

<http://www.whatwg.org/specs/web-apps/current-work/>

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

HTML

Living Standard — Last Updated 2 January 2013



Web developer edition:

<http://developers.whatwg.org/>



Multiple-page

http Too slow? Try reading the multipage copy of
 One-page v the spec instead: <https://web.archive.org>

Participate: /web/20130103030411/http:
 wh_{at} http://whatwg.org/html
 IRC:

FAQ:

<http://wiki.whatwg.org/wiki/FAQ>

Version history:

Twitter updates: <http://twitter.com/WWHATWG>

Commit-Watchers mailing list: <http://lists.whatwg.org/listinfo.cgi/commit-watchers@whatwg.org>

Interactive Web interface: <http://html5.org/tools/web-apps-tracker>

Subversion interface: <http://svn.whatwg.org/>

Issues:

To view the open bugs: HTML bug list

To file bugs, use the "submit review comments" feature at the bottom of the window

To send feedback by e-mail: whatwg@whatwg.org

To view and vote on e-mail feedback: <http://www.whatwg.org/issues/>

Editor:

Ian Hickson, Google, ian@hixie.ch

© Copyright 2004-2011 Apple Computer, Inc., Mozilla Foundation, and Opera Software ASA.
 You are granted a license to use, reproduce and create derivative works of this document.

Table of contents

- 1 Introduction
 - 1.1 Where does this specification fit?
 - 1.2 Is this HTML5?
 - 1.3 Background
 - 1.4 Audience
 - 1.5 Scope
 - 1.6 History
 - 1.7 Design notes
 - 1.7.1 Serializability of script execution
 - 1.7.2 Compliance with other specifications
 - 1.8 HTML vs XHTML
 - 1.9 Structure of this specification
 - 1.9.1 How to read this specification
 - 1.9.2 Typographic conventions
 - 1.10 Privacy concerns
 - 1.11 A quick introduction to HTML
 - 1.11.1 Writing secure applications with HTML
 - 1.11.2 Common pitfalls to avoid when using the scripting APIs
 - 1.12 Conformance requirements for authors
 - 1.12.1 Presentational markup

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 2.1 Terminology
 - 2.1.1 Resources
 - 2.1.2 XML
 - 2.1.3 DOM trees
 - 2.1.4 Scripting
 - 2.1.5 Plugins
 - 2.1.6 Character encodings
- 2.2 Conformance requirements
 - 2.2.1 Conformance classes



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 2.5.4.1 Signed integers
- 2.5.4.2 Non-negative integers
- 2.5.4.3 Floating-point numbers
- 2.5.4.4 Percentages and lengths
- 2.5.4.5 Lists of integers
- 2.5.4.6 Lists of dimensions
- 2.5.5 Dates and times
 - 2.5.5.1 Months
 - 2.5.5.2 Dates
 - 2.5.5.3 Yearless dates
 - 2.5.5.4 Times
 - 2.5.5.5 Local dates and times
 - 2.5.5.6 Time zones
 - 2.5.5.7 Global dates and times
 - 2.5.5.8 Weeks
 - 2.5.5.9 Durations
 - 2.5.5.10 Vaguer moments in time
- 2.5.6 Colors
- 2.5.7 Space-separated tokens
- 2.5.8 Comma-separated tokens
- 2.5.9 References
- 2.5.10 Media queries
- 2.6 URLs
 - 2.6.1 Terminology
 - 2.6.2 Parsing URLs
 - 2.6.3 Base URLs
 - 2.6.4 Resolving URLs
 - 2.6.5 URL manipulation and creation
 - 2.6.6 Dynamic changes to base URLs
 - 2.6.7 Interfaces for URL manipulation
- 2.7 Fetching resources
 - 2.7.1 Terminology
 - 2.7.2 Processing model
 - 2.7.3 Encrypted HTTP and related security concerns
 - 2.7.4 Determining the type of a resource
 - 2.7.5 Extracting character encodings from `meta` elements
 - 2.7.6 CORS settings attributes
 - 2.7.7 CORS-enabled fetch
- 2.8 Common DOM interfaces
 - 2.8.1 Reflecting content attributes in IDL attributes
 - 2.8.2 Collections
 - 2.8.2.1 `HTMLAllCollection`
 - 2.8.2.2 `HTMLFormControlsCollection`
 - 2.8.2.3 `HTMLOptionsCollection`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2.8.7 Garbage collection

2.9 Namespaces

3 Semantics, structure, and APIs of HTML documents

3.1 Documents

3.1.1 The `Document` object

3.1.2 Security

3.1.3 Resource metadata management

3.1.4 DOM tree accessors



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

3.2.3.5 The `xml:base` attribute (XML only)

3.2.3.6 The `dir` attribute

3.2.3.7 The `class` attribute

3.2.3.8 The `style` attribute

3.2.3.9 Embedding custom non-visible data with the `data-*` attributes

3.2.4 Element definitions

3.2.4.1 Attributes

3.2.5 Content models

3.2.5.1 Kinds of content

3.2.5.1.1 Metadata content

3.2.5.1.2 Flow content

3.2.5.1.3 Sectioning content

3.2.5.1.4 Heading content

3.2.5.1.5 Phrasing content

3.2.5.1.6 Embedded content

3.2.5.1.7 Interactive content

3.2.5.1.8 Palpable content

3.2.5.2 Transparent content models

3.2.5.3 Paragraphs

3.2.6 Requirements relating to bidirectional-algorithm formatting characters

3.2.7 WAI-ARIA

3.3 Interactions with XPath and XSLT

3.4 Dynamic markup insertion

3.4.1 Opening the input stream

3.4.2 Closing the input stream

3.4.3 `document.write()`

3.4.4 `document.writeln()`

4 The elements of HTML

4.1 The root element

4.1.1 The `html` element

4.2 Document metadata

4.2.1 The `head` element

4.2.2 The `title` element

4.2.3 The `base` element

4.2.4 The `link` element

4.2.5 The `meta` element

4.2.5.1 Standard metadata names

4.2.5.2 Other metadata names

4.2.5.3 Pragma directives

4.2.5.4 Other pragma directives

4.2.5.5 Specifying the document's character encoding

4.2.6 The `style` element

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4.3.1.2 Restrictions for contents of `script` elements

- 4.3.1.3 Inline documentation for external scripts
- 4.3.1.4 Interaction of `script` elements and XSLT

4.3.2 The `noscript` element

4.4 Sections

- 4.4.1 The `body` element
- 4.4.2 The `article` element
- 4.4.3 The `section` element
- 4.4.4 The `nav` element



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4.5 Grouping content

- 4.5.1 The `p` element
- 4.5.2 The `hr` element
- 4.5.3 The `pre` element
- 4.5.4 The `blockquote` element
- 4.5.5 The `ol` element
- 4.5.6 The `ul` element
- 4.5.7 The `li` element
- 4.5.8 The `dl` element
- 4.5.9 The `dt` element
- 4.5.10 The `dd` element
- 4.5.11 The `figure` element
- 4.5.12 The `figcaption` element
- 4.5.13 The `div` element

4.6 Text-level semantics

- 4.6.1 The `a` element
- 4.6.2 The `em` element
- 4.6.3 The `strong` element
- 4.6.4 The `small` element
- 4.6.5 The `s` element
- 4.6.6 The `cite` element
- 4.6.7 The `q` element
- 4.6.8 The `dfn` element
- 4.6.9 The `abbr` element
- 4.6.10 The `data` element
- 4.6.11 The `time` element
- 4.6.12 The `code` element
- 4.6.13 The `var` element
- 4.6.14 The `samp` element
- 4.6.15 The `kbd` element
- 4.6.16 The `sub` and `sup` elements
- 4.6.17 The `i` element
- 4.6.18 The `b` element
- 4.6.19 The `u` element
- 4.6.20 The `mark` element
- 4.6.21 The `ruby` element
- 4.6.22 The `rt` element
- 4.6.23 The `rp` element
- 4.6.24 The `bdi` element
- 4.6.25 The `bdo` element
- 4.6.26 The `span` element

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 4.7.1 The `ins` element
- 4.7.2 The `del` element
- 4.7.3 Attributes common to `ins` and `del` elements
- 4.7.4 Edits and paragraphs
- 4.7.5 Edits and lists
- 4.7.6 Edits and tables

4.8 Embedded content

4.8.1 The `img` element

4.8.1.1 Requirements for providing text to act as an alternative to images

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

presentation: charts,

presentation: icons, logos

ical effect

text

ation

n no links

4.8.1.1.9 A group of images that form a single larger picture with links

4.8.1.1.10 A key part of the content

4.8.1.1.11 An image not intended for the user

4.8.1.1.12 An image in an e-mail or private document intended for a specific person who is known to be able to view images

4.8.1.1.13 Guidance for markup generators

4.8.1.1.14 Guidance for conformance checkers

4.8.1.2 Adaptive images

4.8.2 The `iframe` element

4.8.3 The `embed` element

4.8.4 The `object` element

4.8.5 The `param` element

4.8.6 The `video` element

4.8.7 The `audio` element

4.8.8 The `source` element

4.8.9 The `track` element

4.8.10 Media elements

4.8.10.1 Error codes

4.8.10.2 Location of the media resource

4.8.10.3 MIME types

4.8.10.4 Network states

4.8.10.5 Loading the media resource

4.8.10.6 Offsets into the media resource

4.8.10.7 Ready states

4.8.10.8 Playing the media resource

4.8.10.9 Seeking

4.8.10.10 Media resources with multiple media tracks

4.8.10.10.1 `AudioTrackList` and `VideoTrackList` objects

4.8.10.10.2 Selecting specific audio and video tracks declaratively

4.8.10.11 Synchronising multiple media elements

4.8.10.11.1 Introduction

4.8.10.11.2 Media controllers

4.8.10.11.3 Assigning a media controller declaratively

4.8.10.12 Timed text tracks

4.8.10.12.1 Text track model

4.8.10.12.2 Sourcing in-band text tracks

4.8.10.12.3 Sourcing out-of-band text tracks

4.8.10.12.4 Guidelines for exposing cues in various formats as text track cues

4.8.10.12.5 Text track API

4.8.10.12.6 Text tracks describing chapters

4.8.10.12.7 Event definitions

4.8.10.13 User interface

4.8.10.14 Time ranges

4.8.10.15 Event definitions

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 4.8.11.1 Proxying canvases to workers
- 4.8.11.2 The 2D rendering context
 - 4.8.11.2.1 Implementation notes
 - 4.8.11.2.2 The canvas state
 - 4.8.11.2.3 DrawingStyle objects
 - 4.8.11.2.4 Line styles
 - 4.8.11.2.5 Text styles
 - 4.8.11.2.6 Building paths

1 0 11 2 7 10 11 12 objects

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 4.8.11.2.17 Compositing
- 4.8.11.2.18 Image smoothing
- 4.8.11.2.19 Shadows
- 4.8.11.2.20 Drawing model
- 4.8.11.2.21 Best practices
- 4.8.11.2.22 Examples
- 4.8.11.3 Pixel density
- 4.8.11.4 Color spaces and color correction
- 4.8.11.5 Serializing bitmaps to a file
- 4.8.11.6 Security with canvas elements
- 4.8.12 The map element
- 4.8.13 The area element
- 4.8.14 Image maps
 - 4.8.14.1 Authoring
 - 4.8.14.2 Processing model
- 4.8.15 MathML
- 4.8.16 SVG
- 4.8.17 Dimension attributes
- 4.9 Tabular data
 - 4.9.1 The table element
 - 4.9.1.1 Techniques for describing tables
 - 4.9.1.2 Techniques for table design
 - 4.9.2 The caption element
 - 4.9.3 The colgroup element
 - 4.9.4 The col element
 - 4.9.5 The tbody element
 - 4.9.6 The thead element
 - 4.9.7 The tfoot element
 - 4.9.8 The tr element
 - 4.9.9 The td element
 - 4.9.10 The th element
 - 4.9.11 Attributes common to td and th elements
 - 4.9.12 Processing model
 - 4.9.12.1 Forming a table
 - 4.9.12.2 Forming relationships between data cells and header cells
- 4.9.13 Table sorting model
- 4.9.14 Examples
- 4.10 Forms
 - 4.10.1 Introduction
 - 4.10.1.1 Writing a form's user interface
 - 4.10.1.2 Implementing the server-side processing for a form
 - 4.10.1.3 Configuring a form to communicate with a server
 - 4.10.1.4 Client-side form validation

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 4.10.3 The `form` element
- 4.10.4 The `fieldset` element
- 4.10.5 The `legend` element
- 4.10.6 The `label` element
- 4.10.7 The `input` element
 - 4.10.7.1 States of the `type` attribute
 - 4.10.7.1.1 Hidden state (`type=hidden`)
 - 4.10.7.1.2 Text (`type=text`) state and Search state (`type=search`)

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 4.10.7.1.12 Local Date and Time state (`type=datetime-local`)
- 4.10.7.1.13 Number state (`type=number`)
- 4.10.7.1.14 Range state (`type=range`)
- 4.10.7.1.15 Color state (`type=color`)
- 4.10.7.1.16 Checkbox state (`type=checkbox`)
- 4.10.7.1.17 Radio Button state (`type=radio`)
- 4.10.7.1.18 File Upload state (`type=file`)
- 4.10.7.1.19 Submit Button state (`type=submit`)
- 4.10.7.1.20 Image Button state (`type=image`)
- 4.10.7.1.21 Reset Button state (`type=reset`)
- 4.10.7.1.22 Button state (`type=button`)
- 4.10.7.2 Implementation notes regarding localization of form controls
- 4.10.7.3 Common `input` element attributes
 - 4.10.7.3.1 The `maxlength` attribute
 - 4.10.7.3.2 The `size` attribute
 - 4.10.7.3.3 The `readonly` attribute
 - 4.10.7.3.4 The `required` attribute
 - 4.10.7.3.5 The `multiple` attribute
 - 4.10.7.3.6 The `pattern` attribute
 - 4.10.7.3.7 The `min` and `max` attributes
 - 4.10.7.3.8 The `step` attribute
 - 4.10.7.3.9 The `list` attribute
 - 4.10.7.3.10 The `placeholder` attribute
- 4.10.7.4 Common `input` element APIs
- 4.10.7.5 Common event behaviors
- 4.10.8 The `button` element
- 4.10.9 The `select` element
- 4.10.10 The `datalist` element
- 4.10.11 The `optgroup` element
- 4.10.12 The `option` element
- 4.10.13 The `textarea` element
- 4.10.14 The `keygen` element
- 4.10.15 The `output` element
- 4.10.16 The `progress` element
- 4.10.17 The `meter` element
- 4.10.18 Form control infrastructure
 - 4.10.18.1 A form control's value
 - 4.10.18.2 Mutability
 - 4.10.18.3 Association of controls and forms
- 4.10.19 Attributes common to form controls
 - 4.10.19.1 Naming form controls: the `name` attribute
 - 4.10.19.2 Submitting element directionality: the `dirname` attribute

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4.10.19.7 Input modalities: the `inputmode` attribute4.10.19.8 Autofilling form controls: the `autocomplete` attribute

4.10.20 APIs for the text field selections

4.10.21 Constraints

4.10.21.1 Definitions

4.10.21.2 Constraint validation

4.10.21.3 The constraint validation API

4.10.21.4 Security

4.10.22 Form submission



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4.11 The details element

4.11.2 The `summary` element4.11.3 The `menu` element4.11.4 The `menuitem` element

4.11.5 Context menus

4.11.5.1 Declaring a context menu

4.11.5.2 Processing model

4.11.6 Commands

4.11.6.1 Facets

4.11.6.2 Using the `a` element to define a command4.11.6.3 Using the `button` element to define a command4.11.6.4 Using the `input` element to define a command4.11.6.5 Using the `option` element to define a command4.11.6.6 Using the `menuitem` element to define a command4.11.6.7 Using the `command` attribute on `menuitem` elements to define a command indirectly4.11.6.8 Using the `accesskey` attribute on a `label` element to define a command4.11.6.9 Using the `accesskey` attribute on a `legend` element to define a command4.11.6.10 Using the `accesskey` attribute to define a command on other elements4.11.7 The `dialog` element

4.11.7.1 Anchor points

4.12 Links

4.12.1 Introduction

4.12.2 Links created by `a` and `area` elements

4.12.3 Following hyperlinks

4.12.4 Downloading resources

4.12.4.1 Hyperlink auditing

4.12.5 Link types

4.12.5.1 Link type "alternate"

4.12.5.2 Link type "author"

4.12.5.3 Link type "bookmark"

4.12.5.4 Link type "help"

4.12.5.5 Link type "icon"

4.12.5.6 Link type "license"

4.12.5.7 Link type "nofollow"

4.12.5.8 Link type "noreferrer"

4.12.5.9 Link type "prefetch"

4.12.5.10 Link type "search"

4.12.5.11 Link type "stylesheet"

4.12.5.12 Link type "tag"

4.12.5.13 Sequential link types

4.12.5.13.1 Link type "next"

4.12.5.13.2 Link type "prev"

4.12.5.14 Other link types

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 4.13.5 Footnotes
- 4.14 Disabled elements
- 4.15 Matching HTML elements using selectors
 - 4.15.1 Case-sensitivity
 - 4.15.2 Pseudo-classes

5 Microdata

- 5.1 Introduction
 - 5.1.1 Overview



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 5.2.4 Values
- 5.2.5 Associating names with items
- 5.2.6 Microdata and other namespaces
- 5.3 Microdata DOM API
- 5.4 Sample microdata vocabularies
 - 5.4.1 vCard
 - 5.4.1.1 Conversion to vCard
 - 5.4.1.2 Examples
 - 5.4.2 vEvent
 - 5.4.2.1 Conversion to iCalendar
 - 5.4.2.2 Examples
 - 5.4.3 Licensing works
 - 5.4.3.1 Examples
- 5.5 Converting HTML to other formats
 - 5.5.1 JSON

6 Loading Web pages

- 6.1 Browsing contexts
 - 6.1.1 Nested browsing contexts
 - 6.1.1.1 Navigating nested browsing contexts in the DOM
 - 6.1.2 Auxiliary browsing contexts
 - 6.1.2.1 Navigating auxiliary browsing contexts in the DOM
 - 6.1.3 Secondary browsing contexts
 - 6.1.4 Security
 - 6.1.5 Groupings of browsing contexts
 - 6.1.6 Browsing context names
- 6.2 The Window object
 - 6.2.1 Security
 - 6.2.2 APIs for creating and navigating browsing contexts by name
 - 6.2.3 Accessing other browsing contexts
 - 6.2.4 Named access on the Window object
 - 6.2.5 Garbage collection and browsing contexts
 - 6.2.6 Closing browsing contexts
 - 6.2.7 Browser interface elements
 - 6.2.8 The WindowProxy object
- 6.3 Origin
 - 6.3.1 Relaxing the same-origin restriction
- 6.4 Sandboxing
- 6.5 Session history and navigation
 - 6.5.1 The session history of browsing contexts
 - 6.5.2 The History interface
 - 6.5.3 The Location interface
 - 6.5.3.1 Security

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- 6.6.3 Page load processing model for XML files
- 6.6.4 Page load processing model for text files
- 6.6.5 Page load processing model for multipart/x-mixed-replace resources
- 6.6.6 Page load processing model for media
- 6.6.7 Page load processing model for content that uses plugins
- 6.6.8 Page load processing model for inline content that doesn't have a DOM
- 6.6.9 Navigating to a fragment identifier
- 6.6.10 History traversal
 - 6.6.10.1 Event definitions



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 6.7.3.1 Some sample manifests
- 6.7.3.2 Writing cache manifests
- 6.7.3.3 Parsing cache manifests
- 6.7.4 Downloading or updating an application cache
- 6.7.5 The application cache selection algorithm
- 6.7.6 Changes to the networking model
- 6.7.7 Expiring application caches
- 6.7.8 Disk space
- 6.7.9 Application cache API
- 6.7.10 Browser state

7 Web application APIs

7.1 Scripting

- 7.1.1 Introduction
- 7.1.2 Enabling and disabling scripting
- 7.1.3 Processing model
 - 7.1.3.1 Definitions
 - 7.1.3.2 Calling scripts
 - 7.1.3.3 Creating scripts
 - 7.1.3.4 Killing scripts
 - 7.1.3.5 Runtime script errors
 - 7.1.3.5.1 Runtime script errors in documents
- 7.1.4 Event loops
 - 7.1.4.1 Definitions
 - 7.1.4.2 Processing model
 - 7.1.4.3 Generic task sources
- 7.1.5 The javascript: URL scheme
- 7.1.6 Events
 - 7.1.6.1 Event handlers
 - 7.1.6.2 Event handlers on elements, Document objects, and Window objects
 - 7.1.6.3 Event firing
 - 7.1.6.4 Events and the Window object

7.2 Base64 utility methods

7.3 Timers

7.4 User prompts

- 7.4.1 Simple dialogs
- 7.4.2 Printing
- 7.4.3 Dialogs implemented using separate documents

7.5 System state and capabilities

- 7.5.1 The Navigator object
 - 7.5.1.1 Client identification
 - 7.5.1.2 Language preferences
 - 7.5.1.3 Custom scheme and content handlers
 - 7.5.1.3.1 Security and privacy
 - 7.5.1.3.2 Sample user interface

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- 8.1 The `hidden` attribute
- 8.2 Inert subtrees
 - 8.2.1 The `inert` attribute
- 8.3 Activation
- 8.4 Focus
 - 8.4.1 Sequential focus navigation and the `tabindex` attribute
 - 8.4.2 Focus management
 - 8.4.3 Document-level focus APIs



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ite

- 8.6.4 Editing APIs
- 8.6.5 Spelling and grammar checking
- 8.7 Drag and drop
 - 8.7.1 Introduction
 - 8.7.2 The drag data store
 - 8.7.3 The `DataTransfer` interface
 - 8.7.3.1 The `DataTransferItemList` interface
 - 8.7.3.2 The `DataTransferItem` interface
 - 8.7.4 The `DragEvent` interface
 - 8.7.5 Drag-and-drop processing model
 - 8.7.6 Events summary
 - 8.7.7 The `draggable` attribute
 - 8.7.8 The `dropzone` attribute
 - 8.7.9 Security risks in the drag-and-drop model

9 Web workers

- 9.1 Introduction
 - 9.1.1 Scope
 - 9.1.2 Examples
 - 9.1.2.1 A background number-crunching worker
 - 9.1.2.2 Worker used for background I/O
 - 9.1.2.3 Shared workers introduction
 - 9.1.2.4 Shared state using a shared worker
 - 9.1.2.5 Delegation
 - 9.1.3 Tutorials
 - 9.1.3.1 Creating a dedicated worker
 - 9.1.3.2 Communicating with a dedicated worker
 - 9.1.3.3 Shared workers
- 9.2 Infrastructure
 - 9.2.1 The global scope
 - 9.2.1.1 The `WorkerGlobalScope` common interface
 - 9.2.1.2 Dedicated workers and the `DedicatedWorkerGlobalScope` interface
 - 9.2.1.3 Shared workers and the `SharedWorkerGlobalScope` interface
 - 9.2.2 The event loop
 - 9.2.3 The worker's lifetime
 - 9.2.4 Processing model
 - 9.2.5 Runtime script errors
 - 9.2.6 Creating workers
 - 9.2.6.1 The `AbstractWorker` abstract interface
 - 9.2.6.2 Dedicated workers and the `Worker` interface
 - 9.2.6.3 Shared workers and the `SharedWorker` interface
- 9.3 APIs available to workers
 - 9.3.1 Importing scripts and libraries

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 10.1 Event definitions
- 10.2 Server-sent events
 - 10.2.1 Introduction
 - 10.2.2 The `EventSource` interface
 - 10.2.3 Processing model
 - 10.2.4 Parsing an event stream
 - 10.2.5 Interpreting an event stream
 - 10.2.6 Notes
 - 10.2.7 Connectionless push and other features



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 10.3.5 Parsing WebSocket URLs
- 10.3.6 Event definitions
- 10.3.7 Garbage collection
- 10.4 Cross-document messaging
 - 10.4.1 Introduction
 - 10.4.2 Security
 - 10.4.2.1 Authors
 - 10.4.2.2 User agents
 - 10.4.3 Posting messages
- 10.5 Channel messaging
 - 10.5.1 Introduction
 - 10.5.1.1 Examples
 - 10.5.1.2 Ports as the basis of an object-capability model on the Web
 - 10.5.1.3 Ports as the basis of abstracting out service implementations
 - 10.5.2 Message channels
 - 10.5.3 Message ports
 - 10.5.4 Broadcasting to many ports
 - 10.5.5 Ports and garbage collection

11 Web storage

- 11.1 Introduction
- 11.2 The API
 - 11.2.1 The `Storage` interface
 - 11.2.2 The `sessionStorage` attribute
 - 11.2.3 The `localStorage` attribute
 - 11.2.3.1 Security
 - 11.2.4 The `storage` event
 - 11.2.4.1 Event definition
 - 11.2.5 Threads
- 11.3 Disk space
- 11.4 Privacy
 - 11.4.1 User tracking
 - 11.4.2 Sensitivity of data
- 11.5 Security
 - 11.5.1 DNS spoofing attacks
 - 11.5.2 Cross-directory attacks
 - 11.5.3 Implementation risks

12 The HTML syntax

- 12.1 Writing HTML documents
 - 12.1.1 The DOCTYPE
 - 12.1.2 Elements
 - 12.1.2.1 Start tags
 - 12.1.2.2 End tags

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 12.1.3.1 Newlines
- 12.1.4 Character references
- 12.1.5 CDATA sections
- 12.1.6 Comments
- 12.2 Parsing HTML documents
 - 12.2.1 Overview of the parsing model
 - 12.2.2 The input byte stream
 - 12.2.2.1 Determining the character encoding



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 12.2.4.1 Data state
- 12.2.4.2 Character reference in data state
- 12.2.4.3 RCDATA state
- 12.2.4.4 Character reference in RCDATA state
- 12.2.4.5 RAWTEXT state
- 12.2.4.6 Script data state
- 12.2.4.7 PLAINTEXT state
- 12.2.4.8 Tag open state
- 12.2.4.9 End tag open state
- 12.2.4.10 Tag name state
- 12.2.4.11 RCDATA less-than sign state
- 12.2.4.12 RCDATA end tag open state
- 12.2.4.13 RCDATA end tag name state
- 12.2.4.14 RAWTEXT less-than sign state
- 12.2.4.15 RAWTEXT end tag open state
- 12.2.4.16 RAWTEXT end tag name state
- 12.2.4.17 Script data less-than sign state
- 12.2.4.18 Script data end tag open state
- 12.2.4.19 Script data end tag name state
- 12.2.4.20 Script data escape start state
- 12.2.4.21 Script data escape start dash state
- 12.2.4.22 Script data escaped state
- 12.2.4.23 Script data escaped dash state
- 12.2.4.24 Script data escaped dash dash state
- 12.2.4.25 Script data escaped less-than sign state
- 12.2.4.26 Script data escaped end tag open state
- 12.2.4.27 Script data escaped end tag name state
- 12.2.4.28 Script data double escape start state
- 12.2.4.29 Script data double escaped state
- 12.2.4.30 Script data double escaped dash state
- 12.2.4.31 Script data double escaped dash dash state
- 12.2.4.32 Script data double escaped less-than sign state
- 12.2.4.33 Script data double escape end state
- 12.2.4.34 Before attribute name state
- 12.2.4.35 Attribute name state
- 12.2.4.36 After attribute name state
- 12.2.4.37 Before attribute value state
- 12.2.4.38 Attribute value (double-quoted) state
- 12.2.4.39 Attribute value (single-quoted) state
- 12.2.4.40 Attribute value (unquoted) state
- 12.2.4.41 Character reference in attribute value state
- 12.2.4.42 After attribute value (quoted) state
- 12.2.4.43 Self-closing start tag state
- 12.2.4.44 Bogus comment state

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- 12.2.4.50 Comment end state
- 12.2.4.51 Comment end bang state
- 12.2.4.52 DOCTYPE state
- 12.2.4.53 Before DOCTYPE name state
- 12.2.4.54 DOCTYPE name state
- 12.2.4.55 After DOCTYPE name state
- 12.2.4.56 After DOCTYPE public keyword state
- 12.2.4.57 Before DOCTYPE public identifier state

12.2.4.58 DOCTYPE public identifier (double-quoted) state



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 12.2.4.68 CDATA section state
- 12.2.4.69 Tokenizing character references
- 12.2.5 Tree construction
 - 12.2.5.1 Creating and inserting elements
 - 12.2.5.2 Closing elements that have implied end tags
 - 12.2.5.3 Foster parenting
 - 12.2.5.4 The rules for parsing tokens in HTML content
 - 12.2.5.4.1 The "initial" insertion mode
 - 12.2.5.4.2 The "before html" insertion mode
 - 12.2.5.4.3 The "before head" insertion mode
 - 12.2.5.4.4 The "in head" insertion mode
 - 12.2.5.4.5 The "in head noscript" insertion mode
 - 12.2.5.4.6 The "after head" insertion mode
 - 12.2.5.4.7 The "in body" insertion mode
 - 12.2.5.4.8 The "text" insertion mode
 - 12.2.5.4.9 The "in table" insertion mode
 - 12.2.5.4.10 The "in table text" insertion mode
 - 12.2.5.4.11 The "in caption" insertion mode
 - 12.2.5.4.12 The "in column group" insertion mode
 - 12.2.5.4.13 The "in table body" insertion mode
 - 12.2.5.4.14 The "in row" insertion mode
 - 12.2.5.4.15 The "in cell" insertion mode
 - 12.2.5.4.16 The "in select" insertion mode
 - 12.2.5.4.17 The "in select in table" insertion mode
 - 12.2.5.4.18 The "after body" insertion mode
 - 12.2.5.4.19 The "in frameset" insertion mode
 - 12.2.5.4.20 The "after frameset" insertion mode
 - 12.2.5.4.21 The "after after body" insertion mode
 - 12.2.5.4.22 The "after after frameset" insertion mode
 - 12.2.5.5 The rules for parsing tokens in foreign content
- 12.2.6 The end
- 12.2.7 Coercing an HTML DOM into an infoset
- 12.2.8 An introduction to error handling and strange cases in the parser
 - 12.2.8.1 Misnested tags: <i></i>
 - 12.2.8.2 Misnested tags: <p></p>
 - 12.2.8.3 Unexpected markup in tables
 - 12.2.8.4 Scripts that modify the page as it is being parsed
 - 12.2.8.5 The execution of scripts that are moving across multiple documents
 - 12.2.8.6 Unclosed formatting elements
- 12.3 Serializing HTML fragments
- 12.4 Parsing HTML fragments
- 12.5 Named character references
- 13 The XHTML syntax
 - 13.1 Writing XHTML documents

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- 14.1 Introduction
- 14.2 The CSS user agent style sheet and presentational hints
- 14.3 Non-replaced elements
 - 14.3.1 Hidden elements
 - 14.3.2 The page
 - 14.3.3 Flow content
 - 14.3.4 Phrasing content
 - 14.3.5 Bidirectional text
 - 14.3.6 Quotes



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- 14.4.3 Attributes for embedded content and images
- 14.4.4 Image maps
- 14.5 Bindings
 - 14.5.1 Introduction
 - 14.5.2 The `button` element
 - 14.5.3 The `details` element
 - 14.5.4 The `input` element as a text entry widget
 - 14.5.5 The `input` element as domain-specific widgets
 - 14.5.6 The `input` element as a range control
 - 14.5.7 The `input` element as a color well
 - 14.5.8 The `input` element as a checkbox and radio button widgets
 - 14.5.9 The `input` element as a file upload control
 - 14.5.10 The `input` element as a button
 - 14.5.11 The `marquee` element
 - 14.5.12 The `meter` element
 - 14.5.13 The `progress` element
 - 14.5.14 The `select` element
 - 14.5.15 The `textarea` element
 - 14.5.16 The `keygen` element
- 14.6 Frames and framesets
- 14.7 Interactive media
 - 14.7.1 Links, forms, and navigation
 - 14.7.2 The `title` attribute
 - 14.7.3 Editing hosts
 - 14.7.4 Text rendered in native user interfaces
- 14.8 Print media
- 14.9 Unstyled XML documents

15 Obsolete features

- 15.1 Obsolete but conforming features
 - 15.1.1 Warnings for obsolete but conforming features
- 15.2 Non-conforming features
- 15.3 Requirements for implementations
 - 15.3.1 The `applet` element
 - 15.3.2 The `marquee` element
 - 15.3.3 Frames
 - 15.3.4 Other elements, attributes and APIs

16 IANA considerations

- 16.1 `text/html`
- 16.2 `multipart/x-mixed-replace`
- 16.3 `application/xhtml+xml`
- 16.4 `application/x-www-form-urlencoded`

3,445 captures

5 Jun 2004 - 22 Aug 2020

16.9 Ping-10
16.10 web+ scheme prefix

Go NOV JAN FEB
◀ 03 ▶
2012 2013 2014

About this capture

Index

- [Elements](#)
- [Element content categories](#)
- [Attributes](#)
- [Element Interfaces](#)
- [All Interfaces](#)



Ref

Ack Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

This specification defines a big part of the Web platform, in lots of detail. Its place in the Web platform Specification Stack relative to other specifications can be best summed up as follows:



1.2 Is this HTML5?

This section is non-normative.

In short: Yes.

In more length: The term "HTML5" is widely used as a buzzword to refer to modern Web technologies, many of which (though by no means all) are developed at the WHATWG, in some cases in conjunction with the W3C and IETF.

The WHATWG work is all published in one specification (the one you are reading right now), parts of which are republished in an edition optimized for Web developers (known as HTML5).

The W3C also publishes parts of this specification as separate documents. One of these parts is called "HTML5"; it is a subset of this specification (the HTML Living Standard) as it stood in late June 2012.

1.2.1 How do the WHATWG and W3C specifications differ?

The features present in both the WHATWG and W3C specifications are specified using identical text, except for the following differences:

- The W3C HTML specification does not include this section.
- The W3C HTML specification lacks some of the fixes made to this specification since late June 2012. Specifically, after revision r7148 changes were only adopted into the W3C HTML specification on a piecemeal basis. An approximate list of fixes that have not been included in the W3C specification is available. (There is also a list of changes that have been made to the W3C version since r7148.)
- The W3C HTML specification refers to the technology as HTML5, rather than just HTML.
- The W3C HTML specification omits mentions of microdata.
- The W3C HTML specification defines conformance for documents in a more traditional (version-orientated) way, because of a working group decision from March 2011. This specification, in part driven by its versionless development model, instead uses a conformance definition that more closely models how specifications are used in practice.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

- The W3C HTML specification gives incomplete advice regarding the `alt` attribute and instead references other documents on the matter because of a working group decision from March 2011.

- The W3C HTML specification includes a link to an incomplete document that contradicts this specification because of a working group decision from February 2011.

- The W3C HTML specification has different prose regarding the use of tables for layout purposes because of a working group decision from March 2011. In contrast, this specification unambiguously disallows the use of `table` elements for layout purposes.

- The **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

- The **/web/20130103030411/http://whatwg.org/html**

- The **authors**

- The W3C HTML specification is missing some conformance constraints that would make documents misusing `canvas` invalid, because of a working group chair decision from June 2011.

- The W3C HTML specification does not define the term sidebar hyperlink, forcing the definition of the `sidebar` link type in the microformats wiki to rely on this copy of the spec instead, because of a working group chair decision from November 2011.

- The W3C HTML specification omits a number of suggestions regarding using the `title` attribute, and makes using the `title` attribute for captions non-conforming in certain specific cases, because of a number of working group chair decisions from March 2012: first, second, third.

- The W3C HTML specification omits some advice about how to remove focus outlines in an accessible manner (instead only urging authors not to remove them without giving an alternative to address the "they're ugly" use case), because of a working group chair decision from March 2012.

- The W3C HTML specification uses the term "caption file" in place of "WebVTT file" in certain places, because of a working group chair decision from March 2012.

- The W3C HTML specification redundantly repeats requirements from the ARIA specification relating to the ARIA attributes, due to a working group chair decision.

- In the W3C 2D Context specification, the `createPattern()` method's definition treats a null second argument as the empty string, like IE and WebKit, rather than as an unknown value, like Opera and Firefox. (This specification treats it as an unknown value, since there doesn't seem to be any value to treating it as the empty string.)

- The W3C HTML specification sometimes does not give the official Unicode names of characters, due to a working group chair decision.

- The W3C HTML specification occasionally replaces "social commentary" in the spec's non-normative prose with more bland text.

- The W3C HTML specification allows `rel` attributes to contain unregistered URLs as tokens, even though they are not treated as URLs (e.g. they are handled case-insensitively), due to a working group chair decision.

- The W3C HTML specification has a nonsensical content model for the `object` element, due to a working group chair decision from July 2012.

- The W3C HTML specification has a different approach to topics relating to registering extensibility tokens.

- The W3C HTML specification avoids mentioning `MouseEventInit`, resulting in a different definition for `DragEventInit`.

- The W3C HTML specification has slightly less accurate prose, slightly more inconsistent style, and has some more bugs in examples in a few places.

- The W3C HTML specification does not handle high-density backing stores for `canvas` in a manner compatible with existing content.

The following sections are only published in the WHATWG specifications and are not currently available anywhere else:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- The `dialog` keyword on the `form` element's `method` attribute, for a simpler way to close dialogs.
- The `cssElementMap` feature for defining CSS element reference identifiers.
- Some predefined sample Microdata vocabularies.
- The `data` element for marking up machine-readable data.
- The application cache feature's prefer-online mode.

• The **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**



ast.

1.3 Background

This section is non-normative.

The World Wide Web's markup language has always been HTML. HTML was primarily designed as a language for semantically describing scientific documents, although its general design and adaptations over the years have enabled it to be used to describe a number of other types of documents.

The main area that has not been adequately addressed by HTML is a vague subject referred to as Web Applications. This specification attempts to rectify this, while at the same time updating the HTML specifications to address issues raised in the past few years.

1.4 Audience

This section is non-normative.

This specification is intended for authors of documents and scripts that use the features defined in this specification, implementors of tools that operate on pages that use the features defined in this specification, and individuals wishing to establish the correctness of documents or implementations with respect to the requirements of this specification.

This document is probably not suited to readers who do not already have at least a passing familiarity with Web technologies, as in places it sacrifices clarity for precision, and brevity for completeness. More approachable tutorials and authoring guides can provide a gentler introduction to the topic.

In particular, familiarity with the basics of DOM is necessary for a complete understanding of some of the more technical parts of this specification. An understanding of Web IDL, HTTP, XML, Unicode, character encodings, JavaScript, and CSS will also be helpful in places but is not essential.

1.5 Scope

This section is non-normative.

This specification is limited to providing a semantic-level markup language and associated semantic-level scripting APIs for authoring accessible pages on the Web ranging from static documents to dynamic applications.

The scope of this specification does not include providing mechanisms for media-specific customization of presentation (although default rendering rules for Web browsers are included at the end of this specification, and several mechanisms for hooking into CSS are provided as part of the language).

The scope of this specification is not to describe an entire operating system. In particular, hardware configuration software, image manipulation tools, and applications that users would be expected to use with high-end workstations on a daily basis are out of scope. In terms of applications, this specification is targeted specifically at applications that would be expected to be used by users on an occasional basis, or regularly but from disparate locations, with low CPU requirements. Examples of such applications include online purchasing systems, searching systems, games (especially multiplayer online games), public telephone books or address books, communications software (e-mail clients, instant messaging clients, discussion software), document editing software, etc.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

hosted first at CERN, and then at the IETF.

With the creation of the W3C, HTML's development changed venue again. A first abortive attempt at extending HTML in 1995 known as HTML 3.0 then made way to a more pragmatic approach known as HTML 3.2, which was completed in 1997. HTML4 quickly followed later that same year.

The following year, the W3C membership decided to stop evolving HTML and instead begin work on an XML-based equivalent, called XHTML. This effort started with a reformulation of HTML4 in XML, known as XHTML 1.0, which added no new features except the new serialization, and which was completed in 2000. After XHTML 1.0, the W3C's focus turned to making it easier for other working groups to extend XHTML on a new language that was not compatible with HTML.

Around the time and publisher culminating in being closed off, the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

In 2003, the interest in evolving HTML

deployment as a web technology was limited to entirely new technologies (like RSS and later Atom), rather than as a replacement for existing deployed technologies (like HTML).

A proof of concept to show that it was possible to extend HTML4's forms to provide many of the features that XForms 1.0 introduced, without requiring browsers to implement rendering engines that were incompatible with existing HTML Web pages, was the first result of this renewed interest. At this early stage, while the draft was already publicly available, and input was already being solicited from all sources, the specification was only under Opera Software's copyright.

The idea that HTML's evolution should be reopened was tested at a W3C workshop in 2004, where some of the principles that underlie the HTML5 work (described below), as well as the aforementioned early draft proposal covering just forms-related features, were presented to the W3C jointly by Mozilla and Opera. The proposal was rejected on the grounds that the proposal conflicted with the previously chosen direction for the Web's evolution; the W3C staff and membership voted to continue developing XML-based replacements instead.

Shortly thereafter, Apple, Mozilla, and Opera jointly announced their intent to continue working on the effort under the umbrella of a new venue called the WHATWG. A public mailing list was created, and the draft was moved to the WHATWG site. The copyright was subsequently amended to be jointly owned by all three vendors, and to allow reuse of the specification.

The WHATWG was based on several core principles, in particular that technologies need to be backwards compatible, that specifications and implementations need to match even if this means changing the specification rather than the implementations, and that specifications need to be detailed enough that implementations can achieve complete interoperability without reverse-engineering each other.

The latter requirement in particular required that the scope of the HTML5 specification include what had previously been specified in three separate documents: HTML4, XHTML1, and DOM2 HTML. It also meant including significantly more detail than had previously been considered the norm.

In 2006, the W3C indicated an interest to participate in the development of HTML5 after all, and in 2007 formed a working group chartered to work with the WHATWG on the development of the HTML5 specification. Apple, Mozilla, and Opera allowed the W3C to publish the specification under the W3C copyright, while keeping a version with the less restrictive license on the WHATWG site.

For a number of years, both groups then worked together. In 2011, however, the groups came to the conclusion that they had different goals: the W3C wanted to publish a "finished" version of "HTML5", while the WHATWG wanted to continue working on a Living Standard for HTML, continuously maintaining the specification rather than freezing it in a state with known problems, and adding new features as needed to evolve the platform.

Since then, the WHATWG has been working on this specification (amongst others), and the W3C has been copying fixes made by the WHATWG into their fork of the document, as well as making other changes, many of which are described above.

A separate document has been published by the W3C HTML working group to document the differences between the HTML specified in this document and the language described in the HTML4 specification. [HTMLDIFF]

1.7 Design notes

This section is non-normative.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Features have thus arisen from many sources, and have not always been designed in especially consistent ways. Furthermore, because of the unique characteristics of the Web, implementation bugs have often become de-facto, and now de-jure, standards, as content is often unintentionally written in ways that rely on them before they can be fixed.

Despite all this, efforts have been made to adhere to certain design goals. These are described in the next few subsections.

1.7.1 Serializability of script execution

This section is non-normative.

To avoid expensive detection of a thought of a: **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

Note: This section is non-normative. It is provided for information only and does not define any requirements for implementors. It is not part of the specification and is not subject to review or approval by the WHATWG. It is intended to be removed from the specification in the future.

ed such that no script can be run without causing other scripts to

1.7.2 Compliance with other specifications

This section is non-normative.

This specification interacts with and relies on a wide variety of other specifications. In certain circumstances, unfortunately, conflicting needs have led to this specification violating the requirements of these other specifications. Whenever this has occurred, the transgressions have each been noted as a "willful violation", and the reason for the violation has been noted.

1.8 HTML vs XHTML

This section is non-normative.

This specification defines an abstract language for describing documents and applications, and some APIs for interacting with in-memory representations of resources that use this language.

The in-memory representation is known as "DOM HTML", or "the DOM" for short.

There are various concrete syntaxes that can be used to transmit resources that use this abstract language, two of which are defined in this specification.

The first such concrete syntax is the HTML syntax. This is the format suggested for most authors. It is compatible with most legacy Web browsers. If a document is transmitted with the `text/html` MIME type, then it will be processed as an HTML document by Web browsers. This specification defines the latest HTML syntax, known simply as "HTML".

The second concrete syntax is the XHTML syntax, which is an application of XML. When a document is transmitted with an XML MIME type, such as `application/xhtml+xml`, then it is treated as an XML document by Web browsers, to be parsed by an XML processor. Authors are reminded that the processing for XML and HTML differs; in particular, even minor syntax errors will prevent a document labeled as XML from being rendered fully, whereas they would be ignored in the HTML syntax. This specification defines the latest XHTML syntax, known simply as "XHTML".

The DOM, the HTML syntax, and the XHTML syntax cannot all represent the same content. For example, namespaces cannot be represented using the HTML syntax, but they are supported in the DOM and in the XHTML syntax. Similarly, documents that use the `noscript` feature can be represented using the HTML syntax, but cannot be represented with the DOM or in the XHTML syntax. Comments that contain the string "`-->`" can only be represented in the DOM, not in the HTML and XHTML syntaxes.

1.9 Structure of this specification

This section is non-normative.

This specification is divided into the following major sections:

Introduction

Non-normative materials providing a context for the HTML standard.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

as well as introducing the features common to all elements, and the concepts used in defining elements.

The elements of HTML

Each element has a predefined meaning, which is explained in this section. Rules for authors on how to use the element, along with user agent requirements for how to handle each element, are also given. This includes large signature features of HTML such as video playback and subtitles, form controls and form submission, and a 2D graphics API known as the HTML canvas.

Microdata

This specification introduces a mechanism for adding machine-readable annotations to documents, so that tools can extract trees algorithms that can be used to catalogues for contact

con info

Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org>

Loading We

HTM

mul

/web/20130103030411/http:

Web applic: //whatwg.org/html

This

environments that deal with

User interaction

HTML documents can provide a number of mechanisms for users to interact with and modify content, which are described in this section, such as how focus works, and drag-and-drop.

Web workers

This section defines an API for background threads in JavaScript.

The communication APIs

This section describes some mechanisms that applications written in HTML can use to communicate with other applications from different domains running on the same client. It also introduces a server-push event stream mechanism known as Server Sent Events or EventSource, and a two-way full-duplex socket protocol for scripts known as Web Sockets.

Web storage

This section defines a client-side storage mechanism based on name-value pairs.

The HTML syntax

The XHTML syntax

All of these features would be for naught if they couldn't be represented in a serialized form and sent to other people, and so these sections define the syntaxes of HTML and XHTML, along with rules for how to parse content using those syntaxes.

Rendering

This section defines the default rendering rules for Web browsers.

There are also some appendices, listing obsolete features and IANA considerations, and several indices.

1.9.1 How to read this specification

This specification should be read like all other specifications. First, it should be read cover-to-cover, multiple times. Then, it should be read backwards at least once. Then it should be read by picking random sections from the contents list and following all the cross-references.

As described in the conformance requirements section below, this specification describes conformance criteria for a variety of conformance classes. In particular, there are conformance requirements that apply to *producers*, for example authors and the documents they create, and there are conformance requirements that apply to *consumers*, for example Web browsers. They can be distinguished by what they are requiring: a requirement on a producer states what is allowed, while a requirement on a consumer states how software is to act.

For example, "the `foo` attribute's value must be a valid integer" is a requirement on producers, as it lays out the allowed values; in contrast, the requirement "the `foo` attribute's value must be parsed using the rules for parsing integers" is a requirement on consumers, as it describes how to process the content.

Requirements on producers have no bearing whatsoever on consumers.

Continuing the above example, a requirement stating that a particular attribute's value is constrained to being a valid integer emphatically does *not* imply anything about the requirements on consumers. It might be that the consumers are in fact required

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1.9.2 Typographic conventions

This is a definition, requirement, or explanation.

Note: *This is a note.*

|| This is an example.

This is an optional note.

⚠ Warning! Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

variable = object . method([optionalArgument])

This is a note to authors describing the usage of an interface.

/* this is a CSS fragment */

The defining instance of a term is marked up like **this**. Uses of that term are marked up like `this` or like *this*.

The defining instance of an element, attribute, or API is marked up like **this**. References to that element, attribute, or API are marked up like `this`.

Other code fragments are marked up like `this`.

Variables are marked up like *this*.

This is an implementation requirement.

1.10 Privacy concerns

This section is non-normative.

Some features of HTML trade user convenience for a measure of user privacy.

In general, due to the Internet's architecture, a user can be distinguished from another by the user's IP address. IP addresses do not perfectly match to a user; as a user moves from device to device, or from network to network, their IP address will change; similarly, NAT routing, proxy servers, and shared computers enable packets that appear to all come from a single IP address to actually map to multiple users. Technologies such as onion routing can be used to further anonymize requests so that requests from a single user at one node on the Internet appear to come from many disparate parts of the network.

However, the IP address used for a user's requests is not the only mechanism by which a user's requests could be related to each other. Cookies, for example, are designed specifically to enable this, and are the basis of most of the Web's session features that enable you to log into a site with which you have an account.

There are other mechanisms that are more subtle. Certain characteristics of a user's system can be used to distinguish groups of users from each other; by collecting enough such information, an individual user's browser's "digital fingerprint" can be computed, which can be as good, if not better, as an IP address in ascertaining which requests are from the same user.

Grouping requests in this manner, especially across multiple sites, can be used for both benign (and even arguably positive) purposes, as well as for malevolent purposes. An example of a reasonably benign purpose would be determining whether a particular person seems to prefer sites with dog illustrations as opposed to sites with cat illustrations (based on how often they visit the sites in question) and then automatically using the preferred illustrations on subsequent visits to participating sites. Malevolent purposes, however, could include governments combining information such as the person's home address (determined from the addresses they use when getting driving directions on one site) with their apparent political affiliations (determined by examining the forum sites that they participate in).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

Unfortunately, as the first paragraph in this section implies, sometimes there is great benefit to be derived from exposing the very information that can also be used for fingerprinting purposes, so it's not as easy as simply blocking all possible leaks. For instance, the ability to log into a site to post under a specific identity requires that the user's requests be identifiable as all being from the same user, more or less by definition. More subtly, though, information such as how wide text is, which is necessary for many effects that involve drawing text onto a canvas (e.g. any effect that involves drawing a border around the text) also leaks information that can be used to group a user's requests. (In this case, by potentially exposing, via a brute force search, which fonts a user has installed, information which can vary considerably from user to user.)

Features in this specification which can be used to fingerprint the user are marked as this paragraph is.

Other features

- The [Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org](#)
- The [Feature /web/20130103030411/http://whatwg.org/html](#)
- The [CSSOMVIEW](#)



1.11 A quick introduction to HTML

This section is non-normative.

A basic HTML document looks like this:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>Sample page</h1>
    <p>This is a <a href="demo.html">simple</a> sample.</p>
    <!-- this is a comment -->
  </body>
</html>
```

HTML documents consist of a tree of elements and text. Each element is denoted in the source by a start tag, such as "<body>", and an end tag, such as "</body>". (Certain start tags and end tags can in certain cases be omitted and are implied by other tags.)

Tags have to be nested such that elements are all completely within each other, without overlapping:

```
<p>This is <em>very <strong>wrong</em></strong></p>
<p>This <em>is <strong>correct</strong>.</em></p>
```

This specification defines a set of elements that can be used in HTML, along with rules about the ways in which the elements can be nested.

Elements can have attributes, which control how the elements work. In the example below, there is a hyperlink, formed using the `a` element and its `href` attribute:

```
<a href="demo.html">simple</a>
```

Attributes are placed inside the start tag, and consist of a name and a value, separated by an "=" character. The attribute value can remain unquoted if it doesn't contain space characters or any of " ' ` = < or >. Otherwise, it has to be quoted using either single or double quotes. The value, along with the "=" character, can be omitted altogether if the value is the empty string.

```
<!-- empty attributes -->
<input name=address disabled>
<input name=address disabled="">

<!-- attributes with a value -->
<input name=address maxlength=200>
<input name=address maxlength='200'>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

DOM trees contain several kinds of nodes, in particular a `DocumentType` node, Element nodes, Text nodes, Comment nodes, and in some cases ProcessingInstruction nodes.

The markup snippet at the top of this section would be turned into the following DOM tree:



The root element of this tree is the `html` element, which is the element always found at the root of HTML documents. It contains two elements, `head` and `body`, as well as a `Text` node between them.

There are many more `Text` nodes in the DOM tree than one would initially expect, because the source contains a number of spaces (represented here by " ") and line breaks ("") that all end up as `Text` nodes in the DOM. However, for historical reasons not all of the spaces and line breaks in the original markup appear in the DOM. In particular, all the whitespace before `head` start tag ends up being dropped silently, and all the whitespace after the `body` end tag ends up placed at the end of the `body`.

The `head` element contains a `title` element, which itself contains a `Text` node with the text "Sample page". Similarly, the `body` element contains an `h1` element, a `p` element, and a comment.

This DOM tree can be manipulated from scripts in the page. Scripts (typically in JavaScript) are small programs that can be embedded using the `script` element or using event handler content attributes. For example, here is a form with a script that sets the value of the form's `output` element to say "Hello World":

```

<form name="main">
  Result: <output name="result"></output>
  <script>
    document.forms.main.elements.result.value = 'Hello World';
  </script>
</form>
  
```

Each element in the DOM tree is represented by an object, and these objects have APIs so that they can be manipulated. For instance, a link (e.g. the `a` element in the tree above) can have its `"href"` attribute changed in several ways:

```

var a = document.links[0]; // obtain the first link in the document
a.href = 'sample.html'; // change the destination URL of the link
a.protocol = 'https'; // change just the scheme part of the URL
a.setAttribute('href', 'http://example.com/'); // change the content attribute directly
  
```

Since DOM trees are used as the way to represent HTML documents when they are processed and presented by implementations (especially interactive implementations like Web browsers), this specification is mostly phrased in terms of DOM trees, instead of the markup described above.

HTML documents represent a media-independent description of interactive content. HTML documents might be rendered to a screen,

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample styled page</title>
    <style>
      body { background: navy; color: yellow; }
    </style>
  </head>
  <body>
    <h1>S
    <p>T
  </body>
</html>
```

For more detailed specification level of data
[/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

examples included in this defines the language with a



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1.11.1 Writing secure applications with HTML

This section is non-normative.

When HTML is used to create interactive sites, care needs to be taken to avoid introducing vulnerabilities through which attackers can compromise the integrity of the site itself or of the site's users.

A comprehensive study of this matter is beyond the scope of this document, and authors are strongly encouraged to study the matter in more detail. However, this section attempts to provide a quick introduction to some common pitfalls in HTML application development.

The security model of the Web is based on the concept of "origins", and correspondingly many of the potential attacks on the Web involve cross-origin actions. [ORIGIN]

Not validating user input

Cross-site scripting (XSS)

SQL injection

When accepting untrusted input, e.g. user-generated content such as text comments, values in URL parameters, messages from third-party sites, etc, it is imperative that the data be validated before use, and properly escaped when displayed. Failing to do this can allow a hostile user to perform a variety of attacks, ranging from the potentially benign, such as providing bogus user information like a negative age, to the serious, such as running scripts every time a user looks at a page that includes the information, potentially propagating the attack in the process, to the catastrophic, such as deleting all data in the server.

When writing filters to validate user input, it is imperative that filters always be whitelist-based, allowing known-safe constructs and disallowing all other input. Blacklist-based filters that disallow known-bad inputs and allow everything else are not secure, as not everything that is bad is yet known (for example, because it might be invented in the future).

For example, suppose a page looked at its URL's query string to determine what to display, and the site then redirected the user to that page to display a message, as in:

```
<ul>
  <li><a href="message.cgi?say=Hello">Say Hello</a>
  <li><a href="message.cgi?say=Welcome">Say Welcome</a>
  <li><a href="message.cgi?say=Kittens">Say Kittens</a>
</ul>
```

If the message was just displayed to the user without escaping, a hostile attacker could then craft a URL that contained a script element:

```
http://example.com/message.cgi?say=%3Cscript%3Ealert%28%27oh%20no%21%27%29%3C
/script%3E
```

If the attacker then convinced a victim user to visit this page, a script of the attacker's choosing would run on the page. Such a script could do any number of hostile actions, limited only by what the site offers: if the site is an e-commerce shop, for instance, such a script could cause the user to unknowingly make arbitrarily many unwanted purchases.

This is called a cross-site scripting attack.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

- When allowing URLs to be provided (e.g. for links), the scheme of each URL also needs to be explicitly whitelisted, as there are many schemes that can be abused. The most prominent example is "javascript:", but user agents can implement (and indeed, have historically implemented) others.
- Allowing a `base` element to be inserted means any `script` elements in the page with relative links can be hijacked, and similarly that any form submissions can get redirected to a hostile site.

Cross-site request forgery (CSRF)

If a site is slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

This is a site requirement for Clickjacking.

Origin headers on all requests.

Clickjacking is a page that provides users with an interface to perform actions that the user might not wish to perform needs to be designed so as to avoid the possibility that users can be tricked into activating the interface.

One way that a user could be so tricked is if a hostile site places the victim site in a small `iframe` and then convinces the user to click, for instance by having the user play a reaction game. Once the user is playing the game, the hostile site can quickly position the `iframe` under the mouse cursor just as the user is about to click, thus tricking the user into clicking the victim site's interface.

To avoid this, sites that do not expect to be used in frames are encouraged to only enable their interface if they detect that they are not in a frame (e.g. by comparing the `window` object to the value of the `top` attribute).

1.11.2 Common pitfalls to avoid when using the scripting APIs

This section is non-normative.

Scripts in HTML have "run-to-completion" semantics, meaning that the browser will generally run the script uninterrupted before doing anything else, such as firing further events or continuing to parse the document.

On the other hand, parsing of HTML files happens asynchronously and incrementally, meaning that the parser can pause at any point to let scripts run. This is generally a good thing, but it does mean that authors need to be careful to avoid hooking event handlers after the events could have possibly fired.

There are two techniques for doing this reliably: use event handler content attributes, or create the element and add the event handlers in the same script. The latter is safe because, as mentioned earlier, scripts are run to completion before further events can fire.

One way this could manifest itself is with `img` elements and the `load` event. The event could fire as soon as the element has been parsed, especially if the image has already been cached (which is common).

Here, the author uses the `onload` handler on an `img` element to catch the `load` event:

```

```

If the element is being added by script, then so long as the event handlers are added in the same script, the event will still not be missed:

```
<script>
  var img = new Image();
  img.src = 'games.png';
  img.alt = 'Games';
  img.onload = gamesLogoHasLoaded;
  // img.addEventListener('load', gamesLogoHasLoaded, false); // would work also
</script>
```

However, if the author first created the `img` element and then in a separate script added the event listeners, there's a chance that the `load` event would be fired in between, leading it to be missed:

```
<!-- Do not use this style, it has a race condition! -->
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
|| img.onload = gamesLogoHasLoaded; // might never fire!
</script>
```

1.12 Conformance requirements for authors

This section is non-normative.

Unlike previous versions of the HTML specification, this specification defines in some detail the required processing for invalid documents.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1.12.1 Pre

This section is non-normative.

The majority of presentational features from previous versions of HTML are no longer allowed. Presentational markup in general has been found to have a number of problems:

The use of presentational elements leads to poorer accessibility

While it is possible to use presentational markup in a way that provides users of assistive technologies (ATs) with an acceptable experience (e.g. using ARIA), doing so is significantly more difficult than doing so when using semantically-appropriate markup. Furthermore, even using such techniques doesn't help make pages accessible for non-AT non-graphical users, such as users of text-mode browsers.

Using media-independent markup, on the other hand, provides an easy way for documents to be authored in such a way that they work for more users (e.g. text browsers).

Higher cost of maintenance

It is significantly easier to maintain a site written in such a way that the markup is style-independent. For example, changing the color of a site that uses `` throughout requires changes across the entire site, whereas a similar change to a site based on CSS can be done by changing a single file.

Larger document sizes

Presentational markup tends to be much more redundant, and thus results in larger document sizes.

For those reasons, presentational markup has been removed from HTML in this version. This change should not come as a surprise; HTML4 deprecated presentational markup many years ago and provided a mode (HTML4 Transitional) to help authors move away from presentational markup; later, XHTML 1.1 went further and obsoleted those features altogether.

The only remaining presentational markup features in HTML are the `style` attribute and the `style` element. Use of the `style` attribute is somewhat discouraged in production environments, but it can be useful for rapid prototyping (where its rules can be directly moved into a separate style sheet later) and for providing specific styles in unusual cases where a separate style sheet would be inconvenient. Similarly, the `style` element can be useful in syndication or for page-specific styles, but in general an external style sheet is likely to be more convenient when the styles apply to multiple pages.

It is also worth noting that some elements that were previously presentational have been redefined in this specification to be media-independent: `b`, `i`, `hr`, `s`, `small`, and `u`.

1.12.2 Syntax errors

This section is non-normative.

The syntax of HTML is constrained to avoid a wide variety of problems.

Unintuitive error-handling behavior

Certain invalid syntax constructs, when parsed, result in DOM trees that are highly unintuitive.

```
|| For example, the following markup fragment results in a DOM with an hr element that is an earlier sibling of the
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

To allow user agents to be used in controlled environments without having to implement the more bizarre and convoluted error handling rules, user agents are permitted to fail whenever encountering a parse error.

Errors where the error-handling behavior is not compatible with streaming user agents

Some error-handling behavior, such as the behavior for the `<table><hr>...` example mentioned above, are incompatible with streaming user agents (user agents that process HTML files in one pass, without storing state). To avoid interoperability problems with such user agents, any syntax resulting in such behavior is considered invalid.

Errors that can result in infoset coercion

When such constructs are encountered, user agents must reconstruct the document tree. This is because different variants that XML enforces, such as well-formedness, can require different handling. Some of these variants require such handling are:

Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

Errors that can result in infoset coercion

Certain types of constructs, they are reconstructed in each paragraph, resulting in progressively more elements in each paragraph:

```
<p><i>He dreamt.
<p><i>He dreamt that he ate breakfast.
<p><i>Then lunch.
<p><i>And finally dinner.
```

The resulting DOM for this fragment would be:

```

  p
    i
      #text: He dreamt.
  p
    i
      i
        #text: He dreamt that he ate breakfast.
  p
    i
      i
        i
          #text: Then lunch.
  p
    i
      i
        i
          #text: And finally dinner.

```

Errors involving fragile syntax constructs

There are syntax constructs that, for historical reasons, are relatively fragile. To help reduce the number of users who accidentally run into such problems, they are made non-conforming.

For example, the parsing of certain named character references in attributes happens even with the closing semicolon being omitted. It is safe to include an ampersand followed by letters that do not form a named character reference, but if the letters are changed to a string that *does* form a named character reference, they will be interpreted as that character instead.

In this fragment, the attribute's value is "?bill&ted":

```
<a href="?bill&ted">Bill and Ted</a>
```

In the following fragment, however, the attribute's value is actually "?art©", *not* the intended "?art©", because even without the final semicolon, "©" is handled the same as "©" and thus gets interpreted as "©":

```
<a href="?art&copy">Art and Copy</a>
```


Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

<input type=radio role=progressbar>

Another example is the restrictions on the content models of the `ul` element, which only allows `li` element children. Lists by definition consist just of zero or more list items, so if a `ul` element contains something other than an `li` element, it's not clear what was meant.

Cases where the default styles are likely to lead to confusion

Certain elements have default styles or behaviors that make certain combinations likely to lead to confusion. Where these have equivalent alternatives without this problem, the confusing combinations are disallowed.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

yes. Putting a block box in an `t` `span` elements, or nesting in a `span` element, but only

`button` element cannot
ive elements would be highly

Errors that indicate a likely misunderstanding of the specification

Sometimes, something is disallowed because allowing it would likely cause author confusion.

For example, setting the `disabled` attribute to the value "false" is disallowed, because despite the appearance of meaning that the element is enabled, it in fact means that the element is *disabled* (what matters for implementations is the presence of the attribute, not its value).

Errors involving limits that have been imposed merely to simplify the language

Some conformance errors simplify the language that authors need to learn.

For example, the `area` element's `shape` attribute, despite accepting both `circ` and `circle` values in practice as synonyms, disallows the use of the `circ` value, so as to simplify tutorials and other learning aids. There would be no benefit to allowing both, but it would cause extra confusion when teaching the language.

Errors that involve peculiarities of the parser

Certain elements are parsed in somewhat eccentric ways (typically for historical reasons), and their content model restrictions are intended to avoid exposing the author to these issues.

For example, a `form` element isn't allowed inside phrasing content, because when parsed as HTML, a `form` element's start tag will imply a `p` element's end tag. Thus, the following markup results in two paragraphs, not one:

```
<p>Welcome. <form><label>Name:</label> <input></form>
```

It is parsed exactly like the following:

```
<p>Welcome. </p><form><label>Name:</label> <input></form>
```

Errors that would likely result in scripts failing in hard-to-debug ways

Some errors are intended to help prevent script problems that would be hard to debug.

This is why, for instance, it is non-conforming to have two `id` attributes with the same value. Duplicate IDs lead to the wrong element being selected, with sometimes disastrous effects whose cause is hard to determine.

Errors that waste authoring time

Some constructs are disallowed because historically they have been the cause of a lot of wasted authoring time, and by encouraging authors to avoid making them, authors can save time in future efforts.

For example, a `script` element's `src` attribute causes the element's contents to be ignored. However, this isn't obvious, especially if the element's contents appear to be executable script — which can lead to authors spending a lot of time trying to debug the inline script without realizing that it is not executing. To reduce this problem, this specification makes it non-conforming to have executable script in a `script` element when the `src` attribute is present. This means that authors who are validating their documents are less likely to waste time with this kind of mistake.

Errors that involve areas that affect authors migrating to and from XHTML

Some authors like to write files that can be interpreted as both XML and HTML with similar results. Though this practice is discouraged in general due to the myriad of subtle complications involved (especially when involving scripting, styling, or any

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Another example would be the restrictions on the values of `xmlns` attributes in the HTML serialization, which are intended to ensure that elements in conforming documents end up in the same namespaces whether processed as HTML or XML.

Errors that involve areas reserved for future expansion

As with the restrictions on the syntax intended to allow for new syntax in future revisions of the language, some restrictions on the content models of elements and values of attributes are intended to allow for future expansion of the HTML vocabulary.

Errors that

Cer

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

reinforces the importance of

1.13 Suggested reading

This section is non-normative.

The following documents might be of interest to readers of this specification.

Character Model for the World Wide Web 1.0: Fundamentals [CHARMOD]

This Architectural Specification provides authors of specifications, software developers, and content developers with a common reference for interoperable text manipulation on the World Wide Web, building on the Universal Character Set, defined jointly by the Unicode Standard and ISO/IEC 10646. Topics addressed include use of the terms 'character', 'encoding' and 'string', a reference processing model, choice and identification of character encodings, character escaping, and string indexing.

Unicode Security Considerations [UTR36]

Because Unicode contains such a large number of characters and incorporates the varied writing systems of the world, incorrect usage can expose programs or systems to possible security attacks. This is especially important as more and more products are internationalized. This document describes some of the security considerations that programmers, system analysts, standards developers, and users should take into account, and provides specific recommendations to reduce the risk of problems.

Web Content Accessibility Guidelines (WCAG) 2.0 [WCAG]

Web Content Accessibility Guidelines (WCAG) 2.0 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also often make your Web content more usable to users in general.

Authoring Tool Accessibility Guidelines (ATAG) 2.0 [ATAG]

This specification provides guidelines for designing Web content authoring tools that are more accessible for people with disabilities. An authoring tool that conforms to these guidelines will promote accessibility by providing an accessible user interface to authors with disabilities as well as by enabling, supporting, and promoting the production of accessible Web content by all authors.

User Agent Accessibility Guidelines (UAAG) 2.0 [UAAG]

This document provides guidelines for designing user agents that lower barriers to Web accessibility for people with disabilities. User agents include browsers and other types of software that retrieve and render Web content. A user agent that conforms to these guidelines will promote accessibility through its own user interface and through other internal facilities, including its ability to communicate with other technologies (especially assistive technologies). Furthermore, all users, not just users with disabilities, should find conforming user agents to be more usable.

Polyglot Markup: HTML-Compatible XHTML Documents [POLYGLOT]

A document that uses polyglot markup is a document that is a stream of bytes that parses into identical document trees (with the exception of the `xmlns` attribute on the root element) when processed as HTML and when processed as XML. Polyglot markup that meets a well defined set of constraints is interpreted as compatible, regardless of whether they are

3,445 captures

5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB
◀ 03 ▶
2012 2013 2014

About this capture



Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

This specification refers to both HTML and XML attributes and IDL attributes, often in the same context. When it is not clear which is being referred to, they are referred to as **content attributes** for HTML and XML attributes, and **IDL attributes** for those defined on IDL interfaces. Similarly, the term "properties" is used for both JavaScript object properties and CSS properties. When these are ambiguous they are qualified as **object properties** and **CSS properties** respectively.

Generally, when the specification states that a feature applies to the HTML syntax or the XHTML syntax, it also includes the other. When a feature specifically only applies to one of the two languages, it is called out by explicitly stating that it does not apply to the other format, as in "for HTML, ... (this does not apply to XHTML)".

This specification uses the term **document** to refer to any use of HTML, ranging from short static documents to long essays or reports with rich multimedia content.

descendant Too slow? Try reading the multipage copy of

In the context of the document objects and their descendants, refer specifically to the spec instead: <https://web.archive.org>

/web/20130103030411/http://whatwg.org/html

In the context of the XML document, the term XML document refers to resources.

The term **XHTML document** is used to refer to both **Documents** in the XML document mode that contains element nodes in the HTML namespace, and byte streams labeled with an XML MIME type that contain elements from the HTML namespace, depending on context.

For simplicity, terms such as **shown**, **displayed**, and **visible** might sometimes be used when referring to the way a document is rendered to the user. These terms are not meant to imply a visual medium; they must be considered to apply to other media in equivalent ways.

When an algorithm B says to return to another algorithm A, it implies that A called B. Upon returning to A, the implementation must continue from where it left off in calling B.

The term "transparent black" refers to the color with red, green, blue, and alpha channels all set to zero.

2.1.1 Resources

The specification uses the term **supported** when referring to whether a user agent has an implementation capable of decoding the semantics of an external resource. A format or type is said to be **supported** if the implementation can process an external resource of that format or type without critical aspects of the resource being ignored. Whether a specific resource is **supported** can depend on what features of the resource's format are in use.

For example, a PNG image would be considered to be in a supported format if its pixel data could be decoded and rendered, even if, unbeknownst to the implementation, the image also contained animation data.

An MPEG-4 video file would not be considered to be in a supported format if the compression format used was not supported, even if the implementation could determine the dimensions of the movie from the file's metadata.

What some specifications, in particular the HTTP and URI specifications, refer to as a **representation** is referred to in this specification as a **resource**. [HTTP] [RFC3986]

The term **MIME type** is used to refer to what is sometimes called an *Internet media type* in protocol literature. The term **media type** in this specification is used to refer to the type of media intended for presentation, as used by the CSS specifications. [RFC2046] [MQ]

A string is a **valid MIME type** if it matches the `media-type` rule defined in section 3.7 "Media Types" of RFC 2616. In particular, a valid MIME type may include MIME type parameters. [HTTP]

A string is a **valid MIME type with no parameters** if it matches the `media-type` rule defined in section 3.7 "Media Types" of RFC 2616, but does not contain any U+003B SEMICOLON characters (:). In other words, if it consists only of a type and subtype, with no MIME Type parameters. [HTTP]

The term **HTML MIME type** is used to refer to the MIME type `text/html`.

A resource's **critical subresources** are those that the resource needs to have available to be correctly processed. Which resources are considered critical or not is defined by the specification that defines the resource's format.

The term **data: URL** refers to URLs that use the `data:` scheme. [RFC2397]

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Except where otherwise stated, all elements defined or mentioned in this specification are in the HTML namespace ("http://www.w3.org/1999/xhtml"), and all attributes defined or mentioned in this specification have no namespace.

The term **element type** is used to refer to the set of elements that have a given local name and namespace. For example, `button` elements are elements with the element type `button`, meaning they have the local name "`button`" and (implicitly as defined above) the HTML namespace.

Attribute names are said to be **XML-compatible** if they match the `Name` production defined in XML they contain no U+003A COLON characters (:). [XML]

The term **XML** Too slow? Try reading the multipage copy of ME type whose subtype ends with th the spec instead: <https://web.archive.org>

2.1.3 DOI /web/20130103030411/http:

The **root element** //whatwg.org/html
no root elem

one then the Document has

The term **root element**, when not referring to a `Document` object's root element, means the furthest ancestor element node of whatever node is being discussed, or the node itself if it has no ancestors. When the node is a part of the document, then the node's root element is indeed the document's root element; however, if the node is not currently part of the document tree, the root element will be an orphaned node.

When an element's root element is the root element of a `Document` object, it is said to be **in a Document**. An element is said to have been **inserted into a document** when its root element changes and is now the document's root element. Analogously, an element is said to have been **removed from a document** when its root element changes from being the document's root element to being another element.

A node's **home subtree** is the subtree rooted at that node's root element. When a node is in a `Document`, its home subtree is that `Document`'s tree.

The `Document` of a `Node` (such as an element) is the `Document` that the `Node`'s `ownerDocument` IDL attribute returns. When a `Node` is in a `Document` then that `Document` is always the `Node`'s `Document`, and the `Node`'s `ownerDocument` IDL attribute thus always returns that `Document`.

The `Document` of a content attribute is the `Document` of the attribute's element.

The term **tree order** means a pre-order, depth-first traversal of DOM nodes involved (through the `parentNode/childNodes` relationship).

When it is stated that some element or attribute is **ignored**, or treated as some other value, or handled as if it was something else, this refers only to the processing of the node after it is in the DOM. A user agent must not mutate the DOM in such situations.

A content attribute is said to **change** value only if its new value is different than its previous value; setting an attribute to a value it already has does not change it.

The term **empty**, when used of an attribute value, `Text` node, or string, means that the length of the text is zero (i.e. not even containing spaces or control characters).

2.1.4 Scripting

The construction "a `Foo` object", where `Foo` is actually an interface, is sometimes used instead of the more accurate "an object implementing the interface `Foo`".

An IDL attribute is said to be **getting** when its value is being retrieved (e.g. by author script), and is said to be **setting** when a new value is assigned to it.

If a DOM object is said to be **live**, then the attributes and methods on that object must operate on the actual underlying data, not a snapshot of the data.

In the contexts of events, the terms *fire* and *dispatch* are used as defined in the DOM specification: **firing** an event means to create and dispatch it, and **dispatching** an event means to follow the steps that propagate the event through the tree. The term **trusted event** is used to refer to events whose `isTrusted` attribute is initialized to true. [DOM]

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Typically such content handlers are provided by third parties, though a user agent can also designate built-in content handlers as plugins.

A user agent must not consider the types `text/plain` and `application/octet-stream` as having a registered plugin.

One example of a plugin would be a PDF viewer that is instantiated in a browsing context when the user navigates to a PDF file. This would count as a plugin regardless of whether the party that implemented the PDF viewer component was the same as that which implemented the user agent itself. However, a PDF viewer application that launches separate from the user agent (as opposed to using the same interface) is not a plugin by this definition.

Note: Plugins and platforms might be required Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

A plugin can

For example, a plugin is instantiated inside a sandboxed iframe.

cted to be user-agent-specific API; others specification doesn't

⚠ Warning! Browsers should take extreme care when interacting with external content intended for plugins. When third-party software is run with the same privileges as the user agent itself, vulnerabilities in the third-party software become as dangerous as those in the user agent.

2.1.6 Character encodings

The preferred MIME name of a character encoding is the name or alias labeled as "preferred MIME name" in the IANA Character Sets registry, if there is one, or the encoding's name, if none of the aliases are so labeled. [IANACHARSET]

An ASCII-compatible character encoding is a single-byte or variable-length encoding in which the bytes 0x09, 0x0A, 0x0C, 0x0D, 0x20 - 0x22, 0x26, 0x27, 0x2C - 0x3F, 0x41 - 0x5A, and 0x61 - 0x7A, ignoring bytes that are the second and later bytes of multibyte sequences, all correspond to single-byte sequences that map to the same Unicode characters as those bytes in ANSI_X3.4-1968 (US-ASCII). [RFC1345]

Note: This includes such encodings as Shift_JIS, HZ-GB-2312, and variants of ISO-2022, even though it is possible in these encodings for bytes like 0x70 to be part of longer sequences that are unrelated to their interpretation as ASCII. It excludes such encodings as UTF-7, UTF-16, GSM03.38, and EBCDIC variants.

The term a **UTF-16 encoding** refers to any variant of UTF-16: self-describing UTF-16 with a BOM, ambiguous UTF-16 without a BOM, raw UTF-16LE, and raw UTF-16BE. [RFC2781]

The term **code unit** is used as defined in the Web IDL specification: a 16 bit unsigned integer, the smallest atomic component of a DOMString. (This is a narrower definition than the one used in Unicode.) [WEBIDL]

The term **Unicode code point** means a *Unicode scalar value* where possible, and an isolated surrogate code point when not. When a conformance requirement is defined in terms of characters or Unicode code points, a pair of code units consisting of a high surrogate followed by a low surrogate must be treated as the single code point represented by the surrogate pair, but isolated surrogates must each be treated as the single code point with the value of the surrogate. [UNICODE]

In this specification, the term **character**, when not qualified as *Unicode character*, is synonymous with the term Unicode code point.

The term **Unicode character** is used to mean a *Unicode scalar value* (i.e. any Unicode code point that is not a surrogate code point). [UNICODE]

The **code-unit length** of a string is the number of code units in that string.

Note: This complexity results from the historical decision to define the DOM API in terms of 16 bit (UTF-16) code units, rather than in terms of Unicode characters.

2.2 Conformance requirements

Go

NOV JAN FEB

03

2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

to be interpreted with the same normative meaning as "MAY" and "OPTIONAL". For readability, these words do not appear in all uppercase letters in this specification. [RFC2119]

Requirements phrased in the imperative as part of algorithms (such as "strip any leading space characters" or "return false and abort these steps") are to be interpreted with the meaning of the key word ("must", "should", "may", etc) used in introducing the algorithm.

For example, were the spec to say:

To eat a kiwi, the user must:

1. Peel the kiwi.

2.

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Here

The former (imperative) style is generally preferred in this specification for stylistic reasons.

Conformance requirements phrased as algorithms or specific steps may be implemented in any manner, so long as the end result is equivalent. (In particular, the algorithms defined in this specification are intended to be easy to follow, and not intended to be performant.)

2.2.1 Conformance classes

This specification describes the conformance criteria for user agents (relevant to implementors) and documents (relevant to authors and authoring tool implementors).

Conforming documents are those that comply with all the conformance criteria for documents. For readability, some of these conformance requirements are phrased as conformance requirements on authors; such requirements are implicitly requirements on documents: by definition, all documents are assumed to have had an author. (In some cases, that author may itself be a user agent — such user agents are subject to additional rules, as explained below.)

For example, if a requirement states that "authors must not use the `foobar` element", it would imply that documents are not allowed to contain elements named `foobar`.

Note: *There is no implied relationship between document conformance requirements and implementation conformance requirements. User agents are not free to handle non-conformant documents as they please; the processing model described in this specification applies to implementations regardless of the conformity of the input documents.*

User agents fall into several (overlapping) categories with different conformance requirements.

Web browsers and other interactive user agents

Web browsers that support the XHTML syntax must process elements and attributes from the HTML namespace found in XML documents as described in this specification, so that users can interact with them, unless the semantics of those elements have been overridden by other specifications.

A conforming XHTML processor would, upon finding an XHTML `script` element in an XML document, execute the `script` contained in that element. However, if the element is found within a transformation expressed in XSLT (assuming the user agent also supports XSLT), then the processor would instead treat the `script` element as an opaque element that forms part of the transform.

Web browsers that support the HTML syntax must process documents labeled with an HTML MIME type as described in this specification, so that users can interact with them.

User agents that support scripting must also be conforming implementations of the IDL fragments in this specification, as described in the Web IDL specification. [WEBIDL]

Note: *Unless explicitly stated, specifications that override the semantics of HTML elements do not override the requirements on DOM objects representing those elements. For example, the `script` element in the*

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

same conformance criteria as Web browsers, except that they are exempt from requirements regarding user interaction.

Note: Typical examples of non-interactive presentation user agents are printers (static UAs) and overhead displays (dynamic UAs). It is expected that most static non-interactive presentation user agents will also opt to lack scripting support.

A non-interactive but dynamic presentation UA would still execute scripts, allowing forms to be dynamically submitted, and so forth. However, since the concept of "focus" is irrelevant when the user cannot interact with the document, the UA would not need to support any of the focus-related DOM APIs.



Visual user

User
ren-This
sett
styl

User

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ng the suggested default

ring are encouraged to offer contrast, using different focus

.signated, implement the rules in the rendering section that that section defines as the behavior that user agents are *expected* to implement.

User agents with no scripting support

Implementations that do not support scripting (or which have their scripting features disabled entirely) are exempt from supporting the events and DOM interfaces mentioned in this specification. For the parts of this specification that are defined in terms of an events model or in terms of the DOM, such user agents must still act as if events and the DOM were supported.

Note: Scripting can form an integral part of an application. Web browsers that do not support scripting, or that have scripting disabled, might be unable to fully convey the author's intent.

Conformance checkers

Conformance checkers must verify that a document conforms to the applicable conformance criteria described in this specification. Automated conformance checkers are exempt from detecting errors that require interpretation of the author's intent (for example, while a document is non-conforming if the content of a `blockquote` element is not a quote, conformance checkers running without the input of human judgement do not have to check that `blockquote` elements only contain quoted material).

Conformance checkers must check that the input document conforms when parsed without a browsing context (meaning that no scripts are run, and that the parser's scripting flag is disabled), and should also check that the input document conforms when parsed with a browsing context in which scripts execute, and that the scripts never cause non-conforming states to occur other than transiently during script execution itself. (This is only a "SHOULD" and not a "MUST" requirement because it has been proven to be impossible. [COMPUTABLE])

The term "HTML validator" can be used to refer to a conformance checker that itself conforms to the applicable requirements of this specification.

XML DTDs cannot express all the conformance requirements of this specification. Therefore, a validating XML processor and a DTD cannot constitute a conformance checker. Also, since neither of the two authoring formats defined in this specification are applications of SGML, a validating SGML system cannot constitute a conformance checker either.

To put it another way, there are three types of conformance criteria:

1. Criteria that can be expressed in a DTD.
2. Criteria that cannot be expressed by a DTD, but can still be checked by a machine.
3. Criteria that can only be checked by a human.

A conformance checker must check for the first two. A simple DTD-based validator only checks for the first class of errors and is therefore not a conforming conformance checker according to this specification.

Data mining tools

Applications and tools that process HTML and XHTML documents for reasons other than to either render the documents or check them for conformance should act in accordance with the semantics of the documents that they process.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- custom request headers
- simple cross-origin request
- redirect steps
- omit credentials flag
- resource sharing check

Web IDL

The IDL fragments in this specification must be interpreted as required for conforming IDL fragments, as described in the Web IDL specification. [WEBIDL]

The term **IDL** refers to the IDL defined in the Web IDL specification. [WEBIDL] The term **scriptable object** refers to objects that support named properties, indexed properties, and read-only (when applicable) properties. [WEBIDL] The term **DOMString** refers to a sequence of characters. [DOM] The term **time value** refers to a Date object. [DOM] The term **time value** refers to a Date object created representing the time 2000-01-01T00:00:00Z, then the Date object created representing that time would represent the same time as that created representing the time 2000-01-01T00:00:00.023Z, 45 millionths earlier. If the given time is NaN, then the result is a Date object that represents a time value NaN (indicating that the object does not represent a specific instant of time).

Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

2000-01-01T00:00:00.023045Z, then the Date object created representing that time would represent the same time as that created representing the time 2000-01-01T00:00:00.023Z, 45 millionths earlier. If the given time is NaN, then the result is a Date object that represents a time value NaN (indicating that the object does not represent a specific instant of time).

JavaScript

Some parts of the language described by this specification only support JavaScript as the underlying scripting language. [ECMA262]

Note: The term "JavaScript" is used to refer to ECMA262, rather than the official term ECMAScript, since the term JavaScript is more widely known. Similarly, the MIME type used to refer to JavaScript in this specification is text/javascript, since that is the most commonly used type, despite it being an officially obsoleted type according to RFC 4329. [RFC4329]

The term **JavaScript global environment** refers to the *global environment* concept defined in the ECMAScript specification.

DOM

The Document Object Model (DOM) is a representation — a model — of a document and its content. The DOM is not just an API; the conformance criteria of HTML implementations are defined, in this specification, in terms of operations on the DOM. [DOM]

Implementations must support DOM and the events defined in DOM Events, because this specification is defined in terms of the DOM, and some of the features are defined as extensions to the DOM interfaces. [DOM] [DOMEVENTS]

In particular, the following features are defined in the DOM specification: [DOM]

- Attr interface
- Comment interface
- DOMImplementation interface
- Document interface
- DocumentFragment interface
- DocumentType interface
- DOMException interface
- Element interface
- Node interface
- NodeList interface
- ProcessingInstruction interface
- Text interface
- HTMLCollection interface, and the terms **collections** and **represented by the collection**
- DOMTokenList interface
- DOMSettableTokenList interface
- createDocument() method
- createHTMLDocument() method
- createElement() method
- createElementNS() method
- getElementById() method
- insertBefore() method
- ownerDocument attribute

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- `id` attribute
- `textContent` attribute
- The `insert`, `append`, `remove`, and `replace` algorithms for nodes
- The **nodes are inserted** and **nodes are removed** concepts
- `Event` interface
- `EventTarget` interface
- `EventInit` dictionary type
- `target` attribute
- `isTrusted` attribute
- The `type` of an event
- The concept of an `event listener` and the event listeners associated with an `EventTarget`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ment is affected by a base

to ranges.

- The `invoke MutationObserver objects` algorithm

The term **throw** in this specification is used as defined in the DOM specification. The following `DOMException` types are defined in the DOM specification: [DOM]

1. `IndexSizeError`
3. `HierarchyRequestError`
4. `WrongDocumentError`
5. `InvalidCharacterError`
7. `NoModificationAllowedError`
8. `NotFoundError`
9. `NotSupportedError`
11. `InvalidStateError`
12. `SyntaxError`
13. `InvalidModificationError`
14. `NamespaceError`
15. `InvalidAccessError`
18. `SecurityError`
19. `NetworkError`
20. `AbortError`
21. `URLMismatchError`
22. `QuotaExceededError`
23. `TimeoutError`
24. `InvalidNodeTypeError`
25. `DataCloneError`

For example, to *throw a `TimeoutError` exception*, a user agent would construct a `DOMException` object whose type was the string "`TimeoutError`" (and whose code was the number 23, for legacy reasons) and actually throw that object as an exception.

The URL associated with a `Document`, as defined in the DOM specification, is referred to in this specification as the document's address.

The following features are defined in the DOM Events specification: [DOMEVENTS]

- `MouseEvent` interface
- `MouseEventInit` dictionary type
- The `UIEvent` interface's `detail` attribute
- `click` event

This specification sometimes uses the term **name** to refer to the event's `type`; as in, "an event named `click`" or "if the event name is `keypress`". The terms "name" and "type" for events are synonymous.

The following features are defined in the DOM Parsing and Serialization specification: [DOMPARSING]

- `innerHTML`
- `outerHTML`

Note: User agents are also encouraged to implement the features described in the HTML Editing APIs and

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- The top layer concept
- `requestFullScreen()`
- The fullscreen enabled flag
- The fully exit fullscreen algorithm

Typed Arrays

The **ArrayBuffer** and **ArrayBufferView** interfaces and underlying concepts from the Typed Array Specification are used for several features in this specification. The **Uint8ClampedArray** interface type is specifically used in the definition of the `canvas` element's 2D API. [TYPEDARRAY]



File API

This Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

It also uses the following interface defined in the File System API specification: [FILESYSTEMAPI]

- `FileCallback`

XMLHttpRequest

This specification references the XMLHttpRequest specification to define how the two specifications interact. The terms **document response entity body**, **XMLHttpRequest base URL**, **XMLHttpRequest origin**, and **XMLHttpRequest referrer source** are defined in that specification. [XHR]

Media Queries

Implementations must support the Media Queries language. [MQ]

CSS modules

While support for CSS as a whole is not required of implementations of this specification (though it is encouraged, at least for Web browsers), some features are defined in terms of specific CSS requirements.

In particular, some features require that a string be **parsed as a CSS <color> value**. When parsing a CSS value, user agents are required by the CSS specifications to apply some error handling rules. These apply to this specification also. [CSSCOLOR] [CSS]

For example, user agents are required to close all open constructs upon finding the end of a style sheet unexpectedly. Thus, when parsing the string "`rgb(0, 0, 0"` (with a missing close-parenthesis) for a color value, the close parenthesis is implied by this error handling rule, and a value is obtained (the color 'black'). However, the similar construct "`rgb(0, 0, "` (with both a missing parenthesis and a missing "blue" value) cannot be parsed, as closing the open construct does not result in a viable value.

The term **CSS element reference identifier** is used as defined in the *CSS Image Values and Replaced Content* specification to define the API that declares identifiers for use with the CSS `'element()'` function. [CSSIMAGES]

Similarly, the term **provides a paint source** is used as defined in the *CSS Image Values and Replaced Content* specification to define the interaction of certain HTML elements with the CSS `'element()'` function. [CSSIMAGES]

Support for the CSS Object Model is required for implementations that support scripting. The following features and terms are defined in the CSSOM specifications: [CSSOM] [CSSOMVIEW]

- `Screen`
- `LinkStyle`
- `CSSStyleDeclaration`
- `cssText` attribute of `CSSStyleDeclaration`
- `StyleSheet`
- `sheet`
- `disabled`
- **Alternative style sheet sets and the preferred style sheet set**
- **Serializing a CSS value**
- **Scroll an element into view**

The term **CSS styling attribute** is defined in the *CSS Style Attributes* specification. [CSSATTR]

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

- [SVGMATRIX](#)

WebGL

The following interface is defined in the WebGL specification: [WEBGL]

- [WebGLRenderingContext](#)

WebVTT

Implementations may support WebVTT as a text track format for subtitles, captions, closed titles, metadata, etc, for media

rec

Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

- [The rules for updating the display of WebVTT text tracks](#)
- [The WebVTT cue text DOM construction rules](#)
- [The WebVTT cue text rendering rules](#)
- [The WebVTT cue text parsing rules](#)

The WebSocket protocol

The following terms are defined in the WebSocket protocol specification: [WSP]

- [establish a WebSocket connection](#)
- [the WebSocket connection is established](#)
- [validate the server's response](#)
- [extensions in use](#)
- [subprotocol in use](#)
- [headers to send appropriate cookies](#)
- [cookies set during the server's opening handshake](#)
- [a WebSocket message has been received](#)
- [fail the WebSocket connection](#)
- [close the WebSocket connection](#)
- [start the WebSocket closing handshake](#)
- [the WebSocket closing handshake is started](#)
- [the WebSocket connection is closed \(possibly *cleanly*\)](#)
- [the WebSocket connection close code](#)
- [the WebSocket connection close reason](#)

ARIA

The terms **strong native semantics** is used as defined in the ARIA specification. The term **default implicit ARIA semantics** has the same meaning as the term *implicit WAI-ARIA semantics* as used in the ARIA specification. [ARIA]

The `role` and `aria-*` attributes are defined in the ARIA specification. [ARIA]

This specification does not *require* support of any particular network protocol, style sheet language, scripting language, or any of the DOM specifications beyond those required in the list above. However, the language described by this specification is biased towards CSS as the styling language, JavaScript as the scripting language, and HTTP as the network protocol, and several features assume that those languages and protocols are in use.

A user agent that implements the HTTP protocol must implement the Web Origin Concept specification and the HTTP State Management Mechanism specification (Cookies) as well. [HTTP] [ORIGIN] [COOKIES]

Note: This specification might have certain additional requirements on character encodings, image formats, audio formats, and video formats in the respective sections.

2.2.3 Extensibility

HTML has a wide number of extensibility mechanisms that can be used for adding semantics in a safe manner:

- Authors can use the `class` attribute to extend elements, effectively creating their own elements, while using the most applicable existing "real" HTML element, so that browsers and other tools that don't know of the extension can still support it somewhat well. This is the tack used by microformats, for example.

Go

NOV JAN FEB



2012 2013 2014

3,445 captures

5 Jun 2004 - 22 Aug 2020

the predefined set of metadata names

- Authors can use the `rel=""` mechanism to annotate links with specific meanings by registering extensions to the predefined set of link types. This is also used by microformats.
 - Authors can embed raw data using the `<script type="">` mechanism with a custom type, for further handling by inline or server-side scripts.
 - Authors can create plugins and invoke them using the `embed` element. This is how Flash works.

- Aut
- Aut
- Aut

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org>

/web/20130103030411/http:
<http://whatwg.org/html>

ot libraries, for instance.
ed nested name-value pairs

s must not use such
ic user agents to access the

If such extensions are nonetheless needed, e.g. for experimental purposes, then vendors are strongly urged to use one of the following extension mechanisms:

For markup-level features that can be limited to the XML serialization and need not be supported in the HTML serialization, vendors should use the namespace mechanism to define custom namespaces in which the non-standard elements and attributes are supported.

For markup-level features that are intended for use with the HTML syntax, extensions should be limited to new attributes of the form "`x-vendor-feature`", where *vendor* is a short string that identifies the vendor responsible for the extension, and *feature* is the name of the feature. New element names should not be created. Using attributes for such extensions exclusively allows extensions from multiple vendors to co-exist on the same element, which would not be possible with elements. Using the "`x-vendor-feature`" form allows extensions to be made without risk of conflicting with future additions to the specification.

For instance, a browser named "FerretBrowser" could use "ferret" as a vendor prefix, while a browser named "Mellblom Browser" could use "mb". If both of these browsers invented extensions that turned elements into scratch-and-sniff areas, an author experimenting with these features could write:

```
<p>This smells of lemons!
<span x-ferret-smellovision x-ferret-smellcode="LEM01"
      x-mb-outputsmell x-mb-smell="lemon juice"></span></p>
```

Attribute names beginning with the two characters "x–" are reserved for user agent use and are guaranteed to never be formally added to the HTML language. For flexibility, attribute names containing underscores (the U+005F LOW LINE character) are also reserved for experimental purposes and are guaranteed to never be formally added to the HTML language.

Note: Pages that use such attributes are by definition non-conforming

For DOM extensions, e.g. new methods and IDL attributes, the new members should be prefixed by vendor-specific strings to prevent clashes with future versions of this specification.

For events, experimental event types should be prefixed with vendor-specific strings

For example, if a user agent called "Pleasold" were to add an event to indicate when the user is going up in an elevator, it could use the prefix "pleasold" and thus name the event "pleasoldgoingup", possibly with an event handler attribute named "onpleasoldgoingup".

All extensions must be defined so that the use of extensions neither contradicts nor causes the non-conformance of functionality defined in the specification

For example, while strongly discouraged from doing so, an implementation "Foo Browser" could add a new IDL attribute "fooTypeTime" to a control's DOM interface that returned the time it took the user to select the current value of a control (say). On the other hand, defining a new control that appears in a form's `elements` array would be in violation of the above requirement, as it would violate the definition of `elements` given in this specification.

When adding new reflecting IDL attributes corresponding to content attributes of the form "`x-vendor-feature`", the IDL attribute should be named "`vendorFeature`" (i.e. the "`x`" is dropped from the IDL attribute's name).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Note: Someone could write a specification that defines any arbitrary byte stream as conforming, and then claim that their random junk is conforming. However, that does not mean that their random junk actually is conforming for everyone's purposes: if someone else decides that that specification does not apply to their work, then they can quite legitimately say that the aforementioned random junk is just that, junk, and not conforming at all. As far as conformance goes, what matters in a particular community is what that community agrees is applicable.

User agents must treat elements and attributes that they do not understand as semantically neutral; leaving them in the DOM (for DOM processors), and styling them according to CSS (for CSS processors) but not inferring any meaning from them.

When supported for performance mentioned in the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

aid in development, or for the feature was not implemented, the attribute itself making it return null or throw

2.3 Case

Comparing two strings in a **case-sensitive** manner means comparing them exactly, code point for code point.

Comparing two strings in an **ASCII case-insensitive** manner means comparing them exactly, code point for code point, except that the characters in the range U+0041 to U+005A (i.e. LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER Z) and the corresponding characters in the range U+0061 to U+007A (i.e. LATIN SMALL LETTER A to LATIN SMALL LETTER Z) are considered to also match.

Comparing two strings in a **compatibility caseless** manner means using the Unicode *compatibility caseless match* operation to compare the two strings. [UNICODE]

Except where otherwise stated, string comparisons must be performed in a case-sensitive manner.

Converting a string to ASCII uppercase means replacing all characters in the range U+0061 to U+007A (i.e. LATIN SMALL LETTER A to LATIN SMALL LETTER Z) with the corresponding characters in the range U+0041 to U+005A (i.e. LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER Z).

Converting a string to ASCII lowercase means replacing all characters in the range U+0041 to U+005A (i.e. LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER Z) with the corresponding characters in the range U+0061 to U+007A (i.e. LATIN SMALL LETTER A to LATIN SMALL LETTER Z).

A string *pattern* is a **prefix match** for a string *s* when *pattern* is not longer than *s* and truncating *s* to *pattern*'s length leaves the two strings as matches of each other.

2.4 UTF-8

When a user agent is required to **decode a byte string as UTF-8, with error handling**, it means that the byte stream must be converted to a Unicode string by interpreting it as UTF-8, except that any errors must be handled as described in the following list. Bytes in the following list are represented in hexadecimal. [RFC3629]

- ↳ One byte in the range FE to FF
- ↳ Overlong forms (e.g. F0 80 80 A0)
- ↳ One byte in the range C0 to C1, followed by one byte in the range 80 to BF
- ↳ One byte in the range F0 to F4, followed by three bytes in the range 80 to BF that represent a code point above U+10FFFF
- ↳ One byte in the range F5 to F7, followed by three bytes in the range 80 to BF
- ↳ One byte in the range F8 to FB, followed by four bytes in the range 80 to BF
- ↳ One byte in the range FC to FD, followed by five bytes in the range 80 to BF
- ↳ One byte in the range C0 to FD that is not followed by a byte in the range 80 to BF
- ↳ One byte in the range E0 to FD, followed by a byte in the range 80 to BF that is not followed by a byte in the range 80 to BF
- ↳ One byte in the range F0 to FD, followed by two bytes in the range 80 to BF, the last of which is not followed by a byte in the range 80 to BF
- ↳ One byte in the range F8 to FD, followed by three bytes in the range 80 to BF, the last of which is not followed by a byte in the range 80 to BF

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

- ↳ One byte in the range 80 to BF not preceded by a byte in the range 80 to FD
- ↳ One byte in the range 80 to BF preceded by a byte that is part of a complete UTF-8 sequence that does not include this byte
- ↳ One byte in the range 80 to BF preceded by a byte that is part of a sequence that has been replaced by a U+FFFD REPLACEMENT CHARACTER, either alone or as part of a sequence

Each such byte must be replaced with a U+FFFD REPLACEMENT CHARACTER.

For the purposes of the above requirements, an **overlong form** in UTF-8 is a sequence that encodes a code point using more bytes than the minimum required to encode that code point in UTF-8.

|| For e Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

2.5 Conformance There are various ways to conform to this specification. n describes what the

Note: *Implementors are strongly urged to carefully examine any third-party libraries they might consider using to implement the parsing of syntaxes described below. For example, date libraries are likely to implement error handling behavior that differs from what is required in this specification, since error-handling behavior is often not defined in specifications that describe date syntaxes similar to those used in this specification, and thus implementations tend to vary greatly in how they handle errors.*

2.5.1 Common parser idioms

The **space characters**, for the purposes of this specification, are U+0020 SPACE, U+0009 CHARACTER TABULATION (tab), U+000A LINE FEED (LF), U+000C FORM FEED (FF), and U+000D CARRIAGE RETURN (CR).

The **White_Space characters** are those that have the Unicode property "White_Space" in the Unicode PropList.txt data file. [UNICODE]

Note: *This should not be confused with the "White_Space" value (abbreviated "WS") of the "Bidi_Class" property in the Unicode.txt data file.*

The **uppercase ASCII letters** are the characters in the range U+0041 LATIN CAPITAL LETTER A to U+005A LATIN CAPITAL LETTER Z.

The **lowercase ASCII letters** are the characters in the range U+0061 LATIN SMALL LETTER A to U+007A LATIN SMALL LETTER Z.

The **ASCII digits** are the characters in the range U+0030 DIGIT ZERO (0) to U+0039 DIGIT NINE (9).

The **alphanumeric ASCII characters** are those that are either uppercase ASCII letters, lowercase ASCII letters, or ASCII digits.

The **ASCII hex digits** are the characters in the ranges U+0030 DIGIT ZERO (0) to U+0039 DIGIT NINE (9), U+0041 LATIN CAPITAL LETTER A to U+0046 LATIN CAPITAL LETTER F, and U+0061 LATIN SMALL LETTER A to U+0066 LATIN SMALL LETTER F.

The **uppercase ASCII hex digits** are the characters in the ranges U+0030 DIGIT ZERO (0) to U+0039 DIGIT NINE (9) and U+0041 LATIN CAPITAL LETTER A to U+0046 LATIN CAPITAL LETTER F only.

The **lowercase ASCII hex digits** are the characters in the ranges U+0030 DIGIT ZERO (0) to U+0039 DIGIT NINE (9) and U+0061 LATIN SMALL LETTER A to U+0066 LATIN SMALL LETTER F only.

Some of the micro-parsers described below follow the pattern of having an *input* variable that holds the string being parsed, and having a *position* variable pointing at the next character to parse in *input*.

For parsers based on this pattern, a step that requires the user agent to **collect a sequence of characters** means that the following algorithm must be run, with *characters* being the set of characters that can be collected:

1. Let *input* and *position* be the same variables as those of the same name in the algorithm that invoked these steps.
2. Let *result* be the empty string.
3. While *position* doesn't point past the end of *input* and the character at *position* is one of the *characters*, append that character

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

White_Space characters means that the user agent must collect a sequence of characters that are White_Space characters. In both cases, the collected characters are not used. [UNICODE]

When a user agent is to **strip line breaks** from a string, the user agent must remove any U+000A LINE FEED (LF) and U+000D CARRIAGE RETURN (CR) characters from that string.

When a user agent is to **strip leading and trailing whitespace** from a string, the user agent must remove all space characters that are at the start or end of the string.

When a user agent has to **strictly split a string** on a particular delimiter character *delimiter*, it ~~must~~ use the following algorithm:

1. Let **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**
2. Let
3. Let
4. Whi

2. Append the string collected in the previous step to *tokens*.

3. Advance *position* to the next character in *input*.

5. Return *tokens*.

Note: For the special cases of splitting a string on spaces and on commas, this algorithm does not apply (those algorithms also perform whitespace trimming).

2.5.2 Boolean attributes

A number of attributes are **boolean attributes**. The presence of a boolean attribute on an element represents the true value, and the absence of the attribute represents the false value.

If the attribute is present, its value must either be the empty string or a value that is an ASCII case-insensitive match for the attribute's canonical name, with no leading or trailing whitespace.

Note: The values "true" and "false" are not allowed on boolean attributes. To represent a false value, the attribute has to be omitted altogether.

Here is an example of a checkbox that is checked and disabled. The `checked` and `disabled` attributes are the boolean attributes.

```
<label><input type=checkbox checked name=cheese disabled> Cheese</label>
```

This could be equivalently written as this:

```
<label><input type=checkbox checked=checked name=cheese disabled=disabled> Cheese</label>
```

You can also mix styles; the following is still equivalent:

```
<label><input type='checkbox' checked name=cheese disabled=""> Cheese</label>
```

2.5.3 Keywords and enumerated attributes

Some attributes are defined as taking one of a finite set of keywords. Such attributes are called **enumerated attributes**. The keywords are each defined to map to a particular *state* (several keywords might map to the same state, in which case some of the keywords are synonyms of each other; additionally, some of the keywords can be said to be non-conforming, and are only in the specification for historical reasons). In addition, two default states can be given. The first is the *invalid value default*, the second is the *missing value default*.

If an enumerated attribute is specified, the attribute's value must be an ASCII case-insensitive match for one of the given keywords that are not said to be non-conforming, with no leading or trailing whitespace.

When the attribute is specified, if its value is an ASCII case-insensitive match for one of the given keywords then that keyword's state is

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the attribute is *not* specified, if there is a *missing value default* state defined, then that is the state represented by the (missing) attribute. Otherwise, the absence of the attribute means that there is no state represented.

Note: The empty string can be a valid keyword.

2.5.4 Numbers

2.5.4.1 Signed integers

A string is a

A valid integer
digits. A valid
that follows:

The rules for
abortion at t

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

HEN-MINUS character (-).

base ten by that string of
e ten by the string of digits

ollowed in the order given,

1. Let *input* be the string being parsed.

2. Let *position* be a pointer into *input*, initially pointing at the start of the string.

3. Let *sign* have the value "positive".

4. Skip whitespace.

5. If *position* is past the end of *input*, return an error.

6. If the character indicated by *position* (the first character) is a U+002D HYPHEN-MINUS character (-):

1. Let *sign* be "negative".

2. Advance *position* to the next character.

3. If *position* is past the end of *input*, return an error.

Otherwise, if the character indicated by *position* (the first character) is a U+002B PLUS SIGN character (+):

1. Advance *position* to the next character. (The "+" is ignored, but it is not conforming.)

2. If *position* is past the end of *input*, return an error.

7. If the character indicated by *position* is not an ASCII digit, then return an error.

8. Collect a sequence of characters that are ASCII digits, and interpret the resulting sequence as a base-ten integer. Let *value* be that integer.

9. If *sign* is "positive", return *value*, otherwise return the result of subtracting *value* from zero.

2.5.4.2 Non-negative integers

A string is a **valid non-negative integer** if it consists of one or more ASCII digits.

A valid non-negative integer represents the number that is represented in base ten by that string of digits.

The **rules for parsing non-negative integers** are as given in the following algorithm. When invoked, the steps must be followed in the order given, aborting at the first step that returns a value. This algorithm will return either zero, a positive integer, or an error.

1. Let *input* be the string being parsed.

2. Let *value* be the result of parsing *input* using the rules for parsing integers.

3. If *value* is an error, return an error.

4. If *value* is less than zero, return an error.

5. Return *value*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

1. A series of one or more ASCII digits.
2. 1. A single U+002E FULL STOP character (.).
 2. A series of one or more ASCII digits.
3. Optionally:
 1. Either a U+0065 LATIN SMALL LETTER E character (e) or a U+0045 LATIN CAPITAL LETTER E character (E).
 2. Optionally, a U+002D HYPHEN-MINUS character (-) or U+002B PLUS SIGN character (+).
 3. A series of one or more ASCII digits.

A valid floating-point number represents the number obtained by multiplying the significand by ten raised to the power of the exponent, where the significand is the first number, interpreted as base ten (including the decimal point and the number after the decimal point, if any, and intervening characters), and the exponent is the second number, interpreted as a base ten integer. The sign of the exponent is given by the character at position `position`: if it is a U+002D HYPHEN-MINUS character (-) and the number after the decimal point is zero, then the number is negative; otherwise, it is positive. The sign of the significand is given by the character at position `position`: if it is a U+002B PLUS SIGN character (+) or a U+0065 LATIN SMALL LETTER E character (e), then the significand is positive; otherwise, it is negative.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Note: 7

The **best representation** of a floating-point number is the string returned by the `toString()` method of the `Number` object. This method returns strings that could be obtained from the `toString()` method of the `Object` object for a particular value, but user agents must always return the same string for that value (though it may differ from the value used by other user agents).

The **rules for parsing floating-point number values** are as given in the following algorithm. This algorithm must be aborted at the first step that returns something. This algorithm will return either a number or an error.

1. Let `input` be the string being parsed.
2. Let `position` be a pointer into `input`, initially pointing at the start of the string.
3. Let `value` have the value 1.
4. Let `divisor` have the value 1.
5. Let `exponent` have the value 1.
6. Skip whitespace.
7. If `position` is past the end of `input`, return an error.
8. If the character indicated by `position` is a U+002D HYPHEN-MINUS character (-):
 1. Change `value` and `divisor` to -1.
 2. Advance `position` to the next character.
 3. If `position` is past the end of `input`, return an error.

Otherwise, if the character indicated by `position` (the first character) is a U+002B PLUS SIGN character (+):

1. Advance `position` to the next character. (The "+" is ignored, but it is not conforming.)
2. If `position` is past the end of `input`, return an error.
9. If the character indicated by `position` is a U+002E FULL STOP (.), and that is not the last character in `input`, and the character after the character indicated by `position` is an ASCII digit, then set `value` to zero and jump to the step labeled `fraction`.
10. If the character indicated by `position` is not an ASCII digit, then return an error.
11. Collect a sequence of characters that are ASCII digits, and interpret the resulting sequence as a base-ten integer. Multiply `value` by that integer.
12. If `position` is past the end of `input`, jump to the step labeled `conversion`.
13. *Fraction:* If the character indicated by `position` is a U+002E FULL STOP (.), run these substeps:
 1. Advance `position` to the next character.
 2. If `position` is past the end of `input`, or if the character indicated by `position` is not an ASCII digit, U+0065 LATIN SMALL LETTER E (e), or U+0045 LATIN CAPITAL LETTER E (E), then jump to the step labeled `conversion`.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

5. Add the value of the character indicated by *position*, interpreted as a base-ten digit (0..9) and divided by *divisor*, to *value*.
6. Advance *position* to the next character.
7. If *position* is past the end of *input*, then jump to the step labeled *conversion*.
8. If the character indicated by *position* is an ASCII digit, jump back to the step labeled *fraction loop* in these substeps.
14. If the character indicated by *position* is a U+0065 LATIN SMALL LETTER E character (e) or a U+0045 LATIN CAPITAL LETTER E (E), then do the following:
 3. If *position* is past the end of *input*, then jump to the step labeled *conversion*.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

3. If *position* is past the end of *input*, then jump to the step labeled *conversion*.
- Otherwise, if the character indicated by *position* is a U+002B PLUS SIGN character (+):
1. Advance *position* to the next character.
 2. If *position* is past the end of *input*, then jump to the step labeled *conversion*.
 4. If the character indicated by *position* is not an ASCII digit, then jump to the step labeled *conversion*.
 5. Collect a sequence of characters that are ASCII digits, and interpret the resulting sequence as a base-ten integer. Multiply *exponent* by that integer.
 6. Multiply *value* by ten raised to the *exponent*th power.
 15. *Conversion*: Let S be the set of finite IEEE 754 double-precision floating-point values except -0, but with two special values added: 2^{1024} and -2^{1024} .
 16. Let *rounded-value* be the number in S that is closest to *value*, selecting the number with an even significand if there are two equally close values. (The two special values 2^{1024} and -2^{1024} are considered to have even significands for this purpose.)
 17. If *rounded-value* is 2^{1024} or -2^{1024} , return an error.
 18. Return *rounded-value*.

2.5.4.4 Percentages and lengths

The **rules for parsing dimension values** are as given in the following algorithm. When invoked, the steps must be followed in the order given, aborting at the first step that returns a value. This algorithm will return either a number greater than or equal to 1.0, or an error; if a number is returned, then it is further categorized as either a percentage or a length.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Skip whitespace.
4. If *position* is past the end of *input*, return an error.
5. If the character indicated by *position* is a U+002B PLUS SIGN character (+), advance *position* to the next character.
6. Collect a sequence of characters that are U+0030 DIGIT ZERO (0) characters, and discard them.
7. If *position* is past the end of *input*, return an error.
8. If the character indicated by *position* is not one of U+0031 DIGIT ONE (1) to U+0039 DIGIT NINE (9), then return an error.
9. Collect a sequence of characters that are ASCII digits, and interpret the resulting sequence as a base-ten integer. Let *value* be

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. Advance *position* to the next character.
2. If *position* is past the end of *input*, or if the character indicated by *position* is not an ASCII digit, then return *value* as a length.
3. Let *divisor* have the value 1.
4. *Fraction loop*: Multiply *divisor* by ten.

5 Add the value of the character indicated by *position* interpreted as a base-ten digit (Q) and divided by *divisor*, to

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

12. If *pos* is past the end of *input*, return *value* as a length.
13. If the character indicated by *position* is a U+0025 PERCENT SIGN character (%), return *value* as a percentage.
14. Return *value* as a length.

2.5.4.5 Lists of integers

A **valid list of integers** is a number of valid integers separated by U+002C COMMA characters, with no other characters (e.g. no space characters). In addition, there might be restrictions on the number of integers that can be given, or on the range of values allowed.

The **rules for parsing a list of integers** are as follows:

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Let *numbers* be an initially empty list of integers. This list will be the result of this algorithm.
4. If there is a character in the string *input* at position *position*, and it is either a U+0020 SPACE, U+002C COMMA, or U+003B SEMICOLON character, then advance *position* to the next character in *input*, or to beyond the end of the string if there are no more characters.
5. If *position* points to beyond the end of *input*, return *numbers* and abort.
6. If the character in the string *input* at position *position* is a U+0020 SPACE, U+002C COMMA, or U+003B SEMICOLON character, then return to step 4.
7. Let *negated* be false.
8. Let *value* be 0.
9. Let *started* be false. This variable is set to true when the parser sees a number or a U+002D HYPHEN-MINUS character (-).
10. Let *got number* be false. This variable is set to true when the parser sees a number.
11. Let *finished* be false. This variable is set to true to switch parser into a mode where it ignores characters until the next separator.
12. Let *bogus* be false.
13. *Parser*: If the character in the string *input* at position *position* is:

↳ A U+002D HYPHEN-MINUS character

Follow these substeps:

1. If *got number* is true, let *finished* be true.
2. If *finished* is true, skip to the next step in the overall set of steps.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

↳ An ASCII digit

Follow these substeps:

1. If *finished* is true, skip to the next step in the overall set of steps.
2. Multiply *value* by ten.
3. Add the value of the digit, interpreted in base ten, to *value*.



Too slow? Try reading the multipage copy of

- ↳ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
- ↳ /web/20130103030411/http://whatwg.org/html

. entry in the list has no digits,
as in "1, 2, x, 4".

2. If *negated* is true, then negate *value*.
3. Append *value* to the *numbers* list.
4. Jump to step 4 in the overall set of steps.

- ↳ A character in the range U+0001 to U+001F, U+0021 to U+002B, U+002D to U+002F, U+003A, U+003C to U+0040, U+005B to U+0060, U+007b to U+007F (i.e. any other non-alphabetic ASCII character)

Follow these substeps:

1. If *got number* is true, let *finished* be true.
2. If *finished* is true, skip to the next step in the overall set of steps.
3. Let *negated* be false.

↳ Any other character

Follow these substeps:

1. If *finished* is true, skip to the next step in the overall set of steps.
2. Let *negated* be false.
3. Let *bogus* be true.
4. If *started* is true, then return the *numbers* list, and abort. (The value in *value* is not appended to the list first; it is dropped.)
14. Advance *position* to the next character in *input*, or to beyond the end of the string if there are no more characters.
15. If *position* points to a character (and not to beyond the end of *input*), jump to the big *Parser* step above.
16. If *negated* is true, then negate *value*.
17. If *got number* is true, then append *value* to the *numbers* list.
18. Return the *numbers* list and abort.

2.5.4.6 Lists of dimensions

The rules for parsing a list of dimensions are as follows. These rules return a list of zero or more pairs consisting of a number and a unit, the unit being one of *percentage*, *relative*, and *absolute*.

1. Let *raw input* be the string being parsed.
2. If the last character in *raw input* is a U+002C COMMA character (,), then remove that character from *raw input*.

3,445 captures
5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB
◀ 03 ▶
2012 2013 2014

About this capture

1. Let *input* be the token.
 2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
 3. Let *value* be the number 0.
 4. Let *unit* be *absolute*.
 5. If *position* is past the end of *input*, set *unit* to *relative* and jump to the last substep. 
- Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
- CII digits, interpret the . Let *s* be the resulting
3. If *s* is not the empty string, run these subsubsteps:
 1. Let *length* be the number of characters in *s* (after the spaces were removed).
 2. Let *fraction* be the result of interpreting *s* as a base-ten integer, and then dividing that number by 10^{length} .
 3. Increment *value* by *fraction*.
 8. Skip whitespace.
 9. If the character at *position* is a U+0025 PERCENT SIGN character (%), then set *unit* to *percentage*. Otherwise, if the character at *position* is a U+002A ASTERISK character (*), then set *unit* to *relative*.
 10. Add an entry to *result* consisting of the number given by *value* and the unit given by *unit*.
 6. Return the list *result*.

2.5.5 Dates and times

In the algorithms below, the **number of days in month month of year year** is: 31 if *month* is 1, 3, 5, 7, 8, 10, or 12; 30 if *month* is 4, 6, 9, or 11; 29 if *month* is 2 and *year* is a number divisible by 400, or if *year* is a number divisible by 4 but not by 100; and 28 otherwise. This takes into account leap years in the Gregorian calendar. [GREGORIAN]

When ASCII digits are used in the date and time syntaxes defined in this section, they express numbers in base ten.

Note: While the formats described here are intended to be subsets of the corresponding ISO8601 formats, this specification defines parsing rules in much more detail than ISO8601. Implementors are therefore encouraged to carefully examine any date parsing libraries before using them to implement the parsing rules described below; ISO8601 libraries might not parse dates and times in exactly the same manner. [ISO8601]

Note: The use of the Gregorian calendar as the wire format in this specification is an arbitrary choice resulting from the cultural biases of those involved in the decision. See also the section discussing date, time, and number formats in forms (for authors), implementation notes regarding localization of form controls, and the `time` element.

2.5.5.1 Months

A **month** consists of a specific proleptic Gregorian date with no time-zone information and no date information beyond a year and a month. [GREGORIAN]

A string is a **valid month string** representing a year *year* and month *month* if it consists of the following components in the given order:

1. Four or more ASCII digits, representing *year*, where *year* > 0

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Parse a month component to obtain *year* and *month*. If this returns nothing, then fail.
4. If *position* is *not* beyond the end of *input*, then fail.
5. Return *year* and *month*.



The rules to **parse a date string** are as follows. This will return either a date, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Collate the characters of *input* into *year* and *month*.
2. If *year* is either a year and a month, then return *year* and *month*.
3. If *year* is not either a year and a month, then fail.
4. If *month* is either a year and a month, then return *year* and *month*.
5. If *month* is not either a year and a month, then fail.
6. If *month* is either a year and a month, then return *year* and *month*.

If *year* is either a year and a month, then return *year* and *month*. If *month* is either a year and a month, then return *year* and *month*. If *year* is not either a year and a month, then fail.

If *month* is either a year and a month, then return *year* and *month*. If *month* is not either a year and a month, then fail.

If *year* is not either a year and a month, then fail.

2.5.5.2 Dates

A **date** consists of a specific proleptic Gregorian date with no time-zone information, consisting of a year, a month, and a day. [GREGORIAN]

A string is a **valid date string** representing a year *year*, month *month*, and day *day* if it consists of the following components in the given order:

1. A valid month string, representing *year* and *month*
2. A U+002D HYPHEN-MINUS character (-)
3. Two ASCII digits, representing *day*, in the range $1 \leq \text{day} \leq \text{maxday}$ where *maxday* is the number of days in the month *month* and year *year*

The rules to **parse a date string** are as follows. This will return either a date, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Parse a date component to obtain *year*, *month*, and *day*. If this returns nothing, then fail.
4. If *position* is *not* beyond the end of *input*, then fail.
5. Let *date* be the date with year *year*, month *month*, and day *day*.
6. Return *date*.

The rules to **parse a date component**, given an *input* string and a *position*, are as follows. This will return either a year, a month, and a day, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Parse a month component to obtain *year* and *month*. If this returns nothing, then fail.
2. Let *maxday* be the number of days in month *month* of year *year*.
3. If *position* is beyond the end of *input* or if the character at *position* is not a U+002D HYPHEN-MINUS character, then fail.
- Otherwise, move *position* forwards one character.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

6. Return *year*, *month*, and *day*.

2.5.5.3 Yearless dates

A **yearless date** consists of a Gregorian month and a day within that month, but with no associated year. [GREGORIAN]

A string is a **valid yearless date string** representing a month *month* and a day *day* if it consists of the following components in the given order:

1. Option

2. Two *Two* Too slow? Try reading the multipage copy of
 3. A U the spec instead: <https://web.archive.org>
 4. Two /web/20130103030411/http:
 and //whatwg.org/html

f days in the month *month**Note: In**year was a leap year.*

The rules to **parse a yearless date string** are as follows. This will return either a month and a day, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Parse a yearless date component to obtain *month* and *day*. If this returns nothing, then fail.
4. If *position* is not beyond the end of *input*, then fail.
5. Return *month* and *day*.

The rules to **parse a yearless date component**, given an *input* string and a *position*, are as follows. This will return either a month and a day, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Collect a sequence of characters that are U+002D HYPHEN-MINUS characters (-). If the collected sequence is not exactly zero or two characters long, then fail.
2. Collect a sequence of characters that are ASCII digits. If the collected sequence is not exactly two characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *month*.
3. If *month* is not a number in the range $1 \leq \text{month} \leq 12$, then fail.
4. Let *maxday* be the number of days in month *month* of any arbitrary leap year (e.g. 4 or 2000).
5. If *position* is beyond the end of *input* or if the character at *position* is not a U+002D HYPHEN-MINUS character, then fail. Otherwise, move *position* forwards one character.
6. Collect a sequence of characters that are ASCII digits. If the collected sequence is not exactly two characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *day*.
7. If *day* is not a number in the range $1 \leq \text{day} \leq \text{maxday}$, then fail.
8. Return *month* and *day*.

2.5.5.4 Times

A **time** consists of a specific time with no time-zone information, consisting of an hour, a minute, a second, and a fraction of a second.

A string is a **valid time string** representing an hour *hour*, a minute *minute*, and a second *second* if it consists of the following components in the given order:

1. Two ASCII digits, representing *hour*, in the range $0 \leq \text{hour} \leq 23$
2. A U+003A COLON character (:)
3. Two ASCII digits, representing *minute*, in the range $0 \leq \text{minute} \leq 59$

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

2. One, two, or three ASCII digits, representing the fractional part of *second***Note:** The second component cannot be 60 or 61; leap seconds cannot be represented.

The rules to **parse a time string** are as follows. This will return either a time, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *r*
3. Par
4. If *pos*
5. Let */web/20130103030411/http:/*
6. Ret *//whatwg.org/html*

The rules to **parse a time component**, given an *input* string and a position, will return either an hour, a minute, and a second, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Collect a sequence of characters that are ASCII digits. If the collected sequence is not exactly two characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *hour*.
2. If *hour* is not a number in the range $0 \leq \text{hour} \leq 23$, then fail.
3. If *position* is beyond the end of *input* or if the character at *position* is not a U+003A COLON character, then fail. Otherwise, move *position* forwards one character.
4. Collect a sequence of characters that are ASCII digits. If the collected sequence is not exactly two characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *minute*.
5. If *minute* is not a number in the range $0 \leq \text{minute} \leq 59$, then fail.
6. Let *second* be a string with the value "0".
7. If *position* is not beyond the end of *input* and the character at *position* is a U+003A COLON, then run these substeps:
 1. Advance *position* to the next character in *input*.
 2. If *position* is beyond the end of *input*, or at the last character in *input*, or if the next two characters in *input* starting at *position* are not both ASCII digits, then fail.
 3. Collect a sequence of characters that are either ASCII digits or U+002E FULL STOP characters. If the collected sequence is three characters long, or if it is longer than three characters long and the third character is not a U+002E FULL STOP character, or if it has more than one U+002E FULL STOP character, then fail. Otherwise, let the collected string be *second* instead of its previous value.
8. Interpret *second* as a base-ten number (possibly with a fractional part). Let *second* be that number instead of the string version.
9. If *second* is not a number in the range $0 \leq \text{second} < 60$, then fail.
10. Return *hour*, *minute*, and *second*.

2.5.5.5 Local dates and times

A **local date and time** consists of a specific proleptic Gregorian date, consisting of a year, a month, and a day, and a time, consisting of an hour, a minute, a second, and a fraction of a second, but expressed without a time zone. [GREGORIAN]

A string is a **valid local date and time string** representing a date and time if it consists of the following components in the given order:

1. A valid date string representing the date
2. A U+0054 LATIN CAPITAL LETTER T character (T) or a U+0020 SPACE character
3. A valid time string representing the time

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

2. A U+0054 LATIN CAPITAL LETTER T character (T)

3. A valid time string representing the time, expressed as the shortest possible string for the given time (e.g. omitting the seconds component entirely if the given time is zero seconds past the minute)

The rules to **parse a local date and time string** are as follows. This will return either a date and time, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.2. Let *p* ↗3. Parse **Too slow? Try reading the multipage copy of**4. If *p* ↗ **the spec instead: https://web.archive.org**

ITAL LETTER T character

(T) ↗ **/web/20130103030411/http:**5. Parse **//whatwg.org/html**6. If *p* ↗ **/whatwg.org/html**7. Let *date* be the date with year *year*, month *month*, and day *day*.8. Let *time* be the time with hour *hour*, minute *minute*, and second *second*.9. Return *date* and *time*.

2.5.5.6 Time zones

A **time-zone offset** consists of a signed number of hours and minutes.

A string is a **valid time-zone offset string** representing a time-zone offset if it consists of either:

- A U+005A LATIN CAPITAL LETTER Z character (Z), allowed only if the time zone is UTC
- Or, the following components, in the given order:
 1. Either a U+002B PLUS SIGN character (+) or, if the time-zone offset is not zero, a U+002D HYPHEN-MINUS character (-), representing the sign of the time-zone offset
 2. Two ASCII digits, representing the hours component *hour* of the time-zone offset, in the range $0 \leq \text{hour} \leq 23$
 3. Optionally, a U+003A COLON character (:) (
 4. Two ASCII digits, representing the minutes component *minute* of the time-zone offset, in the range $0 \leq \text{minute} \leq 59$

Note: This format allows for time-zone offsets from -23:59 to +23:59. In practice, however, right now the range of offsets of actual time zones is -12:00 to +14:00, and the minutes component of offsets of actual time zones is always either 00, 30, or 45. There is no guarantee that this will remain so forever, however; time zones are used as political footballs and are thus subject to very whimsical policy decisions.

Note: See also the usage notes and examples in the global date and time section below for details on using time-zone offsets with historical times that predate the formation of formal time zones.

The rules to **parse a time-zone offset string** are as follows. This will return either a time-zone offset, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.2. Let *position* be a pointer into *input*, initially pointing at the start of the string.3. Parse a time-zone offset component to obtain *timezone_{hours}* and *timezone_{minutes}*. If this returns nothing, then fail.4. If *position* is not beyond the end of *input*, then fail.5. Return the time-zone offset that is *timezone_{hours}* hours and *timezone_{minutes}* minutes from UTC.

The rules to **parse a time-zone offset component**, given an *input* string and a *position*, are as follows. This will return either time-zone

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

1. Let *timezonehours* be 0.
2. Let *timzoneminutes* be 0.
3. Advance *position* to the next character in *input*.

Otherwise, if the character at *position* is either a U+002B PLUS SIGN (+) or a U+002D HYPHEN-MINUS (-), then:

1. If the character at *position* is a U+002B PLUS SIGN (+), let *sign* be "positive". Otherwise, it's a U+002D HYPHEN-MINUS (-); let *sign* be "negative".



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

A COLON character, then

3. Collect a sequence of characters that are ASCII digits. If the collected sequence is not exactly two characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *timzoneminutes*.

If *s* is exactly four characters long, then run these substeps:

1. Interpret the first two characters of *s* as a base-ten integer. Let that number be the *timezonehours*.
2. Interpret the last two characters of *s* as a base-ten integer. Let that number be the *timzoneminutes*.

Otherwise, fail.

5. If *timezonehours* is not a number in the range $0 \leq \text{timezonehours} \leq 23$, then fail.

6. If *sign* is "negative", then negate *timezonehours*.

7. If *timzoneminutes* is not a number in the range $0 \leq \text{timzoneminutes} \leq 59$, then fail.

8. If *sign* is "negative", then negate *timzoneminutes*.

Otherwise, fail.

2. Return *timezonehours* and *timzoneminutes*.

2.5.5.7 Global dates and times

A **global date and time** consists of a specific proleptic Gregorian date, consisting of a year, a month, and a day, and a time, consisting of an hour, a minute, a second, and a fraction of a second, expressed with a time-zone offset, consisting of a signed number of hours and minutes. [GREGORIAN]

A string is a **valid global date and time string** representing a date, time, and a time-zone offset if it consists of the following components in the given order:

1. A valid date string representing the date
2. A U+0054 LATIN CAPITAL LETTER T character (T) or a U+0020 SPACE character
3. A valid time string representing the time
4. A valid time-zone offset string representing the time-zone offset

Times in dates before the formation of UTC in the mid twentieth century must be expressed and interpreted in terms of UT1 (contemporary Earth solar time at the 0° longitude), not UTC (the approximation of UT1 that ticks in SI seconds). Time before the formation of time zones must be expressed and interpreted as UT1 times with explicit time zones that approximate the contemporary difference between the appropriate local time and the time observed at the location of Greenwich, London.

The following are some examples of dates written as valid global date and time strings.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

One millisecond after noon on October 14th 1979, in the time zone in use on the east coast of the USA during daylight saving time.

"8592-01-01T02:09+02:09"

Midnight UTC on the 1st of January, 8592. The time zone associated with that time is two hours and nine minutes ahead of UTC, which is not currently a real time zone, but is nonetheless allowed.

Several things are notable about these dates:



be a valid date.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

-21 12:00Z" (with two

dar (insofar as moments in converted to the Gregorian era's birth is the 15th of Gregorian Calendar.

- The time and time-zone offset components are not optional.
- Dates before the year one can't be represented as a datetime in this version of HTML.
- Times of specific events in ancient times are, at best, approximations, since time was not well coordinated or measured until relatively recent decades.
- Time-zone offsets differ based on daylight savings time.

A string is a **valid normalized forced-UTC global date and time string** representing a date, time, and a time-zone offset if it consists of the following components in the given order:

1. A valid date string representing the date converted to the UTC time zone
2. A U+0054 LATIN CAPITAL LETTER T character (T)
3. A valid time string representing the time converted to the UTC time zone and expressed as the shortest possible string for the given time (e.g. omitting the seconds component entirely if the given time is zero seconds past the minute)
4. A U+005A LATIN CAPITAL LETTER Z character (Z)

The rules to **parse a global date and time string** are as follows. This will return either a time in UTC, with associated time-zone offset information for round-tripping or display purposes, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Parse a date component to obtain *year*, *month*, and *day*. If this returns nothing, then fail.
4. If *position* is beyond the end of *input* or if the character at *position* is neither a U+0054 LATIN CAPITAL LETTER T character (T) nor a U+0020 SPACE character, then fail. Otherwise, move *position* forwards one character.
5. Parse a time component to obtain *hour*, *minute*, and *second*. If this returns nothing, then fail.
6. If *position* is beyond the end of *input*, then fail.
7. Parse a time-zone offset component to obtain *timezonehours* and *timezoneminutes*. If this returns nothing, then fail.
8. If *position* is not beyond the end of *input*, then fail.
9. Let *time* be the moment in time at year *year*, month *month*, day *day*, hours *hour*, minute *minute*, second *second*, subtracting *timezonehours* hours and *timezoneminutes* minutes. That moment in time is a moment in the UTC time zone.
10. Let *timezone* be *timezonehours* hours and *timezoneminutes* minutes from UTC.
11. Return *time* and *timezone*.

Go

NOV JAN FEB

03

2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

sequentially. The week before the number 1 week in a week-year is the last week in the previous week-year, and vice versa.
[GREGORIAN]

A week-year with a number *year* has 53 weeks if it corresponds to either a year *year* in the proleptic Gregorian calendar that has a Thursday as its first day (January 1st), or a year *year* in the proleptic Gregorian calendar that has a Wednesday as its first day (January 1st) and where *year* is a number divisible by 400, or a number divisible by 4 but not by 100. All other week-years have 52 weeks.

The **week number of the last day** of a week-year with 53 weeks is 53; the week number of the last day of a week-year with 52 weeks is 52.



*Note: T
the pro
Gregor*

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*at contains that day in
first Thursday of the*

Note: F

/web/20130103030411/http:

O 8601. [ISO8601]

A string is a //whatwg.org/html order:

components in the given

1. Four or more ASCII digits, representing *year*, where *year* > 0
2. A U+002D HYPHEN-MINUS character (-)
3. A U+0057 LATIN CAPITAL LETTER W character (W)
4. Two ASCII digits, representing the week *week*, in the range $1 \leq \text{week} \leq \text{maxweek}$, where *maxweek* is the week number of the last day of week-year *year*

The rules to **parse a week string** are as follows. This will return either a week-year number and week number, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Collect a sequence of characters that are ASCII digits. If the collected sequence is not at least four characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *year*.
4. If *year* is not a number greater than zero, then fail.
5. If *position* is beyond the end of *input* or if the character at *position* is not a U+002D HYPHEN-MINUS character, then fail. Otherwise, move *position* forwards one character.
6. If *position* is beyond the end of *input* or if the character at *position* is not a U+0057 LATIN CAPITAL LETTER W character (W), then fail. Otherwise, move *position* forwards one character.
7. Collect a sequence of characters that are ASCII digits. If the collected sequence is not exactly two characters long, then fail. Otherwise, interpret the resulting sequence as a base-ten integer. Let that number be the *week*.
8. Let *maxweek* be the week number of the last day of year *year*.
9. If *week* is not a number in the range $1 \leq \text{week} \leq \text{maxweek}$, then fail.
10. If *position* is not beyond the end of *input*, then fail.
11. Return the week-year number *year* and the week number *week*.

2.5.5.9 Durations

A **duration** consists of a number of seconds.

Note: Since months and seconds are not comparable (a month is not a precise number of seconds, but is instead a period whose exact length depends on the precise day from which it is measured) a duration as defined in this specification cannot include months (or years, which are equivalent to twelve months). Only durations that describe a specific number of seconds can be described.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. One or more ASCII digits followed by a U+0044 LATIN CAPITAL LETTER D character, representing a number of days.
2. A U+0054 LATIN CAPITAL LETTER T character followed by one or more of the following subcomponents, in the order given:
 1. One or more ASCII digits followed by a U+0048 LATIN CAPITAL LETTER H character, representing a number of hours.
 2. One or more ASCII digits followed by a U+004D LATIN CAPITAL LETTER M character, representing a number of minutes.



Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org/
/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

three ASCII digits,

Note: This, as with a number of other date- and time-related microsyntaxes defined in this specification, is based on one of the formats defined in ISO 8601. [ISO8601]

- One or more duration time components, each with a different duration time component scale, in any order; the sum of the represented seconds being equal to the number of seconds in t .

A **duration time component** is a string consisting of the following components:

1. Zero or more space characters.
2. One or more ASCII digits, representing a number of time units, scaled by the duration time component scale specified (see below) to represent a number of seconds.
3. If the duration time component scale specified is 1 (i.e. the units are seconds), then, optionally, a U+002E FULL STOP character (.) followed by one, two, or three ASCII digits, representing a fraction of a second.
4. Zero or more space characters.
5. One of the following characters, representing the **duration time component scale** of the time unit used in the numeric part of the duration time component:

U+0057 LATIN CAPITAL LETTER W character

U+0077 LATIN SMALL LETTER W character

Weeks. The scale is 604800.

U+0044 LATIN CAPITAL LETTER D character

U+0064 LATIN SMALL LETTER D character

Days. The scale is 86400.

U+0048 LATIN CAPITAL LETTER H character

U+0068 LATIN SMALL LETTER H character

Hours. The scale is 3600.

U+004D LATIN CAPITAL LETTER M character

U+006D LATIN SMALL LETTER M character

Minutes. The scale is 60.

U+0053 LATIN CAPITAL LETTER S character

U+0073 LATIN SMALL LETTER S character

Seconds. The scale is 1.

6. Zero or more space characters.

Note: This is not based on any of the formats in ISO 8601. It is intended to be a more human-readable alternative to the ISO 8601 duration format.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Let *months*, *seconds*, and *component count* all be zero.
4. Let *M-disambiguator* be *minutes*.

Note: This flag's other value is months. It is used to disambiguate the "M" unit in ISO8601 durations, which use the same unit for months and minutes. Months are not allowed, but are parsed for future compatibility and to avoid misinterpreting ISO8601 durations that would be valid in other contexts.

5. Skip
6. If *pos*
7. If th
- nex
8. Run
- algc

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

hen advance *position* to the

, or until a step requiring the

1. Let *units* be undefined. It will be assigned one of the following values: *years*, *months*, *weeks*, *days*, *hours*, *minutes*, and *seconds*.
2. Let *next character* be undefined. It is used to process characters from the *input*.
3. If *position* is past the end of *input*, then break the loop.
4. If the character in *input* pointed to by *position* is a U+0054 LATIN CAPITAL LETTER T character, then advance *position* to the next character, set *M-disambiguator* to *minutes*, skip whitespace, and return to the top of the loop.
5. Set *next character* to the character in *input* pointed to by *position*.
6. If *next character* is a U+002E FULL STOP character (.), then let *N* equal zero. (Do not advance *position*. That is taken care of below.)

Otherwise, if *next character* is an ASCII digit, then collect a sequence of characters that are ASCII digits, interpret the resulting sequence as a base-ten integer, and let *N* be that number.

Otherwise *next character* is not part of a number; fail.

7. If *position* is past the end of *input*, then fail.
8. Set *next character* to the character in *input* pointed to by *position*, and this time advance *position* to the next character. (If *next character* was a U+002E FULL STOP character (.) before, it will still be that character this time.)
9. If *next character* is a U+002E FULL STOP character (.), then run these substeps:

1. Collect a sequence of characters that are ASCII digits. Let *s* be the resulting sequence.
2. If *s* is the empty string, then fail.
3. Let *length* be the number of characters in *s*.
4. Let *fraction* be the result of interpreting *s* as a base-ten integer, and then dividing that number by 10^{length} .
5. Increment *N* by *fraction*.
6. Skip whitespace.
7. If *position* is past the end of *input*, then fail.
8. Set *next character* to the character in *input* pointed to by *position*, and advance *position* to the next character.
9. If *next character* is neither a U+0053 LATIN CAPITAL LETTER S character nor a U+0073 LATIN SMALL LETTER S character, then fail.
10. Set *units* to *seconds*.

Otherwise, run these substeps:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If *next character* is a U+004D LATIN CAPITAL LETTER M character or a U+006D LATIN SMALL LETTER M character, and *M-disambiguator* is *months*, then set *units* to *months*.

If *next character* is a U+0057 LATIN CAPITAL LETTER W character or a U+0077 LATIN SMALL LETTER W character, set *units* to *weeks* and set *M-disambiguator* to *minutes*.

If *next character* is a U+0044 LATIN CAPITAL LETTER D character or a U+0064 LATIN SMALL LETTER D character, set *units* to *days* and set *M-disambiguator* to *minutes*.



LATIN SMALL LETTER H

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

LATIN SMALL LETTER M

LATIN SMALL LETTER S

10. Increment *component count*.
11. Let *multiplier* be 1.
12. If *units* is *years*, multiply *multiplier* by 12 and set *units* to *months*.
13. If *units* is *months*, add the product of *N* and *multiplier* to *months*.

Otherwise, run these substeps:

1. If *units* is *weeks*, multiply *multiplier* by 7 and set *units* to *days*.
2. If *units* is *days*, multiply *multiplier* by 24 and set *units* to *hours*.
3. If *units* is *hours*, multiply *multiplier* by 60 and set *units* to *minutes*.
4. If *units* is *minutes*, multiply *multiplier* by 60 and set *units* to *seconds*.
5. Forcibly, *units* is now *seconds*. Add the product of *N* and *multiplier* to *seconds*.

14. Skip whitespace.
9. If *component count* is zero, fail.
10. If *months* is not zero, fail.
11. Return the duration consisting of *seconds* seconds.

2.5.5.10 Vaguer moments in time

A string is a **valid date string with optional time** if it is also one of the following:

- A valid date string
- A valid global date and time string

The rules to **parse a date or time string** are as follows. The algorithm will return either a date, a time, a global date and time, or nothing. If at any point the algorithm says that it "fails", this means that it is aborted at that point and returns nothing.

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Set *start position* to the same position as *position*.
4. Set the *date present* and *time present* flags to true.
5. Parse a date component to obtain *year*, *month*, and *day*. If this fails, then set the *date present* flag to false.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Otherwise, if *date present* is false, set *position* back to the same position as *start position*.

7. If the *time present* flag is true, then parse a time component to obtain *hour*, *minute*, and *second*. If this returns nothing, then fail.

8. If the *date present* and *time present* flags are both true, but *position* is beyond the end of *input*, then fail.

9. If the *date present* and *time present* flags are both true, parse a time-zone offset component to obtain *timezonehours* and *timezoneminutes*. If this returns nothing, then fail.

10. If *pc*

Too slow? Try reading the multipage copy of

11. If th
day

the spec instead: <https://web.archive.org>

year, month *month*, and day

Oth
min

[/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

e with hour *hour*, minute

Oth
min

[/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

min, second *second*,

subtracing *timezonehours* hours and *timezoneminutes* minutes, that moment in time being a moment in the UTC time zone; let *timezone* be *timezonehours* hours and *timezoneminutes* minutes from UTC; and return *time* and *timezone*.

2.5.6 Colors

A **simple color** consists of three 8-bit numbers in the range 0..255, representing the red, green, and blue components of the color respectively, in the sRGB color space. [SRGB]

A string is a **valid simple color** if it is exactly seven characters long, and the first character is a U+0023 NUMBER SIGN character (#), and the remaining six characters are all ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component, in hexadecimal.

A string is a **valid lowercase simple color** if it is a valid simple color and doesn't use any characters in the range U+0041 LATIN CAPITAL LETTER A to U+0046 LATIN CAPITAL LETTER F.

The **rules for parsing simple color values** are as given in the following algorithm. When invoked, the steps must be followed in the order given, aborting at the first step that returns a value. This algorithm will return either a simple color or an error.

1. Let *input* be the string being parsed.
2. If *input* is not exactly seven characters long, then return an error.
3. If the first character in *input* is not a U+0023 NUMBER SIGN character (#), then return an error.
4. If the last six characters of *input* are not all ASCII hex digits, then return an error.
5. Let *result* be a simple color.
6. Interpret the second and third characters as a hexadecimal number and let the result be the red component of *result*.
7. Interpret the fourth and fifth characters as a hexadecimal number and let the result be the green component of *result*.
8. Interpret the sixth and seventh characters as a hexadecimal number and let the result be the blue component of *result*.
9. Return *result*.

The **rules for serializing simple color values** given a simple color are as given in the following algorithm:

1. Let *result* be a string consisting of a single U+0023 NUMBER SIGN character (#).
2. Convert the red, green, and blue components in turn to two-digit hexadecimal numbers using lowercase ASCII hex digits, zero-padding if necessary, and append these numbers to *result*, in the order red, green, blue.
3. Return *result*, which will be a valid lowercase simple color.

Some obsolete legacy attributes parse colors in a more complicated manner, using the **rules for parsing a legacy color value**, which are given in the following algorithm. When invoked, the steps must be followed in the order given, aborting at the first step that returns a value. This algorithm will return either a simple color or an error.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

4. If *input* is an ASCII case-insensitive match for the string "transparent", then return an error.
5. If *input* is an ASCII case-insensitive match for one of the keywords listed in the SVG color keywords section of the CSS3 Color specification, then return the simple color corresponding to that keyword. [CSSCOLOR]

Note: CSS2 System Colors are not recognised.

6. If *input* is four characters long, and the first character in *input* is a U+0023 NUMBER SIGN character (#), and the last three characters in *input* are "ACCOLA", then let *result* be the resulting number.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

multiplied by 17.

5. Return *result*.
7. Replace any characters in *input* that have a Unicode code point greater than U+FFFF (i.e. any characters that are not in the basic multilingual plane) with the two-character string "00".
8. If *input* is longer than 128 characters, truncate *input*, leaving only the first 128 characters.
9. If the first character in *input* is a U+0023 NUMBER SIGN character (#), remove it.
10. Replace any character in *input* that is not an ASCII hex digit with the character U+0030 DIGIT ZERO (0).
11. While *input*'s length is zero or not a multiple of three, append a U+0030 DIGIT ZERO (0) character to *input*.
12. Split *input* into three strings of equal length, to obtain three components. Let *length* be the length of those components (one third the length of *input*).
13. If *length* is greater than 8, then remove the leading *length*-8 characters in each component, and let *length* be 8.
14. While *length* is greater than two and the first character in each component is a U+0030 DIGIT ZERO (0) character, remove that character and reduce *length* by one.
15. If *length* is still greater than two, truncate each component, leaving only the first two characters in each.
16. Let *result* be a simple color.
17. Interpret the first component as a hexadecimal number; let the red component of *result* be the resulting number.
18. Interpret the second component as a hexadecimal number; let the green component of *result* be the resulting number.
19. Interpret the third component as a hexadecimal number; let the blue component of *result* be the resulting number.
20. Return *result*.

Note: The 2D graphics context has a separate color syntax that also handles opacity.

2.5.7 Space-separated tokens

A **set of space-separated tokens** is a string containing zero or more words (known as tokens) separated by one or more space characters, where words consist of any string of one or more characters, none of which are space characters.

A string containing a set of space-separated tokens may have leading or trailing space characters.

An **unordered set of unique space-separated tokens** is a set of space-separated tokens where none of the tokens are duplicated.

An **ordered set of unique space-separated tokens** is a set of space-separated tokens where none of the tokens are duplicated but

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Note: How tokens in a set of space-separated tokens are to be compared (e.g. case-sensitively or not) is defined on a per-set basis.

When a user agent has to **split a string on spaces**, it must use the following algorithm:

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Let *tokens* be an ordered list of tokens, initially empty.
4. Skip whitespace around each token.
5. While *position* is not past the end of *input*:
 1. Let *start* be the value of *position*.
 2. Let *end* be the first character in *input* after *start* that is not whitespace.
 3. Append the substring from *start* to *end* to *tokens*.
 4. Advance *position* to *end*.
6. Return *tokens*.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

2.5.8 Comma-separated tokens

A **set of comma-separated tokens** is a string containing zero or more tokens each separated from the next by a single U+002C COMMA character (,), where tokens consist of any string of zero or more characters, neither beginning nor ending with space characters, nor containing any U+002C COMMA characters (,), and optionally surrounded by space characters.

For instance, the string " a ,b,,d d " consists of four tokens: "a", "b", the empty string, and "d d". Leading and trailing whitespace around each token doesn't count as part of the token, and the empty string can be a token.

Sets of comma-separated tokens sometimes have further restrictions on what counts as a valid token. When such restrictions are defined, the tokens must all fit within those restrictions; other values are non-conforming. If no such restrictions are specified, then all values are conforming.

When a user agent has to **split a string on commas**, it must use the following algorithm:

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Let *tokens* be an ordered list of tokens, initially empty.
4. *Token*: If *position* is past the end of *input*, jump to the last step.
5. Collect a sequence of characters that are not U+002C COMMA characters (,). Let *s* be the resulting sequence (which might be the empty string).
6. Strip leading and trailing whitespace from *s*.
7. Append *s* to *tokens*.
8. If *position* is not past the end of *input*, then the character at *position* is a U+002C COMMA character (,); advance *position* past that character.
9. Jump back to the step labeled *token*.
10. Return *tokens*.

2.5.9 References

A **valid hash-name reference** to an element of type *type* is a string consisting of a U+0023 NUMBER SIGN character (#) followed by a string which exactly matches the value of the `name` attribute of an element with type *type* in the document.

The **rules for parsing a hash-name reference** to an element of type *type* are as follows:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. Return the first element of type `type` that has an `id` attribute whose value is a case-sensitive match for `s` or a `name` attribute whose value is a compatibility caseless match for `s`.

2.5.10 Media queries

A string is a **valid media query** if it matches the `media_query_list` production of the Media Queries specification. [MQ]

A string matches the environment of the user if it is the empty string, a string consisting of whitespace characters, or is a media query that matches

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org>

2.6 URL

This specific
defined by the
with Web co

/web/20130103030411/http:
/whatwg.org/html

historical reasons the rules
implement to be compatible

Note: The term "URL" in this specification is used in a manner distinct from the precise technical meaning it is given in RFC 3986. Readers familiar with that RFC will find it easier to read this specification if they pretend the term "URL" as used herein is really called something else altogether. This is a willful violation of RFC 3986. [RFC3986]

2.6.1 Terminology

A **URL** is a string used to identify a resource.

A URL is a **valid URL** if at least one of the following conditions holds:

- The URL is a valid URI reference [RFC3986].
- The URL is a valid IRI reference and it has no query component. [RFC3987]
- The URL is a valid IRI reference and its query component contains no unescaped non-ASCII characters. [RFC3987]
- The URL is a valid IRI reference and the character encoding of the URL's Document is UTF-8 or a UTF-16 encoding. [RFC3987]

A string is a **valid non-empty URL** if it is a valid URL but it is not the empty string.

A string is a **valid URL potentially surrounded by spaces** if, after stripping leading and trailing whitespace from it, it is a valid URL.

A string is a **valid non-empty URL potentially surrounded by spaces** if, after stripping leading and trailing whitespace from it, it is a valid non-empty URL.

This specification defines the URL `about:legacy-compat` as a reserved, though unresolvable, `about:` URI, for use in DOCTYPES in HTML documents when needed for compatibility with XML tools. [ABOUT]

This specification defines the URL `about:srcdoc` as a reserved, though unresolvable, `about:` URI, that is used as the document's address of `iframe srcdoc` documents. [ABOUT]

2.6.2 Parsing URLs

To **parse a URL** `url` into its component parts, the user agent must use the following steps:

1. Strip leading and trailing whitespace from `url`.
2. Parse `url` in the manner defined by RFC 3986, with the following exceptions:
 - o Add all characters with code points less than or equal to U+0020 or greater than or equal to U+007F to the `<unreserved>` production.
 - o Add the characters U+0022, U+003C, U+003E, U+005B .. U+005E, U+0060, and U+007B .. U+007D to the `<unreserved>` production.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. If *url* doesn't match the <URI-reference> production, even after the above changes are made to the ABNF definitions, then parsing the URL fails with an error. [RFC3986]

Otherwise, parsing *url* was successful; the components of the URL are substrings of *url* defined as follows:

<scheme>

The substring matched by the <scheme> production, if any.

<host>

The substring matched by the <host> production, if any.



<pc> Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

<hc> /web/20130103030411/http://whatwg.org/html

rt> component is different
>stport> is the substring that attached by the <port>
.nosc> component.

<path>

The substring matched by one of the following productions, if one of them was matched:

- o <path-abempty>
- o <path-absolute>
- o <path-noscheme>
- o <path-rootless>
- o <path-empty>

<query>

The substring matched by the <query> production, if any.

<fragment>

The substring matched by the <fragment> production, if any.

<host-specific>

The substring that follows the substring matched by the <authority> production, or the whole string if the <authority> production wasn't matched.

Note: These parsing rules are a willful violation of RFC 3986 and RFC 3987 (which do not define error handling), motivated by a desire to handle legacy content. [RFC3986] [RFC3987]

2.6.3 Base URLs

The **fallback base URL** of a `Document` object is the absolute URL obtained by running these substeps:

1. If the `Document` is an `iframe srcdoc` document, then return the document base URL of the `Document`'s browsing context's browsing context container's `Document` and abort these steps.
2. If the document's address is `about:blank`, and the `Document`'s browsing context has a creator browsing context, then return the document base URL of the creator `Document`, and abort these steps.
3. Return the document's address.

The **document base URL** of a `Document` object is the absolute URL obtained by running these substeps:

1. Let *fallback base url* be the `Document`'s fallback base URL.
2. If there is no `base` element that has an `href` attribute, then the document base URL is *fallback base url*; abort these steps.
Otherwise, let *url* be the value of the `href` attribute of the first such element.
3. Resolve *url* relative to *fallback base url* (thus, the `base href` attribute isn't affected by `xml:base` attributes).
4. The document base URL is the result of the previous step if it was successful; otherwise it is *fallback base url*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

steps. Resolving a URL can result in an error, in which case the URL is not resolvable.

1. Let *url* be the URL being resolved.2. Let *encoding* be determined as follows:

↳ If the URL had a character encoding defined when the URL was created or defined

The URL character encoding is as defined.

↳ If the URL came from a script (e.g. as an argument to a method)

↳ Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org

acter encoding.

3. If either /web/20130103030411/http:

4. If there is no base URL, then the URL.

Otherwise, let *base* be the base URI of the document entity being defined as the document base URL or the document that owns the element. [XMLBASE]

For the purposes of the XML Base specification, user agents must act as if all Document objects represented XML documents.

Note: It is possible for `xml:base` attributes to be present even in HTML fragments, as such attributes can be added dynamically using script. (Such scripts would not be conforming, however, as `xml:base` attributes are not allowed in HTML documents.)

5. Parse *url* into its component parts.6. If parsing *url* resulted in a <host> component, then replace the matching substring of *url* with the string that results from expanding any sequences of percent-encoded octets in that component that are valid UTF-8 sequences into Unicode characters as defined by UTF-8.

If any percent-encoded octets in that component are not valid UTF-8 sequences (e.g. sequences of percent-encoded octets that expand to surrogate code points), then return an error and abort these steps.

Apply the IDNA ToASCII algorithm to the matching substring, with both the AllowUnassigned and UseSTD3ASCIIRules flags set. Replace the matching substring with the result of the ToASCII algorithm.

If ToASCII fails to convert one of the components of the string, e.g. because it is too long or because it contains invalid characters, then return an error and abort these steps. [RFC3490]

7. If parsing *url* resulted in a <path> component, then replace the matching substring of *url* with the string that results from applying the following steps to each character other than U+0025 PERCENT SIGN (%) that doesn't match the original <path> production defined in RFC 3986:

1. Encode the character into a sequence of octets as defined by UTF-8.

2. Replace the character with the percent-encoded form of those octets. [RFC3986]

For instance if *url* was "://example.com/a^b@c%FFd%z/?e", then the <path> component's substring would be "/a^b@c%FFd%z/" and the two characters that would have to be escaped would be "^" and "@". The result after this step was applied would therefore be that *url* now had the value "://example.com/a%5Eb%2F98%BAc%FFd%z/?e".

8. If parsing *url* resulted in a <query> component, then replace the matching substring of *url* with the string that results from applying the following steps to each character other than U+0025 PERCENT SIGN (%) that doesn't match the original <query> production defined in RFC 3986:1. If the character in question cannot be expressed in the encoding *encoding*, then replace it with a single 0x3F octet (an ASCII question mark) and skip the remaining substeps for this character.2. Encode the character into a sequence of octets as defined by the encoding *encoding*.

3. Replace the character with the percent-encoded form of those octets. [RFC3986]

9. Apply the algorithm described in RFC 3986 section 5.2 Relative Resolution, using *url* as the potentially relative URI reference

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

For instance, if an absolute URI that would be returned by the above algorithm violates the restrictions specific to its scheme, e.g. a `data:` URI using the “//” server-based naming authority syntax, then user agents are to treat this as an error instead.

11. Let `result` be the target URI (`T`) returned by the Relative Resolution algorithm.
12. If `result` uses a scheme with a server-based naming authority, replace all U+005C REVERSE SOLIDUS (/) characters in `result` with U+002F SOLIDUS (/) characters.
13. Return `result`.



**Note: S
a willfu** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

US (/) characters, are
RFC3986] [RFC3987]

A URL is an
failure.

to, and that output is not a

An absolute
component ↵

ly after the <scheme>

An absolute URL is an **authority-based URL** if, when resolved and then parsed, there are two characters immediately after the <scheme> component and they are both U+002F SOLIDUS characters (/).

2.6.5 URL manipulation and creation

To **fragment-escape** a string `input`, a user agent must run the following steps:

1. Let `input` be the string to be escaped.
2. Let `position` point at the first character of `input`.
3. Let `output` be an empty string.
4. *Loop:* If `position` is past the end of `input`, then jump to the step labeled *end*.
5. If the character in `input` pointed to by `position` is in the range U+0000 to U+0020 or is one of the following characters:
 - U+0022 QUOTATION MARK character (“)
 - U+0023 NUMBER SIGN character (#)
 - U+0025 PERCENT SIGN character (%)
 - U+003C LESS-THAN SIGN character (<)
 - U+003E GREATER-THAN SIGN character (>)
 - U+005B LEFT SQUARE BRACKET character ([)
 - U+005C REVERSE SOLIDUS character (/)
 - U+005D RIGHT SQUARE BRACKET character (])
 - U+005E CIRCUMFLEX ACCENT character (^)
 - U+007B LEFT CURLY BRACKET character ({})
 - U+007C VERTICAL LINE character (|)
 - U+007D RIGHT CURLY BRACKET character (})

...then append the percent-encoded form of the character to `output`. [RFC3986]

Otherwise, append the character itself to `output`.

Note: This escapes any ASCII characters that are not valid in the URI <fragment> production without being escaped.

6. Advance `position` to the next character in `input`.
7. Return to the step labeled *loop*.
8. *End:* Return `output`.

2.6.6 Dynamic changes to base URLs

When an `xml:base` attribute is set, changed, or removed, the attribute's element, and all descendant elements, are affected by a base URL change.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

↳ If the element creates a hyperlink

If the absolute URL identified by the hyperlink is being shown to the user, or if any data derived from that URL is affecting the display, then the `href` attribute should be re-resolved relative to the element and the UI updated appropriately.

|| For example, the CSS `:link/:visited` pseudo-classes might have been affected.

If the hyperlink has a `ping` attribute and its absolute URL(s) are being shown to the user, then the `ping` attribute's tokens should be re-resolved relative to the element and the UI updated appropriately.

↳ If the element is a `link` element or a `script` element with a `src` attribute

Too slow? Try reading the multipage copy of

↳ Otherwise, the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

derived from that URL is updated appropriately.

/web/20130103030411/http://whatwg.org/html

events, although subsequent events no longer correspond to the

2.6.7 Interfaces for URL manipulation

An interface that has a complement of **URL decomposition IDL attributes** has seven attributes with the following definitions:

```
attribute DOMString protocol;
attribute DOMString host;
attribute DOMString hostname;
attribute DOMString port;
attribute DOMString pathname;
attribute DOMString search;
attribute DOMString hash;
```

This box is non-normative. Implementation requirements are given below this box.

`o.protocol [= value]`

Returns the current scheme of the underlying URL.

Can be set, to change the underlying URL's scheme.

`o.host [= value]`

Returns the current host and port (if it's not the default port) in the underlying URL.

Can be set, to change the underlying URL's host and port.

The host and the port are separated by a colon. The port part, if omitted, will be assumed to be the current scheme's default port.

`o.hostname [= value]`

Returns the current host in the underlying URL.

Can be set, to change the underlying URL's host.

`o.port [= value]`

Returns the current port in the underlying URL.

Can be set, to change the underlying URL's port.

`o.pathname [= value]`

Returns the current path in the underlying URL.

Can be set, to change the underlying URL's path.

`o.search [= value]`

Returns the current query component in the underlying URL.

Can be set, to change the underlying URL's query component.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The attributes defined to be URL decomposition IDL attributes must act as described for the attributes with the same corresponding names in this section.

In addition, an interface with a complement of URL decomposition IDL attributes defines an **input**, which is a URL that the attributes act on, and a **common setter action**, which is a set of steps invoked when any of the attributes' setters are invoked.

The seven URL decomposition IDL attributes have similar requirements.

On getting, if
in the table l
the "prefix" c
the end of th

**Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org/
/web/20130103030411/http:
//whatwg.org/html**

On setting, t
characters ir
absolute UR
string *output*
the user agent
output.

corresponding to the attribute
with any prefixes specified in
lumn appropriately added to

itated by %-escaping any
nn. Then, if the input is an
er agent must make a new
th the new value; otherwise,
er action with the value of
output.

When replacing a component in the URL, if the component is part of an optional group in the URL syntax consisting of a character followed by the component, the component (including its prefix character) must be included even if the new value is the empty string.

Note: The previous paragraph applies in particular to the ":" before a <port> component, the "?" before a <query> component, and the "#" before a <fragment> component.

For the purposes of the above definitions, URLs must be parsed using the URL parsing rules defined in this specification.

Attribute	Component	Getter Condition	Prefix	Suffix	Setter Preprocessor	Setter Condition
protocol	<scheme>	—	—	U+003A COLON (:)	Remove all trailing U+003A COLON characters (:)	The new value is not the empty string
host	<hostport>	input is an authority-based URL	—	—	—	The new value is not the empty string and input is an authority-based URL
hostname	<host>	input is an authority-based URL	—	—	Remove all leading U+002F SOLIDUS characters (/)	The new value is not the empty string and input is an authority-based URL
port	<port>	input is an authority-based URL, and contained a <port> component (possibly an empty one)	—	—	Remove all characters in the new value from the first that is not an ASCII digit, if any. Remove any leading U+0030 DIGIT ZERO characters (0) in the new value. If the resulting string is empty, set it to a single U+0030 DIGIT ZERO character (0).	input is an authority-based URL, and the new value, when interpreted as a base-ten integer, is less than or equal to 65535
pathname	<path>	input is a hierarchical URL	—	—	If it has no leading U+002F SOLIDUS character (/), prepend a U+002F SOLIDUS character (/) to the new value	input is hierarchical
search	<query>	input is a hierarchical URL, and contained a <query> component (possibly an empty one)	U+003F QUESTION MARK (?)	—	Remove one leading U+003F QUESTION MARK (?), if any	input is a hierarchical URL
hash	<fragment>	input contained a non-empty <fragment> component	U+0023 NUMBER SIGN (#)	—	Remove one leading U+0023 NUMBER SIGN (#), if any	—

The table below demonstrates how the getter condition for `search` results in different results depending on the exact original syntax of the URL:

Input URL	search value	Explanation
http://example.com/	empty string	No <query> component in input URL.
http://example.com/?	?	There is a <query> component, but it is empty. The question mark in the resulting value is the prefix.
http://example.com/?test	?test	The <query> component has the value "test".
http://example.com/?test#	?test	The (empty) <fragment> component is not part of the <query> component.

3,445 captures
5 Jun 2004 - 22 Aug 2020

/carrot#question%3f				(empty string)	string)
https://www.example.com:4443?	https:	www.example.com:4443	www.example.com	4443 /	? (empty string)

2.7 Fetching resources

2.7.1 Terminology

User agents ↴



in terms of HTTP. [HTTP]

The **HTTP copy action** is idempotent; it can be repeated without changing the result.

FTP. Such actions are

The **HTTP redirect action** is found" error /web/20130103030411/http://whatwg.org/html

gs. For example, a "file not

The **HTTP authentication action** is authenticatio

iple, the HTTP

A **referrer source** is either a **Document** or a **URL**.

2.7.2 Processing model

When a user agent is to **fetch** a resource or URL, optionally **from** an origin *origin*, optionally **using** a specific referrer source as an **override referrer source**, and optionally with any of a **synchronous flag**, a **manual redirect flag**, a **force same-origin flag**, and a **block cookies flag**, the following steps must be run. (When a **URL** is to be fetched, the URL identifies a resource to be obtained.)

1. If there is a specific **override referrer source**, and it is a URL, then let **referrer** be the **override referrer source**, and jump to the step labeled **clean referrer**.
2. Let **document** be the appropriate **Document** as given by the following list:
 - ↳ **If there is a specific override referrer source**
The **override referrer source**.
 - ↳ **When navigating**
The active document of the source browsing context.
 - ↳ **When fetching resources for an element**
The element's **Document**.
3. While **document** is an **iframe srcdoc** **document**, let **document** be **document**'s browsing context's browsing context container's **Document** instead.
4. If the origin of **Document** is not a scheme/host/port tuple, then set **referrer** to the empty string and jump to the step labeled **clean referrer**.
5. Let **referrer** be the document's address of **document**.
6. **Clean referrer**: Remove any **<fragment>** component from **referrer**.
7. If **referrer** is not the empty string, is not a **data:** URL, is not a **javascript:** URL, and is not the URL "about:blank", then generate the **address of the resource from which Request-URIs are obtained** as required by HTTP for the **Referer** (sic) header from **referrer**. [HTTP]

Otherwise, the **Referer** (sic) header must be omitted, regardless of its value.
8. If the algorithm was not invoked with the **synchronous flag**, perform the remaining steps asynchronously.
9. If the **Document** with which any tasks queued by this algorithm would be associated doesn't have an associated browsing context, then abort these steps.
10. This is the **main step**.

If the resource is identified by an absolute URL, and the resource is to be obtained using an idempotent action (such as an

Go

NOV JAN FEB

03



2012 2013 2014

About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Otherwise, if the resource is identified by an absolute URL with a scheme that does not define a mechanism to obtain the resource (e.g. it is a `mailto:` URL) or that the user agent does not support, then act as if the resource was an HTTP 204 No Content response with no other metadata.

Otherwise, if the resource is identified by the URL `about:blank`, then the resource is immediately available and consists of the empty string, with no metadata.

Otherwise, at a time convenient to the user and the user agent, download (or otherwise obtain) the resource, applying the semantics of the relevant specifications (e.g. performing an HTTP GET or POST operation or reading the file from disk, deref-

For **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**
gen

st-URIs are obtained

For **/web/20130103030411/http:**
igin, then the origin that [ORIGIN]

11. If the `//whatwg.org/html`
follows

he user agent must run the

1. Wait until ownership of the storage mutex can be taken by this instance of the fetching algorithm.
2. Take ownership of the storage mutex.
3. Update the cookies. [COOKIES]
4. Release the storage mutex so that it is once again free.

12. If the fetched resource is an HTTP redirect or equivalent, then:



↳ If the **force same-origin flag** is set and the URL of the target of the redirect does not have the same origin as the URL for which the fetch algorithm was invoked

Abort these steps and return failure from this algorithm, as if the remote host could not be contacted.

↳ If the **manual redirect flag** is set

Continue, using the fetched resource (the redirect) as the result of the algorithm. If the calling algorithm subsequently requires the user agent to **transparently follow the redirect**, then the user agent must resume this algorithm from the *main step*, but using the target of the redirect as the resource to fetch, rather than the original resource.

↳ Otherwise

First, apply any relevant requirements for redirects (such as showing any appropriate prompts). Then, redo *main step*, but using the target of the redirect as the resource to fetch, rather than the original resource. For HTTP requests, the new request must include the same headers as the original request, except for headers for which other requirements are specified (such as the `Host` header). [HTTP]

Note: The HTTP specification requires that 301, 302, and 307 redirects, when applied to methods other than the safe methods, not be followed without user confirmation. That would be an appropriate prompt for the purposes of the requirement in the paragraph above. [HTTP]

13. If the algorithm was not invoked with the **synchronous flag**: When the resource is available, or if there is an error of some description, queue a task that uses the resource as appropriate. If the resource can be processed incrementally, as, for instance, with a progressively interlaced JPEG or an HTML file, additional tasks may be queued to process the data as it is downloaded. The task source for these tasks is the networking task source.

Otherwise, return the resource or error information to the calling algorithm.

If the user agent can determine the actual length of the resource being fetched for an instance of this algorithm, and if that length is finite, then that length is the file's **size**. Otherwise, the subject of the algorithm (that is, the resource being fetched) has no known size. (For example, the HTTP `Content-Length` header might provide this information.)

The user agent must also keep track of the **number of bytes downloaded** for each instance of this algorithm. This number must exclude any out-of-band metadata, such as HTTP headers.

Note: The application cache processing model introduces some changes to the networking model to handle the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Note: Whether the type sniffing rules apply to the fetched resource depends on the algorithm that invokes the rules — they are not always applicable.

2.7.3 Encrypted HTTP and related security concerns

Anything in this specification that refers to HTTP also applies to HTTP-over-TLS, as represented by URLs representing the https scheme. [HTTPC¹]

⚠ Warning! Too slow? Try reading the multipage copy of [bad resources sent with erroneous content](https://web.archive.org/web/20130103030411/http://whatwg.org/html) instead: https://web.archive.org

User agents if the page user has previously visited,

Not doing so /web/20130103030411/http:

If a user connects to a server with full encryption, but just act as if there had been no encryption. If the user agent instead allowed the user to override the problem and then displayed the page as if it was fully and safely encrypted, the user could be easily tricked into accepting man-in-the-middle connections.

If a user connects to a server with full encryption, but the page then refers to an external resource that has an expired certificate, then the user agent will act as if the resource was unavailable, possibly also reporting the problem to the user. If the user agent instead allowed the resource to be used, then an attacker could just look for "secure" sites that used resources from a different host and only apply man-in-the-middle attacks to that host, for example taking over scripts in the page.

If a user bookmarks a site that uses a CA-signed certificate, and then later revisits that site directly but the site has started using a self-signed certificate, the user agent could warn the user that a man-in-the-middle attack is likely underway, instead of simply acting as if the page was not encrypted.

2.7.4 Determining the type of a resource

The **Content-Type metadata** of a resource must be obtained and interpreted in a manner consistent with the requirements of the Media Type Sniffing specification. [MIMESNIFF]

The **sniffed type of a resource** must be found in a manner consistent with the requirements given in the Media Type Sniffing specification for finding the *sniffed-type* of the relevant sequence of octets. [MIMESNIFF]

The **rules for sniffing images specifically** and the **rules for distinguishing if a resource is text or binary** are also defined in the Media Type Sniffing specification. Both sets of rules return a MIME type as their result. [MIMESNIFF]

⚠ Warning! It is imperative that the rules in the Media Type Sniffing specification be followed exactly. When a user agent uses different heuristics for content type detection than the server expects, security problems can occur. For more details, see the Media Type Sniffing specification. [MIMESNIFF]

2.7.5 Extracting character encodings from meta elements

The **algorithm for extracting a character encoding from a meta element**, given a string s, is as follows. It either returns a character encoding or nothing.

1. Let *position* be a pointer into s, initially pointing at the start of the string.
2. *Loop*: Find the first seven characters in s after *position* that are an ASCII case-insensitive match for the word "charset". If no such match is found, return nothing and abort these steps.
3. Skip any space characters that immediately follow the word "charset" (there might not be any).
4. If the next character is not a U+003D EQUALS SIGN (=), then move *position* to point just before that next character, and jump back to the step labeled *loop*.
5. Skip any space characters that immediately follow the equals sign (there might not be any).
6. Process the next character as follows:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

↪ If it is an unmatched U+0022 QUOTATION MARK character (")

↪ If it is an unmatched U+0027 APOSTROPHE character (')

↪ If there is no next character

 Return nothing.

↪ Otherwise

 Return the encoding corresponding to the string from this character to the first space character or U+003B SEMICOLON character (;), or the end of s, whichever comes first.



*Note: T
single
the alg
diverge*

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*'t allow the use of
y this algorithm). While
ed by implementations*

2.7.6 **co** [//whatwg.org/html](#)

A **CORS settings** attribute is an enumerated attribute. The following table lists the keywords and states for the attribute — the keywords in the left column map to the states in the cell in the second column on the same row as the keyword.

Keyword	State	Brief description
anonymous	Anonymous	Cross-origin CORS requests for the element will have the <i>omit credentials flag</i> set.
use-credentials	Use Credentials	Cross-origin CORS requests for the element will not have the <i>omit credentials flag</i> set.

The empty string is also a valid keyword, and maps to the Anonymous state. The attribute's *invalid value default* is the Anonymous state. The *missing value default*, used when the attribute is omitted, is the **No CORS** state.

2.7.7 CORS-enabled fetch

When the user agent is required to perform a **potentially CORS-enabled fetch** of an absolute URL *URL* with a mode *mode* that is either "No CORS", "Anonymous", or "Use Credentials", optionally using areferrer source *referrer source*, with an origin *origin*, and with a default origin behaviour *default* which is either "taint" or "fail", it must run the first applicable set of steps from the following list. The default origin behaviour is only used if *mode* is "No CORS". This algorithm wraps the fetch algorithm above, and labels the obtained resource as either **CORS-same-origin** or **CORS-cross-origin**, or blocks the resource entirely.

- ↪ If the *URL* has the same origin as *origin*
- ↪ If the *URL* is a `data:` URL
- ↪ If the *URL* is a `javascript:` URL
- ↪ If the *URL* is `about:blank`

Run these substeps:

1. Fetch *URL*, using *referrer source* if one was specified, with the *manual redirect flag* set.
2. Loop: Wait for the fetch algorithm to know if the result is a redirect or not.
3. Follow the first appropriate steps from the following list:

- ↪ If the result of the fetch is a redirect, and the origin of the target URL of the redirect is not the same origin as *origin*

 Set *URL* to the the target URL of the redirect and return to the top of the potentially CORS-enabled fetch algorithm (this time, one of the other branches below might be taken, based on the value of *mode*).

- ↪ If the result of the fetch is a redirect

Note: The origin of the target URL of the redirect is the same origin as origin.

 Transparently follow the redirect and jump to the step labeled *loop* above.

- ↪ Otherwise

Note: The resource is available, it is not a redirect, and its origin is the same origin as origin.

 The tasks from the fetch algorithm are queued normally, and for the purposes of the calling

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Fetch *URL*, using *referrer source* if one was specified.

The tasks from the fetch algorithm are queued normally, but for the purposes of the calling algorithm, the obtained resource is CORS-cross-origin. The user agent may report a cross-origin resource access failure to the user (e.g. in a debugging console).

↳ If *mode* is "No CORS"

Note: The URL does not have the same origin as origin, and default is fail.

Discard any data fetched as part of this algorithm, and prevent any tasks from being queued by the fetch algorithm from occurring as a fatal network error and to the user (e.g. in a debugging console).

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If performing a cross-origin request with the request *URL* set to *URL*, with the CORS *referrer source* set to *referrer source* if one was specified, the *source origin* set to *origin*, and with the *omit credentials* flag set if *mode* is "Anonymous" and not set otherwise. [CORS]

2. Wait for the CORS cross-origin request status to have a value.

3. Jump to the appropriate step from the following list:

↳ If the CORS cross-origin request status is not success

Discard all fetched data and prevent any tasks from the fetch algorithm from being queued. For the purposes of the calling algorithm, the user agent must act as if there was a fatal network error and no resource was obtained. If a CORS resource sharing check failed, the user agent may report a cross-origin resource access failure to the user (e.g. in a debugging console).

↳ If the CORS cross-origin request status is success

The tasks from the fetch algorithm are queued normally, and for the purposes of the calling algorithm, the obtained resource is CORS-same-origin.

2.8 Common DOM interfaces

2.8.1 Reflecting content attributes in IDL attributes

Some IDL attributes are defined to reflect a particular content attribute. This means that on getting, the IDL attribute returns the current value of the content attribute, and on setting, the IDL attribute changes the value of the content attribute to the given value.

In general, on getting, if the content attribute is not present, the IDL attribute must act as if the content attribute's value is the empty string; and on setting, if the content attribute is not present, it must first be added.

If a reflecting IDL attribute is a `DOMString` attribute whose content attribute is defined to contain a URL, then on getting, the IDL attribute must resolve the value of the content attribute relative to the element and return the resulting absolute URL if that was successful, or the empty string otherwise; and on setting, must set the content attribute to the specified literal value. If the content attribute is absent, the IDL attribute must return the default value, if the content attribute has one, or else the empty string.

If a reflecting IDL attribute is a `DOMString` attribute whose content attribute is defined to contain one or more URLs, then on getting, the IDL attribute must split the content attribute on spaces and return the concatenation of resolving each token URL to an absolute URL relative to the element, with a single U+0020 SPACE character between each URL, ignoring any tokens that did not resolve successfully. If the content attribute is absent, the IDL attribute must return the default value, if the content attribute has one, or else the empty string. On setting, the IDL attribute must set the content attribute to the specified literal value.

If a reflecting IDL attribute is a `DOMString` attribute whose content attribute is an enumerated attribute, and the IDL attribute is limited to only known values, then, on getting, the IDL attribute must return the conforming value associated with the state the attribute is in (in its canonical case), if any, or the empty string if the attribute is in a state that has no associated keyword value or if the attribute is not in a state (e.g. the attribute is missing and there is no *missing value default*); and on setting, the content attribute must be set to the specified new value.

If a reflecting IDL attribute is a `DOMString` attribute but doesn't fall into any of the above categories, then the getting and setting must

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

If a reflecting IDL attribute has a signed integer type (`long`) then, on getting, the content attribute must be parsed according to the rules for parsing signed integers, and if that is successful, and the value is in the range of the IDL attribute's type, the resulting value must be returned. If, on the other hand, it fails or returns an out of range value, or if the attribute is absent, then the default value must be returned instead, or 0 if there is no default value. On setting, the given value must be converted to the shortest possible string representing the number as a valid integer and then that string must be used as the new content attribute value.

If a reflecting IDL attribute has a signed integer type (`long`) that is **limited to only non-negative numbers** then, on getting, the content attribute must be parsed according to the rules for parsing non-negative integers, and if ~~it is successful, and the value is in the range of the IDL attribute's type, the resulting value must be returned~~, or if the attribute is absent, then the default value must be returned instead, or 0 if there is no default value. On setting, if the value is negative, the user agent must throw an `IndexSizeError` exception. Otherwise, the given value must be converted to the shortest possible string representing the number as a valid non-negative integer and that string must be used as the new content attribute value.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If a reflecting IDL attribute has a signed integer type (`long`) that is **inclusive**, then the resulting value must be returned instead, or 0 if there is no default value. On setting, if the value is in the range 0 to 2147483647, then let n be the new value, otherwise let n be the default value, or 0 if there is no default value; then, n must be converted to the shortest possible string representing the number as a valid non-negative integer and that string must be used as the new content attribute value.

If a reflecting IDL attribute has an unsigned integer type (`unsigned long`) that is **inclusive**, then the resulting value must be returned instead, or 0 if there is no default value. On setting, if the value is in the range 0 to 2147483647, then let n be the new value, otherwise let n be the default value, or 0 if there is no default value; then, n must be converted to the shortest possible string representing the number as a valid non-negative integer and that string must be used as the new content attribute value.

If a reflecting IDL attribute has an unsigned integer type (`unsigned long`) that is **limited to only non-negative numbers greater than zero**, then the behavior is similar to the previous case, but zero is not allowed. On getting, the content attribute must first be parsed according to the rules for parsing non-negative integers, and if that is successful, and the value is in the range 1 to 2147483647 inclusive, the resulting value must be returned. If, on the other hand, it fails or returns an out of range value, or if the attribute is absent, the default value must be returned instead, or 1 if there is no default value. On setting, if the value is zero, the user agent must throw an `IndexSizeError` exception. Otherwise, first, if the new value is in the range 1 to 2147483647, then let n be the new value, otherwise let n be the default value, or 1 if there is no default value; then, n must be converted to the shortest possible string representing the number as a valid non-negative integer and that string must be used as the new content attribute value.

If a reflecting IDL attribute has a floating-point number type (`double` or `unrestricted double`), then, on getting, the content attribute must be parsed according to the rules for parsing floating-point number values, and if that is successful, the resulting value must be returned. If, on the other hand, it fails, or if the attribute is absent, the default value must be returned instead, or 0.0 if there is no default value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then that string must be used as the new content attribute value.

If a reflecting IDL attribute has a floating-point number type (`double` or `unrestricted double`) that is **limited to numbers greater than zero**, then the behavior is similar to the previous case, but zero and negative values are not allowed. On getting, the content attribute must be parsed according to the rules for parsing floating-point number values, and if that is successful and the value is greater than 0.0, the resulting value must be returned. If, on the other hand, it fails or returns an out of range value, or if the attribute is absent, the default value must be returned instead, or 0.0 if there is no default value. On setting, if the value is less than or equal to zero, then the value must be ignored. Otherwise, the given value must be converted to the best representation of the number as a floating-point number and then that string must be used as the new content attribute value.

Note: The values `Infinity` and `Not-a-Number (NaN)` values throw an exception on setting, as defined in the Web IDL specification. [WEBIDL]

If a reflecting IDL attribute has the type `DOMTokenList` or `DOMSettableTokenList`, then on getting it must return a `DOMTokenList` or `DOMSettableTokenList` object (as appropriate) whose underlying string is the element's corresponding content attribute. When the object mutates its underlying string, the content attribute must itself be immediately mutated. When the attribute is absent, then the string represented by the object is the empty string; when the object mutates this empty string, the user agent must add the corresponding content attribute, with its value set to the value it would have been set to after mutating the empty string. The same `DOMTokenList` or `DOMSettableTokenList` object must be returned every time for each attribute.

If an element with no attributes has its `element.classList.remove()` method invoked, the underlying string won't be changed, since the result of removing any token from the empty string is still the empty string. However, if the `element.classList.add()` method is then invoked, a `class` attribute will be added to the element with the value of the token to be added.

If a reflecting IDL attribute has the type `HTMLElement`, or an interface that descends from `HTMLElement`, then, on getting, it must run the following algorithm (stopping at the first point where a value is returned):

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

3. If `candidate` is null, or if it is not type-compatible with the IDL attribute, then the IDL attribute must return null.

4. Otherwise, it must return `candidate`.

On setting, if the given element has an `id` attribute, and has the same home subtree as the element of the attribute being set, and the given element is the first element in that home subtree whose ID is the value of that `id` attribute, then the content attribute must be set to the value of that `id` attribute. Otherwise, the content attribute must be set to the empty string.

2.8.2 Collections



The `HTMLAllCollection` interface is a multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

PropertiesCollection

2.8.2.1 `HTMLAllCollection`

The `HTMLAllCollection` interface is a multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

can be used as a synonym for its `namedItem()` method. It is intended only for the legacy `document.all` attribute.

```
interface HTMLAllCollection : HTMLCollection {
    // inherits length and item(unsigned long index)
    object? item(DOMString name);
    legacycaller getter object? namedItem(DOMString name); // overrides inherited namedItem()
    HTMLAllCollection tags(DOMString tagName);
};
```

This box is non-normative. Implementation requirements are given below this box.

`collection.length`

Returns the number of elements in the collection.

`element = collection.item(index)`

`collection[index]`

`collection(index)`

Returns the item with index `index` from the collection. The items are sorted in tree order.

`element = collection.item(name)`

`collection = collection.item(name)`

`element = collection.namedItem(name)`

`collection = collection.namedItem(name)`

`collection[name]`

`collection(name)`

Returns the item with ID or name `name` from the collection.

If there are multiple matching items, then an `HTMLCollection` object containing all those elements is returned.

Only `a`, `applet`, `area`, `embed`, `form`, `frame`, `frameset`, `iframe`, `img`, and `object` elements can have a name for the purpose of this method; their name is given by the value of their `name` attribute.

`collection = collection.tags(tagName)`

Returns a collection that is a filtered view of the current collection, containing only elements with the given tag name.

The object's supported property indices and supported property names are as defined for `HTMLCollection` objects.

The `item(name)` and `namedItem(name)` methods must act according to the following algorithm:

1. Let `collection` be an `HTMLCollection` object rooted at the same node as the `HTMLAllCollection` object on which the method was invoked, whose filter matches only elements that already match the filter of the `HTMLAllCollection` object on which the method was invoked and that are either:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

2. If, at the time the method is called, there is exactly one node in *collection*, then return that node and stop the algorithm.
3. Otherwise, if, at the time the method is called, *collection* is empty, return null and stop the algorithm.
4. Otherwise, return *collection*.

The `tags(tagName)` method must return an `HTMLAllCollection` rooted at the same node as the `HTMLAllCollection` object on which the method was invoked, whose filter matches only HTML elements whose local name is the `tagName` argument and that already match the filter of the `HTMLAllCollection` object on which the method was invoked. In HTML documents, the argument must first be converted to ASCII lowercase.



2.8.2.2 HT Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

The `HTMLFC`

eldset elements.

```
int
// le
};

interface RadioNodeList : NodeList {
    attribute DOMString value;
};
```

erited namedItem()

This box is non-normative. Implementation requirements are given below this box.

`collection.length`

Returns the number of elements in the collection.

`element = collection.item(index)`

`collection[index]`

`collection(index)`

Returns the item with index `index` from the collection. The items are sorted in tree order.

`element = collection.namedItem(name)`

`radioNodeList = collection.namedItem(name)`

`collection[name]`

`collection(name)`

Returns the item with ID or `name` from the collection.

If there are multiple matching items, then a `RadioNodeList` object containing all those elements is returned.

`radioNodeList.value [= value]`

Returns the value of the first checked radio button represented by the object.

Can be set, to check the first radio button with the given value represented by the object.

The object's supported property indices are as defined for `HTMLCollection` objects.

The supported property names consist of the values of all the `id` and `name` attributes of all the elements represented by the collection.

The `namedItem(name)` method must act according to the following algorithm:

1. If, at the time the method is called, there is exactly one node in the collection that has either an `id` attribute or a `name` attribute equal to `name`, then return that node and stop the algorithm.
2. Otherwise, if there are no nodes in the collection that have either an `id` attribute or a `name` attribute equal to `name`, then return null and stop the algorithm.
3. Otherwise, create a new `RadioNodeList` object representing a live view of the `HTMLFormControlsCollection` object, further filtered so that the only nodes in the `RadioNodeList` object are those that have either an `id` attribute or a `name`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Members of the `RadioNodeList` interface inherit from the `NodeList` interface must behave as they would on a `NodeList` object.

The `value` IDL attribute on the `RadioNodeList` object, on getting, must return the value returned by running the following steps:

1. Let `element` be the first element in tree order represented by the `RadioNodeList` object that is an `input` element whose `type` attribute is in the Radio Button state and whose checkedness is true. Otherwise, let it be null.
2. If `element` is null, or if it is an element with no `value` attribute, return the empty string.
3. Otherwise, return the value of `element`'s `value` attribute.



On setting, t

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↑ input element whose
↑ to the new value, if any.

2.8.2.3 HTMLOptionsCollection

The `HTMLOptionsCollection` interface is used for collections of `option` elements. It is always rooted on a `select` element and has attributes and methods that manipulate that element's descendants.

```
interface HTMLOptionsCollection : HTMLCollection {
    // inherits item()
    attribute unsigned long length; // overrides inherited length
    legacycaller getter object? namedItem(DOMString name); // overrides inherited namedItem()
    setter creator void (unsigned long index, HTMLOptionElement? option);
    void add((HTMLOptionElement or HTMLOptGroupElement) element, optional (HTMLElement or
    long) before = null);
    void remove(long index);
    attribute long selectedIndex;
};
```

This box is non-normative. Implementation requirements are given below this box.

`collection.length [= value]`

Returns the number of elements in the collection.

When set to a smaller number, truncates the number of `option` elements in the corresponding container.

When set to a greater number, adds new blank `option` elements to that container.

`element = collection.item(index)`

`collection[index]`

`collection(index)`

Returns the item with index `index` from the collection. The items are sorted in tree order.

`element = collection.namedItem(name)`

`nodeList = collection.namedItem(name)`

`collection[name]`

`collection(name)`

Returns the item with ID or name `name` from the collection.

If there are multiple matching items, then the first is returned.

`collection.add(element [, before])`

Inserts `element` before the node given by `before`.

The `before` argument can be a number, in which case `element` is inserted before the item with that number, or an element from the collection, in which case `element` is inserted before that element.

If `before` is omitted, null, or a number out of range, then `element` will be added at the end of the list.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Returns the index of the first selected item, if any, or -1 if there is no selected item.

Can be set, to change the selection.

The object's supported property indices are as defined for `HTMLCollection` objects.

On getting, the `length` attribute must return the number of nodes represented by the collection.

On setting, the behavior depends on whether the new value is equal to, greater than, or less than the number of nodes represented by the collection.

~~option elements must be inserted at the end of the collection. If the new value is greater than the number of nodes represented by the collection, then the last `length` nodes must be removed from the collection, and the new value must be inserted at the end of the collection. If the new value is lower than the number of nodes represented by the collection, then the last `length` nodes must be removed from the collection, and the new value must be inserted at the beginning of the collection. If the new value is equal to the number of nodes represented by the collection, then the new value must be inserted at the end of the collection.~~

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Note: Some `optgroup` elements must be inserted at the end of the collection. If the new value is greater than the number of nodes represented by the collection, then the last `length` nodes must be removed from the collection, and the new value must be inserted at the end of the collection. If the new value is lower than the number of nodes represented by the collection, then the last `length` nodes must be removed from the collection, and the new value must be inserted at the beginning of the collection. If the new value is equal to the number of nodes represented by the collection, then the new value must be inserted at the end of the collection.

Idren to existing

The supported property names consist of the values of all the `id` and `name` attributes of all the elements represented by the collection.

The `namedItem(name)` method must return the first node in the collection that has either an `id` attribute or a `name` attribute equal to `name`, if there is one; otherwise, it must return null.

When the user agent is to **set the value of a new indexed property** for a given property index `index` to a new value `value`, it must run the following algorithm:

1. If `value` is null, invoke the steps for the `remove` method with `index` as the argument, and abort these steps.
2. Let `length` be the number of nodes represented by the collection.
3. Let `n` be `index` minus `length`.
4. If `n` is greater than zero, then append a `DocumentFragment` consisting of `n-1` new `option` elements with no attributes and no child nodes to the `select` element on which the `HTMLOptionsCollection` is rooted.
5. If `n` is greater than or equal to zero, append `value` to the `select` element. Otherwise, replace the `index`th element in the collection by `value`.

The `add(element, before)` method must act according to the following algorithm:

1. If `element` is an ancestor of the `select` element on which the `HTMLOptionsCollection` is rooted, then throw a `HierarchyRequestError` exception.
2. If `before` is an element, but that element isn't a descendant of the `select` element on which the `HTMLOptionsCollection` is rooted, then throw a `NotFoundError` exception.
3. If `element` and `before` are the same element, then return and abort these steps.
4. If `before` is a node, then let `reference` be that node. Otherwise, if `before` is an integer, and there is a `before`th node in the collection, let `reference` be that node. Otherwise, let `reference` be null.
5. If `reference` is not null, let `parent` be the parent node of `reference`. Otherwise, let `parent` be the `select` element on which the `HTMLOptionsCollection` is rooted.
6. Act as if the DOM `insertBefore()` method was invoked on the `parent` node, with `element` as the first argument and `reference` as the second argument.

The `remove(index)` method must act according to the following algorithm:

1. If the number of nodes represented by the collection is zero, abort these steps.
2. If `index` is not a number greater than or equal to 0 and less than the number of nodes represented by the collection, abort these steps.
3. Let `element` be the `index`th element in the collection.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2.8.2.4 HTMLPropertiesCollection

The `HTMLPropertiesCollection` interface is used for collections of elements that add name-value pairs to a particular item in the microdata model.

```
interface HTMLPropertiesCollection : HTMLCollection {
    // inherits length and item()
    getter PropertyNodeList? namedItem(DOMString name); // override inherited namedItem()
    ...
};

Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html
```

This box is non-normative. Implementation requirements are given below this box.

collection.length

Returns the number of elements in the collection.

element = collection.item(index)

collection[index]

Returns the element with index *index* from the collection. The items are sorted in tree order.

propertyNodeList = collection.namedItem(name)

Returns a `PropertyNodeList` object containing any elements that add a property named *name*.

collection[name]

Returns a `PropertyNodeList` object containing any elements that add a property named *name*. The *name* index has to be one of the values listed in the `names` list.

collection.names

Returns an array with the property names of the elements in the collection.

propertyNodeList.getValues()

Returns an array of the various values that the relevant elements have.

The object's supported property indices are as defined for `HTMLCollection` objects.

The supported property names consist of the property names of all the elements represented by the collection.

The `names` attribute must return a live read only array object giving the property names of all the elements represented by the collection, listed in tree order, but with duplicates removed, leaving only the first occurrence of each name. The same object must be returned each time.

The `namedItem(name)` method must return a `PropertyNodeList` object representing a live view of the `HTMLPropertiesCollection` object, further filtered so that the only nodes in the `PropertyNodeList` object are those that have a property name equal to *name*. The nodes in the `PropertyNodeList` object must be sorted in tree order, and the same object must be returned each time a particular *name* is queried.

Members of the `PropertyNodeList` interface inherited from the `NodeList` interface must behave as they would on a `NodeList` object.

The `getValues` method the `PropertyNodeList` object must return a newly constructed array whose values are the values obtained from the `itemValue` DOM property of each of the elements represented by the object, in tree order.

Go

NOV JAN FEB

03

2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

When a `DOMStringMap` object is instantiated, it is associated with three algorithms, one for getting the list of name-value pairs, one for setting names to certain values, and one for deleting names.

```
interface DOMStringMap {
    getter DOMString (DOMString name);
    setter void (DOMString name, DOMString value);
    creator void (DOMString name, DOMString value);
    deleter void (DOMString name);
};
```



The support getting the li
To determine name-value
To set the va
run, passing

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ned from the algorithm for

value component of the name-value pairs.

; to certain values must be

To **delete an existing named property** `name`, the algorithm for deleting names must be run, passing `name` as the name.

Note: The `DOMStringMap` interface definition here is only intended for JavaScript environments. Other language bindings will need to define how `DOMStringMap` is to be implemented for those languages.

The `dataset` attribute on elements exposes the `data-*` attributes on the element.

Given the following fragment and elements with similar constructions:

```

```

...one could imagine a function `splashDamage()` that takes some arguments, the first of which is the element to process:

```
function splashDamage(node, x, y, damage) {
    if (node.classList.contains('tower') && // checking the 'class' attribute
        node.dataset.x == x && // reading the 'data-x' attribute
        node.dataset.y == y) { // reading the 'data-y' attribute
        var hp = parseInt(node.dataset.hp); // reading the 'data-hp' attribute
        hp = hp - damage;
        if (hp < 0) {
            hp = 0;
            node.dataset.ai = 'dead'; // setting the 'data-ai' attribute
            delete node.dataset.ability; // removing the 'data-ability' attribute
        }
        node.dataset.hp = hp; // setting the 'data-hp' attribute
    }
}
```

2.8.4 DOMElementMap

The `DOMElementMap` interface represents a set of name-element mappings. It exposes these using the scripting language's native mechanisms for property access.

When a `DOMElementMap` object is instantiated, it is associated with three algorithms, one for getting the list of name-element mappings, one for mapping a name to a certain element, and one for deleting mappings by name.

```
interface DOMElementMap {
    getter Element (DOMString name);
    setter void (DOMString name, Element value);
    creator void (DOMString name, Element value);
    deleter void (DOMString name);
};
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

mappings.

To set the value of a **new** or **existing** named property *name* to value *value*, the algorithm for mapping a name to a certain element must be run, passing *name* as the name *value* as the element.

To **delete an existing named property** *name*, the algorithm for deleting mappings must be run, passing *name* as the name component of the mapping to be deleted.

Note: The DOMElementMap interface definition here is only intended for JavaScript environments. Other language bindings

Too slow? Try reading the multipage copy of
2.8.5 Transferable objects instead: <https://web.archive.org>

Some objects transfer ownership to another object. This is used in particular to

//whatwg.org/html

```
[NoI1]
interface Transferable { };
```

To **transfer a Transferable object** to a new owner, the user agent must run the steps defined for the type of object in question. The steps will return a new object of the same type, and will permanently **neuter** the original object. (This is an irreversible and non-idempotent operation; once an object has been transferred, it cannot be transferred, or indeed used, again.)

The following Transferable types exist:

- MessagePort
- ArrayBuffer [TYPEDARRAY]
- CanvasProxy

2.8.6 Safe passing of structured data

When a user agent is required to obtain a **structured clone** of a value, optionally with a *transfer map*, it must run the following algorithm, which either returns a separate value, or throws an exception. If a *transfer map* is provided, it consists of an association list of Transferable objects to placeholder objects.

1. Let *input* be the value being cloned.
2. Let *transfer map* be the *transfer map* passed to the algorithm, if any, or the empty list otherwise.
3. Let *memory* be an association list of pairs of objects, initially empty. This is used to handle duplicate references. In each pair of objects, one is called the *source* object and the other the *destination* object.
4. For each mapping in *transfer map*, add a mapping from the Transferable object (the source object) to the placeholder object (the destination object) to *memory*.
5. Let *output* be the value resulting from calling the internal structured cloning algorithm with *input* as the "input" argument, and *memory* as the "memory" argument.
6. Return *output*.

The **internal structured cloning algorithm** is always called with two arguments, *input* and *memory*, and its behavior is as follows:

1. If *input* is the source object of a pair of objects in *memory*, then return the destination object in that pair of objects and abort these steps.
2. If *input* is a primitive value, then return that value and abort these steps.
3. The *input* value is an object. Jump to the appropriate step below:

↳ If *input* is a Boolean object

Let *output* be a newly constructed Boolean object with the same value as *input*.

↳ If *input* is a Number object

Let *output* be a newly constructed Number object with the same value as *input*.

Go NOV JAN FEB
 03 < >
 2012 2013 2014 ▾ About this capture

[3,445 captures](#)
 5 Jun 2004 - 22 Aug 2020

Let `output` be a newly constructed `DataObject` with the same value as `input`.

↳ If `input` is a `RegExp` object

Let `output` be a newly constructed `RegExp` object with the same pattern and flags as `input`.

Note: The value of the `lastIndex` property is not copied.

↳ If `input` is a `File` object

Let `output` be a newly constructed `File` object corresponding to the same underlying data.

↳ Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ /web/20130103030411/http://whatwg.org/html

selected `File` objects in order.

↳

Let `output` be a newly constructed `ImageData` object whose `width` and `height` attributes have values equal to the corresponding attributes on `input`, and whose `data` attribute has the value obtained from invoking the internal structured cloning algorithm recursively with the value of the `data` attribute on `input` as the new "input" argument and `memory` as the new "memory" argument.

↳ If `input` is an `ImageBitmap` object

Let `output` be a newly constructed `ImageBitmap` object whose bitmap data is a copy of `input`'s bitmap data.

↳ If `input` is an `Array` object

Let `output` be a newly constructed empty `Array` object whose `length` is equal to the `length` of `input`.

Note: This means that the length of sparse arrays is preserved.

↳ If `input` is an `Object` object

Let `output` be a newly constructed empty `Object` object.

↳ If `input` is an object that another specification defines how to clone

Let `output` be a clone of the object as defined by the other specification.

↳ If `input` is another native object type (e.g. `Error`, `Function`)

↳ If `input` is a host object (e.g. a DOM node)

Throw a `DataCloneError` exception and abort the overall structured clone algorithm.

4. Add a mapping from `input` (the source object) to `output` (the destination object) to `memory`.

5. If `input` is an `Array` object or an `Object` object, then, for each enumerable property in `input`, add a new property to `output` having the same name, and having a value created from invoking the internal structured cloning algorithm recursively with the value of the property as the "input" argument and `memory` as the "memory" argument. The order of the properties in the `input` and `output` objects must be the same, and any properties that involve running script must be processed in that same order. If obtaining the value of the property involved executing script, and that script threw an uncaught exception, then abort the overall structured clone algorithm, with that exception being passed through to the caller.

Note: This does not walk the prototype chain.

Note: Property descriptors, setters, getters, and analogous features are not copied in this process. For example, the property in the `input` could be marked as read-only, but in the `output` it would just have the default state (typically read-write, though that could depend on the scripting environment).

Note: Properties of `Array` objects are not treated any differently than those of other Objects. In particular, this means that non-index properties of arrays are copied as well.

6. Return `output`.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

There is an **implied strong reference** from any IDL attribute that returns a pre-existing object to that object.

For example, the `document.location` attribute means that there is a strong reference from a `Document` object to its `Location` object. Similarly, there is always a strong reference from a `Document` to any descendant nodes, and from any node to its owner `Document`.

2.9 Namespaces



The `HTML` r Too slow? Try reading the multipage copy of

The `MathML` the spec instead: [https://web.archive.org/](https://web.archive.org/web/20130103030411/http://www.whatwg.org/)

The `SVG` na /[web/20130103030411/http:](https://web.archive.org/web/20130103030411/http://www.whatwg.org/)

The `XLink` r //whatwg.org/html

The `XML` na.

The `XMLNS namespace` is: <http://www.w3.org/2000/xmlns/>

Data mining tools and other user agents that perform operations on content without running scripts, evaluating CSS or XPath expressions, or otherwise exposing the resulting DOM to arbitrary content, may "support namespaces" by just asserting that their DOM node analogues are in certain namespaces, without actually exposing the above strings.

Note: In the HTML syntax, namespace prefixes and namespace declarations do not have the same effect as in XML.
For instance, the colon has no special meaning in HTML element names.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Every XML and HTML document in an HTML UA is represented by a Document object. [DOM]

The **document's address** is an absolute URL that is initially set when the `Document` is created but that can change during the lifetime of the `Document`, for example when the user navigates to a fragment identifier on the page or when `pushState()` method is called with a new URL.

⚠Warning! *Interactive user agents typically expose the document's address in their user interface. This is the primary mechanism by which a user can tell if a site is attempting to impersonate another.*

When a `Document` is created by a script using the `createDocument()` or `createHTMLDocument()` APIs, the document's address is the same ↗

loaded immediately. **Too slow? Try reading the multipage copy of**

Each `Document` document. **the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Unicode strikethrough. **/web/20130103030411/http://whatwg.org/html**

When the user uses <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. Let `s` ↗

2. Navigate the browsing context to a resource whose source is `source`, with replacement enabled. When the navigate algorithm creates a `Document` object for this purpose, set that `Document`'s reload override flag and set its reload override buffer to `source`.

3.1.1 The Document object

The DOM specification defines a `Document` interface, which this specification extends significantly:

```
[OverrideBuiltins]
partial /*sealed*/ interface Document {
    // resource metadata management
    [PutForwards:href, Unforgeable] readonly attribute Location? location;
        attribute DOMString domain;
    readonly attribute DOMString referrer;
        attribute DOMString cookie;
    readonly attribute DOMString lastModified;
    readonly attribute DOMString readyState;

    // DOM tree accessors
    getter object (DOMString name);
        attribute DOMString title;
        attribute DOMString dir;
        attribute HTMLElement? body;
    readonly attribute HTMLHeadElement? head;
    readonly attribute HTMLCollection images;
    readonly attribute HTMLCollection embeds;
    readonly attribute HTMLCollection plugins;
    readonly attribute HTMLCollection links;
    readonly attribute HTMLCollection forms;
    readonly attribute HTMLCollection scripts;
    NodeList getElementsByName(DOMString elementName);
    NodeList getItems(optional DOMString typeNames); // microdata
    readonly attribute DOMElementMap cssElementMap;
    readonly attribute HTMLScriptElement? currentScript;

    // dynamic markup insertion
    Document open(optional DOMString type, optional DOMString replace);
    WindowProxy open(DOMString url, DOMString name, DOMString features, optional boolean
replace);
    void close();
    void write(DOMString... text);
    void writeln(DOMString... text);
}
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
        attribute DOMString designMode;
        boolean execCommand(DOMString commandId);
        boolean execCommand(DOMString commandId, boolean showUI);
        boolean execCommand(DOMString commandId, boolean showUI, DOMString value);
        boolean queryCommandEnabled(DOMString commandId);
        boolean queryCommandIndeterm(DOMString commandId);
        boolean queryCommandState(DOMString commandId);
        boolean queryCommandSupported(DOMString commandId);
        DOMString queryCommandValue(DOMString commandId);

re   Too slow? Try reading the multipage copy of
//   the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

        attribute EventHandler onclick;
        attribute EventHandler onclose;
        attribute EventHandler oncontextmenu;
        attribute EventHandler oncuechange;
        attribute EventHandler ondblclick;
        attribute EventHandler ondrag;
        attribute EventHandler ondragend;
        attribute EventHandler ondragenter;
        attribute EventHandler ondragleave;
        attribute EventHandler ondragover;
        attribute EventHandler ondragstart;
        attribute EventHandler ondrop;
        attribute EventHandler ondurationchange;
        attribute EventHandler onemptied;
        attribute EventHandler onended;
        attribute OnErrorEventHandler onerror;
        attribute EventHandler onfocus;
        attribute EventHandler oninput;
        attribute EventHandler oninvalid;
        attribute EventHandler onkeydown;
        attribute EventHandler onkeypress;
        attribute EventHandler onkeyup;
        attribute EventHandler onload;
        attribute EventHandler onloadeddata;
        attribute EventHandler onloadedmetadata;
        attribute EventHandler onloadstart;
        attribute EventHandler onmousedown;
        attribute EventHandler onmousemove;
        attribute EventHandler onmouseout;
        attribute EventHandler onmouseover;
        attribute EventHandler onmouseup;
        attribute EventHandler onmousewheel;
        attribute EventHandler onpause;
        attribute EventHandler onplay;
        attribute EventHandler onplaying;
        attribute EventHandler onprogress;
        attribute EventHandler onratechange;
        attribute EventHandler onreset;
        attribute EventHandler onscroll;
        attribute EventHandler onseeked;
        attribute EventHandler onseeking;
        attribute EventHandler onselect;
        attribute EventHandler onshow;
        attribute EventHandler onsort;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
attribute EventHandler onwaiting;

// special event handler IDL attributes that only apply to Document objects
[LenientThis] attribute EventHandler onreadystatechange;
};
```

3.1.2 Security

User agents
effective scr

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org>

accessed by scripts whose

3.1.3 Res

/web/20130103030411/http:
//whatwg.org/html
docume

ts are given below this box.

Returns the address of the `Document` from which the user navigated to this one, unless it was blocked or there was no such document, in which case it returns the empty string.

The `noreferrer` link type can be used to block the referrer.

The `referrer` attribute must return either the address of the active document of the source browsing context *at the time the navigation was started* (that is, the page which navigated the browsing context to the current document), with any `<fragment>` component removed; or the empty string if there is no such originating page, or if the UA has been configured not to report referrers in this case, or if the navigation was initiated for a hyperlink with a `noreferrer` keyword.

Note: In the case of HTTP, the `referrer` IDL attribute will match the `Referer` (sic) header that was sent when fetching the current page.

Note: Typically user agents are configured to not report referrers in the case where the referrer uses an encrypted protocol and the current page does not (e.g. when navigating from an `https`: page to an `http`: page).

This box is non-normative. Implementation requirements are given below this box.

`document.cookie [= value]`

Returns the HTTP cookies that apply to the `Document`. If there are no cookies or cookies can't be applied to this resource, the empty string will be returned.

Can be set, to add a new cookie to the element's set of HTTP cookies.

If the contents are sandboxed into a unique origin (e.g. in an `iframe` with the `sandbox` attribute), a `SecurityError` exception will be thrown on getting and setting.

The `cookie` attribute represents the cookies of the resource identified by the document's address.

A `Document` object that falls into one of the following conditions is a **cookie-averse Document object**:

- A `Document` that has no browsing context.
- A `Document` whose address does not use a server-based naming authority.

On getting, if the document is a cookie-averse `Document` object, then the user agent must return the empty string. Otherwise, if the `Document`'s origin is not a scheme/host/port tuple, the user agent must throw a `SecurityError` exception. Otherwise, the user agent must first obtain the storage mutex and then return the cookie-string for the document's address for a "non-HTTP" API, decoded as UTF-8, with error handling. [COOKIES]

On setting, if the document is a cookie-averse `Document` object, then the user agent must do nothing. Otherwise, if the `Document`'s origin is not a scheme/host/port tuple, the user agent must throw a `SecurityError` exception. Otherwise, the user agent must obtain

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

help manage which cookies are sent to which parts of the site, and are not in any way a security feature.

This box is non-normative. Implementation requirements are given below this box.

`document.lastModified`

Returns the date of the last modification to the document, as reported by the server, in the form "MM/DD/YYYY hh:mm:ss", in the user's local time zone.

If the last modification date is not known, the current time is returned instead.



The `lastModified` attribute returns the date of the last modification to the document, as reported by the server, in the user's local time zone. It is defined as follows:

1. The value of the `Last-Modified` header from the document's source file.
2. A URL consisting of the document's base URL followed by `/whatwg.org/html`.
3. The character `/`.
4. A U+002F SOLIDUS character (/).
5. The year component of the date.
6. A U+0020 SPACE character.
7. The hours component of the time.
8. A U+003A COLON character (:).
9. The minutes component of the time.
10. A U+003A COLON character (:).
11. The seconds component of the time.

All the numeric components above, other than the year, must be given as two ASCII digits representing the number in base ten, zero-padded if necessary. The year must be given as the shortest possible string of four or more ASCII digits representing the number in base ten, zero-padded if necessary.

The `Document`'s source file's last modification date and time must be derived from relevant features of the networking protocols used, e.g. from the value of the `HTTP Last-Modified` header of the document, or from metadata in the file system for local files. If the last modification date and time are not known, the attribute must return the current date and time in the above format.

This box is non-normative. Implementation requirements are given below this box.

`document.readyState`

Returns "loading" while the `Document` is loading, "interactive" once it is finished parsing but still loading sub-resources, and "complete" once it has loaded.

The `readystatechange` event fires on the `Document` object when this value changes.

Each document has a **current document readiness**. When a `Document` object is created, it must have its current document readiness set to the string "loading" if the document is associated with an HTML parser, an XML parser, or an XSLT processor, and to the string "complete" otherwise. Various algorithms during page loading affect this value. When the value is set, the user agent must fire a simple event named `readystatechange` at the `Document` object.

A `Document` is said to have an **active parser** if it is associated with an HTML parser or an XML parser that has not yet been stopped or aborted.

The `readyState` IDL attribute must, on getting, return the current document readiness.

3.1.4 DOM tree accessors

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Returns the `head` element.

The `head` element of a document is the first `head` element that is a child of the `html` element, if there is one, or null otherwise.

The `head` attribute, on getting, must return the `head` element of the document (a `head` element or null).

This box is non-normative. Implementation requirements are given below this box.

document.head Too slow? Try reading the multipage copy of
 Ref the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `title` attribute must, on getting, return the value that would have been returned by the `IDL` attribute of the same name on the `Document` interface, or null otherwise.

The `title` attribute must, on getting, run the following algorithm:

1. If the root element is an `svg` element in the "`http://www.w3.org/2000/svg`" namespace, and the user agent supports SVG, then return the value that would have been returned by the `IDL` attribute of the same name on the `SVGDocument` interface. [SVG]
2. Otherwise, let `value` be a concatenation of the data of all the child `Text` nodes of the `title` element, in tree order, or the empty string if the `title` element is null.
3. Replace any sequence of one or more consecutive space characters in `value` with a single U+0020 SPACE character.
4. Strip leading and trailing whitespace from `value`.
5. Return `value`.

On setting, the following algorithm must be run. Mutation events must be fired as appropriate.

1. If the root element is an `svg` element in the "`http://www.w3.org/2000/svg`" namespace, and the user agent supports SVG, then the setter must act as if it was the setter for the `IDL` attribute of the same name on the `Document` interface defined by the SVG specification. Stop the algorithm here. [SVG]
2. If the `title` element is null and the `head` element is null, then the attribute must do nothing. Stop the algorithm here.
3. If the `title` element is null, then a new `title` element must be created and appended to the `head` element. Let `element` be that element. Otherwise, let `element` be the `title` element.
4. The children of `element` (if any) must all be removed.
5. A single `Text` node whose data is the new value being assigned must be appended to `element`.

The `title` `IDL` attribute defined above must replace the attribute of the same name on the `Document` interface defined by the SVG specification when the user agent supports both HTML and SVG. [SVG]

This box is non-normative. Implementation requirements are given below this box.

document.body [= value]

Returns the `body` element.

Can be set, to replace the `body` element.

If the new value is not a `body` or `frameset` element, this will throw a `HierarchyRequestError` exception.

The `body` element of a document is the first child of the `html` element that is either a `body` element or a `frameset` element. If there is no such element, it is null.

The `body` attribute, on getting, must return the `body` element of the document (either a `body` element, a `frameset` element, or null).

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2. Otherwise, if the new value is the same as the body element, do nothing. Abort these steps.
3. Otherwise, if the body element is not null, then replace that element with the new value in the DOM, as if the root element's `replaceChild()` method had been called with the new value and the incumbent body element as its two arguments respectively, then abort these steps.
4. Otherwise, the body element is null. Append the new value to the root element.

This box is non-normative. Implementation requirements are given below this box.

`document`

Ref

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`document`

Ref

/web/20130103030411/http://whatwg.org/html

`document`

Ref

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

On getting, the `cssElementMap` IDL attribute must return a `DOMElementMap` object, associated with the following algorithms, which expose the current mappings:

The algorithm for getting the list of name-element mappings

Return the `Document`'s CSS ID overrides list, maintaining the order in which the entries were originally added to the list.

The algorithm for mapping a name to a certain element

Let `name` be the name passed to the algorithm and `element` be the `Element` passed to the algorithm.

```
If eler
  Oth Too slow? Try reading the multipage copy of
  eler the spec instead: https://web.archive.org
  Oth /web/20130103030411/http://whatwg.org/html
The algorit If th also ↵
```

ace its current value with
use element is `element`.
s algorithm, remove it. This

The same object must be returned each time.

This box is non-normative. Implementation requirements are given below this box.

`document.currentScript`

Returns the `script` element that is currently executing. In the case of reentrant `script` execution, returns the one that most recently started executing amongst those that have not yet finished executing.

Returns null if the `Document` is not currently executing a `script` element (e.g. because the running script is an event handler, or a timeout).

The `currentScript` attribute, on getting, must return the value to which it was most recently initialized. When the `Document` is created, the `currentScript` must be initialized to null.

The `Document` interface supports named properties. The supported property names at any moment consist of the values of the `name` content attributes of all the `applet`, `exposed embed`, `form`, `iframe`, `img`, and `exposed object` elements in the `Document` that have `name` content attributes, and the values of the `id` content attributes of all the `applet` and `exposed object` elements in the `Document` that have `id` content attributes, and the values of the `id` content attributes of all the `img` elements in the `Document` that have both `name` content attributes and `id` content attributes.

To determine the value of a named property `name` when the `Document` object is indexed for property retrieval, the user agent must return the value obtained using the following steps:

1. Let `elements` be the list of named elements with the name `name` in the `Document`.

Note: There will be at least one such element, by definition.

2. If `elements` has only one element, and that element is an `iframe` element, then return the `WindowProxy` object of the nested browsing context represented by that `iframe` element, and abort these steps.
3. Otherwise, if `elements` has only one element, return that element and abort these steps.
4. Otherwise return an `HTMLCollection` rooted at the `Document` node, whose filter matches only named elements with the name `name`.

Named elements with the name `name`, for the purposes of the above algorithm, are those that are either:

- `applet`, `exposed embed`, `form`, `iframe`, `img`, or `exposed object` elements that have a `name` content attribute whose value is `name`, or
- `applet` or `exposed object` elements that have an `id` content attribute whose value is `name`, or
- `img` elements that have an `id` content attribute whose value is `name`, and that have a `name` content attribute present also.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3.1.5 Loading XML documents

```
partial interface XMLHttpRequest {
    boolean load(DOMString url);
};
```

The `load(u)`

1. Let `url` be the value of `u`. If `url` is `null`, throw a `SyntaxError`.
2. Resolve `url` to a `Document` object. If the resolution fails, abort these steps.
3. If the `Document` object has an `XMLHttpRequest` interface, then run the steps below. Otherwise, if the `Document` object has an `XMLHttpRequest` attribute, then set the `XMLHttpRequest` attribute to `null` and abort these steps.
4. Remove the `XMLHttpRequest` attribute from the `Document` object.
5. Set the current document readiness of `document` to "loading".
6. Run the remainder of these steps asynchronously, and return true from the method.
7. Let `result` be a `Document` object.
8. Let `success` be false.
9. Fetch `url` from the origin of `document`, using the entry script's referrer source, with the `synchronous` flag set and the `force same-origin` flag set.
10. If the fetch attempt was successful, and the resource's Content-Type metadata is an XML MIME type, then run these substeps:
 1. Create a new XML parser associated with the `result` document.
 2. Pass this parser the fetched document.
 3. If there is an XML well-formedness or XML namespace well-formedness error, then remove all child nodes from `result`. Otherwise let `success` be true.
11. Queue a task to run the following steps.
 1. Set the current document readiness of `document` to "complete".
 2. Replace all the children of `document` by the children of `result` (even if it has no children), firing mutation events as if a `DocumentFragment` containing the new children had been inserted.
 3. Fire a simple event named `load` at `document`.

3.2 Elements

3.2.1 Semantics

Elements, attributes, and attribute values in HTML are defined (by this specification) to have certain meanings (semantics). For example, the `ol` element represents an ordered list, and the `lang` attribute represents the language of the content.

These definitions allow HTML processors, such as Web browsers or search engines, to present and use documents and applications in a wide variety of contexts that the author might not have considered.

As a simple example, consider a Web page written by an author who only considered desktop computer Web browsers. Because HTML conveys *meaning*, rather than presentation, the same page can also be used by a small browser on a mobile phone, without any change to the page. Instead of headings being in large letters as on the desktop, for example, the browser on the mobile phone might use the same size text for the whole the page, but with the headings in bold.

But it goes further than just differences in screen size: the same page could equally be used by a blind user using a browser based around speech synthesis, which instead of displaying the page on a screen, reads the page to the user, e.g. using

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

page quite difficult.

Even beyond browsers, software can make use of this information. Search engines can use the headings to more effectively index a page, or to provide quick links to subsections of the page from their results. Tools can use the headings to create a table of contents (that is in fact how this very specification's table of contents is generated).

This example has focused on headings, but the same principle applies to all of the semantics in HTML.

Authors must not use elements, attributes, or attribute values for purposes other than their appropriate intended semantic purpose, as doing so prevents software from correctly processing the page



For e Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<td>
  <a href="http://example.org/~ernest/"><cite>Ernest</cite></a>,
  in an essay from 1992
</td>
</tr>
</table>
</body>
</html>
```

...because the data placed in the cells is clearly not tabular data (and the `cite` element mis-used). This would make software that relies on these semantics fail: for example, a speech browser that allowed a blind user to navigate tables in the document would report the quote above as a table, confusing the user; similarly, a tool that extracted titles of works from pages would extract "Ernest" as the title of a work, even though it's actually a person's name, not a title.

A corrected version of this document might be:

```
<!DOCTYPE HTML>
<html lang="en-GB">
  <head> <title> Demonstration </title> </head>
  <body>
    <blockquote>
      <p> My favourite animal is the cat. </p>
    </blockquote>
    <p>
      <a href="http://example.org/~ernest/">Ernest</a>,
      in an essay from 1992
    </p>
  </body>
</html>
```

This next document fragment, intended to represent the heading of a corporate site, is similarly non-conforming because the second line is not intended to be a heading of a subsection, but merely a subheading or subtitle (a subordinate heading for the same section).

```
<body>
  <h1>ABC Company</h1>
  <h2>Leading the way in widget design since 1432</h2>
  ...

```

The `hgroup` element is intended for these kinds of situations:

```
<body>
  <hgroup>
    <h1>ABC Company</h1>
    <h2>Leading the way in widget design since 1432</h2>
  </hgroup>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

In the next example, there is a non-conforming attribute value ("carpet") and a non-conforming attribute ("texture"), which is not permitted by this specification:

```
<label>Carpet: <input type="carpet" name="c" texture="deep pile"></label>
```

Here would be an alternative and correct way to mark this up:

```
<label>Carpet: <input type="text" class="carpet" name="c" data-texture="deep pile"></label>
```

Through scripting and using other mechanisms, the values of attributes, text, and indeed the entire structure of the document may change dynamically. User agents must update their state of the document to reflect these changes.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

HTM would be updated by a script, the UA

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

3.2.2 Elements

The nodes representing HTML elements in the DOM must implement, and expose to scripts, the interfaces listed for them in the relevant sections of this specification. This includes HTML elements in XML documents, even when those documents are in another context (e.g. inside an XSLT transform).

Elements in the DOM represent things; that is, they have intrinsic *meaning*, also known as semantics.

For example, an `ol` element represents an ordered list.

The basic interface, from which all the HTML elements' interfaces inherit, and which must be used by elements that have no additional requirements, is the `HTMLElement` interface.

```
interface HTMLElement : Element {
    // metadata attributes
    attribute DOMString title;
    attribute DOMString lang;
    attribute boolean translate;
    attribute DOMString dir;
    readonly attribute DOMStringMap dataset;

    // microdata
    attribute boolean itemScope;
    [PutForwards=value] readonly attribute DOMSettableTokenList itemType;
    attribute DOMString itemId;
    [PutForwards=value] readonly attribute DOMSettableTokenList itemRef;
    [PutForwards=value] readonly attribute DOMSettableTokenList itemProp;
    readonly attribute HTMLPropertiesCollection properties;
    attribute any itemValue; // acts as DOMString on setting

    // user interaction
    attribute boolean hidden;
    void click();
    attribute long tabIndex;
    void focus();
    void blur();
    attribute DOMString accessKey;
    readonly attribute DOMString accessKeyLabel;
    attribute boolean draggable;
    [PutForwards=value] readonly attribute DOMSettableTokenList dropzone;
    attribute DOMString contentEditable;
    readonly attribute boolean isContentEditable;
    attribute HTMLMenuElement? contextMenu;
    attribute boolean spellcheck;
    void forceSpellCheck();

    // command API
}
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
readonly attribute boolean? commandChecked;  
  
// styling  
[PutForwards=cssText] readonly attribute CSSStyleDeclaration style;  
  
// event handler IDL attributes  
attribute EventHandler onabort;  
attribute EventHandler onblur;  
attribute EventHandler oncancel;
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
attribute EventHandler ondragend;  
attribute EventHandler ondragenter;  
attribute EventHandler ondragleave;  
attribute EventHandler ondragover;  
attribute EventHandler ondragstart;  
attribute EventHandler ondrop;  
attribute EventHandler ondurationchange;  
attribute EventHandler onemptied;  
attribute EventHandler onended;  
attribute OnErrorEventHandler onerror;  
attribute EventHandler onfocus;  
attribute EventHandler oninput;  
attribute EventHandler oninvalid;  
attribute EventHandler onkeydown;  
attribute EventHandler onkeypress;  
attribute EventHandler onkeyup;  
attribute EventHandler onload;  
attribute EventHandler onloadeddata;  
attribute EventHandler onloadedmetadata;  
attribute EventHandler onloadstart;  
attribute EventHandler onmousedown;  
attribute EventHandler onmousemove;  
attribute EventHandler onmouseout;  
attribute EventHandler onmouseover;  
attribute EventHandler onmouseup;  
attribute EventHandler onmousewheel;  
attribute EventHandler onpause;  
attribute EventHandler onplay;  
attribute EventHandler onplaying;  
attribute EventHandler onprogress;  
attribute EventHandler onratechange;  
attribute EventHandler onreset;  
attribute EventHandler onscroll;  
attribute EventHandler onseeked;  
attribute EventHandler onseeking;  
attribute EventHandler onselect;  
attribute EventHandler onshow;  
attribute EventHandler onsort;  
attribute EventHandler onstalled;  
attribute EventHandler onsubmit;  
attribute EventHandler onsuspend;  
attribute EventHandler ontimeupdate;  
attribute EventHandler onvolumechange;  
attribute EventHandler onwaiting;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

The `HTMLElement` interface holds methods and attributes related to a number of disparate features, and the members of this interface are therefore described in various different sections of this specification.

The `HTMLUnknownElement` interface must be used for HTML elements that are not defined by this specification (or other applicable specifications).

3.2.3 Global attributes

The following attributes are common to and may be specified on all HTML elements (even those defined in this specification):

- `acc`
- `cla`
- `con`
- `cor`
- `dir`
- `dra`
- `drc`
- `hic`
- `id`
- `ineru`
- `itemid`
- `itemprop`
- `itemref`
- `itemscope`
- `itemtype`
- `lang`
- `spellcheck`
- `style`
- `tabindex`
- `title`
- `translate`

These attributes are only defined by this specification as attributes for HTML elements. When this specification refers to elements having these attributes, elements from namespaces that are not defined as having these attributes must not be considered as being elements with these attributes.

For example, in the following XML fragment, the "bogus" element does not have a `dir` attribute as defined in this specification, despite having an attribute with the literal name "dir". Thus, the directionality of the inner-most `span` element is 'rtl', inherited from the `div` element indirectly through the "bogus" element.

```
<div xmlns="http://www.w3.org/1999/xhtml" dir="rtl">
  <bogus xmlns="http://example.net/ns" dir="ltr">
    <span xmlns="http://www.w3.org/1999/xhtml">
      </span>
    </bogus>
  </div>
```

The following event handler content attributes may be specified on any HTML element:

- `onabort`
- `onblur*`
- `oncancel`
- `oncanplay`
- `oncanplaythrough`
- `onchange`
- `onclick`
- `onclose`
- `oncontextmenu`
- `oncuechange`
- `ondblclick`
- `ondrag`
- `ondragend`
- `ondragenter`
- `ondragleave`
- `ondragover`
- `ondragstart`
- `ondrop`
- `ondurationchange`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

- oninvalid
- onkeydown
- onkeypress
- onkeyup
- onload*
- onloadeddata
- onloadedmetadata
- onloadstart
- onmousedown
- onmousemove
- onmouseout
- onmouseover
- onscroll
- onseeked
- onseeking
- onselect
- onshow
- onsort
- onstalled
- onsubmit
- onsuspend
- ontimeupdate
- onvolumechange
- onwaiting

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



Note: The attributes marked with an asterisk have a different meaning when specified on body elements as those elements expose event handlers of the Window object with the same names.

Note: While these attributes apply to all elements, they are not useful on all elements. For example, only media elements will ever receive a volumechange event fired by the user agent.

Custom data attributes (e.g. data-foldername or data-msgid) can be specified on any HTML element, to store custom data specific to the page.

In HTML documents, elements in the HTML namespace may have an xmlns attribute specified, if, and only if, it has the exact value "http://www.w3.org/1999/xhtml". This does not apply to XML documents.

Note: In HTML, the xmlns attribute has absolutely no effect. It is basically a talisman. It is allowed merely to make migration to and from XHTML mildly easier. When parsed by an HTML parser, the attribute ends up in no namespace, not the "http://www.w3.org/2000/xmlns/" namespace like namespace declaration attributes in XML do.

Note: In XML, an xmlns attribute is part of the namespace declaration mechanism, and an element cannot actually have an xmlns attribute in no namespace specified.

The XML specification also allows the use of the xml:space attribute in the XML namespace on any element in an XML document. This attribute has no effect on HTML elements, as the default behavior in HTML is to preserve whitespace. [XML]

Note: There is no way to serialize the xml:space attribute on HTML elements in the text/html syntax.

To enable assistive technology products to expose a more fine-grained interface than is otherwise possible with HTML elements and attributes, a set of annotations for assistive technology products can be specified (the ARIA role and aria-* attributes).

3.2.3.1 The id attribute

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Note: An element's unique identifier can be used for a variety of purposes, most notably as a way to link to specific parts of a document using fragment identifiers, as a way to target an element when scripting, and as a way to style a specific element from CSS.

Identifiers are opaque strings. Particular meanings should not be derived from the value of the `id` attribute.

3.2.3.2 The `title` attribute

The `title` attribute

the title or a
it could be a
could be a la

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org>

Note: If the `title` attribute's value contains U+000A LINE FEED (LF) characters, the content is split into multiple lines. Each U+000A LINE FEED (LF) character represents a line break.

tooltip. On a link, this could be
the image; on a paragraph,
etc; on interactive content, it

use the attribute in an
mouse to cause a
with a modern phone or

If this attribute is omitted from an element, then it implies that the `title` attribute of the nearest ancestor HTML element with a `title` attribute set is also relevant to this element. Setting the attribute overrides this, explicitly stating that the advisory information of any ancestors is not relevant to this element. Setting the attribute to the empty string indicates that the element has no advisory information.

If the `title` attribute's value contains U+000A LINE FEED (LF) characters, the content is split into multiple lines. Each U+000A LINE FEED (LF) character represents a line break.

Caution is advised with respect to the use of newlines in `title` attributes.

For instance, the following snippet actually defines an abbreviation's expansion *with a line break in it*:

```
<p>My logs show that there was some interest in <abbr title="Hypertext  
Transport Protocol">HTTP</abbr> today.</p>
```

Some elements, such as `link`, `abbr`, and `input`, define additional semantics for the `title` attribute beyond the semantics described above.

The **advisory information** of an element is the value that the following algorithm returns, with the algorithm being aborted once a value is returned. When the algorithm returns the empty string, then there is no advisory information.

1. If the element is a `link`, `style`, `dfn`, `abbr`, or `title` element, then: if the element has a `title` attribute, return the value of that attribute, otherwise, return the empty string.
2. Otherwise, if the element has a `title` attribute, then return its value.
3. Otherwise, if the element has a parent element, then return the parent element's advisory information.
4. Otherwise, return the empty string.

User agents should inform the user when elements have advisory information, otherwise the information would not be discoverable.

The `title` IDL attribute must reflect the `title` content attribute.

3.2.3.3 The `lang` and `xml:lang` attributes

The `lang` attribute (in no namespace) specifies the primary language for the element's contents and for any of the element's attributes that contain text. Its value must be a valid BCP 47 language tag, or the empty string. Setting the attribute to the empty string indicates that the primary language is unknown. [BCP47]

The `lang` attribute in the XML namespace is defined in XML. [XML]

If these attributes are omitted from an element, then the language of this element is the same as the language of its parent element, if any.

The `lang` attribute in no namespace may be used on any HTML element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Authors must not use the `lang` attribute in the XML namespace on HTML elements in HTML documents. To ease migration to and from XHTML, authors may specify an attribute in no namespace with no prefix and with the literal localname "`xml:lang`" on HTML elements in HTML documents, but such attributes must only be specified if a `lang` attribute in no namespace is also specified, and both attributes must have the same value when compared in an ASCII case-insensitive manner.

Note: The attribute in no namespace with no prefix and with the literal localname "`xml:lang`" has no effect on language processing.



To determine the language of an element, user agents must use the `lang` attribute in no namespace if the node is in no namespace attribute set. That attribute's value is the language of the node.

If both the `lang` attribute and the `xml:lang` attribute are present on the element, user agents must use the `lang` attribute.

If neither the `lang` attribute nor the `xml:lang` attribute is present on the element, user agents must use the `lang` attribute.

If there is a pragma-set or a default language set, then that is the language of the node. If there is no pragma-set or default language set, then language information from a higher-level protocol (such as HTTP), if any, must be used as the final fallback language instead. In the absence of any such language information, and in cases where the higher-level protocol reports multiple languages, the language of the node is unknown, and the corresponding language tag is the empty string.

If the resulting value is not a recognized language tag, then it must be treated as an unknown language having the given language tag, distinct from all other languages. For the purposes of round-tripping or communicating with other services that expect language tags, user agents should pass unknown language tags through unmodified, and tagged as being BCP 47 language tags, so that subsequent services do not interpret the data as another type of language description. [BCP47]

Thus, for instance, an element with `lang="xyzzy"` would be matched by the selector `:lang(xyzzy)` (e.g. in CSS), but it would not be matched by `:lang(abcd)`, even though both are equally invalid. Similarly, if a Web browser and screen reader working in unison communicated about the language of the element, the browser would tell the screen reader that the language was "xyzzy", even if it knew it was invalid, just in case the screen reader actually supported a language with that tag after all. Even if the screen reader supported both BCP 47 and another syntax for encoding language names, and in that other syntax the string "xyzzy" was a way to denote the Belarusian language, it would be *incorrect* for the screen reader to then start treating text as Belarusian, because "xyzzy" is not how Belarusian is described in BCP 47 codes (BCP 47 uses the code "be" for Belarusian).

If the resulting value is the empty string, then it must be interpreted as meaning that the language of the node is explicitly unknown.

User agents may use the element's language to determine proper processing or rendering (e.g. in the selection of appropriate fonts or pronunciations, for dictionary selection, or for the user interfaces of form controls such as date pickers).

The `lang` IDL attribute must reflect the `lang` content attribute in no namespace.

3.2.3.4 The `translate` attribute

The `translate` attribute is an enumerated attribute that is used to specify whether an element's attribute values and the values of its `Text` node children are to be translated when the page is localized, or whether to leave them unchanged.

The attribute's keywords are the empty string, `yes`, and `no`. The empty string and the `yes` keyword map to the `yes` state. The `no` keyword maps to the `no` state. In addition, there is a third state, the `inherit` state, which is the *missing value default* (and the *invalid value default*).

Each element has a **translation mode**, which is in either the translate-enabled state or the no-translate state. If the element's `translate` attribute is in the `yes` state, then the element's translation mode is in the translate-enabled state. Otherwise, if the element's `translate` attribute is in the `no` state, then the element's translation mode is in the no-translate state. Otherwise, the element's `translate` attribute is in the `inherit` state; in that case, the element's translation mode is in the same state as its parent element, if any, or in the translate-enabled state, if the element is a root element.

When an element is in the **translate-enabled** state, the element's attribute values and the values of its `Text` node children are to be translated when the page is localized.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

On setting, it must set the content attribute's value to "yes" if the new value is true, and set the content attribute's value to "no" otherwise.

In this example, everything in the document is to be translated when the page is localised, except the sample keyboard input and sample program output:

```
<!DOCTYPE HTML>
<html> <!-- default on the root element is translate=yes -->
<head>
<!-- example Doc comes from / --> <!-- and translation inherited from ancestors -->
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

>re>

3.2.3.5 The `xml:base` attribute (XML only)

The `xml:base` attribute is defined in XML Base. [XMLBASE]

The `xml:base` attribute may be used on HTML elements of XML documents. Authors must not use the `xml:base` attribute on HTML elements in HTML documents.

3.2.3.6 The `dir` attribute

The `dir` attribute specifies the element's text directionality. The attribute is an enumerated attribute with the following keywords and states:

The `ltr` keyword, which maps to the `ltr` state

Indicates that the contents of the element are explicitly directionally embedded left-to-right text.

The `rtl` keyword, which maps to the `rtl` state

Indicates that the contents of the element are explicitly directionally embedded right-to-left text.

The `auto` keyword, which maps to the `auto` state

Indicates that the contents of the element are explicitly embedded text, but that the direction is to be determined programmatically using the contents of the element (as described below).

Note: *The heuristic used by this state is very crude (it just looks at the first character with a strong directionality, in a manner analogous to the Paragraph Level determination in the bidirectional algorithm). Authors are urged to only use this value as a last resort when the direction of the text is truly unknown and no better server-side heuristic can be applied. [BIDI]*

Note: *For `textarea` and `pre` elements, the heuristic is applied on a per-paragraph level.*

The attribute has no *invalid value default* and no *missing value default*.

The directionality of an element (any element, not just an HTML element) is either '`ltr`' or '`rtl`', and is determined as per the first appropriate set of steps from the following list:

↳ If the element's `dir` attribute is in the `ltr` state

The directionality of the element is '`ltr`'.

↳ If the element's `dir` attribute is in the `rtl` state

The directionality of the element is '`rtl`'.

↳ If the element is an `input` element whose `type` attribute is in the Text, Search, Telephone, URL, or E-mail state, and the `dir` attribute is in the `auto` state

↳ If the element is a `textarea` element and the `dir` attribute is in the `auto` state

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

↳ If the element is a `bdi` element and the `dir` attribute is not in a defined state (i.e. it is not present or has an invalid value)

Find the first character in tree order that matches the following criteria:

- The character is from a `Text` node that is a descendant of the element whose directionality is being determined.
- The character is of bidirectional character type L, AL, or R. [BIDI]
- The character is not in a `Text` node that has an ancestor element that is a descendant of the element whose directionality is being determined and that is either:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

if the element is 'rtl'.

↳ If the element is a root element and the `dir` attribute is not in a defined state (i.e. it is not present or has an invalid value)

The directionality of the element is 'ltr'.

↳ If the element has a parent element and the `dir` attribute is not in a defined state (i.e. it is not present or has an invalid value)

The directionality of the element is the same as the element's parent element's directionality.

Note: Since the `dir` attribute is only defined for HTML elements, it cannot be present on elements from other namespaces. Thus, elements from other namespaces always just inherit their directionality from their parent element, or, if they don't have one, default to 'ltr'.

The effect of this attribute is primarily on the presentation layer. For example, the rendering section in this specification defines a mapping from this attribute to the CSS 'direction' and 'unicode-bidi' properties, and CSS defines rendering in terms of those properties.

This box is non-normative. Implementation requirements are given below this box.

`document.dir [= value]`

Returns the `html` element's `dir` attribute's value, if any.

Can be set, to either "ltr", "rtl", or "auto" to replace the `html` element's `dir` attribute's value.

If there is no `html` element, returns the empty string and ignores new values.

The `dir` IDL attribute on an element must reflect the `dir` content attribute of that element, limited to only known values.

The `dir` IDL attribute on `Document` objects must reflect the `dir` content attribute of the `html` element, if any, limited to only known values. If there is no such element, then the attribute must return the empty string and do nothing on setting.

Note: Authors are strongly encouraged to use the `dir` attribute to indicate text direction rather than using CSS, since that way their documents will continue to render correctly even in the absence of CSS (e.g. as interpreted by search engines).

This markup fragment is of an IM conversation.

```
<p dir="auto class="u1"><b><bdi>Student</bdi></b> How do you write "What's your name?" in Arabic?</p>
<p dir="auto class="u2"><b><bdi>Teacher</bdi></b> م&#177; اسمك؟</p>
<p dir="auto class="u1"><b><bdi>Student</bdi></b> Thanks.</p>
<p dir="auto class="u2"><b><bdi>Teacher</bdi></b> That's written "ش&#177;كرًا".</p>
<p dir="auto class="u2"><b><bdi>Teacher</bdi></b> Do you know how to write "Please"?</p>
<p dir="auto class="u1"><b><bdi>Student</bdi></b> من فضلك "، right?</p>
```

Given a suitable style sheet and the default alignment styles for the `p` element, namely to align the text to the `start` edge of the paragraph, the resulting rendering could be as follows:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Student: Thanks.**Teacher:** That's written "شكراً".**Teacher:** Do you know how to write "Please"??right , " من فضلك" : **Student**

As noted earlier, the `auto` value is not a panacea. The final paragraph in this example is misinterpreted as being right-to-left text, since it begins with an Arabic character which causes the "right?" to be to the left of the Arabic text.

3.2.3.7 Th Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

Every HTML

The attribute belongs to: /web/20130103030411/http://whatwg.org/html

The classes `l` on spaces. (Duplicates are ignored.)

us classes that the element

of the `class` attribute is split

Note: Assigning classes to an element affects class matching in selectors in CSS, the `getElementsByClassName()` method in the DOM, and other such features.

There are no additional restrictions on the tokens authors can use in the `class` attribute, but authors are encouraged to use values that describe the nature of the content, rather than values that describe the desired presentation of the content.

Note: The `className` and `classList` IDL attributes, defined in the DOM specification, reflect the `class` content attribute. [DOM]

3.2.3.8 The `style` attribute

All HTML elements may have the `style` content attribute set. This is a CSS styling attribute as defined by the CSS Styling Attribute Syntax specification. [CSSATTR]

In user agents that support CSS, the attribute's value must be parsed when the attribute is added or has its value changed, according to the rules given for CSS styling attributes. [CSSATTR]

Documents that use `style` attributes on any of their elements must still be comprehensible and usable if those attributes were removed.

Note: In particular, using the `style` attribute to hide and show content, or to convey meaning that is otherwise not included in the document, is non-conforming. (To hide and show content, use the `hidden` attribute.)

This box is non-normative. Implementation requirements are given below this box.

`element.style`

Returns a `CSSStyleDeclaration` object for the element's `style` attribute.

The `style` IDL attribute must return a `CSSStyleDeclaration` whose value represents the declarations specified in the attribute. (If the attribute is absent, the object represents an empty declaration.) Mutating the `CSSStyleDeclaration` object must create a `style` attribute on the element (if there isn't one already) and then change its value to be a value representing the serialized form of the `CSSStyleDeclaration` object. The same object must be returned each time. [CSSOM]

In the following example, the words that refer to colors are marked up using the `span` element and the `style` attribute to make those words show up in the relevant colors in visual media.

```
<p>My sweat suit is <span style="color: green; background: transparent">green</span> and my eyes are <span style="color: blue; background: transparent">blue</span>.</p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Note: All attribute names on HTML elements in HTML documents get ASCII-lowercased automatically, so the restriction on ASCII uppercase letters doesn't affect such documents.

Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.

These attributes are not intended for use by software that is independent of the site that uses the attributes.

For instance, a site about music could annotate list items representing tracks in an album with custom data attributes containing the letter `l` to filter the list by track length, or

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

It would be easier to search for tracks of a certain length on a basic site.

This is because these attributes are intended for use by the site's own scripts, and are not a generic extension mechanism for publicly-visible metadata.

Every HTML element may have any number of custom data attributes specified, with any value.

This box is non-normative. Implementation requirements are given below this box.

`element.dataset`

Returns a `DOMStringMap` object for the element's `data-*` attributes.

Hyphenated names become camel-cased. For example, `data-foo-bar=""` becomes `element.dataset.fooBar`.

The `dataset` IDL attribute provides convenient accessors for all the `data-*` attributes on an element. On getting, the `dataset` IDL attribute must return a `DOMStringMap` object, associated with the following algorithms, which expose these attributes on their element:

The algorithm for getting the list of name-value pairs

1. Let `list` be an empty list of name-value pairs.
2. For each content attribute on the element whose first five characters are the string "data-" and whose remaining characters (if any) do not include any uppercase ASCII letters, add a name-value pair to `list` whose name is the attribute's name with the first five characters removed and whose value is the attribute's value.
3. For each name `list`, for each U+002D HYPHEN-MINUS character (-) in the name that is followed by a lowercase ASCII letter, remove the U+002D HYPHEN-MINUS character (-) and replace the character that followed it by the same character converted to ASCII uppercase.
4. Return `list`.

The algorithm for setting names to certain values

1. Let `name` be the name passed to the algorithm.
2. Let `value` be the value passed to the algorithm.
3. If `name` contains a U+002D HYPHEN-MINUS character (-) followed by a lowercase ASCII letter, throw a `SyntaxError` exception and abort these steps.
4. For each uppercase ASCII letter in `name`, insert a U+002D HYPHEN-MINUS character (-) before the character and replace the character with the same character converted to ASCII lowercase.
5. Insert the string `data-` at the front of `name`.
6. Set the value of the attribute with the name `name`, to the value `value`, replacing any previous value if the attribute already existed. If `setAttribute()` would have thrown an exception when setting an attribute with the name `name`, then this must throw the same exception.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

replace the character with the same character converted to ASCII lowercase.

3. Insert the string `data-` at the front of `name`.
4. Remove the attribute with the name `name`, if such an attribute exists. Do nothing otherwise.

The same object must be returned each time.

If a Web page wanted an element to represent a space ship, e.g. as part of a game, it would have to use the `class` attribute along with `data-*` attributes:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Notice how the hyphenated attribute name becomes camel-cased in the API.

Authors should carefully design such extensions so that when the attributes are ignored and any associated CSS dropped, the page is still usable.

User agents must not derive any implementation behavior from these attributes or values. Specifications intended for user agents must not define these attributes to have any meaningful values.

JavaScript libraries may use the custom data attributes, as they are considered to be part of the page on which they are used. Authors of libraries that are reused by many authors are encouraged to include their name in the attribute names, to reduce the risk of clashes. Where it makes sense, library authors are also encouraged to make the exact name used in the attribute names customizable, so that libraries whose authors unknowingly picked the same name can be used on the same page, and so that multiple versions of a particular library can be used on the same page even when those versions are not mutually compatible.

For example, a library called "DoQuery" could use attribute names like `data-doquery-range`, and a library called "jJo" could use attributes names like `data-jjo-range`. The jJo library could also provide an API to set which prefix to use (e.g. `J.setDataPrefix('j2')`, making the attributes have names like `data-j2-range`).

3.2.4 Element definitions

Each element in this specification has a definition that includes the following information:

Categories

A list of categories to which the element belongs. These are used when defining the content models for each element.

Contexts in which this element can be used

A non-normative description of where the element can be used. This information is redundant with the content models of elements that allow this one as a child, and is provided only as a convenience.

Note: For simplicity, only the most specific expectations are listed. For example, an element that is both flow content and phrasing content can be used anywhere that either flow content or phrasing content is expected, but since anywhere that flow content is expected, phrasing content is also expected (since all phrasing content is flow content), only "where phrasing content is expected" will be listed.

Content model

A normative description of what content must be included as children and descendants of the element.

Content attributes

A normative list of attributes that may be specified on the element (except where otherwise disallowed).

DOM interface

A normative definition of a DOM interface that such elements must implement.

This is then followed by a description of what the element represents, along with any additional normative conformance criteria that may apply to authors and implementations. Examples are sometimes also included.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3.2.5 Content models

Each element defined in this specification has a content model: a description of the element's expected contents. An HTML element must have contents that match the requirements described in the element's content model.

The space characters are always allowed between elements. User agents represent these characters between elements in the source markup as `Text` nodes in the DOM. Empty `Text` nodes and `Text` nodes consisting of just sequences of those characters are considered **inter-element whitespace**.

Inter-element
contents ma
semantics.

**Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org/
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

lishing whether an element's
fine document and element

*Note: T
node a
Similar
whites*

*have the same parent
e) between them.
han inter-element*

Authors must not use HTML elements anywhere except where they are explicitly allowed, as defined for each element, or as explicitly required by other specifications. For XML compound documents, these contexts could be inside elements from other namespaces, if those elements are defined as providing the relevant contexts.

For example, the Atom specification defines a `content` element. When its `type` attribute has the value `xhtml`, the Atom specification requires that it contain a single HTML `div` element. Thus, a `div` element is allowed in that context, even though this is not explicitly normatively stated by this specification. [ATOM]

In addition, HTML elements may be orphan nodes (i.e. without a parent node).

For example, creating a `td` element and storing it in a global variable in a script is conforming, even though `td` elements are otherwise only supposed to be used inside `tr` elements.

```
var data = {
  name: "Banana",
  cell: document.createElement('td'),
};
```

3.2.5.1 Kinds of content

Each element in HTML falls into zero or more **categories** that group elements with similar characteristics together. The following broad categories are used in this specification:

- Metadata content
- Flow content
- Sectioning content
- Heading content
- Phrasing content
- Embedded content
- Interactive content

Note: Some elements also fall into other categories, which are defined in other parts of this specification.

These categories are related as follows:

Go

NOV JAN FEB

03

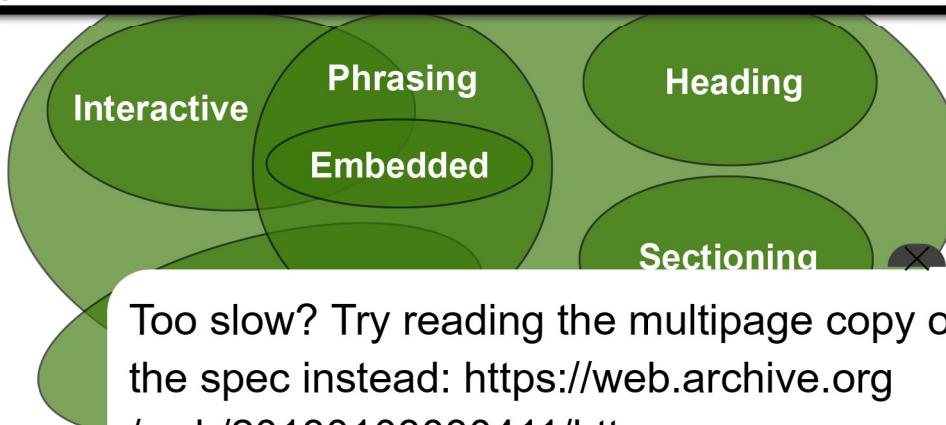
2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020



Sectioning c
Metadata is s
type of phrasing content, and sometimes is interactive content.

types of flow content.
Embedded content is also a

Other categories are also used for specific purposes, e.g. form controls are specified using a number of categories to define common requirements. Some elements have unique requirements and do not fit into any particular category.

3.2.5.1.1 Metadata content

Metadata content is content that sets up the presentation or behavior of the rest of the content, or that sets up the relationship of the document with other documents, or that conveys other "out of band" information.

⇒ base, link, meta, noscript, script, style, title

Elements from other namespaces whose semantics are primarily metadata-related (e.g. RDF) are also metadata content.

Thus, in the XML serialization, one can use RDF, like this:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:r="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
<head>
  <title>Hedral's Home Page</title>
  <r:RDF>
    <Person xmlns="http://www.w3.org/2000/10/swap/pim/contact#"
            r:about="http://hedral.example.com/#">
      <fullName>Cat Hedral</fullName>
      <mailbox r:resource="mailto:hedral@damowmow.com"/>
      <personalTitle>Sir</personalTitle>
    </Person>
  </r:RDF>
</head>
<body>
  <h1>My home page</h1>
  <p>I like playing with string, I guess. Sister says squirrels are fun
     too so sometimes I follow her to play with them.</p>
</body>
</html>
```

This isn't possible in the HTML serialization, however.

3.2.5.1.2 Flow content

Most elements that are used in the body of documents and applications are categorized as **flow content**.

⇒ a, abbr, address, area (if it is a descendant of a map element), article, aside, audio, b, bdi, bdo, blockquote, br, button, canvas, cite, code, data, datalist, del, details, dfn, dialog, div, dl, em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, keygen, label,

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3.2.5.1.3 Sectioning content

Sectioning content is content that defines the scope of headings and footers.

⇒ `article, aside, nav, section`

Each sectioning content element potentially has a heading and an outline. See the section on headings and sections for further details.

Note: There are also certain elements that are sectioning roots. These are distinct from sectioning content, but they can also ...

Too slow? Try reading the multipage copy of

3.2.5.1.4 *He* the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Heading co
heading con /web/20130103030411/http:
ements, or implied by the

⇒¹ `//whatwg.org/html`

3.2.5.1.5 Phrasing content

Phrasing content is the text of the document, as well as elements that mark up that text at the intra-paragraph level. Runs of phrasing content form paragraphs.

⇒ `a, abbr, area (if it is a descendant of a map element), audio, b, bdi, bdo, br, button, canvas, cite, code, data, datalist, del, dfn, em, embed, i, iframe, img, input, ins, kbd, keygen, label, link (if the itemprop attribute is present), map, mark, math, meta (if the itemprop attribute is present), meter, noscript, object, output, progress, q, ruby, s, samp, script, select, small, span, strong, sub, sup, svg, textarea, time, u, var, video, wbr, text`

As a general rule, elements whose content model allows any phrasing content should have either at least one descendant `Text` node that is not inter-element whitespace, or at least one descendant element node that is embedded content. For the purposes of this requirement, nodes that are descendants of `del` elements must not be counted as contributing to the ancestors of the `del` element.

Note: Most elements that are categorized as phrasing content can only contain elements that are themselves categorized as phrasing content, not any flow content.

Text, in the context of content models, means either nothing, or `Text` nodes. `Text` is sometimes used as a content model on its own, but is also phrasing content, and can be inter-element whitespace (if the `Text` nodes are empty or contain just space characters).

`Text` nodes and attribute values must consist of Unicode characters, must not contain U+0000 characters, must not contain permanently undefined Unicode characters (noncharacters), and must not contain control characters other than space characters. This specification includes extra constraints on the exact value of `Text` nodes and attribute values depending on their precise context.

3.2.5.1.6 Embedded content

Embedded content is content that imports another resource into the document, or content from another vocabulary that is inserted into the document.

⇒ `audio, canvas, embed, iframe, img, math, object, svg, video`

Elements that are from namespaces other than the HTML namespace and that convey content but not metadata, are embedded content for the purposes of the content models defined in this specification. (For example, MathML, or SVG.)

Some embedded content elements can have **fallback content**: content that is to be used when the external resource cannot be used (e.g. because it is of an unsupported format). The element definitions state what the fallback is, if any.

3.2.5.1.7 Interactive content

Interactive content is content that is specifically intended for user interaction.

⇒ `a, audio (if the controls attribute is present), button, details, embed, iframe, img (if the usemap attribute is present), input (if the type attribute is not in the hidden state), keygen, label, object (if the usemap attribute is present), select, textarea, video (if the controls attribute is present)`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

clicking it, the default action of the interaction event must be to run synthetic click activation steps on the element.

Each element has a *click in progress* flag, initially set to false.

When a user agent is to **run synthetic click activation steps** on an element, the user agent must run the following steps:

1. If the element's *click in progress* flag is set to true, then abort these steps.
2. Set the *click in progress* flag on the element to true.
3. Run ↻
4. Fire met
5. If th If th
6. Set ↻

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

I because the `click()`

When a pointing device is clicked, the user agent must run these steps:

1. If the element's *click in progress* flag is set to true, then abort these steps.
2. Set the *click in progress* flag on the element to true.
3. Let e be the nearest activatable element of the element designated by the user (defined below), if any.
4. If there is an element e , run pre-click activation steps on it.
5. Dispatch the required `click` event.

If there is an element e and the `click` event is not canceled, run post-click activation steps on element e .

If there is an element e and the event is canceled, run canceled activation steps on element e .

6. Set the *click in progress* flag on the element to false.

Note: The above doesn't happen for arbitrary synthetic events dispatched by author script. However, the `click()` method can be used to make it happen programmatically.

Note: Click-focusing behavior (e.g. the focusing of a text field when user clicks in one) typically happens before the click, when the mouse button is first depressed, and is therefore not discussed here.

Given an element $target$, the **nearest activatable element** is the element returned by the following algorithm:

1. If $target$ has a defined activation behavior, then return $target$ and abort these steps.
2. If $target$ has a parent element, then set $target$ to that parent element and return to the first step.
3. Otherwise, there is no nearest activatable element.

When a user agent is to **run pre-click activation steps** on an element, it must run the **pre-click activation steps** defined for that element, if any.

When a user agent is to **run canceled activation steps** on an element, it must run the **canceled activation steps** defined for that element, if any.

When a user agent is to **run post-click activation steps** on an element, it must run the **activation behavior** defined for that element, if any. Activation behaviors can refer to the `click` event that was fired by the steps above leading up to this point.

3.2.5.1.8 Palpable content

As a general rule, elements whose content model allows any flow content or phrasing content should have at least one child node that is **palpable content** and that does not have the `hidden` attribute specified.

This requirement is not a hard requirement, however, as there are many cases where an element can be empty legitimately, for

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The following elements are palpable content:

⇒ a, abbr, address, article, aside, audio (if the controls attribute is present), b, bdi, bdo, blockquote, button, canvas, cite, code, data, details, dfn, div, dl (if the element's children include at least one name-value group), em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, i, iframe, img, input (if the type attribute is not in the hidden state), ins, kbd, keygen, label, map, mark, math, menu (if the type attribute is in the toolbar state), meter, nav, object, ol (if the element's children include at least one li element), output, p, pre, progress, q, ruby, s, samp, section, select, small, span, strong, sub, sup, svq, table, t, time, u, ul (if the element's child'

Too slow? Try reading the multipage copy of

3.2.5.2 Tr: the spec instead: <https://web.archive.org>

Some elements are transparent. This means that they are "transparent" in the sense that they do not affect the content model of the document. The content model of a transparent element is the content model of the element itself. The content model of a non-transparent element is the content model of the element plus the content model of its children.

model. The content model of a transparent element is the content model of the element itself. The content model of a non-transparent element is the content model of the element plus the content model of its children.

For instance, an ins element inside a ruby element cannot contain an rt element, because the part of the ruby element's content model that allows ins elements is the part that allows phrasing content, and the rt element is not phrasing content.

Note: In some cases, where transparent elements are nested in each other, the process has to be applied iteratively.

Consider the following markup fragment:

```
<p><object><param><ins><map><a href="/">Apples</a></map></ins></object></p>
```

To check whether "Apples" is allowed inside the a element, the content models are examined. The a element's content model is transparent, as is the map element's, as is the ins element's, as is the part of the object element's in which the ins element is found. The object element is found in the p element, whose content model is phrasing content. Thus, "Apples" is allowed, as text is phrasing content.

When a transparent element has no parent, then the part of its content model that is "transparent" must instead be treated as accepting any flow content.

3.2.5.3 Paragraphs

Note: The term **paragraph** as defined in this section is used for more than just the definition of the p element. The paragraph concept defined here is used to describe how to interpret documents. The p element is merely one of several ways of marking up a paragraph.

A **paragraph** is typically a run of phrasing content that forms a block of text with one or more sentences that discuss a particular topic, as in typography, but can also be used for more general thematic grouping. For instance, an address is also a paragraph, as is a part of a form, a byline, or a stanza in a poem.

In the following example, there are two paragraphs in a section. There is also a heading, which contains phrasing content that is not a paragraph. Note how the comments and inter-element whitespace do not form paragraphs.

```
<section>
  <h1>Example of paragraphs</h1>
  This is the <em>first</em> paragraph in this example.
  <p>This is the second.</p>
  <!-- This is not a paragraph. -->
</section>
```

Paragraphs in flow content are defined relative to what the document looks like without the a, ins, del, and map elements complicating matters, since those elements, with their hybrid content models, can straddle paragraph boundaries, as shown in the first two examples below.

Note: Generally, having elements straddle paragraph boundaries is best avoided. Maintaining such markup can be difficult.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<section>
  <ins><h1>Example of paragraphs</h1>
  This is the <em>first</em> paragraph in</ins> this example<del>.
  <p>This is the second.</p></del>
  <!-- This is not a paragraph. -->
</section>
```

Let `view` be a view of the DOM that replaces all `a`, `ins`, `del`, and `map` elements in the document with their contents. Then, in `view`, for each run of sibling phrasing content nodes uninterrupted by other types of content, in an element `X` that accepts content other than phrasing content, the first node in the run that contains `X` is the first node in the run. For each such run, if there is no `del` element in the run, then the first node in the run is the first node in the run. For each such run, if there is a `del` element in the run, then the first node in the run is the first node in the run.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Conformance
video, audio
therein, like
A paragraph

of the run. For each such run, if there is no `del` element in the run, then the first node in the run is the first node in the run. For each such run, if there is a `del` element in the run, then the first node in the run is the first node in the run.

This can happen with `object`,
'ML to be further embedded

Note: The `p` element can be used to wrap individual paragraphs when there would otherwise not be any content other than phrasing content to separate the paragraphs from each other.

In the following example, the link spans half of the first paragraph, all of the heading separating the two paragraphs, and half of the second paragraph. It straddles the paragraphs and the heading.

```
<header>
  Welcome!
  <a href="about.html">
    This is home of...
    <h1>The Falcons!</h1>
    The Lockheed Martin multirole jet fighter aircraft!
  </a>
  This page discusses the F-16 Fighting Falcon's innermost secrets.
</header>
```

Here is another way of marking this up, this time showing the paragraphs explicitly, and splitting the one link element into three:

```
<header>
  <p>Welcome! <a href="about.html">This is home of...</a></p>
  <h1><a href="about.html">The Falcons!</a></h1>
  <p><a href="about.html">The Lockheed Martin multirole jet
  fighter aircraft!</a> This page discusses the F-16 Fighting
  Falcon's innermost secrets.</p>
</header>
```

It is possible for paragraphs to overlap when using certain elements that define fallback content. For example, in the following section:

```
<section>
  <h1>My Cats</h1>
  You can play with my cat simulator.
  <object data="cats.sim">
    To see the cat simulator, use one of the following links:
    <ul>
      <li><a href="cats.sim">Download simulator file</a>
      <li><a href="http://sims.example.com/watch?v=LYds5xY4INU">Use online simulator</a>
    </ul>
    Alternatively, upgrade to the Mellblom Browser.
  </object>
  I'm quite proud of it.
</section>
```

There are five paragraphs:

1. The paragraph that says "You can play with my cat simulator. `object` I'm quite proud of it.", where `object` is the `object`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The first paragraph is overlapped by the other four. A user agent that supports the "cats.sim" resource will only show the first one, but a user agent that shows the fallback will confusingly show the first sentence of the first paragraph as if it was in the same paragraph as the second one, and will show the last paragraph as if it was at the start of the second sentence of the first paragraph.

To avoid this confusion, explicit `p` elements can be used. For example:

```
<section>
  <h1>My Cats</h1>
  <p>
```

Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
</p>
</section>
```

ie simulator

3.2.6 Requirements relating to bidirectional-algorithm formatting characters

Text content in HTML elements with child `Text` nodes, and text in attributes of HTML elements that allow free-form text, may contain characters in the range U+202A to U+202E (the bidirectional-algorithm formatting characters). However, the use of these characters is restricted so that any embedding or overrides generated by these characters do not start and end with different parent elements, and so that all such embeddings and overrides are explicitly terminated by a U+202C POP DIRECTIONAL FORMATTING character. This helps reduce incidences of text being reused in a manner that has unforeseen effects on the bidirectional algorithm. [BIDI]

The aforementioned restrictions are defined by specifying that certain parts of documents form bidirectional-algorithm formatting character ranges, and then imposing a requirement on such ranges.

The strings resulting from applying the following algorithm to an HTML element `element` are bidirectional-algorithm formatting character ranges:

1. Let `output` be an empty list of strings.
2. Let `string` be an empty string.
3. Let `node` be the first child node of `element`, if any, or null otherwise.
4. *Loop*: If `node` is null, jump to the step labeled *end*.
5. Process `node` according to the first matching step from the following list:
 - ↪ If `node` is a `Text` node
 - Append the text data of `node` to `string`.
 - ↪ If `node` is a `br` element
 - ↪ If `node` is an HTML element that is flow content but that is not also phrasing content
 - If `string` is not the empty string, push `string` onto `output`, and let `string` be empty string.
 - ↪ Otherwise
 - Do nothing.
6. Let `node` be `node`'s next sibling, if any, or null otherwise.
7. Jump to the step labeled *loop*.
8. *End*: If `string` is not the empty string, push `string` onto `output`.
9. Return `output` as the bidirectional-algorithm formatting character ranges.

The value of a namespace-less attribute of an HTML element is a bidirectional-algorithm formatting character range.

Any strings that, as described above, are **bidirectional-algorithm formatting character ranges** must match the `string` production in

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```

lre      = %x202A ; U+202A LEFT-TO-RIGHT EMBEDDING
rle      = %x202B ; U+202B RIGHT-TO-LEFT EMBEDDING
lro      = %x202D ; U+202D LEFT-TO-RIGHT OVERRIDE
rlo      = %x202E ; U+202E RIGHT-TO-LEFT OVERRIDE
pdf      = %x202C ; U+202C POP DIRECTIONAL FORMATTING
plaintext = * ( %x0000-2029 / %x202F-10FFFF )
           ; any string with no bidirectional-algorithm formatting characters

```

Note: Authors are encouraged to use the `dir` attribute, the `bdo` element, and the `bdi` element, rather than maintaining the `bidi` poorly. These characters interact

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

3.2.7 WA!

Authors may ARIA specific described below.

states that do not represent the actual state of the document. [ARIA]

ments described in the It implicit ARIA semantics products report nonsensical

Authors must not set the ARIA role and aria-* attributes in a manner that conflicts with the semantics described in the following table, except that the presentation role may always be used. Authors must not set the ARIA role and aria-* attributes to values that match the default implicit ARIA semantics defined in the following two tables.

User agents are required to implement ARIA semantics on all HTML elements, as defined in the ARIA specifications. The default implicit ARIA semantics defined below must be recognized by implementations for the purposes of ARIA processing. [ARIAIMPL]

Note: The ARIA attributes defined in the ARIA specifications, and the strong native semantics and default implicit ARIA semantics defined below, do not have any effect on CSS pseudo-class matching, user interface modalities that don't use assistive technologies, or the default actions of user interaction events as described in this specification.

The following table defines the strong native semantics and corresponding default implicit ARIA semantics that apply to HTML elements. Each language feature (element or attribute) in a cell in the first column implies the ARIA semantics (any role, states, and properties) given in the cell in the second column of the same row. When multiple rows apply to an element, the role from the last row to define a role must be applied, and the states and properties from all the rows must be combined.

Language feature	Strong native semantics and default implicit ARIA semantics
area element that creates a hyperlink	link role
base element	No role
datalist element	listbox role, with the aria-multiselectable property set to "false"
details element	aria-expanded state set to "true" if the element's open attribute is present, and set to "false" otherwise
dialog element without an open attribute	The aria-hidden state set to "true"
head element	No role
hgroup element	heading role, with the aria-level property set to the element's outline depth
hr element	separator role
html element	No role
img element whose alt attribute's value is empty	presentation role
input element with a type attribute in the Checkbox state	aria-checked state set to "mixed" if the element's indeterminate IDL attribute is true, or "true" if the element's checkedness is true, or "false" otherwise
input element with a type attribute in the Color state	No role
input element with a type attribute in the Date state	No role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Date and Time state	No role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Local Date and Time state	No role, with the aria-readonly property set to "true" if the element has a readonly attribute

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

the File Upload state	
input element with a type attribute in the Hidden state	No role
input element with a type attribute in the Month state	No role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Number state	spinbutton role, with the aria-readonly property set to "true" if the element has a readonly attribute, the aria-valuemax property set to the element's maximum, the aria-valuemin property set to the element's minimum, and, if the result of applying the rules for parsing floating-point number values to the element's value is a number, with the aria-valuenow property set to that number
input element the Password input elem the Radio Butt input elem the Range stat input elem the Reset Butt	readonly attribute otherwise aria-valuemin property set to ig the rules for parsing floating- ue otherwise
input element with a type attribute in the Search state with no suggestions source element	textbox role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Submit Button state	button role
input element with a type attribute in the Telephone state with no suggestions source element	textbox role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Text state with no suggestions source element	textbox role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Text, Search, Telephone, URL, or E-mail states with a suggestions source element	combobox role, with the aria-owns property set to the same value as the list attribute, and the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Time state	No role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the URL state with no suggestions source element	textbox role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element with a type attribute in the Week state	No role, with the aria-readonly property set to "true" if the element has a readonly attribute
input element that is required	The aria-required state set to "true"
keygen element	No role
label element	No role
link element that creates a hyperlink	link role
menu element with a type attribute in the popup menu state	No role
meta element	No role
meter element	No role
nav element	navigation role
noscript element	No role
optgroup element	No role
option element that is in a list of options or that represents a suggestion in a datalist element	option role, with the aria-selected state set to "true" if the element's selectedness is true, or "false" otherwise.
param element	No role
progress element	progressbar role, with, if the progress bar is determinate, the aria-valuemax property set to the maximum value of the progress bar, the aria-valuemin property set to zero, and the aria-valuenow property set to the current value of the progress bar
script element	No role
select element with a multiple attribute	listbox role, with the aria-multiselectable property set to "true"

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

source element	No role	
style element	No role	
summary element	No role	
textarea element	textbox role, with the aria-multiline property set to "true", and the ariareadonly property set to "true" if the element has a readonly attribute	
textarea element with a required attribute	The aria-required state set to "true"	
title element	No role	X

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Some HTML ARIA semantics

first column implies, unless otherwise overridden, the ARIA semantic (role, state, or property) given in the cell in the second column of the same row, but this semantic may be overridden under the conditions listed in the cell in the third column of that row. In addition, any element may be given the presentation role, regardless of the restrictions below.

Language feature	Default implicit ARIA semantic	Restrictions
a element that creates a hyperlink	link role	Role must be either link, menuitem, tab, or treeitem
address element	No role	If specified, role must be contentinfo
article element	article role	Role must be either article, document, application, or main
aside element	complementary role	Role must be either complementary, note, or search
audio element	No role	If specified, role must be application
button element	button role	Role must be either button, menuitem
details element	group role	Role must be a role that supports aria-expanded
dialog element	dialog role	Role must be either alert, alertdialog, application, contentinfo, dialog, document, log, main, marquee, region, search, or status
embed element	No role	If specified, role must be either application, document, or img
footer element	No role	If specified, role must be contentinfo
h1 element that does not have an hgroup ancestor	heading role, with the aria-level property set to the element's outline depth	Role must be either heading or tab
h2 element that does not have an hgroup ancestor	heading role, with the aria-level property set to the element's outline depth	Role must be either heading or tab
h3 element that does not have an hgroup ancestor	heading role, with the aria-level property set to the element's outline depth	Role must be either heading or tab
h4 element that does not have an hgroup ancestor	heading role, with the aria-level property set to the element's outline depth	Role must be either heading or tab
h5 element that does not have an hgroup ancestor	heading role, with the aria-level property set to the element's outline depth	Role must be either heading or tab
h6 element that does not have an hgroup ancestor	heading role, with the aria-level property set to the element's outline depth	Role must be either heading or tab
header element	No role	If specified, role must be banner
iframe element	No role	If specified, role must be either application, document, or img
img element whose alt attribute's value is absent	img role	No restrictions
img element whose alt attribute's value is present and not empty	img role	No restrictions
input element with a type attribute in the Button state	button role	Role must be either button, menuitem
input element with a type attribute in the Checkbox state	checkbox role	Role must be either checkbox or menuitemcheckbox
input element with a type attribute in the Image Button state	button role	Role must be either button, menuitem
input element with a type attribute in the Radio Button state	radio role	Role must be either radio or menuitemradio

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

object element	No role	If specified, role must be either application, document, or img
ol element	list role	Role must be either directory, list, listbox, menu, menubar, tablist, toolbar, or tree
output element	status role	No restrictions
section element	region role	Role must be either alert, alertdialog, application, contentinfo, dialog, document, log, main, marquee, region, search, or status
ul element	list role	Role must be either directory, list, listbox, menu, menubar, tablist, toolbar, tree

video element

The body elem

Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>Conformance /whatwg.org/html

than remove could say "Use a more appropriate element to represent a button, for example a button element or an input element" rather than "The button role cannot be used with a elements".

These features can be used to make accessibility tools render content to their users in more useful ways. For example, ASCII art, which is really an image, appears to be text, and in the absence of appropriate annotations would end up being rendered by screen readers as a very painful reading of lots of punctuation. Using the features described in this section, one can instead make the ATs skip the ASCII art and just read the caption:

```
<figure role="img" aria-labelledby="fish-caption">
  <pre>
    o .-'`/
      '   / (
        o .-'` ``'-._   .
          _/ (o)   . ' / 
            ))))   >< <
              ``\ |_\ \
                '-._ _.-'   '..
      jgs   ``\_\ \
  </pre>
  <figcaption id="fish-caption">
    Joan G. Stark, "<cite>fish</cite>".
    October 1997. ASCII on electrons. 28x8.
  </figcaption>
</figure>
```

3.3 Interactions with XPath and XSLT

Implementations of XPath 1.0 that operate on HTML documents parsed or created in the manners described in this specification (e.g. as part of the `document.evaluate()` API) must act as if the following edit was applied to the XPath 1.0 specification.

First, remove this paragraph:

A QName in the node test is expanded into an expanded-name using the namespace declarations from the expression context. This is the same way expansion is done for element type names in start and end-tags except that the default namespace declared with `xmlns` is not used: if the QName does not have a prefix, then the namespace URI is null (this is the same way attribute names are expanded). It is an error if the QName has a prefix for which there is no namespace declaration in the expression context.

Then, insert in its place the following:

A QName in the node test is expanded into an expanded-name using the namespace declarations from the expression context. If the QName has a prefix, then there must be a namespace declaration for this prefix in the expression context, and the corresponding namespace URI is the one that is associated with this prefix. It is an error if the QName has a prefix for which there is no namespace declaration in the expression context.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. If the context node is from an HTML DOM, the default element namespace is "http://www.w3.org/1999/xhtml".

2. Otherwise, the default element namespace URI is null.

Note: This is equivalent to adding the default element namespace feature of XPath 2.0 to XPath 1.0, and using the HTML namespace as the default element namespace for HTML documents. It is motivated by the desire to have implementations be compatible with legacy HTML content while still supporting the changes that this specification introduces to HTML regarding the namespace used for HTML elements, and by the desire to use XPath 1.0 rather than XPath 2.0.



Note: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ve implementations be
uces to HTML

XSLT 1.0 pr
affected as f

ulting rule in XSLT 1.0) are

If the transformation program outputs an element in no namespace, the processor must, prior to constructing the corresponding DOM element node, change the namespace of the element to the HTML namespace, ASCII-lowercase the element's local name, and ASCII-lowercase the names of any non-namespaced attributes on the element.

Note: This requirement is a willful violation of the XSLT 1.0 specification, required because this specification changes the namespaces and case-sensitivity rules of HTML in a manner that would otherwise be incompatible with DOM-based XSLT transformations. (Processors that serialize the output are unaffected.) [XSLT10]

This specification does not specify precisely how XSLT processing interacts with the HTML parser infrastructure (for example, whether an XSLT processor acts as if it puts any elements into a stack of open elements). However, XSLT processors must stop parsing if they successfully complete, and must set the current document readiness first to "interactive" and then to "complete" if they are aborted.

This specification does not specify how XSLT interacts with the navigation algorithm, how it fits in with the event loop, nor how error pages are to be handled (e.g. whether XSLT errors are to replace an incremental XSLT output, or are rendered inline, etc).

Note: There are also additional non-normative comments regarding the interaction of XSLT and HTML in the script element section.

3.4 Dynamic markup insertion

Note: APIs for dynamically inserting markup into the document interact with the parser, and thus their behavior varies depending on whether they are used with HTML documents (and the HTML parser) or XHTML in XML documents (and the XML parser).

3.4.1 Opening the input stream

The `open()` method comes in several variants with different numbers of arguments.

This box is non-normative. Implementation requirements are given below this box.

`document = document . open([type [, replace]])`

Causes the `Document` to be replaced in-place, as if it was a new `Document` object, but reusing the previous object, which is then returned.

If the `type` argument is omitted or has the value "text/html", then the resulting `Document` has an HTML parser associated with it, which can be given data to parse using `document.write()`. Otherwise, all content passed to `document.write()` will be parsed as plain text.

If the `replace` argument is present and has the value "replace", the existing entries in the session history for the `Document` object are removed.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Works like the `window.open()` method.

Document objects have an **ignore-opens-during-unload counter**, which is used to prevent scripts from invoking the `document.open()` method (directly or indirectly) while the document is being unloaded. Initially, the counter must be set to zero.

When called with two or fewer arguments, the `document.open()` method must act as follows:

1. If the `Document` object is not flagged as an HTML document, throw an `InvalidStateError` exception and abort these steps.
2. If the `Document` object is flagged as an HTML document, then:
 3. Let `value` be the first argument passed to the `open()` method.
 4. Let `other` be the second argument passed to the `open()` method.
If `value` is the empty string, let `value` be `"replace"`, and let `other` be `false`.
 5. If the `Document` object's ignore-opens-during-unload counter is greater than zero, then the method does nothing. Abort these steps and return the `Document` object on which the method was invoked.

Note: This basically causes `document.open()` to be ignored when it's called in an inline script found during the parsing of data sent over the network, while still letting it have an effect when called asynchronously or on a document that is itself being spoon-fed using these APIs.

6. Similarly, if the `Document`'s ignore-opens-during-unload counter is greater than zero, then the method does nothing. Abort these steps and return the `Document` object on which the method was invoked.

Note: This basically causes `document.open()` to be ignored when it's called from a `beforeunload` pagehide, or `unload` event handler while the `Document` is being unloaded.

7. Release the storage mutex.
8. Set the `Document`'s `salvageable` state to `false`.
9. Prompt to unload the `Document` object. If the user refused to allow the document to be unloaded, then abort these steps and return the `Document` object on which the method was invoked.
10. Unload the `Document` object, with the `recycle` parameter set to `true`.
11. Abort the `Document`.
12. Unregister all event listeners registered on the `Document` node and its descendants.
13. Remove any tasks associated with the `Document` in any task source.
14. Remove all child nodes of the document, without firing any mutation events.
15. Replace the `Document`'s singleton objects with new instances of those objects. (This includes in particular the `Window`, `Location`, `History`, `ApplicationCache`, and `Navigator` objects, the various `BarProp` objects, the two `Storage` objects, the various `HTMLCollection` objects, and objects defined by other specifications, like `Selection` and the `document`'s `UndoManager`. It also includes all the Web IDL prototypes in the JavaScript binding, including the `Document` object's `prototype`.)
16. Change the document's character encoding to UTF-8.
17. If the `Document` is ready for post-load tasks, then set the `Document` object's `reload` override flag and set the `Document`'s `reload` override buffer to the empty string.
18. Set the `Document`'s `salvageable` state back to `true`.
19. Change the document's address to the entry script's document's address.
20. Create a new HTML parser and associate it with the document. This is a **script-created parser** (meaning that it can be closed

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

22. If the `type` string contains a U+003B SEMICOLON character (;), remove the first such character and all characters from it up to the end of the string.

23. Strip leading and trailing whitespace from `type`.

If `type` is *not* now an ASCII case-insensitive match for the string "text/html", then act as if the tokenizer had emitted a start tag token with the tag name "pre" followed by a single U+000A LINE FEED (LF) character, then switch the HTML parser's tokenizer to the PLAINTEXT state.

24. Remove all the entries in the browsing context's session history after the current entry current entry is the last entry in the

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

25. Rer bro objects in the top-level

26. Rer [//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

27. If `replace` is false, then add a new entry, just before the last entry, and associate with the new entry the text that was parsed by the previous parser associated with the `Document` object, as well as the state of the document at the start of these steps. This allows the user to step backwards in the session history to see the page before it was blown away by the `document.open()` call. This new entry does not have a `Document` object, so a new one will be created if the session history is traversed to that entry.

28. Finally, set the insertion point to point at just before the end of the input stream (which at this point will be empty).

29. Return the `Document` on which the method was invoked.

Note: The `document.open()` method does not affect whether a `Document` is ready for post-load tasks or completely loaded.

When called with three or more arguments, the `open()` method on the `Document` object must call the `open()` method on the `Window` object of the `Document` object, with the same arguments as the original call to the `open()` method, and return whatever that method returned. If the `Document` object has no `Window` object, then the method must throw an `InvalidAccessError` exception.

3.4.2 Closing the input stream

This box is non-normative. Implementation requirements are given below this box.

`document.close()`

Closes the input stream that was opened by the `document.open()` method.

Throws an `InvalidStateError` exception if the `Document` is an XML document.

The `close()` method must run the following steps:

1. If the `Document` object is not flagged as an HTML document, throw an `InvalidStateError` exception and abort these steps.
2. If there is no script-created parser associated with the document, then abort these steps.
3. Insert an explicit "EOF" character at the end of the parser's input stream.
4. If there is a pending parsing-blocking script, then abort these steps.
5. Run the tokenizer, processing resulting tokens as they are emitted, and stopping when the tokenizer reaches the explicit "EOF" character or spins the event loop.

3.4.3 `document.write()`

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

⚠Warning: This method has very idiosyncratic behavior. In some cases, this method can affect the state of the HTML parser while the parser is running, resulting in a DOM that does not correspond to the source of the document (e.g. if the string written is the string "<plaintext>" or "<!--"). In other cases, the call can clear the current page first, as if `document.open()` had been called. In yet more cases, the method is simply ignored, or throws an exception. To make matters worse, the exact behavior of this method can in some cases be dependent on network latency, which can lead to failures that are very hard to debug. For all these reasons, use of this method is strongly discouraged.

This method throws an `InvalidStateError` exception when invoked on XML documents.



Document code to prevent errors from occurring. The document object. If the document's ignore-opens-during-unload counter is greater than zero or the Document's ignore-destructive-writes counter is greater than zero, abort these steps. If the insertion point is undefined, call the `open()` method on the `document` object (with no arguments). If the user refused to allow the document to be unloaded, then abort these steps. Otherwise, the insertion point will point at just before the end of the (empty) input stream.

1. If the `document.write()` method was called from script executing inline (i.e. executing because the parser parsed a set of `script` tags), then this is a reentrant invocation of the parser.
2. If the insertion point is undefined and either the Document's ignore-opens-during-unload counter is greater than zero or the Document's ignore-destructive-writes counter is greater than zero, abort these steps.
3. If the insertion point is undefined, call the `open()` method on the `document` object (with no arguments). If the user refused to allow the document to be unloaded, then abort these steps. Otherwise, the insertion point will point at just before the end of the (empty) input stream.
4. Insert the string consisting of the concatenation of all the arguments to the method into the input stream just before the insertion point.
5. If the `Document` object's reload override flag is set, then append the string consisting of the concatenation of all the arguments to the method to the `Document`'s reload override buffer.
6. If there is no pending parsing-blocking script, have the HTML parser process the characters that were inserted, one at a time, processing resulting tokens as they are emitted, and stopping when the tokenizer reaches the insertion point or when the processing of the tokenizer is aborted by the tree construction stage (this can happen if a `script` end tag token is emitted by the tokenizer).

Note: If the `document.write()` method was called from script executing inline (i.e. executing because the parser parsed a set of `script` tags), then this is a reentrant invocation of the parser.

7. Finally, return from the method.

3.4.4 `document.writeln()`

This box is non-normative. Implementation requirements are given below this box.

`document.writeln(text...)`

Adds the given string(s) to the `Document`'s input stream, followed by a newline character. If necessary, calls the `open()` method implicitly first.

This method throws an `InvalidStateError` exception when invoked on XML documents.

The `document.writeln(...)` method, when invoked, must act as if the `document.write()` method had been invoked with the same argument(s), plus an extra argument consisting of a string containing a single line feed character (U+000A).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

4.1.1 The `html` element

Categories:

None.

Contexts in which this element can be used:

As the root element of a document.

Wherever a subdocument fragment is allowed in a compound document.

Content model:

A

Content: Too slow? Try reading the multipage copy of
 the spec instead: <https://web.archive.org>

DOM interface: /web/20130103030411/http://whatwg.org/html

The `html` element represents the root of an HTML document.

Authors are encouraged to specify a `lang` attribute on the root `html` element, giving the document's language. This aids speech synthesis tools to determine what pronunciations to use, translation tools to determine what rules to use, and so forth.

The `manifest` attribute gives the address of the document's application cache manifest, if there is one. If the attribute is present, the attribute's value must be a valid non-empty URL potentially surrounded by spaces.

The `manifest` attribute only has an effect during the early stages of document load. Changing the attribute dynamically thus has no effect (and thus, no DOM API is provided for this attribute).

Note: For the purposes of application cache selection, later base elements cannot affect the resolving of relative URLs in `manifest` attributes, as the attributes are processed before those elements are seen.

Note: The `window.applicationCache.IDL` attribute provides scripted access to the offline application cache mechanism.

The `html` element in the following example declares that the document's language is English.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Swapping Songs</title>
</head>
<body>
<h1>Swapping Songs</h1>
<p>Tonight I swapped some of the songs I wrote with some friends, who
gave me some of the songs they wrote. I love sharing my music.</p>
</body>
</html>
```

4.2 Document metadata

4.2.1 The `head` element

Categories:

None.

Contexts in which this element can be used:As the first element in an `html` element.**Content model:**

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Global attributes

DOM interface:

```
interface HTMLHeadElement : HTMLElement {};
```

The `head` element represents a collection of metadata for the Document.

The `cr`

short one:

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Here is an example of a longer one:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<BASE HREF="http://www.example.com/">
<TITLE>An application with a long head</TITLE>
<LINK REL="STYLESHEET" HREF="default.css">
<LINK REL="STYLESHEET ALTERNATE" HREF="big.css" TITLE="Big Text">
<SCRIPT SRC="support.js"></SCRIPT>
<META NAME="APPLICATION-NAME" CONTENT="Long headed application">
</HEAD>
<BODY>
...
```

Note: The `title` element is a required child in most situations, but when a higher-level protocol provides title information, e.g. in the Subject line of an e-mail when HTML is used as an e-mail authoring format, the `title` element can be omitted.

4.2.2 The `title` element

Categories:

Metadata content.

Contexts in which this element can be used:

In a `head` element containing no other `title` elements.

Content model:

Text.

Content attributes:

Global attributes

DOM interface:

```
IDL interface HTMLTitleElement : HTMLElement {
    attribute DOMString text;
};
```

The `title` element represents the document's title or name. Authors should use titles that identify their documents even when they are used out of context, for example in a user's history or bookmarks, or in search results. The document's title is often different from its first heading, since the first heading does not have to stand alone when taken out of context.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*head element's content model for a description of when the element is required.**This box is non-normative. Implementation requirements are given below this box.***title . text [= value]**

Returns the contents of the element, ignoring child nodes that aren't `Text` nodes.

Can be set, to replace the element's children with the given value.



The IDL attri
(ignoring an
IDL attribute
Here **Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html**

of the `title` element
way as the `textContent`

used on those same pages.

```
<h1>Introduction</h1>
<p>This companion guide to the highly successful
<cite>Introduction to Medieval Bee-Keeping</cite> book is...
```

The next page might be a part of the same site. Note how the title describes the subject matter unambiguously, while the first heading assumes the reader knows what the context is and therefore won't wonder if the dances are Salsa or Waltz:

```
<title>Dances used during bee mating rituals</title>
...
<h1>The Dances</h1>
```

The string to use as the document's title is given by the `document.title` IDL attribute.

User agents should use the document's title when referring to the document in their user interface. When the contents of a `title` element are used in this way, the directionality of that `title` element should be used to set the directionality of the document's title in the user interface.

4.2.3 The base element

Categories:

Metadata content.

Contexts in which this element can be used:

In a `head` element containing no other `base` elements.

Content model:

Empty.

Content attributes:

Global attributes
`href`
`target`

DOM interface:

```
IDL interface HTMLBaseElement : HTMLElement {
    attribute DOMString href;
    attribute DOMString target;
};
```

The `base` element allows authors to specify the document base URL for the purposes of resolving relative URLs, and the name of the default browsing context for the purposes of following hyperlinks. The element does not represent any content beyond this information.

There must be no more than one `base` element per document.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

URLs, except the `html` element (its `manifest` attribute isn't affected by `base` elements).

Note: If there are multiple `base` elements with `href` attributes, all but the first are ignored.

The `target` attribute, if specified, must contain a valid browsing context name or keyword, which specifies which browsing context is to be used as the default when hyperlinks and forms in the Document cause navigation.

A `base` element, if it has a `target` attribute, must come before any elements in the tree that represent hyperlinks.



Note: If

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. If the `href` IDL attribute is set, return the resulting absolute URL and abort these steps.
2. Let `url` be the value of the `href` IDL attribute.
3. Let `url` be the result of step 1.

4. Resolve `url` relative to `fallback base url` (thus, the `base href` attribute isn't affected by `xml:base` attributes or `base` elements).
5. If the previous step was successful, return the resulting absolute URL and abort these steps.
6. Otherwise, return the empty string.

The `href` IDL attribute, on setting, must set the `href` content attribute to the given new value.

The `target` IDL attribute must reflect the content attribute of the same name.

In this example, a `base` element is used to set the document base URL:

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is an example for the &lt;base&gt; element</title>
    <base href="http://www.example.com/news/index.html">
  </head>
  <body>
    <p>Visit the <a href="archives.html">archives</a>.</p>
  </body>
</html>
```

The link in the above example would be a link to "http://www.example.com/news/archives.html".

IDL

4.2.4 The link element

Categories:

Metadata content.

If the `itemprop` attribute is present: flow content.

If the `itemprop` attribute is present: phrasing content.

If the `itemprop` attribute is present: where phrasing content is expected.

Content model:

Empty.

Content attributes:

Global attributes

`href`

`crossorigin`

`rel`

`media`

`hreflang`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
interface HTMLLinkElement : HTMLElement {
    attribute boolean disabled;
    attribute DOMString href;
    attribute DOMString crossOrigin;
    attribute DOMString rel;
    readonly attribute DOMTokenList relList;
    attribute DOMString media;
    attribute DOMString hreflang;
```



Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The link el

The destinat

surrounded by spaces. If the `href` attribute is absent, then the element does not define a link.

.d non-empty URL potentially

A link element must have either a `rel` attribute or an `itemprop` attribute, but not both.

Note: *If the `rel` attribute is used, the element is restricted to the head element. When used with the `itemprop` attribute, the element can be used both in the head element and in the body of the page, subject to the constraints of the microdata model.*

The types of link indicated (the relationships) are given by the value of the `rel` attribute, which, if present, must have a value that is a set of space-separated tokens. The allowed keywords and their meanings are defined in a later section. If the `rel` attribute is absent, has no keywords, or if none of the keywords used are allowed according to the definitions in this specification, then the element does not create any links.

Two categories of links can be created using the `link` element: Links to external resources and hyperlinks. The link types section defines whether a particular link type is an external resource or a hyperlink. One `link` element can create multiple links (of which some might be external resource links and some might be hyperlinks); exactly which and how many links are created depends on the keywords given in the `rel` attribute. User agents must process the links on a per-link basis, not a per-element basis.

Note: *Each link created for a link element is handled separately. For instance, if there are two link elements with `rel="stylesheet"`, they each count as a separate external resource, and each is affected by its own attributes independently. Similarly, if a single link element has a `rel` attribute with the value `next stylesheet`, it creates both a hyperlink (for the `next` keyword) and an external resource link (for the `stylesheet` keyword), and they are affected by other attributes (such as `media` or `title`) differently.*

For example, the following `link` element creates two hyperlinks (to the same page):

```
<link rel="author license" href="/about">
```

The two links created by this element are one whose semantic is that the target page has information about the current page's author, and one whose semantic is that the target page has information regarding the license under which the current page is provided.

The `crossorigin` attribute is a CORS settings attribute. It is intended for use with external resource links.

The exact behavior for links to external resources depends on the exact relationship, as defined for the relevant link type. Some of the attributes control whether or not the external resource is to be applied (as defined below).

For external resources that are represented in the DOM (for example, style sheets), the DOM representation must be made available (modulo cross-origin restrictions) even if the resource is not applied. To obtain the resource, the user agent must run the following steps:

1. If the `href` attribute's value is the empty string, then abort these steps.
2. Resolve the URL given by the `href` attribute, relative to the element.
3. If the previous step fails, then abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

User agents may opt to only try to obtain such resources when they are needed, instead of proactively fetching all the external resources that are not applied.

The semantics of the protocol used (e.g. HTTP) must be followed when fetching external resources. (For example, redirects will be followed and 404 responses will cause the external resource to not be applied.)

Once the attempts to obtain the resource and its critical subresources are complete, the user agent must, if the loads were successful, queue a task to fire a simple event named `load` at the `link` element, or, if the resource or one of its critical subresources failed to completely load for any reason (e.g. DNS error, HTTP 404 response, a connection being promoted to closed, unsupported Content-Type), queue a task to fire a simple event named `error` at the `link` element, passing the resource or its `error` object as arguments.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The element `link` must be present in the document. If it is not, the user agent must queue a task to fire a simple event named `error` at the `link` element, passing the resource or its `error` object as arguments.

Interactive user agents may provide users with a means to follow the hyperlinks created using the `link` element, somewhere within their user interface. The exact interface is not defined by this specification, but it could include the following information (obtained from the element's attributes, again as defined below), in some form or another (possibly simplified), for each hyperlink created with each `link` element in the document:

- The relationship between this document and the resource (given by the `rel` attribute).
- The title of the resource (given by the `title` attribute).
- The address of the resource (given by the `href` attribute).
- The language of the resource (given by the `hreflang` attribute).
- The optimum media for the resource (given by the `media` attribute).

User agents could also include other information, such as the type of the resource (as given by the `type` attribute).

Note: *Hyperlinks created with the `link` element and its `rel` attribute apply to the whole page. This contrasts with the `rel` attribute of `a` and `area` elements, which indicates the type of a link whose context is given by the link's location within the document.*

The `media` attribute says which media the resource applies to. The value must be a valid media query.

If the link is a hyperlink then the `media` attribute is purely advisory, and describes for which media the document in question was designed.

However, if the link is an external resource link, then the `media` attribute is prescriptive. The user agent must apply the external resource when the `media` attribute's value matches the environment and the other relevant conditions apply, and must not apply it otherwise.

Note: *The external resource might have further restrictions defined within that limit its applicability. For example, a CSS style sheet might have some `@media` blocks. This specification does not override such further restrictions or requirements.*

The default, if the `media` attribute is omitted, is "all", meaning that by default links apply to all media.

The `hreflang` attribute on the `link` element has the same semantics as the `hreflang` attribute on `a` and `area` elements.

The `type` attribute gives the MIME type of the linked resource. It is purely advisory. The value must be a valid MIME type.

For external resource links, the `type` attribute is used as a hint to user agents so that they can avoid fetching resources they do not support. If the attribute is present, then the user agent must assume that the resource is of the given type (even if that is not a valid MIME type, e.g. the empty string). If the attribute is omitted, but the external resource link type has a default type defined, then the user agent must assume that the resource is of that type. If the UA does not support the given MIME type for the given link relationship, then the UA should not obtain the resource; if the UA does support the given MIME type for the given link relationship, then the UA should obtain the resource at the appropriate time as specified for the external resource link's particular type. If the attribute is omitted, and the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

resource, not the aforementioned assumed type.

If the external resource link type defines rules for processing the resource's Content-Type metadata, then those rules apply. Otherwise, if the resource is expected to be an image, user agents may apply the image sniffing rules, with the *official type* being the type determined from the resource's Content-Type metadata, and use the resulting sniffed type of the resource as if it was the actual type. Otherwise, if neither of these conditions apply or if the user agent opts not to apply the image sniffing rules, then the user agent must use the resource's Content-Type metadata to determine the type of the resource. If there is no type metadata, but the external resource link type has a default type defined, then the user agent must assume that the resource is of that type.



*data.

Once the us

the other rel

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

s of a supported type and

If a d /web/20130103030411/http://whatwg.org/html

<..

...then a compliant UA that supported only CSS style sheets would fetch the B and C files, and skip the A file (since text/plain is not the MIME type for CSS style sheets).

For files B and C, it would then check the actual types returned by the server. For those that are sent as text/css, it would apply the styles, but for those labeled as text/plain, or any other type, it would not.

If one of the two files was returned without a Content-Type metadata, or with a syntactically incorrect type like Content-Type: "null", then the default type for stylesheet links would kick in. Since that default type is text/css, the style sheet would nonetheless be applied.

The title attribute gives the title of the link. With one exception, it is purely advisory. The value is text. The exception is for style sheet links, where the title attribute defines alternative style sheet sets.

Note: The title attribute on link elements differs from the global title attribute of most other elements in that a link without a title does not inherit the title of the parent element: it merely has no title.

The sizes attribute is used with the icon link type. The attribute must not be specified on link elements that do not have a rel attribute that specifies the icon keyword.

HTTP Link: header fields, if supported, must be assumed to come before any links in the document, in the order that they were given in the HTTP message. These header fields are to be processed according to the rules given in the relevant specifications. [HTTP] [WEBLINK]

Note: Registration of relation types in HTTP Link: header fields is distinct from HTML link types, and thus their semantics can be different from same-named HTML types.

The IDL attributes href, rel, media, hreflang, type, and sizes each must reflect the respective content attributes of the same name.

The crossorigin IDL attribute must reflect the crossorigin content attribute.

The IDL attribute relList must reflect the rel content attribute.

The IDL attribute disabled only applies to style sheet links. When the link element defines a style sheet link, then the disabled attribute behaves as defined for the alternative style sheets DOM. For all other link elements it always return false and does nothing on setting.

The LinkStyle interface is also implemented by this element; the styling processing model defines how. [CSSOM]

Here, a set of link elements provide some style sheets:

```
<!-- a persistent style sheet -->
<link rel="stylesheet" href="default.css">
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<link rel="alternate stylesheet" href="big.css" title="Big fonts">
<link rel="alternate stylesheet" href="wide.css" title="Wide screen">
```

The following example shows how you can specify versions of the page that use alternative formats, are aimed at other languages, and that are intended for other media:

```
<link rel=alternate href="/en/html" hreflang=en type=text/html title="English HTML">
<link rel=alternate href="/fr/html" hreflang=fr type=text/html title="French HTML">
<link rel=alternate href="/en/html/print" hreflang=en type=text/html media=print
t:>
```

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

```
media=print
title="English PDF">
title="French PDF">
```

4.2.5 The `//whatwg.org/html`

Categories.

Metadata content.

If the `itemprop` attribute is present: flow content.

If the `itemprop` attribute is present: phrasing content.

Contexts in which this element can be used:

If the `charset` attribute is present, or if the element's `http-equiv` attribute is in the Encoding declaration state: in a `head` element.

If the `http-equiv` attribute is present but not in the Encoding declaration state: in a `head` element.

If the `http-equiv` attribute is present but not in the Encoding declaration state: in a `noscript` element that is a child of a `head` element.

If the `name` attribute is present: where metadata content is expected.

If the `itemprop` attribute is present: where metadata content is expected.

If the `itemprop` attribute is present: where phrasing content is expected.

Content model:

Empty.

Content attributes:

Global attributes

`name`

`http-equiv`

`content`

`charset`

DOM interface:

```
IDL interface HTMLMetaElement : HTMLElement {
    attribute DOMString name;
    attribute DOMString httpEquiv;
    attribute DOMString content;
};
```

The `meta` element represents various kinds of metadata that cannot be expressed using the `title`, `base`, `link`, `style`, and `script` elements.

The `meta` element can represent document-level metadata with the `name` attribute, pragma directives with the `http-equiv` attribute, and the file's character encoding declaration when an HTML document is serialized to string form (e.g. for transmission over the network or for disk storage) with the `charset` attribute.

Exactly one of the `name`, `http-equiv`, `charset`, and `itemprop` attributes must be specified.

If either `name`, `http-equiv`, or `itemprop` is specified, then the `content` attribute must also be specified. Otherwise, it must be omitted.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Note: The `charset` attribute on the `meta` element has no effect in XML documents, and is only allowed in order to facilitate migration to and from XHTML.

There must not be more than one `meta` element with a `charset` attribute per document.

The `content` attribute gives the value of the document metadata or pragma directive when the element is used for those purposes. The allowed values depend on the exact context, as described in subsequent sections of this specification.

If a `meta` element has a `name` attribute, it sets document metadata. Document metadata is expressed in terms of name-value pairs, the `name` attribute on the `meta` element giving the name, and the `content` attribute on the same element giving the value. The `name` specifies what the `meta` element represents, and the `content` attribute gives the value. The `name` in the following sections. If a `name` attribute is present, its value must be a string.

Too slow? Try reading the multipage copy of

the spec instead: [https://web.archive.org](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

/web/20130103030411/http://whatwg.org/html

4.2.5.1 `St:` <http://whatwg.org/html>

This specification defines a few names for the `name` attribute of the `meta` element.

Names are case-insensitive, and must be compared in an ASCII case-insensitive manner.

`application-name`

The value must be a short free-form string giving the name of the Web application that the page represents. If the page is not a Web application, the `application-name` metadata name must not be used. There must not be more than one `meta` element with its `name` attribute set to the value `application-name` per document. User agents may use the application name in UI in preference to the page's title, since the title might include status messages and the like relevant to the status of the page at a particular moment in time instead of just being the name of the application.

`author`

The value must be a free-form string giving the name of one of the page's authors.

`description`

The value must be a free-form string that describes the page. The value must be appropriate for use in a directory of pages, e.g. in a search engine. There must not be more than one `meta` element with its `name` attribute set to the value `description` per document.

`generator`

The value must be a free-form string that identifies one of the software packages used to generate the document. This value must not be used on pages whose markup is not generated by software, e.g. pages whose markup was written by a user in a text editor.

Here is what a tool called "Frontweaver" could include in its output, in the page's `head` element, to identify itself as the tool used to generate the page:

```
<meta name=generator content="Frontweaver 8.2">
```

`keywords`

The value must be a set of comma-separated tokens, each of which is a keyword relevant to the page.

This page about typefaces on British motorways uses a `meta` element to specify some keywords that users might use to look for the page:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Typefaces on UK motorways</title>
    <meta name="keywords" content="british,type face,font,fonts,highway,highways">
  </head>
  <body>
    ...
  </body>
</html>
```

Note: Many search engines do not consider such keywords, because this feature has historically been used

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

steps.

1. Let *keywords* be an empty list.
2. For each *meta* element with a *name* attribute and a *content* attribute and whose *name* attribute's value is *keywords*, run the following substeps:
 1. Split the value of the element's *content* attribute on commas.
 2. Add the resulting tokens, if any, to *keywords*.



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Use

) the page.

f the value.

I information of pages within aggregator that used this though the use of

||

4.2.5.2 Other metadata names

Extensions to the predefined set of metadata names may be registered in the WHATWG Wiki MetaExtensions page. [WHATWGWIKI]

Anyone is free to edit the WHATWG Wiki MetaExtensions page at any time to add a type. These new names must be specified with the following information:

Keyword

The actual name being defined. The name should not be confusingly similar to any other defined name (e.g. differing only in case).

Brief description

A short non-normative description of what the metadata name's meaning is, including the format the value is required to be in.

Specification

A link to a more detailed description of the metadata name's semantics and requirements. It could be another page on the Wiki, or a link to an external page.

Synonyms

A list of other names that have exactly the same processing requirements. Authors should not use the names defined to be synonyms, they are only intended to allow user agents to support legacy content. Anyone may remove synonyms that are not used in practice; only names that need to be processed as synonyms for compatibility with legacy content are to be registered in this way.

Status

One of the following:

Proposed

The name has not received wide peer review and approval. Someone has proposed it and is, or soon will be, using it.

Ratified

The name has received wide peer review and approval. It has a specification that unambiguously defines how to handle pages that use the name, including when they use it in incorrect ways.

Discontinued

The metadata name has received wide peer review and it has been found wanting. Existing pages are using this metadata name, but new pages should avoid it. The "brief description" and "specification" entries will give details of what authors should use instead, if anything.

If a metadata name is found to be redundant with existing values, it should be removed and listed as a synonym for the existing value.

If a metadata name is registered in the "proposed" state for a period of a month or more without being used or specified, then it may be removed from the registry.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Conformance checkers must use the information given on the WHATWG Wiki MetaExtensions page to establish if a value is allowed or not: values defined in this specification or marked as "proposed" or "ratified" must be accepted, whereas values marked as "discontinued" or not listed in either this specification or on the aforementioned page must be rejected as invalid. Conformance checkers may cache this information (e.g. for performance reasons or to avoid the use of unreliable network connectivity).

When an author uses a new metadata name not defined by either this specification or the Wiki page, conformance checkers should offer to add the value to the Wiki, with the details described above, with the "proposed" status.

Metadata names whose values are to be IRI s must not be proposed or accepted. Links must be represented using the link element, not the meta

Too slow? Try reading the multipage copy of

4.2.5.3 Pr: the spec instead: <https://web.archive.org>

When the h1 /web/20130103030411/http:

The http-equiv attribute in the first `<head>` is attribute. The states given in the last section are non-conforming, as noted in the last section.

State	Keyword	Notes
Content Language	content-language	Non-conforming
Encoding declaration	content-type	
Default style	default-style	
Refresh	refresh	
Cookie setter	set-cookie	Non-conforming

When a meta element is inserted into the document, if its http-equiv attribute is present and represents one of the above states, then the user agent must run the algorithm appropriate for that state, as described in the following list:

Content language state (http-equiv="content-language")

Note: This feature is non-conforming. Authors are encouraged to use the lang attribute instead.

This pragma sets the **pragma-set default language**. Until such a pragma is successfully processed, there is no pragma-set default language.

1. If the meta element has no content attribute, or if that attribute's value is the empty string, then abort these steps.
2. If the element's content attribute contains a U+002C COMMA character (,) then abort these steps.
3. Let *input* be the value of the element's content attribute.
4. Let *position* point at the first character of *input*.
5. Skip whitespace.
6. Collect a sequence of characters that are not space characters.
7. Set the pragma-set default language to the string that resulted from the previous step.

Note: This pragma is not exactly equivalent to the HTTP Content-Language header. [HTTP]

Encoding declaration state (http-equiv="content-type")

The Encoding declaration state is just an alternative form of setting the charset attribute: it is a character encoding declaration. This state's user agent requirements are all handled by the parsing section of the specification.

For meta elements with an http-equiv attribute in the Encoding declaration state, the content attribute must have a value that is an ASCII case-insensitive match for a string that consists of: the literal string "text/html;", optionally followed by any number of space characters, followed by the literal string "charset=", followed by the character encoding name of the character encoding declaration.

A document must not contain both a meta element with an http-equiv attribute in the Encoding declaration state and a meta element with the charset attribute present.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

1. If the `meta` element has no `content` attribute, or if that attribute's value is the empty string, then abort these steps.
2. Set the preferred style sheet set to the value of the element's `content` attribute. [CSSOM]

Refresh state (`http-equiv="refresh"`)

This pragma acts as timed redirect.

1. If another `meta` element with an `http-equiv` attribute in the Refresh state has already been successfully processed (at least one step), then abort these steps.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

6. Collect a sequence of characters that are ASCII digits, and parse the resulting string using the rules for parsing non-negative integers. If the sequence of characters collected is the empty string, then no number will have been parsed; abort these steps. Otherwise, let `time` be the parsed number.
7. Collect a sequence of characters that are ASCII digits and U+002E FULL STOP characters (.). Ignore any collected characters.
8. Skip whitespace.
9. Let `url` be the address of the current page.
10. If the character in `input` pointed to by `position` is a U+003B SEMICOLON (;), then advance `position` to the next character. Otherwise, jump to the last step.
11. Skip whitespace.
12. If the character in `input` pointed to by `position` is a U+0055 LATIN CAPITAL LETTER U character (U) or a U+0075 LATIN SMALL LETTER U character (u), then advance `position` to the next character. Otherwise, jump to the last step.
13. If the character in `input` pointed to by `position` is a U+0052 LATIN CAPITAL LETTER R character (R) or a U+0072 LATIN SMALL LETTER R character (r), then advance `position` to the next character. Otherwise, jump to the last step.
14. If the character in `input` pointed to by `position` is a U+004C LATIN CAPITAL LETTER L character (L) or a U+006C LATIN SMALL LETTER L character (l), then advance `position` to the next character. Otherwise, jump to the last step.
15. Skip whitespace.
16. If the character in `input` pointed to by `position` is a U+003D EQUALS SIGN (=), then advance `position` to the next character. Otherwise, jump to the last step.
17. Skip whitespace.
18. If the character in `input` pointed to by `position` is either a U+0027 APOSTROPHE character ('') or U+0022 QUOTATION MARK character (""), then let `quote` be that character, and advance `position` to the next character. Otherwise, let `quote` be the empty string.
19. Let `url` be equal to the substring of `input` from the character at `position` to the end of the string.
20. If `quote` is not the empty string, and there is a character in `url` equal to `quote`, then truncate `url` at that character, so that it and all subsequent characters are removed.
21. Strip any trailing space characters from the end of `url`.
22. Strip any U+0009 CHARACTER TABULATION (tab), U+000A LINE FEED (LF), and U+000D CARRIAGE RETURN (CR) characters from `url`.
23. Resolve the `url` value to an absolute URL, relative to the `meta` element. If this fails, abort these steps.
24. Perform one or more of the following steps:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

For the purposes of the previous paragraph, a refresh is said to have come due as soon as the *later* of the following two conditions occurs:

- At least *time* seconds have elapsed since the document has completely loaded, adjusted to take into account user or user agent preferences.
- At least *time* seconds have elapsed since the `meta` element was inserted into the Document, adjusted to take into account user or user agent preferences.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

For either

<https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

- just a valid non-negative integer, or
- a valid non-negative integer, followed by a U+003B SEMICOLON character (;), followed by one or more space characters, followed by a substring that is an ASCII case-insensitive match for the string "URL", followed by a U+003D EQUALS SIGN character (=), followed by a valid URL that does not start with a literal U+0027 APOSTROPHE ('') or U+0022 QUOTATION MARK ("") character.

In the former case, the integer represents a number of seconds before the page is to be reloaded; in the latter case the integer represents a number of seconds before the page is to be replaced by the page at the given URL.

A news organization's front page could include the following markup in the page's `head` element, to ensure that the page automatically reloads from the server every five minutes:

```
<meta http-equiv="Refresh" content="300">
```

A sequence of pages could be used as an automated slide show by making each page refresh to the next page in the sequence, using markup such as the following:

```
<meta http-equiv="Refresh" content="20; URL=page4.html">
```

Cookie setter (`http-equiv="set-cookie"`)

This pragma sets an HTTP cookie. [COOKIES]

It is non-conforming. Real HTTP headers should be used instead.

1. If the `meta` element has no `content` attribute, or if that attribute's value is the empty string, then abort these steps.
2. Obtain the storage mutex.
3. Act as if receiving a `set-cookie-string` for the document's address via a "non-HTTP" API, consisting of the value of the element's `content` attribute encoded as UTF-8. [COOKIES] [RFC3629]

There must not be more than one `meta` element with any particular state in the document at a time.

4.2.5.4 Other pragma directives

Extensions to the predefined set of pragma directives may, under certain conditions, be registered in the WHATWG Wiki PragmaExtensions page. [WHATWGWIKI]

Such extensions must use a name that is identical to an HTTP header registered in the Permanent Message Header Field Registry, and must have behavior identical to that described for the HTTP header. [IANAPERMHEADERS]

Pragma directives corresponding to headers describing metadata, or not requiring specific user agent processing, must not be registered; instead, use metadata names. Pragma directives corresponding to headers that affect the HTTP processing model (e.g. caching) must not be registered, as they would result in HTTP-level behavior being different for user agents that implement HTML than for user agents that do not.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Brief description

A short non-normative description of the purpose of the pragma directive.

Specification

A link to the specification defining the corresponding HTTP header.

Conformance checkers must use the information given on the WHATWG Wiki PragmaExtensions page to establish if a value is allowed or not: values defined in this specification or listed on the aforementioned page must be accepted, whereas values not listed in either this specification or performed for performance must be rejected. Cache this information (e.g.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

4.2.5.5 Sp A character /web/20130103030411/http:

The following document is specified.

- The character encoding name given must be the name of the character encoding used to serialize the file.
- The value must be a valid character encoding name, and must be an ASCII case-insensitive match for the preferred MIME name for that encoding. [IANACHARSET]
- The character encoding declaration must be serialized without the use of character references or character escapes of any kind.
- The element containing the character encoding declaration must be serialized completely within the first 1024 bytes of the document.

In addition, due to a number of restrictions on `meta` elements, there can only be one meta-based character encoding declaration per document.

If an HTML document does not start with a BOM, and its encoding is not explicitly given by Content-Type metadata, and the document is not an `iframe srcdoc` document, then the character encoding used must be an ASCII-compatible character encoding, and the encoding must be specified using a `meta` element with a `charset` attribute or a `meta` element with an `http-equiv` attribute in the Encoding declaration state.

Note: A character encoding declaration is required (either in the Content-Type metadata or explicitly in the file) even if the encoding is US-ASCII, because a character encoding is needed to process non-ASCII characters entered by the user in forms, in URLs generated by scripts, and so forth.

If the document is an `iframe srcdoc` document, the document must not have a character encoding declaration. (In this case, the source is already decoded, since it is part of the document that contained the `iframe`.)

If an HTML document contains a `meta` element with a `charset` attribute or a `meta` element with an `http-equiv` attribute in the Encoding declaration state, then the character encoding used must be an ASCII-compatible character encoding.

Authors are encouraged to use UTF-8. Conformance checkers may advise authors against using legacy encodings. [RFC3629]

Authoring tools should default to using UTF-8 for newly-created documents. [RFC3629]

Encodings in which a series of bytes in the range 0x20 to 0x7E can encode characters other than the corresponding characters in the range U+0020 to U+007E represent a potential security vulnerability: a user agent that does not support the encoding (or does not support the label used to declare the encoding, or does not use the same mechanism to detect the encoding of unlabelled content as another user agent) might end up interpreting technically benign plain text content as HTML tags and JavaScript. For example, this applies to encodings in which the bytes corresponding to "<script>" in ASCII can encode a different string. Authors should not use such encodings, which are known to include JIS_C6226-1983, JIS_X0212-1990, HZ-GB-2312, JOHAB (Windows code page 1361), encodings based on ISO-2022, and encodings based on EBCDIC. Furthermore, authors must not use the CESU-8, UTF-7, BOCU-1 and SCSU encodings, which also fall into this category, because these encodings were never intended for use for Web content. [RFC1345] [RFC1842] [RFC1468] [RFC2237] [RFC1554] [CP50220] [RFC1922] [RFC1557] [CESU8] [UTF7] [BOCU1] [SCSU]

Authors should not use UTF-32, as the encoding detection algorithms described in this specification intentionally do not distinguish it from UTF-16. [UNICODE]

Note: Using non-UTF-8 encodings can have unexpected results on form submission and URL encodings, which use

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

document (in the head element):

<meta charset="utf-8">

In XML, the XML declaration would be used instead, at the very top of the markup:

<?xml version="1.0" encoding="utf-8"?>

4.2.6 The style element



Category: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

Contexts: /web/20130103030411/http://whatwg.org/html

If the element is in an element whitespace, and not as the child of an element whose content model is transparent.

Content model:

Depends on the value of the type attribute, but must match requirements described in prose below.

Content attributes:

Global attributes

media

type

scoped

Also, the title attribute has special semantics on this element.

DOM interface:

```
IDL interface HTMLStyleElement : HTMLElement {
    attribute boolean disabled;
    attribute DOMString media;
    attribute DOMString type;
    attribute boolean scoped;
};

HTMLStyleElement implements LinkStyle;
```

The style element allows authors to embed style information in their documents. The style element is one of several inputs to the styling processing model. The element does not represent content for the user.

The type attribute gives the styling language. If the attribute is present, its value must be a valid MIME type that designates a styling language. The charset parameter must not be specified. The default value for the type attribute, which is used if the attribute is absent, is "text/css". [RFC2318]

When examining types to determine if they support the language, user agents must not ignore unknown MIME parameters — types with unknown parameters must be assumed to be unsupported. The charset parameter must be treated as an unknown parameter for the purpose of comparing MIME types here.

The media attribute says which media the styles apply to. The value must be a valid media query. The user agent must apply the styles when the media attribute's value matches the environment and the other relevant conditions apply, and must not apply them otherwise.

Note: The styles might be further limited in scope, e.g. in CSS with the use of @media blocks. This specification does not override such further restrictions or requirements.

The default, if the media attribute is omitted, is "all", meaning that by default styles apply to all media.

The scoped attribute is a boolean attribute. If present, it indicates that the styles are intended just for the subtree rooted at the style element's parent element, as opposed to the whole Document.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Note: This implies that only one scoped style element is allowed at a time, and that such elements cannot be children of, e.g., a or ins elements, even when those are used as flow content containers.

If the `scoped` attribute is present, then the user agent must apply the specified style information only to the `style` element's parent element (if any), and that element's descendants. Otherwise, the specified styles must, if applied, be applied to the entire document.

The following will eventually be moved to a CSS specification; it is specified here only on an interim basis until an editor can be found to own this.



Within scope Too slow? Try reading the multipage copy of follows.
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

global : GI ; //whatwg.org/html

The following rule set is defined:

```
B           b|\\"0{0,4}(42|62)(\r\n|[ \t\r\n\f])?
@{G}{L}{O}{B}{A}{L} {return GLOBAL_SYM;}
```

Simple selectors in rule sets prefixed by the `@global` `@-rule` in scoped CSS resources must be processed in the same way as normal rule sets in non-scoped CSS resources.

Simple selectors in scoped CSS resources that are not prefixed by an `@global` `@-rule` must only match the `style` element's parent element (if any), and that element's descendants.

For scoped CSS resources, the effect of other `@-rules` must be scoped to either the scoped sheet and its subresources or to the subtree rooted at the `style` element's parent (if any), even if the `@-rule` in question would ordinarily apply to all style sheets that affect the Document, or to all nodes in the Document. Any `@page` rules in scoped CSS resources must be ignored.

For example, an '`@font-face`' rule defined in a scoped style sheet would only define the font for the purposes of elements in the scoped section; the font would not be used for elements outside the subtree. However, rules outside the subtree that refer to font family names declared in '`@font-face`' rules in a scoped section, when those rules are inherited by nodes in the scoped section, would end up referring to the fonts declared in that section.

The `title` attribute on `style` elements defines alternative style sheet sets. If the `style` element has no `title` attribute, then it has no title; the `title` attribute of ancestors does not apply to the `style` element. [CSSOM]

Note: The `title` attribute on `style` elements, like the `title` attribute on `link` elements, differs from the global `title` attribute in that a `style` block without a `title` does not inherit the `title` of the parent element: it merely has no title.

The `textContent` of a `style` element must match the `style` production in the following ABNF, the character set for which is Unicode. [ABNF]

```
style      = no-c-start *( c-start no-c-end c-end no-c-start )
no-c-start = <any string that doesn't contain a substring that matches c-start >
c-start    = "<!--"
no-c-end   = <any string that doesn't contain a substring that matches c-end >
c-end     = "-->"
```

All descendant elements must be processed, according to their semantics, before the `style` element itself is evaluated. For styling languages that consist of pure text (as opposed to XML), user agents must evaluate `style` elements by passing the concatenation of the contents of all the `Text` nodes that are children of the `style` element (not any other nodes such as comments or elements), in tree order, to the style system. For XML-based styling languages, user agents must pass all the child nodes of the `style` element to the style system.

All URLs found by the styling language's processor must be resolved, relative to the element (or as defined by the styling language), when the processor is invoked.

Once the attempts to obtain the style sheet's critical subresources, if any, are complete, or, if the style sheet has no critical

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

Subresources (e.g. CSS parse errors, PNG decoding errors) are not failures for the purposes of this paragraph.

The task source for these tasks is the DOM manipulation task source.

The element must delay the load event of the element's document until all the attempts to obtain the style sheet's critical subresources, if any, are complete.

Note: This specification does not specify a style system, but CSS is expected to be supported by most Web browsers. [CSS]

The `media`,The `disabled`The `LinkStyle`

Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org

/web/20130103030411/http:

//whatwg.org/html

<

```
<html lang="en-US">
  <head>
    <title>My favorite book</title>
    <style>
      body { color: black; background: white; }
      em { font-style: normal; color: red; }
    </style>
  </head>
  <body>
    <p>My <em>favorite</em> book of all time has <em>got</em> to be
       <cite>A Cat's Life</cite>. It is a book by P. Rahmel that talks
       about the <i lang="la">Felis Catus</i> in modern human society.</p>
  </body>
</html>
```

me.

[CSSOM]

le leaving titles of works and
of documents.

4.2.7 Styling

The `link` and `style` elements can provide styling information for the user agent to use when rendering the document. The CSS and CSSOM specifications specify what styling information is to be used by the user agent and how it is to be used. [CSS] [CSSOM]

The `style` and `link` elements implement the `LinkStyle` interface. [CSSOM]

For `style` elements, if the user agent does not support the specified styling language, then the `sheet` attribute of the element's `LinkStyle` interface must return null. Similarly, `link` elements that do not represent external resource links that contribute to the styling processing model (i.e. that do not have a `stylesheet` keyword in their `rel` attribute), for which the link is an alternative stylesheet but whose `title` content attribute is absent or empty, or whose resource is CORS-cross-origin, must have their `LinkStyle` interface's `sheet` attribute return null.

Otherwise, the `LinkStyle` interface's `sheet` attribute must return null if the corresponding element is not in a `Document`, and otherwise must return a `StyleSheet` object with the following properties: [CSSOM]

The style sheet type

The style sheet type must be the same as the style's specified type. For `style` elements, this is the same as the `type` content attribute's value, or `text/css` if that is omitted. For `link` elements, this is the Content-Type metadata of the specified resource.

The style sheet location

For `link` elements, the location must be the result of resolving the URL given by the element's `href` content attribute, relative to the element, or the empty string if that fails. For `style` elements, there is no location.

The style sheet media

The media must be the same as the value of the element's `media` content attribute, or the empty string, if the attribute is omitted.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The style sheet alternate flag

For link elements, true if the link is an alternative stylesheet. In all other cases, false.

The same object must be returned each time.

The disabled IDL attribute on link and style elements must return false and do nothing on setting, if the sheet attribute of their LinkStyle interface is null. Otherwise, it must return the value of the StyleSheet interface's disabled attribute on getting, and forward the new value to that same attribute on setting.

The rules for'

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

When a style was an inter available to rendering" style

↳ Link: header, or some

ed no other resources (e.g. it
synchronously made
ches its "update the

A style sheet in the context of the Document of an HTML parser or XML parser is said to be a **style sheet that is blocking scripts** if the element was created by that Document's parser, and the element is either a style element or a link element that was an external resource link that contributes to the styling processing model when the element was created by the parser, and the element's style sheet was enabled when the element was created by the parser, and the element's style sheet ready flag is not yet set, and, the last time the event loop reached step 1, the element was in that Document, and the user agent hasn't given up on that particular style sheet yet. A user agent may give up on a style sheet at any time.

Note: Giving up on a style sheet before the style sheet loads, if the style sheet eventually does still load, means that the script might end up operating with incorrect information. For example, if a style sheet sets the color of an element to green, but a script that inspects the resulting style is executed before the sheet is loaded, the script will find that the element is black (or whatever the default color is), and might thus make poor choices (e.g. deciding to use black as the color elsewhere on the page, instead of green). Implementors have to balance the likelihood of a script using incorrect information with the performance impact of doing nothing while waiting for a slow network request to finish.

A Document has a **style sheet that is blocking scripts** if there is either a style sheet that is blocking scripts in the context of that Document, or if that Document is in a browsing context that has a parent browsing context, and the active document of that parent browsing context itself has a style sheet that is blocking scripts.

A Document has **no style sheet that is blocking scripts** if it does not have a style sheet that is blocking scripts as defined in the previous paragraph.

4.3 Scripting

Scripts allow authors to add interactivity to their documents.

Authors are encouraged to use declarative alternatives to scripting where possible, as declarative mechanisms are often more maintainable, and many users disable scripting.

For example, instead of using script to show or hide a section to show more details, the details element could be used.

Authors are also encouraged to make their applications degrade gracefully in the absence of scripting support.

For example, if an author provides a link in a table header to dynamically resort the table, the link could also be made to function without scripts by requesting the sorted table from the server.

IDL

4.3.1 The script element

Categories:

- Metadata content.
- Flow content.
- Phrasing content.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If there is no `src` attribute, depends on the value of the `type` attribute, but must match script content restrictions.

If there *is* a `src` attribute, the element must be either empty or contain only script documentation that also matches script content restrictions.

Content attributes:

Global attributes

`src`

`type`

`charset`

DOM inter

a d c **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```
attribute DOMString charset;
attribute boolean async;
attribute boolean defer;
attribute DOMString crossOrigin;
attribute DOMString text;
};
```

The `script` element allows authors to include dynamic script and data blocks in their documents. The element does not represent content for the user.

When used to include dynamic scripts, the scripts may either be embedded inline or may be imported from an external file using the `src` attribute. If the language is not that described by "text/javascript", then the `type` attribute must be present, as described below. Whatever language is used, the contents of the `script` element must conform with the requirements of that language's specification.

When used to include data blocks (as opposed to scripts), the data must be embedded inline, the format of the data must be given using the `type` attribute, the `src` attribute must not be specified, and the contents of the `script` element must conform to the requirements defined for the format used.

The `type` attribute gives the language of the script or format of the data. If the attribute is present, its value must be a valid MIME type. The `charset` parameter must not be specified. The default, which is used if the attribute is absent, is "text/javascript".

The `src` attribute, if specified, gives the address of the external script resource to use. The value of the attribute must be a valid non-empty URL potentially surrounded by spaces identifying a script resource of the type given by the `type` attribute, if the attribute is present, or of the type "text/javascript", if the attribute is absent. A resource is a script resource of a given type if that type identifies a scripting language and the resource conforms with the requirements of that language's specification.

The `charset` attribute gives the character encoding of the external script resource. The attribute must not be specified if the `src` attribute is not present. If the attribute is set, its value must be a valid character encoding name, must be an ASCII case-insensitive match for the preferred MIME name for that encoding, and must match the encoding given in the `charset` parameter of the Content-Type metadata of the external file, if any. [IANACHARSET]

The `async` and `defer` attributes are boolean attributes that indicate how the script should be executed. The `defer` and `async` attributes must not be specified if the `src` attribute is not present.

There are three possible modes that can be selected using these attributes. If the `async` attribute is present, then the script will be executed asynchronously, as soon as it is available. If the `async` attribute is not present but the `defer` attribute is present, then the script is executed when the page has finished parsing. If neither attribute is present, then the script is fetched and executed immediately, before the user agent continues parsing the page.

Note: The exact processing details for these attributes are, for mostly historical reasons, somewhat non-trivial, involving a number of aspects of HTML. The implementation requirements are therefore by necessity scattered throughout the specification. The algorithms below (in this section) describe the core of this processing, but these algorithms reference and are referenced by the parsing rules for script start and end tags in HTML, in foreign

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

The `crossorigin` attribute is a CORS settings attribute. It controls, for scripts that are obtained from other origins, whether error information will be exposed.

Changing the `src`, `type`, `charset`, `async`, `defer`, and `crossorigin` attributes dynamically has no direct effect; these attribute are only used at specific times described below.

A `script` element has several associated pieces of state.

The first is a flag indicating whether or not the script block has been "**already started**". Initially, `script` elements must have this flag unset (script blocks, when created, are not "already started").

Too slow? Try reading the multipage copy of

The second set by the `H` tag on the `script` element must have this flag unset. It is unset by those elements.

The third is the `HTML` parser's flag on the `script` element whose "force-async"

The fourth is the `HTML` parser's flag set. It is unset by the `script` element whose "force-async"

The fourth is a flag indicating whether or not the script block is "**ready to be parser-executed**". Initially, `script` elements must have this flag unset (script blocks, when created, are not "ready to be parser-executed"). This flag is used only for elements that are also "parser-inserted", to let the parser know when to execute the script.

The last few pieces of state are ***the script block's type***, ***the script block's character encoding***, and ***the script block's fallback character encoding***. They are determined when the script is prepared, based on the attributes on the element at that time, and the Document of the `script` element.

When a `script` element that is not marked as being "parser-inserted" experiences one of the events listed in the following list, the user agent must synchronously prepare the `script` element:

- The `script` element gets inserted into a document, at the time the node is inserted according to the DOM, after any other `script` elements inserted at the same time that are earlier in the Document in tree order.
- The `script` element is in a Document and a node or document fragment is inserted into the `script` element, after any `script` elements inserted at that time.
- The `script` element is in a Document and has a `src` attribute set where previously the element had no such attribute.

To **prepare a script**, the user agent must act as follows:

1. If the `script` element is marked as having "already started", then the user agent must abort these steps at this point. The script is not executed.
2. If the element has its "parser-inserted" flag set, then set `was-parser-inserted` to true and unset the element's "parser-inserted" flag. Otherwise, set `was-parser-inserted` to false.

Note: *This is done so that if parser-inserted script elements fail to run when the parser tries to run them, e.g. because they are empty or specify an unsupported scripting language, another script can later mutate them and cause them to run again.*

3. If `was-parser-inserted` is true and the element does not have an `async` attribute, then set the element's "force-async" flag to true.

Note: *This is done so that if a parser-inserted script element fails to run when the parser tries to run it, but it is later executed after a script dynamically updates it, it will execute asynchronously even if the `async` attribute isn't set.*

4. If the element has no `src` attribute, and its child nodes, if any, consist only of comment nodes and empty `Text` nodes, then the user agent must abort these steps at this point. The script is not executed.
5. If the element is not in a Document, then the user agent must abort these steps at this point. The script is not executed.
6. If either:
 - the `script` element has a `type` attribute and its value is the empty string, or

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Otherwise, if the `script` element has a `type` attribute, let *the script block's type* for this `script` element be the value of that attribute with any leading or trailing sequences of space characters removed.

Otherwise, if the element has a non-empty `language` attribute; let *the script block's type* for this `script` element be the concatenation of the string "text/" followed by the value of the `language` attribute.

Note: *The language attribute is never conforming, and is always ignored if there is a type attribute present.*

7. If the `script` element has a `scriptType` attribute, then let *the script block's type* be the value of that attribute.
- use **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**
8. If we are in a `script` element, then set its "force-async" flag to `true`.
9. The `script` element is flagged as "parser-inserted", but the element's `Document` is not the `Document` of the parser that created the element, then abort these steps.
10. If scripting is disabled for the `script` element, then the user agent must abort these steps at this point. The script is not executed.

Note: *The definition of scripting is disabled means that, amongst others, the following scripts will not execute: scripts in XMLHttpRequest's responseXML documents, scripts in DOMParser-created documents, scripts in documents created by XMLHttpRequest's transformToDocument feature, and scripts that are first inserted by a script into a Document that was created using the createDocument() API. [XHR] [DOMPARSING] [DOM]*

12. If the `script` element has an `event` attribute and a `for` attribute, then run these substeps:
 1. Let `for` be the value of the `for` attribute.
 2. Let `event` be the value of the `event` attribute.
 3. Strip leading and trailing whitespace from `event` and `for`.
 4. If `for` is not an ASCII case-insensitive match for the string "window", then the user agent must abort these steps at this point. The script is not executed.
 5. If `event` is not an ASCII case-insensitive match for either the string "onload" or the string "onload()", then the user agent must abort these steps at this point. The script is not executed.
13. If the `script` element has a `charset` attribute, then let *the script block's character encoding* for this `script` element be the encoding given by the `charset` attribute.

Otherwise, let *the script block's fallback character encoding* for this `script` element be the same as the encoding of the document itself.

Note: *Only one of these two pieces of state is set.*

14. If the element has a `src` content attribute, run these substeps:
 1. Let `src` be the value of the element's `src` attribute.
 2. If `src` is the empty string, queue a task to fire a simple event named `error` at the element, and abort these steps.
 3. Resolve `src` relative to the element.
 4. If the previous step failed, queue a task to fire a simple event named `error` at the element, and abort these steps.
 5. Do a potentially CORS-enabled fetch of the resulting absolute URL, with the `mode` being the state of the element's `crossorigin` content attribute, the `origin` being the origin of the `script` element's `Document`, and the `default origin`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

For historical reasons, if the URL is a `javascript:` URL, then the user agent must not, despite the requirements in the definition of the fetching algorithm, actually execute the script in the URL; instead the user agent must act as if it had received an empty HTTP 400 response.

For performance reasons, user agents may start fetching the script (as defined above) as soon as the `src` attribute is set, instead, in the hope that the element will be inserted into the document (and that the `crossorigin` attribute won't change value in the meantime). Either way, once the element is inserted into the document, the load must have started as described in this step. If the UA performs such prefetching, but the element is never inserted in the document, or the `src` attribute is dynamically changed, or the `crossorigin` attribute is dynamically changed, then the document has been effectively wasted.

15. The `Too slow? Try reading the multipage copy of`

- ↳ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the document has been flagged as
the document has finished

must set the element's "ready to be parser-executed" flag. The parser will handle executing the script.

- ↳ If the element has a `src` attribute, and the element has been flagged as "parser-inserted", and the element does not have an `async` attribute

The element is the pending parsing-blocking script of the `Document` of the parser that created the element. (There can only be one such script per `Document` at a time.)

The task that the networking task source places on the task queue once the fetching algorithm has completed must set the element's "ready to be parser-executed" flag. The parser will handle executing the script.

- ↳ If the element does not have a `src` attribute, and the element has been flagged as "parser-inserted", and either the parser that created the `script` is an XML parser or it's an HTML parser whose script nesting level is not greater than one, and the `Document` of the HTML parser or XML parser that created the `script` element has a style sheet that is blocking scripts

The element is the pending parsing-blocking script of the `Document` of the parser that created the element. (There can only be one such script per `Document` at a time.)

Set the element's "ready to be parser-executed" flag. The parser will handle executing the script.

- ↳ If the element has a `src` attribute, does not have an `async` attribute, and does not have the "force-async" flag set

The element must be added to the end of the **list of scripts that will execute in order as soon as possible** associated with the `Document` of the `script` element at the time the prepare a script algorithm started.

The task that the networking task source places on the task queue once the fetching algorithm has completed must run the following steps:

1. If the element is not now the first element in the list of scripts that will execute in order as soon as possible to which it was added above, then mark the element as ready but abort these steps without executing the script yet.
2. **Execution:** Execute the script block corresponding to the first script element in this list of scripts that will execute in order as soon as possible.
3. Remove the first element from this list of scripts that will execute in order as soon as possible.
4. If this list of scripts that will execute in order as soon as possible is still not empty and the first entry has already been marked as ready, then jump back to the step labeled **execution**.

- ↳ If the element has a `src` attribute

The element must be added to the **set of scripts that will execute as soon as possible** of the `Document` of the `script` element at the time the prepare a script algorithm started.

The task that the networking task source places on the task queue once the fetching algorithm has completed must execute the script block and then remove the element from the set of scripts that will execute as soon as possible.

Go

NOV JAN FEB



2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

The **pending parsing-blocking script** of a Document is used by the Document's parser(s).

Note: If a `script` element that blocks a parser gets moved to another `Document` before it would normally have stopped blocking that parser, it nonetheless continues blocking that parser until the condition that causes it to be blocking the parser no longer applies (e.g. if the `script` is a pending parsing-blocking `script` because there was a style sheet that is blocking scripts when it was parsed, but then the `script` is moved to another `Document` before the style sheet loads, the `script` still blocks the parser until the style sheets are all loaded, at which time the `script` executes and the parser is unblocked)

When the user says "Too slow? Try reading the multipage copy of..."

1. If there's a newer version of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
 2. Just ↳ <http://whatwg.org/html>

the parser that created the

Executing the script block must just consist of firing a simple event named `error` at the element.

↳ If the load was successful

Executing the script block must consist of running the following steps. For the purposes of these steps, the script is considered to be from an *external file* if, while the prepare a script algorithm above was running for this script, the `script` element had a `src` attribute specified.

1. Initialize ***the script block's source*** as follows:

↪ If the script is from an external file and the script block's type is a text-based language

The contents of that file, interpreted as a Unicode string, are the script source.

To obtain the Unicode string, the user agent run the following steps:

1. If the resource's Content Type metadata, if any, specifies a character encoding, and the user agent supports that encoding, then let *character encoding* be that encoding, and jump to the bottom step in this series of steps.
 2. If the algorithm above set the *script block's character encoding*, then let *character encoding* be that encoding, and jump to the bottom step in this series of steps.
 3. For each of the rows in the following table, starting with the first one and going down, if the file has as many or more bytes available than the number of bytes in the first column, and the first bytes of the file match the bytes given in the first column, then set *character encoding* to the encoding given in the cell in the second column of that row, and jump to the bottom step in this series of steps:

Bytes in Hexadecimal	Encoding
FE FF	Big-endian UTF-16
FF FE	Little-endian UTF-16
EF BB BF	UTF-8

Note: This step looks for Unicode Byte Order Marks (BOMs).

4. Let *character encoding* be the *script block's fallback character encoding*.
 5. Convert the file to Unicode using *character encoding*, following the rules for doing so given by the specification for the *script block's type*.

↳ If the script is from an external file and the script block's type is an XML-based language

The external file is the script source. When it is later executed, it must be interpreted in a manner consistent with the specification defining the language given by *the script block's type*.

↳ If the script is inline and *the script block's type* is a text-based language

The value of the `text` IDL attribute at the time the element's "already started" flag was last

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

2. Fire a simple event named `beforescriptexecute` that bubbles and is cancelable at the `script` element.

If the event is canceled, then abort these steps.

3. If the script is from an external file, then increment the ignore-destructive-writes counter of the `script` element's `Document`. Let *neutralized doc* be that `Document`.

4. Let *old script element* be the value to which the `Document` element's `currentScript` object was most

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ement.

xe, the URL from which the

he resource was CORS-Je algorithm.

Note: This is where the script is compiled and actually executed.

7. Initialize the `Document` element's `currentScript` object to *old script element*.

8. Decrement the ignore-destructive-writes counter of *neutralized doc*, if it was incremented in the earlier step.

9. Fire a simple event named `afterscriptexecute` that bubbles (but is not cancelable) at the `script` element.

10. If the script is from an external file, fire a simple event named `load` at the `script` element.

Otherwise, the script is internal; queue a task to fire a simple event named `load` at the `script` element.

The IDL attributes `src`, `type`, `charset`, `defer`, each must reflect the respective content attributes of the same name.

The `crossOrigin` IDL attribute must reflect the `crossorigin` content attribute.

The `async` IDL attribute controls whether the element will execute asynchronously or not. If the element's "force-async" flag is set, then, on getting, the `async` IDL attribute must return true, and on setting, the "force-async" flag must first be unset, and then the content attribute must be removed if the IDL attribute's new value is false, and must be set to the empty string if the IDL attribute's new value is true. If the element's "force-async" flag is not set, the IDL attribute must reflect the `async` content attribute.

This box is non-normative. Implementation requirements are given below this box.

`script.text [= value]`

Returns the contents of the element, ignoring child nodes that aren't `Text` nodes.

Can be set, to replace the element's children with the given value.

The IDL attribute `text` must return a concatenation of the contents of all the `Text` nodes that are children of the `script` element (ignoring any other nodes such as comments or elements), in tree order. On setting, it must act the same way as the `textContent` IDL attribute.

Note: When inserted using the `document.write()` method, `script` elements execute (typically synchronously), but when inserted using `innerHTML` and `outerHTML` attributes, they do not execute at all.

In this example, two `script` elements are used. One embeds an external script, and the other includes some data.

```
<script src="game-engine.js"></script>
<script type="text/x-game-map">
.....U.....
o.....A....
....A....AAA....
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

used by the site's search engine to help users who are looking for particular features in their game maps.

The following sample shows how a `script` element can be used to define a function that is then used by other parts of the document. It also shows how a `script` element can be used to invoke script while the document is being parsed, in this case to initialize the form's output.

```
<script>
  function calculate(form) {
    var price = 52000;
    ...
  }
</script>
<form name="pricecalc" onsubmit="return false" onchange="calculate(this)">
  <fieldset>
    <legend>Work out the price of your car</legend>
    <p>Base cost: £52000.</p>
    <p>Select additional options:</p>
    <ul>
      <li><label><input type=checkbox name=brakes> Ceramic brakes (£1000)</label></li>
      <li><label><input type=checkbox name=radio> Satellite radio (£2500)</label></li>
      <li><label><input type=checkbox name=turbo> Turbo charger (£5000)</label></li>
      <li><label><input type=checkbox name=sticker> "XZ" sticker (£250)</label></li>
    </ul>
    <p>Total: £<output name=result></output></p>
  </fieldset>
  <script>
    calculate(document.forms.pricecalc);
  </script>
</form>
```

4.3.1.1 Scripting languages

A user agent is said to **support the scripting language** if each component of *the script block's type* is an ASCII case-insensitive match for the corresponding component in the MIME type string of a scripting language that the user agent implements.

The following lists the MIME type strings that user agents must recognize, and the languages to which they refer:

```
"application/ecmascript"
"application/javascript"
"application/x-ecmascript"
"application/x-javascript"
"text/ecmascript"
"text/javascript"
"text/javascript1.0"
"text/javascript1.1"
"text/javascript1.2"
"text/javascript1.3"
"text/javascript1.4"
"text/javascript1.5"
"text/jscript"
"text/livescript"
"text/x-ecmascript"
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The following MIME types (with or without parameters) must not be interpreted as scripting languages:

- "text/plain"
- "text/xml"
- "application/octet-stream"
- "application/xml"

Note: These types are explicitly listed here because they are poorly-defined types that are nonetheless likely to be used as formats for data blocks, and it would be problematic if they were suddenly interpreted as script by a user agent.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

When examined to be compared

IME parameters. Types are

Note: For script tags, any of the

4.3.1.2 Restrictions for contents of script elements

The `textContent` of a `script` element must match the `script` production in the following ABNF, the character set for which is Unicode. [ABNF]

```

script      = data1 *( escape [ script-start data3 ] "-->" data1 ) [ escape ]
escape      = "<!--" data2 *( script-start data3 script-end data2 )

data1       = <any string that doesn't contain a substring that matches not-data1>
not-data1   = "<!--"

data2       = <any string that doesn't contain a substring that matches not-data2>
not-data2   = script-start / "-->"

data3       = <any string that doesn't contain a substring that matches not-data3>
not-data3   = script-end / "-->"

script-start = lt      s c r i p t tag-end
script-end   = lt slash s c r i p t tag-end

lt          = %x003C ; U+003C LESS-THAN SIGN character (<)
slash        = %x002F ; U+002F SOLIDUS character (/)

s           = %x0053 ; U+0053 LATIN CAPITAL LETTER S
s           = / %x0073 ; U+0073 LATIN SMALL LETTER S
c           = %x0043 ; U+0043 LATIN CAPITAL LETTER C
c           = / %x0063 ; U+0063 LATIN SMALL LETTER C
r           = %x0052 ; U+0052 LATIN CAPITAL LETTER R
r           = / %x0072 ; U+0072 LATIN SMALL LETTER R
i           = %x0049 ; U+0049 LATIN CAPITAL LETTER I
i           = / %x0069 ; U+0069 LATIN SMALL LETTER I
p           = %x0050 ; U+0050 LATIN CAPITAL LETTER P
p           = / %x0070 ; U+0070 LATIN SMALL LETTER P
t           = %x0054 ; U+0054 LATIN CAPITAL LETTER T
t           = / %x0074 ; U+0074 LATIN SMALL LETTER T

tag-end     = %x0009 ; U+0009 CHARACTER TABULATION (tab)
tag-end     = / %x000A ; U+000A LINE FEED (LF)
tag-end     = / %x000C ; U+000C FORM FEED (FF)
tag-end     = / %x0020 ; U+0020 SPACE
tag-end     = / %x002F ; U+002F SOLIDUS (/)
tag-end     = / %x003E ; U+003E GREATER-THAN SIGN (>)

```

When a `script` element contains script documentation, there are further restrictions on the contents of the element, as described in the section below.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
documentation = *( *( space / tab / comment ) [ line-comment ] newline )
comment       = slash star *( not-star / star not-slash ) 1*star slash
line-comment  = slash slash *not-newline
```

```
; characters
tab          = %x0009 ; U+0009 CHARACTER TABULATION (tab)
newline      = %x000A ; U+000A LINE FEED (LF)
space        = %x0020 ; U+0020 SPACE
```

star
slash
not-ne
not-st
not-sl

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



Note: This corresponds to putting the contents of the element in JavaScript comments.

Note: This requirement is in addition to the earlier restrictions on the syntax of contents of script elements.

This allows authors to include documentation, such as license information or API information, inside their documents while still referring to external script files. The syntax is constrained so that authors don't accidentally include what looks like valid script while also providing a `src` attribute.

```
<script src="cool-effects.js">
  // create new instances using:
  //   var e = new Effect();
  // start the effect using .play, stop using .stop:
  //   e.play();
  //   e.stop();
</script>
```

4.3.1.4 Interaction of script elements and XSLT

This section is non-normative.

This specification does not define how XSLT interacts with the `script` element. However, in the absence of another specification actually defining this, here are some guidelines for implementors, based on existing implementations:

- When an XSLT transformation program is triggered by an `<?xml-stylesheet?>` processing instruction and the browser implements a direct-to-DOM transformation, `script` elements created by the XSLT processor need to be marked "parser-inserted" and run in document order (modulo scripts marked `defer` or `async`), asynchronously while the transformation is occurring.
- The `XSLTProcessor.transformToDocument()` method adds elements to a `Document` that is not in a browsing context, and, accordingly, any `script` elements they create need to have their "already started" flag set in the prepare a script algorithm and never get executed (scripting is disabled). Such `script` elements still need to be marked "parser-inserted", though, such that their `async` IDL attribute will return `false` in the absence of an `async` content attribute.
- The `XSLTProcessor.transformToFragment()` method needs to create a fragment that is equivalent to one built manually by creating the elements using `document.createElementNS()`. For instance, it needs to create `script` elements that aren't "parser-inserted" and that don't have their "already started" flag set, so that they will execute when the fragment is inserted into a document.

The main distinction between the first two cases and the last case is that the first two operate on `Documents` and the last operates on a fragment.

4.3.2 The noscript element

Categories:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

In a `head` element of an HTML document, if there are no ancestor `noscript` elements.Where phrasing content is expected in HTML documents, if there are no ancestor `noscript` elements.**Content model:**

When scripting is disabled, in a `head` element: in any order, zero or more `link` elements, zero or more `style` elements, and zero or more `meta` elements.

When scripting is disabled, not in a `head` element: transparent, but there must be no `noscript` element descendants.

Otherwise: text that conforms to the requirements given in the prose.

**Content after:**

Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

DOM interface
The noscript element's present different from the XML parser.

disabled. It is used to direct how the document is

When used in HTML documents, the allowed content model is as follows:

In a head element, if scripting is disabled for the noscript element

The `noscript` element must contain only `link`, `style`, and `meta` elements.

In a head element, if scripting is enabled for the noscript element

The `noscript` element must contain only text, except that invoking the HTML fragment parsing algorithm with the `noscript` element as the `context` element and the text contents as the `input` must result in a list of nodes that consists only of `link`, `style`, and `meta` elements that would be conforming if they were children of the `noscript` element, and no parse errors.

Outside of head elements, if scripting is disabled for the noscript element

The `noscript` element's content model is transparent, with the additional restriction that a `noscript` element must not have a `noscript` element as an ancestor (that is, `noscript` can't be nested).

Outside of head elements, if scripting is enabled for the noscript element

The `noscript` element must contain only text, except that the text must be such that running the following algorithm results in a conforming document with no `noscript` elements and no `script` elements, and such that no step in the algorithm causes an HTML parser to flag a parse error:

1. Remove every `script` element from the document.
2. Make a list of every `noscript` element in the document. For every `noscript` element in that list, perform the following steps:
 1. Let the `parent element` be the parent element of the `noscript` element.
 2. Take all the children of the `parent element` that come before the `noscript` element, and call these elements `the before children`.
 3. Take all the children of the `parent element` that come after the `noscript` element, and call these elements `the after children`.
 4. Let `s` be the concatenation of all the `Text` node children of the `noscript` element.
 5. Set the `innerHTML` attribute of the `parent element` to the value of `s`. (This, as a side-effect, causes the `noscript` element to be removed from the document.)
 6. Insert `the before children` at the start of the `parent element`, preserving their original relative order.
 7. Insert `the after children` at the end of the `parent element`, preserving their original relative order.

Note: All these contortions are required because, for historical reasons, the noscript element is handled differently by the HTML parser based on whether scripting was enabled or not when the parser was invoked.

The `noscript` element must not be used in XML documents.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

The `noscript` element has no other requirements. In particular, children of the `noscript` element are not exempt from form submission, scripting, and so forth, even when scripting is enabled for the element.

In the following example, a `noscript` element is used to provide fallback for a script.

```
<form action="calcSquare.php">
<p>
  <label for=x>Number</label>:
  <input id="x" name="x" type="number">
</p>
</form>

</script>
<noscript>
  <input type=submit value="Calculate Square">
</noscript>
</form>
```

When script is disabled, a button appears to do the calculation on the server side. When script is enabled, the value is computed on-the-fly instead.

The `noscript` element is a blunt instrument. Sometimes, scripts might be enabled, but for some reason the page's script might fail. For this reason, it's generally better to avoid using `noscript`, and to instead design the script to change the page from being a scriptless page to a scripted page on the fly, as in the next example:

```
<form action="calcSquare.php">
<p>
  <label for=x>Number</label>:
  <input id="x" name="x" type="number">
</p>
<input id="submit" type=submit value="Calculate Square">
<script>
  var x = document.getElementById('x');
  var output = document.createElement('p');
  output.textContent = 'Type a number; it will be squared right then!';
  x.form.appendChild(output);
  x.form.onsubmit = function () { return false; }
  x.oninput = function () {
    var v = x.valueAsNumber;
    output.textContent = v + ' squared is ' + v * v;
  };
  var submit = document.getElementById('submit');
  submit.parentNode.removeChild(submit);
</script>
</form>
```

The above technique is also useful in XHTML, since `noscript` is not supported in the XHTML syntax.

4.4 Sections

IDL

4.4.1 The `body` element

Categories:

Sectioning root.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Content attributes:

Global attributes

onafterprint

onbeforeprint

onbeforeunload

onblur

onerror

onfocus

onfullscreenchange

onfullscreenerror

o Too slow? Try reading the multipage copy of

o the spec instead: <https://web.archive.org/web/20130103030411/http://www.whatwg.org/html>

o /web/20130103030411/http://whatwg.org/html

o.

onpageshow

onpopstate

onresize

onscroll

onstorage

onunload

DOM interface:

```
interface HTMLBodyElement : HTMLElement {
    attribute EventHandler onafterprint;
    attribute EventHandler onbeforeprint;
    attribute EventHandler onbeforeunload;
    // attribute EventHandler onblur;
    attribute OnErrorEventHandler onerror;
    // attribute EventHandler onfocus;
    attribute EventHandler onfullScreenchange;
    attribute EventHandler onfullScreenerror;
    attribute EventHandler onhashchange;
    // attribute EventHandler onload;
    attribute EventHandler onmessage;
    attribute EventHandler onoffline;
    attribute EventHandler ononline;
    attribute EventHandler onpopstate;
    attribute EventHandler onpagehide;
    attribute EventHandler onpageshow;
    attribute EventHandler onresize;
    // attribute EventHandler onscroll;
    attribute EventHandler onstorage;
    attribute EventHandler onunload;
};
```

The `body` element represents the main content of the document.

In conforming documents, there is only one `body` element. The `document.body` IDL attribute provides scripts with easy access to a document's `body` element.

Note: Some DOM operations (for example, parts of the drag and drop model) are defined in terms of "the body element". This refers to a particular element in the DOM, as per the definition of the term, and not any arbitrary `body` element.

The `body` element exposes as event handler content attributes a number of the event handlers of the `Window` object. It also mirrors their event handler IDL attributes.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

`onerror` event handler content attributes of that element, then that of the root `html` element, and only *then* would it trigger the `onerror` event handler content attribute on the `body` element. This is because the event would bubble from the target, to the `body`, to the `html`, to the `Document`, to the `Window`, and the event handler on the `body` is watching the `Window` not the `body`. A regular event listener attached to the `body` using `addEventListener()`, however, would be run when the event bubbled through the `body` and not when it reaches the `Window` object.

This page updates an indicator to show whether or not the user is online:

```
<!DOCTYPE HTML>
<
  Too slow? Try reading the multipage copy of
  the spec instead: https://web.archive.org
  /web/20130103030411/http:
  //whatwg.org/html

</head>
<body ononline="update(true)"
      onoffline="update(false)"
      onload="update(navigator.onLine)">
  <p>You are: <span id="status">(Unknown)</span></p>
</body>
</html>
```



4.4.2 The `article` element

Categories:

- Flow content.
- Sectioning content.
- Palpable content.

Contexts in which this element can be used:

- Where flow content is expected.

Content model:

- Flow content.

Content attributes:

- Global attributes

DOM interface:

- Uses `HTMLElement`.

The `article` element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

When `article` elements are nested, the inner `article` elements represent articles that are in principle related to the contents of the outer article. For instance, a blog entry on a site that accepts user-submitted comments could represent the comments as `article` elements nested within the `article` element for the blog entry.

Author information associated with an `article` element (q.v. the `address` element) does not apply to nested `article` elements.

Note: When used specifically with content to be redistributed in syndication, the `article` element is similar in purpose to the `entry` element in Atom. [ATOM]

Note: The schema.org microdata vocabulary can be used to provide the publication date for an `article` element, using one of the `CreativeWork` subtypes.

When the main content of the page (i.e. excluding footers, headers, navigation blocks, and sidebars) is all one single self-contained

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<article itemscope itemtype="http://schema.org/BlogPosting">
  <header>
    <h1 itemprop="headline">The Very First Rule of Life</h1>
    <p><time itemprop="datePublished" datetime="2009-10-09">3 days ago</time></p>
    <link itemprop="url" href="?comments=0">
  </header>
  <p>If there's a microphone anywhere near you, assume it's hot and
  sending whatever you're saying to the world. Seriously.</p>
  <p>...</p>
```



**Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

Here
[/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
<p><time itemprop="datePublished" datetime="2009-10-09">3 days ago</time></p>
<link itemprop="url" href="?comments=0">
</header>
<p>If there's a microphone anywhere near you, assume it's hot and
  sending whatever you're saying to the world. Seriously.</p>
<p>...</p>
<section>
  <h1>Comments</h1>
  <article itemprop="comment" itemscope itemtype="http://schema.org/UserComments" id="c1">
    <link itemprop="url" href="#c1">
    <footer>
      <p>Posted by: <span itemprop="creator" itemscope itemtype="http://schema.org/Person">
        <span itemprop="name">George Washington</span>
      </span></p>
      <p><time itemprop="commentTime" datetime="2009-10-10">15 minutes ago</time></p>
    </footer>
    <p>Yeah! Especially when talking about your lobbyist friends!</p>
  </article>
  <article itemprop="comment" itemscope itemtype="http://schema.org/UserComments" id="c2">
    <link itemprop="url" href="#c2">
    <footer>
      <p>Posted by: <span itemprop="creator" itemscope itemtype="http://schema.org/Person">
        <span itemprop="name">George Hammond</span>
      </span></p>
      <p><time itemprop="commentTime" datetime="2009-10-10">5 minutes ago</time></p>
    </footer>
    <p>Hey, you have the same first name as me.</p>
  </article>
  </section>
</article>
```

Notice the use of `footer` to give the information for each comment (such as who wrote it and when): the `footer` element can appear at the start of its section when appropriate, such as in this case. (Using `header` in this case wouldn't be wrong either; it's mostly a matter of authoring preference.)

4.4.3 The `section` element

Categories:

- Flow content.
- Sectioning content.
- Palpable content.

Contexts in which this element can be used:

- Where flow content is expected.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

DOM interface:Uses `HTMLElement`.

The `section` element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading.

Examples of sections would be chapters, the various tabbed pages in a tabbed dialog box, or the numbered sections of a thesis. A Web site's home page could be split into sections for an introduction, news items, and contact information.

Note: A sense of **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

Note: T or as a section **/web/20130103030411/http://whatwg.org/html**

*en it would make
'y for styling purposes
general rule is that the
document's outline.*

In the following example, we see an `article` (part of a larger Web page, about apples), containing two short sections.

```
<article>
  <hgroup>
    <h1>Apples</h1>
    <h2>Tasty, delicious fruit!</h2>
  </hgroup>
  <p>The apple is the pomaceous fruit of the apple tree.</p>
  <section>
    <h1>Red Delicious</h1>
    <p>These bright red apples are the most common found in many
      supermarkets.</p>
  </section>
  <section>
    <h1>Granny Smith</h1>
    <p>These juicy, green apples make a great filling for
      apple pies.</p>
  </section>
</article>
```

Notice how the use of `section` means that the author can use `h1` elements throughout, without having to worry about whether a particular section is at the top level, the second level, the third level, and so on.

Here is a graduation programme with two sections, one for the list of people graduating, and one for the description of the ceremony. (The markup in this example features an uncommon style sometimes used to minimize the amount of inter-element whitespace.)

```
<!DOCTYPE Html>
<Html
  ><Head
    ><Title
      >Graduation Ceremony Summer 2022</Title
    ></Head
  ><Body
    ><H1
      >Graduation</H1
    ><Section
      ><H1
        >Ceremony</H1
      ><P
        >Opening Procession</P
      ><P
        >Speech by Validactorian</P
      ><P
        >Speech by Class President</P
      ><P
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
><H1
    >Graduates</H1
><Ul
    ><Li
        >Molly Carpenter</Li
    ><Li
        >Anastasia Luccio</Li
    ><Li
        >Ebenezer McCove</Li
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

></Html>

In this example, a book author has marked up some sections as chapters and some as appendices, and uses CSS to style the headers in these two classes of section differently. The whole book is wrapped in an `article` element as part of an even larger document containing other books.

```
<article class="book">
    <style>
        section { border: double medium; margin: 2em; }
        section.chapter h1 { font: 2em Roboto, Helvetica Neue, sans-serif; }
        section.appendix h1 { font: small-caps 2em Roboto, Helvetica Neue, sans-serif; }
    </style>
    <header>
        <hgroup>
            <h1>My Book</h1>
            <h2>A sample with not much content</h2>
        </hgroup>
        <p><small>Published by Dummy Publicorp Ltd.</small></p>
    </header>
    <section class="chapter">
        <h1>My First Chapter</h1>
        <p>This is the first of my chapters. It doesn't say much.</p>
        <p>But it has two paragraphs!</p>
    </section>
    <section class="chapter">
        <h1>It Continutes: The Second Chapter</h1>
        <p>Bla dee bla, dee bla dee bla. Boom.</p>
    </section>
    <section class="chapter">
        <h1>Chapter Three: A Further Example</h1>
        <p>It's not like a battle between brightness and earthtones would go unnoticed.</p>
        <p>But it might ruin my story.</p>
    </section>
    <section class="appendix">
        <h1>Appendix A: Overview of Examples</h1>
        <p>These are demonstrations.</p>
    </section>
    <section class="appendix">
        <h1>Appendix B: Some Closing Remarks</h1>
        <p>Hopefully this long example shows that you <em>can</em> style sections, so long as they are used to indicate actual sections.</p>
    </section>
</article>
```

Go

NOV JAN FEB

03

2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

About this capture

Palpable content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Flow content.

Content attributes:

Global attributes



DOM interface Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

The nav element

tion with navigation links.

Note:

that contains

pages or

sufficient for such cases; while a nav element can be used in such cases, it is usually unnecessary.

intended for sections
list of links to various
other elements alone is

Note: User agents (such as screen readers) that are targeted at users who can benefit from navigation information being omitted in the initial rendering, or who can benefit from navigation information being immediately available, can use this element as a way to determine what content on the page to initially skip or provide on request (or both).

In the following example, the page has several places where links are present, but only one of those places is considered a navigation section.

```
<body itemscope itemtype="http://schema.org/Blog">
  <header>
    <h1>Wake up sheeple!</h1>
    <p><a href="news.html">News</a> - 
       <a href="blog.html">Blog</a> -
       <a href="forums.html">Forums</a></p>
    <p>Last Modified: <span itemprop="dateModified">2009-04-01</span></p>
    <nav>
      <h1>Navigation</h1>
      <ul>
        <li><a href="articles.html">Index of all articles</a></li>
        <li><a href="today.html">Things sheeple need to wake up for today</a></li>
        <li><a href="successes.html">Sheeple we have managed to wake</a></li>
      </ul>
    </nav>
  </header>
  <div>
    <article itemprop="blogPosts" itemscope itemtype="http://schema.org/BlogPosting">
      <header>
        <h1 itemprop="headline">My Day at the Beach</h1>
      </header>
      <div itemprop="articleBody">
        <p>Today I went to the beach and had a lot of fun.</p>
        ...more content...
      </div>
      <footer>
        <p>Posted <time itemprop="datePublished" datetime="2009-10-10">Thursday</time>. </p>
      </footer>
    </article>
    ...more blog posts...
  </div>
  <footer>
    <p>Copyright ©
      <span itemprop="copyrightYear">2010</span>
      <span itemprop="copyrightHolder">The Example Company</span>
    </p>
  </footer>

```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

</body>

Notice the `div` elements being used to wrap all the contents of the page other than the header and footer, and all the contents of the blog entry other than its header and footer.

You can also see microdata annotations in the above example that use the schema.org vocabulary to provide the publication date and other metadata about the blog post.

In the following example, there are two `nav` elements, one for primary navigation around the site, and one for secondary navigation.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
</nav>
<article>
  <header>
    <h1>Demos in Exampland</h1>
    <p>Written by A. N. Other.</p>
  </header>
  <nav>
    <ul>
      <li><a href="#public">Public demonstrations</a></li>
      <li><a href="#destroy">Demolitions</a></li>
      ...more...
    </ul>
  </nav>
  <div>
    <section id="public">
      <h1>Public demonstrations</h1>
      <p>...more...</p>
    </section>
    <section id="destroy">
      <h1>Demolitions</h1>
      <p>...more...</p>
    </section>
    ...more...
  </div>
  <footer>
    <p><a href="?edit">Edit</a> | <a href="?delete">Delete</a> | <a href="?Rename">Rename</a></p>
  </footer>
</article>
<footer>
  <p><small>© copyright 1998 Exampland Emperor</small></p>
</footer>
</body>
```

A `nav` element doesn't have to contain a list, it can contain other kinds of content as well. In this navigation block, links are provided in prose:

```
<nav>
  <h1>Navigation</h1>
  <p>You are on my home page. To the north lies <a href="/blog">my blog</a>, from whence the sounds of battle can be heard. To the east you can see a large mountain, upon which many <a href="/school">school papers</a> are littered. Far up thus mountain you can spy a little figure who appears to be me, desperately scribbling a <a href="/school/thesis">thesis</a>. </p>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

page. Cobwebs cover its disused entrance, and at one point you see a rat run quickly out of the page.</p>

</nav>

4.4.5 The aside element

Categories:

Flow content



S

F Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org>

Contexts

V

Content I /web/20130103030411/http:

F //whatwg.org/html

Content a.

Global attributes

DOM interface:

Uses HTMLElement.

The `aside` element represents a section of a page that consists of content that is tangentially related to the content around the `aside` element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.

The element can be used for typographical effects like pull quotes or sidebars, for advertising, for groups of `nav` elements, and for other content that is considered separate from the main content of the page.

Note: It's not appropriate to use the `aside` element just for parentheticals, since those are part of the main flow of the document.

The following example shows how an `aside` is used to mark up background material on Switzerland in a much longer news story on Europe.

```
<aside>
  <h1>Switzerland</h1>
  <p>Switzerland, a land-locked country in the middle of geographic Europe, has not joined the geopolitical European Union, though it is a signatory to a number of European treaties.</p>
</aside>
```

The following example shows how an `aside` is used to mark up a pull quote in a longer article.

...

```
<p>He later joined a large company, continuing on the same work.
<q>I love my job. People ask me what I do for fun when I'm not at work. But I'm paid to do my hobby, so I never know what to answer. Some people wonder what they would do if they didn't have to work... but I know what I would do, because I was unemployed for a year, and I filled that time doing exactly what I do now.</q></p>
```

<aside>

```
  <q> People ask me what I do for fun when I'm not at work. But I'm paid to do my hobby, so I never know what to answer. </q>
</aside>
```

```
<p>Of course his work – or should that be hobby? – isn't his only passion. He also enjoys other pleasures.</p>
```

...

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
<p>My tagline</p>
</header>
<aside>
  <!-- this aside contains two sections that are tangentially related
  to the page, namely, links to other blogs, and links to blog posts
  from this blog -->
<nav>
  <h1>My blogroll</h1>
  <ul>
```



Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
</nav>
</aside>
<aside>
  <!-- this aside is tangentially related to the page also, it
  contains twitter messages from the blog author -->
  <h1>Twitter Feed</h1>
  <blockquote cite="http://twitter.example.net/t31351234">
    I'm on vacation, writing my blog.
  </blockquote>
  <blockquote cite="http://twitter.example.net/t31219752">
    I'm going to go on vacation soon.
  </blockquote>
</aside>
<article>
  <!-- this is a blog post -->
  <h1>My last post</h1>
  <p>This is my last post.</p>
  <footer>
    <p><a href="/last-post" rel=bookmark>Permalink</a>
  </footer>
</article>
<article>
  <!-- this is also a blog post -->
  <h1>My first post</h1>
  <p>This is my first post.</p>
  <aside>
    <!-- this aside is about the blog post, since it's inside the
    <article> element; it would be wrong, for instance, to put the
    blogroll here, since the blogroll isn't really related to this post
    specifically, only to the page as a whole -->
    <h1>Posting</h1>
    <p>While I'm thinking about it, I wanted to say something about
    posting. Posting is fun!</p>
  </aside>
  <footer>
    <p><a href="/first-post" rel=bookmark>Permalink</a>
  </footer>
</article>
<footer>
  <nav>
    <a href="/archives">Archives</a> -
    <a href="/about">About me</a> -
    <a href="/copyright">Copyright</a>
  </nav>
</footer>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Categories:

- Flow content.
- Heading content.
- Palpable content.

Contexts in which this element can be used:

- As a child of an `hgroup` element.
- Where flow content is expected.

Content model:

F Too slow? Try reading the multipage copy of

Content: C the spec instead: <https://web.archive.org>

DOM interface: /web/20130103030411/http://whatwg.org/html



These elements represent headings for their sections.

The semantics and meaning of these elements are defined in the section on headings and sections.

These elements have a **rank** given by the number in their name. The `h1` element is said to have the highest rank, the `h6` element has the lowest rank, and two elements with the same name have equal rank.

As far as their respective document outlines (their heading and section structures) are concerned, these two snippets are semantically equivalent:

```
<body>
<h1>Let's call it a draw(ing surface)</h1>
<h2>Diving in</h2>
<h2>Simple shapes</h2>
<h2>Canvas coordinates</h2>
<h3>Canvas coordinates diagram</h3>
<h2>Paths</h2>
</body>

<body>
<h1>Let's call it a draw(ing surface)</h1>
<section>
  <h1>Diving in</h1>
</section>
<section>
  <h1>Simple shapes</h1>
</section>
<section>
  <h1>Canvas coordinates</h1>
</section>
  <h1>Canvas coordinates diagram</h1>
</section>
<section>
  <h1>Paths</h1>
</section>
</body>
```

Authors might prefer the former style for its terseness, or the latter style for its convenience in the face of heavy editing; which is best is purely an issue of preferred authoring style.

4.4.7 The `hgroup` element

Categories:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Where flow content is expected.

Content model:One or more `h1`, `h2`, `h3`, `h4`, `h5`, and `h6` elements.**Content attributes:**

Global attributes

DOM interface:Uses `HTMLElement`

The `hgroup` has multiple other elements. The rank of such elements defines how `hgroup` elements are assigned to individual sections.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The section on `headings` and `sections` defines how `hgroup` elements are assigned to individual sections.

Here are some examples of valid headings.

```
<hgroup>
  <h1>The reality dysfunction</h1>
  <h2>Space is not the only void</h2>
</hgroup>

<hgroup>
  <h1>Dr. Strangelove</h1>
  <h2>Or: How I Learned to Stop Worrying and Love the Bomb</h2>
</hgroup>
```

The point of using `hgroup` in these examples is to mask the `h2` element (which acts as a secondary title) from the outline algorithm.

How a user agent exposes such multi-level headings in user interfaces (e.g. in tables of contents or search results) is left open to implementors, as it is a user interface issue. The first example above could be rendered as:

The reality dysfunction: Space is not the only void

Alternatively, it could look like this:

The reality dysfunction (Space is not the only void)

In interfaces where a title can be rendered on multiple lines, it could be rendered as follows, maybe with the first line in a bigger font size:

The reality dysfunction
Space is not the only void

4.4.8 The `header` element

Categories:

Flow content.
Palpable content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:Flow content, but with no `header` or `footer` element descendants.**Content attributes:**

Global attributes

DOM interface:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Note: A header element is intended to usually contain the section's heading (an h1–h6 element or an hgroup element), but this is not required. The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos.

Here are some sample headers. This first one is for a game:

```
<header>
<p>Welcome to...</p>
<h1>Voidwars!</h1>
</h1>
```



The 1 Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<ac><this version='</ac>
<dd><a href="http://www.w3.org/TR/2004/WD-SVG12-20041027/">http://www.w3.org/TR/2004/WD-SVG12-20041027/</a></dd>
<dt>Previous version:</dt>
<dd><a href="http://www.w3.org/TR/2004/WD-SVG12-20040510/">http://www.w3.org/TR/2004/WD-SVG12-20040510/</a></dd>
<dt>Latest version of SVG 1.2:</dt>
<dd><a href="http://www.w3.org/TR/SVG12/">http://www.w3.org/TR/SVG12/</a></dd>
<dt>Latest SVG Recommendation:</dt>
<dd><a href="http://www.w3.org/TR/SVG/">http://www.w3.org/TR/SVG/</a></dd>
<dt>Editor:</dt>
<dd>Dean Jackson, W3C, <a href="mailto:dean@w3.org">dean@w3.org</a></dd>
<dt>Authors:</dt>
<dd>See <a href="#authors">Author List</a></dd>
</dl>
<p class="copyright"><a href="http://www.w3.org/Consortium/Legal/ipr-notice...">http://www.w3.org/Consortium/Legal/ipr-notice...</a></p>
</header>
```

Note: The header element is not sectioning content; it doesn't introduce a new section.

In this example, the page has a page heading given by the h1 element, and two subsections whose headings are given by h2 elements. The content after the header element is still part of the last subsection started in the header element, because the header element doesn't take part in the outline algorithm.

```
<body>
<header>
<h1>Little Green Guys With Guns</h1>
<nav>
<ul>
<li><a href="/games">Games</a>
<li><a href="/forum">Forum</a>
<li><a href="/download">Download</a>
</ul>
</nav>
<h2>Important News</h2> <!-- this starts a second subsection -->
<!-- this is part of the subsection entitled "Important News" -->
<p>To play today's games you will need to update your client.</p>
<h2>Games</h2> <!-- this starts a third subsection -->
</header>
<p>You have three active games:</p>
<!-- this is still part of the subsection entitled "Games" -->
...
```

4.4.9 The footer element

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Visible content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Flow content, but with no header or footer element descendants.

Content attributes:

Global attributes

**DOM interface:**

`l Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
/web/20130103030411/http:
When the footer
and other su
//whatwg.org/html`

tent. A footer typically
the like.

verbose license agreements,

Note: Contact information for the author or editor of a section belongs in an address element, possibly itself inside a footer. Bylines and other information that could be suitable for both a header or a footer can be placed in either (or neither). The primary purpose of these elements is merely to help the author write self-explanatory markup that is easy to maintain and style; they are not intended to impose specific structures on authors.

Footers don't necessarily have to appear at the end of a section, though they usually do.

When the nearest ancestor sectioning content or sectioning root element is the body element, then it applies to the whole page.

Note: The footer element is not sectioning content; it doesn't introduce a new section.

Here is a page with two footers, one at the top and one at the bottom, with the same content:

```
<body>
<footer><a href="#">Back to index...</a></footer>
<hgroup>
  <h1>Lorem ipsum</h1>
  <h2>The ipsum of all lorem</h2>
</hgroup>
<p>A dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id est laborum.</p>
<footer><a href="#">Back to index...</a></footer>
</body>
```

Here is an example which shows the footer element being used both for a site-wide footer and for a section footer.

```
<!DOCTYPE HTML>
<HTML><HEAD>
<TITLE>The Ramblings of a Scientist</TITLE>
<BODY>
<H1>The Ramblings of a Scientist</H1>
<ARTICLE>
  <H1>Episode 15</H1>
  <VIDEO SRC="/fm/015.ogv" CONTROLS PRELOAD>
    <P><A HREF="/fm/015.ogv">Download video</A>.</P>
  </VIDEO>
  <FOOTER> <!-- footer for article -->
    <P>Published <TIME DATETIME="2009-10-21T18:26-07:00">on 2009/10/21 at 6:26pm</TIME></P>
  </FOOTER>
</ARTICLE>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<FOOTER> <!-- footer for article -->
<P>Published <TIME DATETIME="2009-09-15T14:54-07:00">on 2009/09/15 at 2:54pm</TIME></P>
</FOOTER>
</ARTICLE>
<FOOTER> <!-- site wide footer -->
<NAV>
<P><A HREF="/credits.html">Credits</A> -
<A HREF="/tos.html">Terms of Service</A> -
<A HREF="/index.html">Home Index</A></P>
```



**Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org/
web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

Some
linksf material, including images,
"front page" in the footer.

This fragment shows the bottom of a page on a site with a footer.

```
...
<footer>
<nav>
<section>
<h1>Articles</h1>
<p> Go to the gym with
our somersaults class! Our teacher Jim takes you through the paces
in this two-part article. <a href="articles/somersaults/1">Part
1</a> · <a href="articles/somersaults/1">Part 2</a></p>
<p> Tired of walking on the edge of
a cliff!-- sic -->? Our guest writer Lara shows you how to bumble
your way through the bars. <a href="articles/kindplus/1">Read
more...</a></p>
<p> The chips are down, now all
that's left is a potato. What can you do with it? <a
href="articles/crisps/1">Read more...</a></p>
</section>
<ul>
<li> <a href="/about">About us...</a>
<li> <a href="/feedback">Send feedback!</a>
<li> <a href="/sitemap">Sitemap</a>
</ul>
</nav>
<p><small>Copyright © 2015 The Snacker -
<a href="/tos">Terms of Service</a></small></p>
</footer>
</body>
```

4.4.10 The address element

Categories:

Flow content.
Palpable content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Flow content, but with no heading content descendants, no sectioning content descendants, and no header, footer, or address element descendants.

Content attributes:

Global attributes

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

element, then the contact information applies to the document as a whole.

For example, a page at the W3C Web site related to HTML might include the following contact information:

```
<ADDRESS>
<A href="../People/Raggett/">Dave Raggett</A>,
<A href="../People/Arnaud/">Arnaud Le Hors</A>,
contact persons for the <A href="Activity">W3C HTML Activity</A>
</ADDRESS>
```

The address:

the relevant **Too slow? Try reading the multipage copy of**

The address **the spec instead: https://web.archive.org**

For e **/web/20130103030411/http://whatwg.org/html**

Typically, the

those addresses are in fact
is in general.)

The contact information for a node *node* is a collection of `address` elements defined by the first applicable entry from the following list:

↳ If *node* is an `article` element

↳ If *node* is a `body` element

The contact information consists of all the `address` elements that have *node* as an ancestor and do not have another `body` or `article` element ancestor that is a descendant of *node*.

↳ If *node* has an ancestor element that is an `article` element

↳ If *node* has an ancestor element that is a `body` element

The contact information of *node* is the same as the contact information of the nearest `article` or `body` element ancestor, whichever is nearest.

↳ If *node*'s Document has a `body` element

The contact information of *node* is the same as the contact information of the `body` element of the `Document`.

↳ Otherwise

There is no contact information for *node*.

User agents may expose the contact information of a node to the user, or use it for other purposes, such as indexing sections based on the sections' contact information.

In this example the footer contains contact information and a copyright notice.

```
<footer>
<address>
  For more details, contact
  <a href="mailto:js@example.com">John Smith</a>.
</address>
<p><small>© copyright 2038 Example Corp.</small></p>
</footer>
```

4.4.11 Headings and sections

The `h1-h6` elements and the `hgroup` element are headings.

The first element of heading content in an element of sectioning content represents the heading for that section. Subsequent headings of equal or higher rank start new (implied) sections, headings of lower rank start implied subsections that are part of the previous one. In both cases, the element represents the heading of the implied section.

Certain elements are said to be **sectioning roots**, including `blockquote` and `td` elements. These elements can have their own outlines, but the sections and headings inside these elements do not contribute to the outlines of their ancestors.

⇒ `blockquote, body, details, dialog, fieldset, figure, td`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<body>
<h1>Foo</h1>
<h2>Bar</h2>
<blockquote>
<h3>Bla</h3>
</blockquote>
<p>Baz</p>
<h2>Quux</h2>
<section>
```



Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
...the /web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

(graph)
(elf)

3. Thud (heading of explicit section section)

Notice how the `section` ends the earlier implicit section so that a later paragraph ("Grunt") is back at the top level.

Sections may contain headings of any rank, but authors are strongly encouraged to either use only `h1` elements, or to use elements of the appropriate rank for the section's nesting level.

Authors are also encouraged to explicitly wrap sections in elements of sectioning content, instead of relying on the implicit sections generated by having multiple headings in one element of sectioning content.

For example, the following is correct:

```
<body>
<h4>Apples</h4>
<p>Apples are fruit.</p>
<section>
<h2>Taste</h2>
<p>They taste lovely.</p>
<h6>Sweet</h6>
<p>Red apples are sweeter than green ones.</p>
<h1>Color</h1>
<p>Apples come in various colors.</p>
</section>
</body>
```

However, the same document would be more clearly expressed as:

```
<body>
<h1>Apples</h1>
<p>Apples are fruit.</p>
<section>
<h2>Taste</h2>
<p>They taste lovely.</p>
<section>
<h3>Sweet</h3>
<p>Red apples are sweeter than green ones.</p>
</section>
</section>
<h2>Color</h2>
<p>Apples come in various colors.</p>
</section>
</body>
```

Both of the documents above are semantically identical and would produce the same outline in compliant user agents.

This third example is also semantically identical, and might be easier to maintain (e.g. if sections are often moved around in

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<section>
  <h1>Taste</h1>
  <p>They taste lovely.</p>
  <section>
    <h1>Sweet</h1>
    <p>Red apples are sweeter than green ones.</p>
  </section>
</section>
<section>
```



Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org/
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

This
would

browsers without CSS support

4.4.11.1 Creating an outline

This section defines an algorithm for creating an outline for a sectioning content element or a sectioning root element. It is defined in terms of a walk over the nodes of a DOM tree, in tree order, with each node being visited when it is *entered* and when it is *exited* during the walk.

The **outline** for a sectioning content element or a sectioning root element consists of a list of one or more potentially nested sections. A **section** is a container that corresponds to some nodes in the original DOM tree. Each section can have one heading associated with it, and can contain any number of further nested sections. The algorithm for the outline also associates each node in the DOM tree with a particular section and potentially a heading. (The sections in the outline aren't `section` elements, though some may correspond to such elements — they are merely conceptual sections.)

The following markup fragment:

```
<body>
  <h1>A</h1>
  <p>B</p>
  <h2>C</h2>
  <p>D</p>
  <h2>E</h2>
  <p>F</p>
</body>
```

...results in the following outline being created for the `body` node (and thus the entire document):

1. Section created for `body` node.
Associated with heading "A".
Also associated with paragraph "B".
Nested sections:
 1. Section implied for first `h2` element.
Associated with heading "C".
Also associated with paragraph "D".
No nested sections.
 2. Section implied for second `h2` element.
Associated with heading "E".
Also associated with paragraph "F".
No nested sections.

The algorithm that must be followed during a walk of a DOM subtree rooted at a sectioning content element or a sectioning root element to determine that element's outline is as follows:

1. Let `current outlinee` be null. (It holds the element whose outline is being created.)
2. Let `current section` be null. (It holds a pointer to a section, so that elements in the DOM can all be associated with a section.)
3. Create a stack to hold elements, which is used to handle nesting. Initialize this stack to empty.
4. Walk over the DOM in tree order, starting with the sectioning content element or sectioning root element at the root of the subtree for which an outline is to be created, and trigger the first relevant step below for each element as the walk enters and

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

attribute.

Pop that element from the stack.

↪ If the top of the stack is a heading content element or an element with a `hidden` attribute

Do nothing.

↪ When entering an element with a `hidden` attribute

Push the element being entered onto the stack. (This causes the algorithm to skip that element and any descendants of the element.)



↪ Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Append heading and let that be the

Associate *current outlinee* with *current section*.

Let there be a new outline for the new *current outlinee*, initialized with just the new *current section* as the only section in the outline.

↪ When exiting a sectioning content element, if the stack is not empty

If the *current section* has no heading, create an implied heading and let that be the heading for the *current section*.

Pop the top element from the stack, and let the *current outlinee* be that element.

Let *current section* be the last section in the outline of the *current outlinee* element.

Append the outline of the sectioning content element being exited to the *current section*. (This does not change which section is the last section in the outline.)

↪ When exiting a sectioning root element, if the stack is not empty

Run these steps:

1. If the *current section* has no heading, create an implied heading and let that be the heading for the *current section*.
2. Pop the top element from the stack, and let the *current outlinee* be that element.
3. Let *current section* be the last section in the outline of the *current outlinee* element.
4. *Finding the deepest child*: If *current section* has no child sections, stop these steps.
5. Let *current section* be the last child section of the current *current section*.
6. Go back to the substep labeled *finding the deepest child*.

↪ When exiting a sectioning content element or a sectioning root element

Note: The current outlinee is the element being exited, and it is the sectioning content element or a sectioning root element at the root of the subtree for which an outline is being generated.

If the *current section* has no heading, create an implied heading and let that be the heading for the *current section*.

Skip to the next step in the overall set of steps. (The walk is over.)

↪ When entering a heading content element

If the *current section* has no heading, let the element being entered be the heading for the *current section*.

Otherwise, if the element being entered has a rank equal to or higher than the heading of the last section of the outline of the *current outlinee*, or if the heading of the last section of the outline of the *current outlinee* is an implied heading, then create a new section and append it to the outline of the *current outlinee* element, so that this new section is the new last section of that outline. Let *current section* be that new section. Let the element

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

2. *Heading loop*: If the element being entered has a rank lower than the rank of the heading of the *candidate section*, then create a new section, and append it to *candidate section*. (This does not change which section is the last section in the outline.) Let *current section* be this new section. Let the element being entered be the new heading for the *current section*. Abort these substeps.
3. Let *new candidate section* be the section that contains *candidate section* in the outline of *current outlinee*.
4. Let *candidate section* be *new candidate section*.



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

In addition, whenever the walk exits a node, after doing the steps above, if the node is not associated with a section yet, associate the node with the section *current section*.

5. Associate all nodes with the heading of the section with which they are associated, if any.

The tree of sections created by the algorithm above, or a proper subset thereof, must be used when generating document outlines, for example when generating tables of contents.

The outline created for the body element of a Document is the outline of the entire document.

When creating an interactive table of contents, entries should jump the user to the relevant sectioning content element, if the section was created for a real element in the original document, or to the relevant heading content element, if the section in the tree was generated for a heading in the above process.

Note: Selecting the first section of the document therefore always takes the user to the top of the document, regardless of where the first heading in the body is to be found.

The **outline depth** of a heading content element associated with a section *section* is the number of sections that are ancestors of *section* in the outermost outline that *section* finds itself in when the outlines of its Document's elements are created, plus 1. The outline depth of a heading content element not associated with a section is 1.

User agents should provide default headings for sections that do not have explicit section headings.

Consider the following snippet:

```
<body>
  <nav>
    <p><a href="/">Home</a></p>
  </nav>
  <p>Hello world.</p>
  <aside>
    <p>My cat is cute.</p>
  </aside>
</body>
```

Although it contains no headings, this snippet has three sections: a document (the `body`) with two subsections (a `nav` and an `aside`). A user agent could present the outline as follows:

1. Untitled document
 1. Navigation
 2. Sidebar

These default headings ("Untitled document", "Navigation", "Sidebar") are not specified by this specification, and might vary with the user's language, the page's language, the user's preferences, the user agent implementor's preferences, etc.

The following JavaScript function shows how the tree walk could be implemented. The root argument is the root of the tree to walk (either a sectioning content element or a sectioning root element), and the enter and exit arguments

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
enter(node);
if (node.firstChild) {
    node = node.firstChild;
    continue start;
}
while (node) {
    exit(node);
    if (node == root) {
        node = null;
    }
}
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4.4.12 Usage summary

This section is non-normative.

Element	Purpose	Example
body	The main content of the document.	<pre><!DOCTYPE HTML> <html> <head> <title>Steve Hill's Home Page</title> </head> <body> <p>Hard Trance is My Life.</p> </body> </html></pre>
article	A self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.	<pre><article> <p>My fave Masif tee so far!</p> <footer>Posted 2 days ago</footer> </article> <article> <p>Happy 2nd birthday Masif Saturdays!!!</p> <footer>Posted 3 weeks ago</footer> </article></pre>
section	A generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading.	<pre><h1>Biography</h1> <section> <h1>The facts</h1> <p>1500+ shows, 14+ countries</p> </section> <section> <h1>2010/2011 figures per year</h1> <p>100+ shows, 8+ countries</p> </section></pre>
nav	A section of a page that links to other pages or to parts within the page: a section with navigation links.	<pre><nav> <p>Home <p>Bio <p>Discog </nav></pre>
aside	A section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.	

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

	<p>This year we played a kind of trance that originated in Belgium, Germany, and the Netherlands in the mid 90s.</p>
h1–h6	A section heading <h1>The Guide To Music On The Playa</h1> <h2>The Main Stage</h2> <p>If you want to play on a stage, you should bring one.</p> <h2>Amplified Music</h2> <p>Amplifiers up to 300W or 90dB are welcome.</p>
hgroup	The heading of a section. The element is used to group a set of h1–h6 elements when the heading has multiple levels, such as subheadings, alternative titles, or <i>ordinals</i> .
	<p>Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html</p> <pre></section> <section> <hgroup> <h1>Loudness!</h1> <h2>Questions About Amplified Music</h2> </hgroup> <p>Amplifiers up to 300W or 90dB are welcome.</p> </section></pre>
header	A group of introductory or navigational aids.
footer	A footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like.

4.5 Grouping content

4.5.1 The p element

Categories:

- Flow content.
- Palpable content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

```
IDL interface HTMLParagraphElement : HTMLElement {};
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

breaks in a different manner, for instance using inline paragraphs (§ 11).

The following examples are conforming HTML fragments:

```
<p>The little kitten gently seated himself on a piece of
carpet. Later in his life, this would be referred to as the time the
cat sat on the mat.</p>

<fieldset>
<legend>Personal information</legend>
```



**Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http:
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

The validator complained, ~~~
So the author was pained,

To move the error from the markup to the rhyming.</p>

The `p` element should not be used when a more specific element is more appropriate.

The following example is technically correct:

```
<section>
<!-- ... -->
<p>Last modified: 2001-04-23</p>
<p>Author: fred@example.com</p>
</section>
```

However, it would be better marked-up as:

```
<section>
<!-- ... -->
<footer>Last modified: 2001-04-23</footer>
<address>Author: fred@example.com</address>
</section>
```

Or:

```
<section>
<!-- ... -->
<footer>
<p>Last modified: 2001-04-23</p>
<address>Author: fred@example.com</address>
</footer>
</section>
```

List elements (in particular, `ol` and `ul` elements) cannot be children of `p` elements. When a sentence contains a bulleted list, therefore, one might wonder how it should be marked up.

For instance, this fantastic sentence has bullets relating to

- **wizards,**
- **faster-than-light travel, and**
- **telepathy,**

and is further discussed below.

The solution is to realise that a paragraph, in HTML terms, is not a logical concept, but a structural one. In the fantastic example above, there are actually five paragraphs as defined by this specification: one before the list, one for each bullet, and one after the list.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<li>faster-than-light travel, and
<li>telepathy,
</ul>
<p>and is further discussed below.</p>
```

Authors wishing to conveniently style such "logical" paragraphs consisting of multiple "structural" paragraphs can use the `div` element instead of the `p` element.

Thus for instance the above example could become the following:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

to consider each part of the example separately.

div instead of having

4.5.2 The `hr` element

Categories:

Flow content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Empty.

Content attributes:

Global attributes

DOM interface:

```
IDL interface HTMLHRElement : HTMLElement {};
```

The `hr` element represents a paragraph-level thematic break, e.g. a scene change in a story, or a transition to another topic within a section of a reference book.

The following fictional extract from a project manual shows two sections that use the `hr` element to separate topics within the section.

```
<section>
  <h1>Communication</h1>
  <p>There are various methods of communication. This section covers a few of the important ones used by the project.</p>
  <hr>
  <p>Communication stones seem to come in pairs and have mysterious properties:</p>
  <ul>
    <li>They can transfer thoughts in two directions once activated if used alone.</li>
    <li>If used with another device, they can transfer one's consciousness to another body.</li>
    <li>If both stones are used with another device, the consciousness switch bodies.</li>
  </ul>
  <hr>
  <p>Radios use the electromagnetic spectrum in the meter range and longer.</p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<h1>Food</h1>
<p>All food at the project is rationed:</p>
<dl>
  <dt>Potatoes</dt>
  <dd>Two per day</dd>
  <dt>Soup</dt>
  <dd>One bowl per day</dd>
</dl>
<br>
```



Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

and the `h1` elements imply

```
<p>Dudley was ninety-two, in his second life, and fast approaching  
time for another rejuvenation. Despite his body having the physical  
age of a standard fifty-year-old, the prospect of a long degrading  
campaign within academia was one he regarded with dread. For a  
supposedly advanced civilization, the Intersolar Commonwealth could be  
appallingly backward at times, not to mention cruel.</p>
<p><i>Maybe it won't be that bad</i>, he told himself. The lie was  
comforting enough to get him through the rest of the night's  
shift.</p>
```

```
<hr>
<p>The Carlton AllLander drove Dudley home just after dawn. Like the  
astronomer, the vehicle was old and worn, but perfectly capable of  
doing its job. It had a cheap diesel engine, common enough on a  
semi-frontier world like Gralmond, although its drive array was a  
thoroughly modern photoneural processor. With its high suspension and  
deep-tread tyres it could plough along the dirt track to the  
observatory in all weather and seasons, including the metre-deep snow  
of Gralmond's winters.</p>
```

Note: The `hr` element does not affect the document's outline.

4.5.3 The `pre` element

Categories:

Flow content.
Palpable content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

```
IDL interface HTMLPreElement : HTMLElement {};
```

The `pre` element represents a block of preformatted text, in which structure is represented by typographic conventions rather than by elements.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- Including an e-mail, with paragraphs indicated by blank lines, lists indicated by lines prefixed with a bullet, and so on.
- Including fragments of computer code, with structure indicated according to the conventions of that language.
- Displaying ASCII art.

Note: Authors are encouraged to consider how preformatted text will be experienced when the formatting is lost, as will be the case for users of speech synthesizers, braille displays, and the like. For cases like ASCII art, it is likely that an alternative presentation, such as a textual description, would be more universally accessible to the readers of the document.

To represent pre elements user is to enable the spec instead: <https://web.archive.org>

A newline in /web/20130103030411/http: be implemented //whatwg.org/html

block of computer output the ent to indicate text that the

rithm. This requirement may fit these requirements by

In the following snippet, a sample of computer code is presented.

```
<p>This is the <code>Panel</code> constructor:</p>
<pre><code>function Panel(element, canClose, closeHandler) {
    this.element = element;
    this.canClose = canClose;
    this.closeHandler = function () { if (closeHandler) closeHandler() };
}</code></pre>
```

In the following snippet, samp and kbd elements are mixed in the contents of a pre element to show a session of Zork I.

```
<pre><samp>You are in an open field west of a big white house with a boarded
front door.
There is a small mailbox here.

></samp> <kbd>open mailbox</kbd>

<samp>Opening the mailbox reveals:
A leaflet.

></samp></pre>
```

The following shows a contemporary poem that uses the pre element to preserve its unusual formatting, which forms an intrinsic part of the poem itself.

```
<pre>
maxling

it is with a      heart
                  heavy

that i admit loss of a feline
so            loved

a friend lost to the
unknown
(night)

~cdr 11dec07</pre>
```

IDL

4.5.4 The blockquote element

Categories:

- Flow content.
- Sectioning root.
- Palpable content.

Go

NOV JAN FEB



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

Digitized by srujanika@gmail.com

▼ About this capture

Content attributes:

Global attributes

cite

DOM interface:

```
interface HTMLQuoteElement : HTMLElement  
        attribute DOMString cite;
```



Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org>

The block [/web/20130103030411/http://whatwg.org/html](http://web/20130103030411/http://whatwg.org/html)
Content insi //whatwg.org/html

cited in the `cite` attribute.

If the `cite` attribute is present, it must be a valid URL potentially surrounded by spaces. To obtain the corresponding citation link, the value of the attribute must be resolved relative to the element. User agents may allow users to follow such citation links, but they are primarily intended for private use (e.g. by server-side scripts collecting statistics about a site's use of quotations), not for readers.

The content of a `blockquote` may be abbreviated or may have context added in the conventional manner for the text's language.

For example, in English this is traditionally done using square brackets. Consider a page with the sentence "Fred ate the cracker. He then said he liked apples and fish"; it could be quoted as follows:

```
<blockquote>
  <p>[Fred] then said he liked [...] fish.</p>
</blockquote>
```

Attribution for the quotation, if any, must be placed outside the `blockquote` element.

For example, here the attribution is given in a paragraph after the quote:

```
<blockquote>
  <p>I contend that we are both atheists. I just believe in one fewer
  god than you do. When you understand why you dismiss all the other
  possible gods, you will understand why I dismiss yours.</p>
</blockquote>
<p>— Stephen Roberts</p>
```

The other examples below show other ways of showing attribution.

The `cite` IDL attribute must reflect the element's `cite` content attribute.

Here a `blockquote` element is used in conjunction with a `figure` element and its `figcaption` to clearly relate a quote to its attribution (which is not part of the quote and therefore doesn't belong inside the `blockquote` itself):

```
<figure>
<blockquote>


The truth may be puzzling. It may take some work to grapple with. It may be counterintuitive. It may contradict deeply held prejudices. It may not be consonant with what we desperately want to be true. But our preferences do not determine what's true. We have a method, and that method helps us to reach not absolute truth, only asymptotic approaches to the truth – never there, just closer and closer, always finding vast new oceans of undiscovered possibilities. Cleverly designed experiments are the key.</p>
</blockquote>
<figcaption>Carl Sagan, in "<cite>Wonder and Skepticism</cite>", from the <cite>Skeptical Enquirer</cite> Volume 19, Issue 1 (January-February 1995)</figcaption>
</figure>


```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Coral is far more red, than her lips red,
...
...

This example shows how a forum post could use `blockquote` to show what post a user is replying to. The `article` element is used for each post, to mark up the threading.

```
<article>
<h1><a href="http://bacon.example.com/?blog=109431">Bacon on a crowbar</a></h1>
<article>
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<blockquote>
  <blockquote><p>I bet a narwhal would love that.</p></blockquote>
  <p>Dude narwhals don't eat bacon.</p>
</blockquote>
<p>Next thing you'll be saying they don't get capes and wizard hats either!</p>
<footer><a href="?pid=29580">permalink</a></footer>
<article>
  <article>
    <header><strong>boing</strong> -5 points 1 hour ago</header>
    <p>narwhals are worse than ceiling cat</p>
    <footer><a href="?pid=29581">permalink</a></footer>
  </article>
  </article>
</article>
<article>
  <header><strong>fred</strong> 1 points 23 minutes ago</header>
  <blockquote><p>I bet a narwhal would love that.</p></blockquote>
  <p>I bet they'd love to peel a banana too.</p>
  <footer><a href="?pid=29582">permalink</a></footer>
</article>
</article>
</article>
```

This example shows the use of a `blockquote` for short snippets, demonstrating that one does not have to use `p` elements inside `blockquote` elements:

```
<p>He began his list of "lessons" with the following:</p>
<blockquote>One should never assume that his side of the issue will be recognized, let alone that it will be conceded to have merits.</blockquote>
<p>He continued with a number of similar points, ending with:</p>
<blockquote>Finally, one should be prepared for the threat of breakdown in negotiations at any given moment and not be cowed by the possibility.</blockquote>
<p>We shall now discuss these points...</p>
```

Note: Examples of how to represent a conversation are shown in a later section; it is not appropriate to use the `cite` and `blockquote` elements for this purpose.

4.5.5 The `ol` element

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the element's children include at least one `li` element, it is a **listable content**.**Contexts in which this element can be used:**

Where flow content is expected.

Content model:Zero or more `li` elements.**Content attributes:**

Global attributes



rf

s

t

DOM interface

Too slow? Try reading the multipage copy of
 the spec instead: [https://web.archive.org
 /web/20130103030411/http:
 //whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

};

----- rendering tips -----

The `ol` element represents a list of items, where the items have been intentionally ordered, such that changing the order would change the meaning of the document.

The items of the list are the `li` element child nodes of the `ol` element, in tree order.

The `reversed` attribute is a boolean attribute. If present, it indicates that the list is a descending list (... , 3, 2, 1). If the attribute is omitted, the list is an ascending list (1, 2, 3, ...).

The `start` attribute, if present, must be a valid integer giving the ordinal value of the first list item.

If the `start` attribute is present, user agents must parse it as an integer, in order to determine the attribute's value. The default value, used if the attribute is missing or if the value cannot be converted to a number according to the referenced algorithm, is 1 if the element has no `reversed` attribute, and is the number of child `li` elements otherwise.

The first item in the list has the ordinal value given by the `ol` element's `start` attribute, unless that `li` element has a `value` attribute with a value that can be successfully parsed, in which case it has the ordinal value given by that `value` attribute.

Each subsequent item in the list has the ordinal value given by its `value` attribute, if it has one, or, if it doesn't, the ordinal value of the previous item, plus one if the `reversed` is absent, or minus one if it is present.

The `type` attribute can be used to specify the kind of marker to use in the list, in the cases where that matters (e.g. because items are to be referenced by their number/letter). The attribute, if specified, must have a value that is a case-sensitive match for one of the characters given in the first cell of one of the rows of the following table. The `type` attribute represents the state given in the cell in the second column of the row whose first cell matches the attribute's value; if none of the cells match, or if the attribute is omitted, then the attribute represents the decimal state.

Keyword	State	Description	Examples for values 1-3 and 3999-4001							
1 (U+0031)	decimal	Decimal numbers	1.	2.	3.	...	3999.	4000.	4001.	...
a (U+0061)	lower-alpha	Lowercase latin alphabet	a.	b.	c.	...	ewu.	ewv.	eww.	...
A (U+0041)	upper-alpha	Uppercase latin alphabet	A.	B.	C.	...	EWU.	EWV.	EWW.	...
i (U+0069)	lower-roman	Lowercase roman numerals	i.	ii.	iii.	...	mmmcxcix.	i ⁻ v ⁻ .	i ⁻ v ⁻ i.	...
I (U+0049)	upper-roman	Uppercase roman numerals	I.	II.	III.	...	MMMCXCIX.	I ⁻ V ⁻ .	I ⁻ V ⁻ I.	...

User agents should render the items of the list in a manner consistent with the state of the `type` attribute of the `ol` element. Numbers less than or equal to zero should always use the decimal system regardless of the `type` attribute.

Note: For CSS user agents, a mapping for this attribute to the 'list-style-type' CSS property is given in the rendering section (the mapping is straightforward: the states above have the same names as their corresponding CSS values).

The `reversed`, `start`, and `type` IDL attributes must reflect the respective content attributes of the same name. The `start` IDL attribute has the same default as its content attribute.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<ol>
<li>Switzerland
<li>United Kingdom
<li>United States
<li>Norway
</ol>
```

Note how changing the order of the list changes the meaning of the document. In the following example, changing the relative order of the first two items has changed the birthplace of the author:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4.5.6 The `ul` element

Categories:

Flow content.

If the element's children include at least one `li` element: Palpable content.

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Zero or more `li` elements.

Content attributes:

Global attributes

DOM interface:

```
IDL interface HTMLULListElement : HTMLElement {};
```

The `ul` element represents a list of items, where the order of the items is not important — that is, where changing the order would not materially change the meaning of the document.

The items of the list are the `li` element child nodes of the `ul` element.

The following markup shows a list where the order does not matter, and where the `ul` element is therefore appropriate. Compare this list to the equivalent list in the `ol` section to see an example of the same items using the `ol` element.

```
<p>I have lived in the following countries:</p>
<ul>
<li>Norway
<li>Switzerland
<li>United Kingdom
<li>United States
</ul>
```

Note that changing the order of the list does not change the meaning of the document. The items in the snippet above are given in alphabetical order, but in the snippet below they are given in order of the size of their current account balance in 2007, without changing the meaning of the document whatsoever:

```
<p>I have lived in the following countries:</p>
<ul>
<li>Switzerland
<li>Norway
<li>United Kingdom
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Categories:

None.

Contexts in which this element can be used:Inside `ol` elements.Inside `ul` elements.Inside `menu` elements whose `type` attribute is in the toolbar state.**Content model:**

F Too slow? Try reading the multipage copy of
Content:
 the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

DOM interface:

If
`li`
 DOM interface
`listItem`

```
attribute long value;
};
```

The `li` element represents a list item. If its parent element is an `ol`, `ul`, or `menu` element, then the element is an item of the parent element's list, as defined for those elements. Otherwise, the list item has no defined list-related relationship to any other `li` element.

If the parent element is an `ol` element, then the `li` element has an **ordinal value**.

The `value` attribute, if present, must be a valid integer giving the ordinal value of the list item.

If the `value` attribute is present, user agents must parse it as an integer, in order to determine the attribute's value. If the attribute's value cannot be converted to a number, the attribute must be treated as if it was absent. The attribute has no default value.

The `value` attribute is processed relative to the element's parent `ol` element (q.v.), if there is one. If there is not, the attribute has no effect.

The `value` IDL attribute must reflect the value of the `value` content attribute.

The following example, the top ten movies are listed (in reverse order). Note the way the list is given a title by using a `figure` element and its `figcaption` element.

```
<figure>
  <figcaption>The top 10 movies of all time</figcaption>
  <ol>
    <li value="10"><cite>Josie and the Pussycats</cite>, 2001</li>
    <li value="9"><cite lang="sh">Црна мајка, бели мајор</cite>, 1998</li>
    <li value="8"><cite>A Bug's Life</cite>, 1998</li>
    <li value="7"><cite>Toy Story</cite>, 1995</li>
    <li value="6"><cite>Monsters, Inc</cite>, 2001</li>
    <li value="5"><cite>Cars</cite>, 2006</li>
    <li value="4"><cite>Toy Story 2</cite>, 1999</li>
    <li value="3"><cite>Finding Nemo</cite>, 2003</li>
    <li value="2"><cite>The Incredibles</cite>, 2004</li>
    <li value="1"><cite>Ratatouille</cite>, 2007</li>
  </ol>
</figure>
```

The markup could also be written as follows, using the `reversed` attribute on the `ol` element:

```
<figure>
  <figcaption>The top 10 movies of all time</figcaption>
  <ol reversed>
    <li><cite>Josie and the Pussycats</cite>, 2001</li>
    <li><cite lang="sh">Црна мајка, бели мајор</cite>, 1998</li>
    <li><cite>A Bug's Life</cite>, 1998</li>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<li><cite>The Incredibles</cite>, 2004</li>
<li><cite>Ratatouille</cite>, 2007</li>
</ol>
</figure>
```

Note: While it is conforming to include heading elements (e.g. `h1`) inside `li` elements, it likely does not convey the semantics that the author intended. A heading starts a new section, so a heading in a list implicitly splits the list into spanning multiple sections.



4.5.8 The `dl` element

Categories: the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

F /web/20130103030411/http:

If If

Contexts: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Within

Content model:

Zero or more groups each consisting of one or more `dt` elements followed by one or more `dd` elements.

Content attributes:

Global attributes

DOM interface:

```
IDL interface HTMLDLListElement : HTMLElement {};
```

The `dl` element represents an association list consisting of zero or more name-value groups (a description list). Each group must consist of one or more names (`dt` elements) followed by one or more values (`dd` elements). Within a single `dl` element, there should not be more than one `dt` element for each name.

Name-value groups may be terms and definitions, metadata topics and values, questions and answers, or any other groups of name-value data.

The values within a group are alternatives; multiple paragraphs forming part of the same value must all be given within the same `dd` element.

The order of the list of groups, and of the names and values within each group, may be significant.

If a `dl` element is empty, it contains no groups.

If a `dl` element has one or more non-whitespace `Text` node children, or has child elements that are neither `dt` nor `dd` elements, all such `Text` nodes and elements, as well as their descendants (including any `dt` or `dd` elements), do not form part of any groups in that `dl`.

If a `dl` element has one or more `dt` element children but no `dd` element children, then it consists of one group with names but no values.

If a `dl` element has one or more `dd` element children but no `dt` element children, then it consists of one group with values but no names.

If a `dl` element's first `dt` or `dd` element child is a `dd` element, then the first group has no associated name.

If a `dl` element's last `dt` or `dd` element child is a `dt` element, then the last group has no associated value.

Note: When a `dl` element doesn't match its content model, it is often due to accidentally using `dd` elements in the place of `dt` elements and vice versa. Conformance checkers can spot such mistakes and might be able to advise authors how to correctly use the markup.

In the following example, one entry ("Authors") is linked to two values ("John" and "Luke").

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<dd> Frank
</dl>
```

In the following example, one definition is linked to two terms.

```
<dl>
<dt lang="en-US"> <dfn>color</dfn> </dt>
<dt lang="en-GB"> <dfn>colour</dfn> </dt>
<dd> A sensation which (in humans) derives from the ability ? X
+<br/>
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<dt> Recommended update interval </dt>
<dd> 60s </dd>
<dt> Authors </dt>
<dt> Editors </dt>
<dd> Robert Rothman </dd>
<dd> Daniel Jackson </dd>
</dl>
```

The following example shows the `dl` element used to give a set of instructions. The order of the instructions here is important (in the other examples, the order of the blocks was not important).

```
<p>Determine the victory points as follows (use the
first matching case):</p>
<dl>
<dt> If you have exactly five gold coins </dt>
<dd> You get five victory points </dd>
<dt> If you have one or more gold coins, and you have one or more silver coins </dt>
<dd> You get two victory points </dd>
<dt> If you have one or more silver coins </dt>
<dd> You get one victory point </dd>
<dt> Otherwise </dt>
<dd> You get no victory points </dd>
</dl>
```

The following snippet shows a `dl` element being used as a glossary. Note the use of `dfn` to indicate the word being defined.

```
<dl>
<dt><dfn>Apartment</dfn>, n.</dt>
<dd>An execution context grouping one or more threads with one or
more COM objects.</dd>
<dt><dfn>Flat</dfn>, n.</dt>
<dd>A deflated tire.</dd>
<dt><dfn>Home</dfn>, n.</dt>
<dd>The user's login directory.</dd>
</dl>
```

Note: The `dl` element is inappropriate for marking up dialogue. Examples of how to mark up dialogue are shown below.

4.5.9 The `dt` element

Categories:

None.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

Content attributes:

Global attributes

DOM interface:

Uses HTMLElement.

The `dt` element represents the term, or name, part of a term-description group in a description list (`dl` element).



a term being defined,

**Note: Tr
but this** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

This elem

or questions and the `dd`

```
<at>What do we want?</at>
<dd>Our data.</dd>
<dt>When do we want it?</dt>
<dd>Now.</dd>
<dt>Where is it?</dt>
<dd>We are not sure.</dd>
</dl>
</article>
```

4.5.10 The `dd` element**Categories:**

None.

Contexts in which this element can be used:After `dt` or `dd` elements inside `dl` elements.**Content model:**

Flow content.

Content attributes:

Global attributes

DOM interface:

Uses HTMLElement.

The `dd` element represents the description, definition, or value, part of a term-description group in a description list (`dl` element).

A `dl` can be used to define a vocabulary list, like in a dictionary. In the following example, each entry, given by a `dt` with a `dfn`, has several `dd`s, showing the various parts of the definition.

```
<dl>
  <dt><dfn>happiness</dfn></dt>
  <dd class="pronunciation">/'hæ p. nes/</dd>
  <dd class="part-of-speech"><i><abbr>n.</abbr></i></dd>
  <dd>The state of being happy.</dd>
  <dd>Good fortune; success. <q>Oh <b>happiness</b>! It worked!</q></dd>
  <dt><dfn>rejoice</dfn></dt>
  <dd class="pronunciation">/ri jois'/</dd>
  <dd class="part-of-speech"><i><abbr>v.intr.</abbr></i> To be delighted oneself.</dd>
  <dd class="part-of-speech"><i><abbr>v.tr.</abbr></i> To cause one to be delighted.</dd>
</dl>
```

4.5.11 The `figure` element

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Contexts in which this element can be used:

Where flow content is expected.

Content model:

Either: One `figcaption` element followed by flow content.
 Or: Flow content followed by one `figcaption` element.
 Or: Flow content.

Content attributes:

Gr



DOM interface `Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org`

The `figure` element represents a single unit of presentation, typically referenced as a single unit from the document's table of contents, or to dedicated pages, or to an appendix.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<figcaption>Exhibit B. The <cite>Rough Copy</cite> trailer.</figcaption>
</figure>
```

<p>The case was resolved out of court.

Here, a part of a poem is marked up using `figure`.

```
<figure>
<p>'Twas brillig, and the slithy toves<br>
r
```



Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

32-98</figcaption>

In thi
both
[/web/20130103030411/http://whatwg.org/html](http://whatwg.org/html)

nents are used to provide

```
<_
<figcaption>The castle through the ages: 1423, 1858, and 1999 respectively.</figcaption>
<figure>
<figcaption>Etching. Anonymous, ca. 1423.</figcaption>

</figure>
<figure>
<figcaption>Oil-based paint on canvas. Maria Towle, 1858.</figcaption>

</figure>
<figure>
<figcaption>Film photograph. Peter Jankle, 1999.</figcaption>

</figure>
</figure>
```

The previous example could also be more succinctly written as follows (using `title` attributes in place of the nested `figure/figcaption` pairs):

```
<figure>



<figcaption>The castle through the ages: 1423, 1858, and 1999 respectively.</figcaption>
</figure>
```

4.5.12 The `figcaption` element

Categories:

None.

Contexts in which this element can be used:

As the first or last child of a `figure` element.

Content model:

Flow content.

Content attributes:

Global attributes

DOM interface:

Uses `HTMLElement`.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

NOTE: This dev element

Categories:

- Flow content.
- Palpable content.

Contexts in which this element can be used:

- Where flow content is expected.

Content model:

Flr



Content: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

DOM interface

The `div` element has no special meaning at all. It represents its children. It can be used with the `class`, `lang`, and `title` attributes to mark up semantics common to a group of consecutive elements.

Note: Authors are strongly encouraged to view the `div` element as an element of last resort, for when no other element is suitable. Use of more appropriate elements instead of the `div` element leads to better accessibility for readers and easier maintainability for authors.

For example, a blog post would be marked up using `article`, a chapter using `section`, a page's navigation aids using `nav`, and a group of form controls using `fieldset`.

On the other hand, `div` elements can be useful for stylistic purposes or to wrap multiple paragraphs within a section that are all to be annotated in a similar way. In the following example, we see `div` elements used as a way to set the language of two paragraphs at once, instead of setting the language on the two paragraph elements separately:

```
<article lang="en-US">
  <h1>My use of language and my cats</h1>
  <p>My cat's behavior hasn't changed much since her absence, except
    that she plays her new physique to the neighbors regularly, in an
    attempt to get pets.</p>
  <div lang="en-GB">
    <p>My other cat, coloured black and white, is a sweetie. He followed
      us to the pool today, walking down the pavement with us. Yesterday
      he apparently visited our neighbours. I wonder if he recognises that
      their flat is a mirror image of ours.</p>
    <p>Hm, I just noticed that in the last paragraph I used British
      English. But I'm supposed to write in American English. So I
      shouldn't say "pavement" or "flat" or "colour"...</p>
  </div>
  <p>I should say "sidewalk" and "apartment" and "color"!</p>
</article>
```

4.6 Text-level semantics

IDL

4.6.1 The `a` element

Categories:

- Flow content.
- Phrasing content.
- Interactive content.
- Palpable content.

Contexts in which this element can be used:

- Where phrasing content is expected.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

href
target
download
ping
rel
hreflang
type

DOM interface



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
attribute DOMString type;

attribute DOMString text;

// URL decomposition IDL attributes
attribute DOMString protocol;
attribute DOMString host;
attribute DOMString hostname;
attribute DOMString port;
attribute DOMString pathname;
attribute DOMString search;
attribute DOMString hash;
};
```

If the `a` element has an `href` attribute, then it represents a hyperlink (a hypertext anchor) labeled by its contents.

If the `a` element has no `href` attribute, then the element represents a placeholder for where a link might otherwise have been placed, if it had been relevant, consisting of just the element's contents.

The `target`, `download`, `ping`, `rel`, `hreflang`, and `type` attributes must be omitted if the `href` attribute is not present.

If the `itemprop` is specified on an `a` element, then the `href` attribute must also be specified.

If a site uses a consistent navigation toolbar on every page, then the link that would normally link to the page itself could be marked up using an `a` element:

```
<nav>
<ul>
<li> <a href="/">Home</a> </li>
<li> <a href="/news">News</a> </li>
<li> <a>Examples</a> </li>
<li> <a href="/legal">Legal</a> </li>
</ul>
</nav>
```

The `href`, `target`, `download`, and `ping` attributes affect what happens when users follow hyperlinks or download hyperlinks created using the `a` element. The `rel`, `hreflang`, and `type` attributes may be used to indicate to the user the likely nature of the target resource before the user follows the link.

The activation behavior of `a` elements that create hyperlinks is to run the following steps:

1. If the `a` element's `Document` is not in a browsing context, then abort these steps.
2. If either the `a` element has a `download` attribute and the algorithm is not allowed to show a popup, or the element's `target` attribute is present and applying the rules for choosing a browsing context given a browsing context name, using the value of

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

1. If the `click` event was a real pointing-device-triggered `click` event on the `img` element, then let `x` be the distance in CSS pixels from the left edge of the image's left border, if it has one, or the left edge of the image otherwise, to the location of the click, and let `y` be the distance in CSS pixels from the top edge of the image's top border, if it has one, or the top edge of the image otherwise, to the location of the click. Otherwise, let `x` and `y` be zero.
2. Let the ***hyperlink suffix*** be a U+003F QUESTION MARK character, the value of `x` expressed as a base-ten integer using ASCII digits, a U+002C COMMA character (,), and the value of `y` expressed as a base-ten integer using ASCII digits.

4. Finally,
down
according to
the spec instead:

Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

a. `textContent`
Sar...

nt, as determined by the
fix, then take that into

ts are given below this box.

The IDL attributes `href`, `download`, `ping`, `target`, `rel`, `hreflang`, and `type`, must reflect the respective content attributes of the same name.

The IDL attribute `relList` must reflect the `rel` content attribute.

The `textContent` IDL attribute, on getting, must return the same value as the `textContent` IDL attribute on the element, and on setting, must act as if the `textContent` IDL attribute on the element had been set to the new value.

The `a` element also supports the complement of URL decomposition IDL attributes, `protocol`, `host`, `port`, `hostname`, `pathname`, `search`, and `hash`. These must follow the rules given for URL decomposition IDL attributes, with the input being the result of resolving the element's `href` attribute relative to the element, if there is such an attribute and resolving it is successful, or the empty string otherwise; and the common setter action being the same as setting the element's `href` attribute to the new output value.

The `a` element may be wrapped around entire paragraphs, lists, tables, and so forth, even entire sections, so long as there is no interactive content within (e.g. buttons or other links). This example shows how this can be used to make an entire advertising block into a link:

```
<aside class="advertising">
  <h1>Advertising</h1>
  <a href="http://ad.example.com/?adid=1929&pubid=1422">
    <section>
      <h1>Mellblomatic 9000!</h1>
      <p>Turn all your widgets into mellbloms!</p>
      <p>Only $9.99 plus shipping and handling.</p>
    </section>
  </a>
  <a href="http://ad.example.com/?adid=375&pubid=1422">
    <section>
      <h1>The Mellblom Browser</h1>
      <p>Web browsing at the speed of light.</p>
      <p>No other browser goes faster!</p>
    </section>
  </a>
</aside>
```

4.6.2 The `em` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Content attributes:

Global attributes

DOM interface:Uses `HTMLElement`.

The `em` element represents stress emphasis of its contents.

The level of stress that a particular piece of content has is given by its number of ancestor `em` elements.

The placement
precise way

**Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

These
stress
. By ei
asser.

```
<p><em>Cats</em> are cute animals.</p>
```

Moving the stress to the verb, one highlights that the truth of the entire sentence is in question (maybe someone is saying cats are not cute):

```
<p>Cats <em>are</em> cute animals.</p>
```

By moving it to the adjective, the exact nature of the cats is reasserted (maybe someone suggested cats were *mean* animals):

```
<p>Cats are <em>cute</em> animals.</p>
```

Similarly, if someone asserted that cats were vegetables, someone correcting this might emphasize the last word:

```
<p>Cats are cute <em>animals</em>.</p>
```

By emphasizing the entire sentence, it becomes clear that the speaker is fighting hard to get the point across. This kind of stress emphasis also typically affects the punctuation, hence the exclamation mark here.

```
<p><em>Cats are cute animals!</em></p>
```

Anger mixed with emphasizing the cuteness could lead to markup such as:

```
<p><em>Cats are <em>cute</em> animals!</em></p>
```

The `em` element isn't a generic "italics" element. Sometimes, text is intended to stand out from the rest of the paragraph, as if it was in a different mood or voice. For this, the `i` element is more appropriate.

The `em` element also isn't intended to convey importance; for that purpose, the `strong` element is more appropriate.

4.6.3 The `strong` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

increases the importance of its contents.

Changing the importance of a piece of text with the `strong` element does not change the meaning of the sentence.

Here is an example of a warning notice in a game, with the various parts marked up according to how important they are:

```
<p><strong>Warning.</strong> This dungeon is dangerous.  
<strong>Avoid the ducks.</strong> Take any gold you find.  
<strong><strong>Do not take any of the diamonds</strong>,  
they are explosive and detonate anything within
```



[†] Too slow? Try reading the multipage copy of

4.6.4 The spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Category: /web/20130103030411/http:

F
F //whatwg.org/html
P&.,

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Uses `HTMLElement`.

The `small` element represents side comments such as small print.

Note: Small print typically features disclaimers, caveats, legal restrictions, or copyrights. Small print is also sometimes used for attribution, or for satisfying licensing requirements.

Note: The `small` element does not "de-emphasize" or lower the importance of text emphasized by the `em` element or marked as important with the `strong` element. To mark text as not emphasized or important, simply do not mark it up with the `em` or `strong` elements respectively.

The `small` element should not be used for extended spans of text, such as multiple paragraphs, lists, or sections of text. It is only intended for short runs of text. The text of a page listing terms of use, for instance, would not be a suitable candidate for the `small` element: in such a case, the text is not a side comment, it is the main content of the page.

In this example, the `small` element is used to indicate that value-added tax is not included in a price of a hotel room:

```
<dl>  
  <dt>Single room  
  <dd>199 € <small>breakfast included, VAT not included</small>  
  <dt>Double room  
  <dd>239 € <small>breakfast included, VAT not included</small>  
</dl>
```

In this second example, the `small` element is used for a side comment in an article.

```
<p>Example Corp today announced record profits for the  
second quarter <small>(Full Disclosure: Foo News is a subsidiary of  
Example Corp)</small>, leading to speculation about a third quarter  
merger with Demo Group.</p>
```

This is distinct from a sidebar, which might be multiple paragraphs long and is removed from the main flow of text. In the following example, we see a sidebar from the same article. This sidebar also has small print, indicating the source of the information in the sidebar.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
and news on a sample basis".</p>
<p><small>Information obtained from <a href="http://example.com/about.html">example.com</a> home page.</small></p>
</aside>
```

In this last example, the `small` element is marked as being *important* small print.

```
<p><strong><small>Continued use of this service will result in a kiss.</small></strong></p>
```

4.6.5 The `s` element

Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

Categories:

- F /web/20130103030411/http://whatwg.org/html

Contexts:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Uses `HTMLElement`.

The `s` element represents contents that are no longer accurate or no longer relevant.

Note: The `s` element is not appropriate when indicating document edits; to mark a span of text as having been removed from a document, use the `del` element.

In this example a recommended retail price has been marked as no longer relevant as the product in question has a new sale price.

```
<p>Buy our Iced Tea and Lemonade!</p>
<p><s>Recommended retail price: $3.99 per bottle</s></p>
<p><strong>Now selling for just $2.99 a bottle!</strong></p>
```

4.6.6 The `cite` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Uses `HTMLElement`.

The `cite` element represents the title of a work (e.g. a book, a paper, an essay, a poem, a score, a song, a script, a film, a TV show, a game, a sculpture, a painting, a theatre production, a play, an opera, a musical, an exhibition, a legal case report, etc). This can be a work that is being quoted or referenced in detail (i.e. a citation), or it can just be a work that is mentioned in passing.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

This next example shows a typical use of the `cite` element:

```
<p>My favorite book is <code><cite>The Reality Dysfunction</cite></code> by Peter F. Hamilton. My favorite comic is <code><cite>Pearls Before Swine</cite></code> by Stephan Pastis. My favorite track is <code><cite>Jive Samba</cite></code> by the Cannonball Adderley Sextet.</p>
```

This is correct usage:

```
<p>Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html
```

The `cite` element is obviously a key part of any citation in a bibliography, but it is only used to mark the title:

```
<p><code><cite>Universal Declaration of Human Rights</cite></code>, United Nations, December 1948. Adopted by General Assembly resolution 217 A (III).</p>
```

Note: A citation is not a quote (for which the `q` element is appropriate).

This is incorrect usage, because `cite` is not for quotes:

```
<p><code><cite>This is wrong!</cite></code>, said Ian.</p>
```

This is also incorrect usage, because a person is not a work:

```
<p><code><q>This is still wrong!</q>, said <code>Ian</code>.</p>
```

The correct usage does not use a `cite` element:

```
<p><code><q>This is correct</q>, said Ian.</code></p>
```

As mentioned above, the `b` element might be relevant for marking names as being keywords in certain kinds of documents:

```
<p>And then <code><b>Ian</b></code> said <code><q>this might be right, in a gossip column, maybe!</q></code>.</p>
```

4.6.7 The `q` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

- Global attributes
- `cite`

DOM interface:

Uses `HTMLQuoteElement`.

The `q` element represents some phrasing content quoted from another source.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the `cite` attribute is present, it must be a valid URL potentially surrounded by spaces. To obtain the corresponding citation link, the value of the attribute must be resolved relative to the element. User agents may allow users to follow such citation links, but they are primarily intended for private use (e.g. by server-side scripts collecting statistics about a site's use of quotations), not for readers.

The `q` element must not be used in place of quotation marks that do not represent quotes; for example, it is inappropriate to use the `q` element for marking up sarcastic statements.

The use of `q` elements to mark up quotations is entirely optional; using explicit quotation punctuation without `q` elements is just as correct.

Here **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

mission to bring the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web. I disagree with this mission.

In the following example, the quotation itself contains a quotation:

```
<p>In <cite>Example One</cite>, he writes <q>The man said <q>Things that are impossible just take longer</q>. I disagree with him</q>. Well, I disagree even more!</p>
```

In the following example, quotation marks are used instead of the `q` element:

```
<p>His best argument was “I disagree”, which I thought was laughable.</p>
```

In the following example, there is no quote — the quotation marks are used to name a word. Use of the `q` element in this case would be inappropriate.

```
<p>The word “ineffable” could have been used to describe the disaster resulting from the campaign’s mismanagement.</p>
```

4.6.8 The `dfn` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content, but there must be no `dfn` element descendants.

Content attributes:

- Global attributes
- Also, the `title` attribute has special semantics on this element.

DOM interface:

Uses `HTMLElement`.

The `dfn` element represents the defining instance of a term. The paragraph, description list group, or section that is the nearest ancestor of the `dfn` element must also contain the definition(s) for the term given by the `dfn` element.

Defining term: If the `dfn` element has a `title` attribute, then the exact value of that attribute is the term being defined. Otherwise, if it

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Note: The `title` attribute of ancestor elements does not affect `dfn` elements.

An `a` element that links to a `dfn` element represents an instance of the term defined by the `dfn` element.

In the following fragment, the term "Garage Door Opener" is first defined in the first paragraph, then used in the second. In both cases, its abbreviation is what is actually displayed.

```
<p>The <dfn><abbr title="Garage Door Opener">GDO</abbr></dfn> X
i
 Too slow? Try reading the multipage copy of
 the spec instead: https://web.archive.org
With /web/20130103030411/http:
 //whatwg.org/html
<:
<p>Teal'c activated his <a href="#gdo><abbr title="Garage Door Opener">GDO</abbr></a>
and so Hammond ordered the iris to be opened.</p>
```

4.6.9 The `abbr` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

- Global attributes
- Also, the `title` attribute has special semantics on this element.

DOM interface:

Uses `HTMLElement`.

The `abbr` element represents an abbreviation or acronym, optionally with its expansion. The `title` attribute may be used to provide an expansion of the abbreviation. The attribute, if specified, must contain an expansion of the abbreviation, and nothing else.

The paragraph below contains an abbreviation marked up with the `abbr` element. This paragraph defines the term "Web Hypertext Application Technology Working Group".

```
<p>The <dfn id=whatwg><abbr
title="Web Hypertext Application Technology Working Group">WHATWG</abbr></dfn>
is a loose unofficial collaboration of Web browser manufacturers and
interested parties who wish to develop new technologies designed to
allow authors to write and deploy Applications over the World Wide
Web.</p>
```

An alternative way to write this would be:

```
<p>The <dfn id=whatwg>Web Hypertext Application Technology
Working Group</dfn> (<abbr
title="Web Hypertext Application Technology Working Group">WHATWG</abbr>)
is a loose unofficial collaboration of Web browser manufacturers and
interested parties who wish to develop new technologies designed to
allow authors to write and deploy Applications over the World Wide
Web.</p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

started working on HTML5 in 2004.</p>

This paragraph links an abbreviation to its definition.

```
<p>The <a href="#whatwg"><abbr title="Web Hypertext Application Technology Working Group">WHATWG</abbr></a> community does not have much representation from Asia.</p>
```

This paragraph marks up an abbreviation without giving an expansion, possibly as a hook to apply styles for abbreviations (e.g. smallcaps)

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If an abbreviation is used, it is the grammatical number of the

contents of the element.

Here is an example:

```
<p>Two <abbr title="Working Group">WG</abbr>s worked on this specification: the <abbr>WHATWG</abbr> and the <abbr>HTMLWG</abbr>.</p>
```

Here the plural is inside the element, so the expansion is in the plural:

```
<p>Two <abbr title="Working Groups">WGs</abbr> worked on this specification: the <abbr>WHATWG</abbr> and the <abbr>HTMLWG</abbr>.</p>
```

Abbreviations do not have to be marked up using this element. It is expected to be useful in the following cases:

- Abbreviations for which the author wants to give expansions, where using the `abbr` element with a `title` attribute is an alternative to including the expansion inline (e.g. in parentheses).
- Abbreviations that are likely to be unfamiliar to the document's readers, for which authors are encouraged to either mark up the abbreviation using an `abbr` element with a `title` attribute or include the expansion inline in the text the first time the abbreviation is used.
- Abbreviations whose presence needs to be semantically annotated, e.g. so that they can be identified from a style sheet and given specific styles, for which the `abbr` element can be used without a `title` attribute.

Providing an expansion in a `title` attribute once will not necessarily cause other `abbr` elements in the same document with the same contents but without a `title` attribute to behave as if they had the same expansion. Every `abbr` element is independent.

4.6.10 The `data` element

Categories:

Flow content.
Phrasing content.
Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes
`value`

DOM interface:

```
IDL interface HTMLDataElement : HTMLElement {
    attribute DOMString value;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `value` attribute must be present. Its value must be a representation of the element's contents in a machine-readable format.

Note: When the value is date- or time-related, the more specific `time` element can be used instead.

The element can be used for several purposes.

When combined with microformats or the microdata attributes defined in this specification, the element serves to provide both a machine-readable value for the purposes of data processors, and a human-readable value for the purposes of rendering in a Web browser. In this case, the format to be used in the `value` attribute is determined by the microformat or microdata vocabulary in use.

The element **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

The `value` | /web/20130103030411/http:

Here on each [/whatwg.org/html](http://whatwg.org/html)

ral value to store alongside a data-* attributes can also

provide a sorting mechanism imposed form in another.

```
<table sortable>
  <thead> <tr> <th> Game <th> Corporations <th> Map Size
  <tbody>
    <tr> <td> 1830 <td> <data value="8">Eight</data> <td> <data value="93">19+74 hexes (93 total)</data>
    <tr> <td> 1856 <td> <data value="11">Eleven</data> <td> <data value="99">12+87 hexes (99 total)</data>
    <tr> <td> 1870 <td> <data value="10">Ten</data> <td> <data value="149">4+145 hexes (149 total)</data>
  </tbody>
```

4.6.11 The `time` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

- Where phrasing content is expected.

Content model:

- Phrasing content.

Content attributes:

- Global attributes
- `datetime`

DOM interface:

```
IDL interface HTMLTimeElement : HTMLElement {
  attribute DOMString datetime;
};
```

The `time` element represents its contents, along with a machine-readable form of those contents in the `datetime` attribute. The kind of content is limited to various kinds of dates, times, time-zone offsets, and durations, as described below.

The `datetime` attribute may be present. If present, its value must be a representation of the element's contents in a machine-readable format.

A `time` element that does not have a `datetime` content attribute must not have any element descendants.

The `datetime` value of a `time` element is the value of the element's `datetime` content attribute, if it has one, or the element's `textContent`, if it does not.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

A valid date string

```
|| <time>2011-11-12</time>
```

A valid yearless date string

```
|| <time>11-12</time>
```

A valid time string

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

A valid local date and time string

```
|| <time>2011-11-12T14:54:39</time>
|| <time>2011-11-12T14:54:39.929</time>
|| <time>2011-11-12 14:54</time>
|| <time>2011-11-12 14:54:39</time>
|| <time>2011-11-12 14:54:39.929</time>
```

A valid time-zone offset string

```
|| <time>Z</time>
|| <time>+0000</time>
|| <time>+00:00</time>
|| <time>-0800</time>
|| <time>-08:00</time>
```

A valid global date and time string

```
|| <time>2011-11-12T14:54Z</time>
|| <time>2011-11-12T14:54:39Z</time>
|| <time>2011-11-12T14:54:39.929Z</time>
|| <time>2011-11-12T14:54+0000</time>
|| <time>2011-11-12T14:54:39+0000</time>
|| <time>2011-11-12T14:54:39.929+0000</time>
|| <time>2011-11-12T14:54+00:00</time>
|| <time>2011-11-12T14:54:39+00:00</time>
|| <time>2011-11-12T14:54:39.929+00:00</time>
|| <time>2011-11-12T06:54-0800</time>
|| <time>2011-11-12T06:54:39-0800</time>
|| <time>2011-11-12T06:54:39.929-0800</time>
|| <time>2011-11-12T06:54-08:00</time>
|| <time>2011-11-12T06:54:39-08:00</time>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
|| <time>2011-11-12 14:54:39.929Z</time>
|| <time>2011-11-12 14:54+0000</time>
|| <time>2011-11-12 14:54:39+0000</time>
|| <time>2011-11-12 14:54:39.929+0000</time>
|| <time>2011-11-12 14:54+00:00</time>
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
|| ~<time>2011-11-12 00:54-00.00~</time>
|| <time>2011-11-12 06:54:39-08:00</time>
|| <time>2011-11-12 06:54:39.929-08:00</time>
```

A valid week string

```
|| <time>2011-W46</time>
```

Four or more ASCII digits, at least one of which is not U+0030 DIGIT ZERO (0)

```
|| <time>2011</time>
|| <time>0001</time>
```

A valid duration string

```
|| <time>PT4H18M3S</time>
|| <time>4h 18m 3s</time>
```

The **machine-readable equivalent of the element's contents** must be obtained from the element's datetime value by using the following algorithm:

1. If parsing a month string from the element's datetime value returns a month, that is the machine-readable equivalent; abort these steps.
2. If parsing a date string from the element's datetime value returns a date, that is the machine-readable equivalent; abort these steps.
3. If parsing a yearless date string from the element's datetime value returns a yearless date, that is the machine-readable equivalent; abort these steps.
4. If parsing a time string from the element's datetime value returns a time, that is the machine-readable equivalent; abort these steps.
5. If parsing a local date and time string from the element's datetime value returns a local date and time, that is the machine-readable equivalent; abort these steps.
6. If parsing a time-zone offset string from the element's datetime value returns a time-zone offset, that is the machine-readable equivalent; abort these steps.
7. If parsing a global date and time string from the element's datetime value returns a global date and time, that is the machine-readable equivalent; abort these steps.
8. If parsing a week string from the element's datetime value returns a week, that is the machine-readable equivalent; abort these steps.
9. If the element's datetime value consists of only ASCII digits, at least one of which is not U+0030 DIGIT ZERO (0), then the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

11. There is no machine-readable equivalent.

Note: The algorithms referenced above are intended to be designed such that for any arbitrary string *s*, only one of the algorithms returns a value. A more efficient approach might be to create a single algorithm that parses all these data types in one pass; developing such an algorithm is left as an exercise to the reader.

The `datetime` IDL attribute must reflect the content attribute of the same name.

The `time` element can be used to encode dates, for example in microformats. The following shows a hypothetical way of encoding:

Too slow? Try reading the multipage copy of the spec instead: ['](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Here, a fictional microdata vocabulary based on the Atom vocabulary is used with the `time` element to mark up a blog post's publication date.

```
<article itemscope itemtype="http://n.example.org/rfc4287">
  <h1 itemprop="title">Big tasks</h1>
  <footer>Published <time itemprop="published" datetime="2009-08-29">two days ago</time>.
  </footer>
  <p itemprop="content">Today, I went out and bought a bike for my kid.</p>
</article>
```

In this example, another article's publication date is marked up using `time`, this time using the schema.org microdata vocabulary:

```
<article itemscope itemtype="http://schema.org/BlogPosting">
  <h1 itemprop="headline">Small tasks</h1>
  <footer>Published <time itemprop="datePublished" datetime="2009-08-30">yesterday</time>.
  </footer>
  <p itemprop="articleBody">I put a bike bell on his bike.</p>
</article>
```

In the following snippet, the `time` element is used to encode a date in the ISO8601 format, for later processing by a script:

```
<p>Our first date was <time datetime="2006-09-23">a Saturday</time>.</p>
```

In this second snippet, the value includes a time:

```
<p>We stopped talking at <time datetime="2006-09-24T05:00-07:00">5am the next morning</time>.</p>
```

A script loaded by the page (and thus privy to the page's internal convention of marking up dates and times using the `time` element) could scan through the page and look at all the `time` elements therein to create an index of dates and times.

For example, this element conveys the string "Tuesday" with the additional semantic that the 12th of November 2011 is the meaning that corresponds to "Tuesday":

```
Today is <time datetime="2011-11-12">Tuesday</time>.
```

In this example, a specific time in the Pacific Standard Time timezone is specified:

```
Your next meeting is at <time datetime="2011-11-12T15:00-08:00">3pm</time>.
```

4.6.12 The `code` element

Categories:

- Flow content.
- Phrasing content.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Uses HTMLElement.

The code element represents a fragment of computer code. This could be an XML element name, a filename, a computer program, or any other string.

Although the code element is intended to be used with the lang attribute, it can also be used with the "language" attribute.

The following example shows how the code element can be used. It includes a class prefix with the language attribute.

```
<p>
<code>
<code>
```

```
<p>When you call the <code>activate()</code> method on the
<code>robotSnowman</code> object, the eyes glow.</p>
```

```
<p>The example below uses the <code>begin</code> keyword to indicate
the start of a statement block. It is paired with an <code>end</code>
keyword, which is followed by the <code>.;</code> punctuation character
(full stop) to indicate the end of the program.</p>
```

wish to mark code elements
with a class prefixed with

nes and computer code,

The following example shows how a block of code could be marked up using the pre and code elements.

```
<pre><code class="language-pascal">var i: Integer;
begin
  i := 1;
end.</code></pre>
```

A class is used in that example to indicate the language used.

Note: See the pre element for more details.

4.6.13 The var element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Uses HTMLElement.

The var element represents a variable. This could be an actual variable in a mathematical expression or programming context, an identifier representing a constant, a symbol identifying a physical quantity, a function parameter, or just be a term used as a placeholder in prose.

In the paragraph below, the letter "n" is being used as a variable in prose:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

element can still be used to refer to specific variables that are then mentioned in MathML expressions.

In this example, an equation is shown, with a legend that references the variables in the equation. The expression itself is marked up with MathML, but the variables are mentioned in the figure's legend using var.

```
<figure>
<math>
<mi>a</mi>
<mo>=</mo>
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<math>
<mi>a</mi>
<mo>=</mo>
<mi>b</mi>
<mo>*</mo>
<mi>c</mi>
<mo>*</mo>
<mi>d</mi>
<mo>*</mo>
<mi>e</mi>
<mo>*</mo>
<mi>f</mi>
```

)f

Here, the equation describing mass-energy equivalence is used in a sentence, and the var element is used to mark the variables and constants in that equation:

<p>Then he turned to the blackboard and picked up the chalk. After a few moment's thought, he wrote <var>E</var> = <var>m</var> <var>c</var>². The teacher looked pleased.</p>

4.6.14 The samp element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

- Where phrasing content is expected.

Content model:

- Phrasing content.

Content attributes:

- Global attributes

DOM interface:

- Uses HTMLElement.

The samp element represents (sample) output from a program or computing system.

Note: See the pre and kbd elements for more details.

This example shows the samp element being used inline:

```
<p>The computer said <samp>Too much cheese in tray
two</samp> but I didn't know what that meant.</p>
```

This second example shows a block of sample output. Nested samp and kbd elements allow for the styling of specific elements of the sample output using a style sheet.

```
<pre><samp><span class="prompt">jdoe@mowmow:~$</span> <kbd>ssh demo.example.com</kbd>
Last login: Tue Apr 12 09:10:17 2005 from mowmow.example.com on pts/1
Linux demo 2.6.10-grsec+gg3+e+fhs6b+nfs+gr0501+++p3+c4a+gr2b-reslog-v6.189 #1 SMP Tue Feb 1
11:22:36 PST 2005 i686 unknown
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phr



Content: Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

DOM interface: /web/20130103030411/http://whatwg.org/html

The `kbd` element represents input based on system output, such as voice commands).

When the `kbd` element is nested inside a `samp` element, it represents the input as it was echoed by the system.

When the `kbd` element contains a `samp` element, it represents input based on system output, for example invoking a menu item.

When the `kbd` element is nested inside another `kbd` element, it represents an actual key or other single unit of input as appropriate for the input mechanism.

Here the `kbd` element is used to indicate keys to press:

```
<p>To make George eat an apple, press <code><ctrl><shift>F3</code></p>
```

In this second example, the user is told to pick a particular menu item. The outer `kbd` element marks up a block of input, with the inner `kbd` elements representing each individual step of the input, and the `samp` elements inside them indicating that the steps are input based on something being displayed by the system, in this case menu labels:

```
<p>To make George eat an apple, select  
<code><ctrl><shift><samp>File</samp></code> | <code><ctrl><shift>A</code></p>
```

Such precision isn't necessary; the following is equally fine:

```
<p>To make George eat an apple, select <code>File | Eat Apple...</code></p>
```

4.6.16 The `sub` and `sup` elements

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:

Use `HTMLElement`.

The `sup` element represents a superscript and the `sub` element represents a subscript.

These elements must be used only to mark up typographical conventions with specific meanings, not for typographical presentation for presentation's sake. For example, it would be inappropriate for the `sub` and `sup` elements to be used in the name of the LaTeX

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
<p>The most beautiful women are
<span lang="fr"><abbr>Mlle</abbr> Gwendoline</span> and
<span lang="fr"><abbr>Mme</abbr> Denise</span>.</p>
```

The `sub` element can be used inside a `var` element, for variables that have subscripts.

Here, the `sub` element is used to represents the subscript that identifies the variable in a family of variables:

```
<p>The coordinate of the <var>i</var>th point is
(<var>x<sub>i</sub></var>, <var>y<sub>i</sub></var>).</p>
```

F

Too slow? Try reading the multipage copy of

Mathematical notation but authors

or marking up mathematics,

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`i</var>_i</var>`

4.6.17 The `i` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

- Phrasing content.

Content attributes:

- Global attributes

DOM interface:

Uses `HTMLElement`.

The `i` element represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose in a manner indicating a different quality of text, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, transliteration, a thought, or a ship name in Western texts.

Terms in languages different from the main text should be annotated with `lang` attributes (or, in XML, `lang` attributes in the XML namespace).

The examples below show uses of the `i` element:

```
<p>The <i class="taxonomy">Felis silvestris catus</i> is cute.</p>
<p>The term <i>prose content</i> is defined above.</p>
<p>There is a certain <i lang="fr">je ne sais quoi</i> in the air.</p>
```

In the following example, a dream sequence is marked up using `i` elements.

```
<p>Raymond tried to sleep.</p>
<p><i>The ship sailed away on Thursday</i>, he
dreamt. <i>The ship had many people aboard, including a beautiful
princess called Carey. He watched her, day-in, day-out, hoping she
would notice him, but she never did.</i></p>
<p><i>Finally one night he picked up the courage to speak with
her</i></p>
<p>Raymond woke with a start as the fire alarm rang out.</p>
```

Authors can use the `class` attribute on the `i` element to identify why the element is being used, so that if the style of a particular use (e.g. dream sequences as opposed to taxonomic terms) is to be changed at a later date, the author doesn't have to go through the entire document (or series of related documents) annotating each use.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

case that content in **i** elements will necessarily be italicized.

4.6.18 The **b** element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.



Contexts

- v Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
- F Content : /web/20130103030411/http://whatwg.org/html
- C Content : //whatwg.org/html

DOM interface

Uses `HTMLElement`.

The **b** element represents a span of text to which attention is being drawn for utilitarian purposes without conveying any extra importance and with no implication of an alternate voice or mood, such as key words in a document abstract, product names in a review, actionable words in interactive text-driven software, or an article lede.

The following example shows a use of the **b** element to highlight key words without marking them up as important:

```
<p>The <b>frobonitor</b> and <b>barbinator</b> components are fried.</p>
```

In the following example, objects in a text adventure are highlighted as being special by use of the **b** element.

```
<p>You enter a small room. Your <b>sword</b> glows  
brighter. A <b>rat</b> scurries past the corner wall.</p>
```

Another case where the **b** element is appropriate is in marking up the lede (or lead) sentence or paragraph. The following example shows how a BBC article about kittens adopting a rabbit as their own could be marked up:

```
<article>  
  <h2>Kittens 'adopted' by pet rabbit</h2>  
  <p><b class="lede">Six abandoned kittens have found an  
unexpected new mother figure – a pet rabbit.</b></p>  
  <p>Veterinary nurse Melanie Humble took the three-week-old  
kittens to her Aberdeen home.</p>  
  [...]
```

As with the **i** element, authors can use the `class` attribute on the **b** element to identify why the element is being used, so that if the style of a particular use is to be changed at a later date, the author doesn't have to go through annotating each use.

The **b** element should be used as a last resort when no other element is more appropriate. In particular, headings should use the **h1** to **h6** elements, stress emphasis should use the **em** element, importance should be denoted with the **strong** element, and text marked or highlighted should use the **mark** element.

The following would be *incorrect* usage:

```
<p><b>WARNING!</b> Do not frob the barbinator!</p>
```

In the previous example, the correct element to use would have been **strong**, not **b**.

Note: Style sheets can be used to format **b** elements, just like any other element can be restyled. Thus, it is not the case that content in **b** elements will necessarily be boldened.

4.6.19 The **u** element

Categories:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:Uses `HTMLElement`

The `u` element is used to mark text as being misspelt. In most cases, marking key titles, the `ci` element should be used; for labeling ship

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

itation, such as labeling the misspelt.

ent should be used; for context; for marking book it should be used; for

Note: The default rendering of the `u` element in visual presentations clashes with the conventional rendering of hyperlinks (underlining). Authors are encouraged to avoid using the `u` element where it could be confused for a hyperlink.

4.6.20 The `mark` element**Categories:**

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Phrasing content.

Content attributes:

Global attributes

DOM interface:Uses `HTMLElement`.

The `mark` element represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context. When used in a quotation or other block of text referred to from the prose, it indicates a highlight that was not originally present but which has been added to bring the reader's attention to a part of the text that might not have been considered important by the original author when the block was originally written, but which is now under previously unexpected scrutiny. When used in the main prose of a document, it indicates a part of the document that has been highlighted due to its likely relevance to the user's current activity.

This example shows how the `mark` element can be used to bring attention to a particular part of a quotation:

```
<p lang="en-US">Consider the following quote:</p>
<blockquote lang="en-GB">
  <p>Look around and you will find, no-one's really
  <mark>colour</mark> blind.</p>
</blockquote>
<p lang="en-US">As we can tell from the <em>spelling</em> of the word,
the person writing this quote is clearly not American.</p>
```

(If the goal was to mark the element as misspelt, however, the `u` element, possibly with a class, would be more appropriate.)

Another example of the `mark` element is highlighting parts of a document that are matching some search string. If someone looked at a document, and the server knew that the user was searching for the word "kitten", then the server might return the document with one paragraph modified as follows:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<p>The highlighted part below is where the error lies:</p>
<pre><code>var i: Integer;
begin
  i := <mark>1.1</mark>;
end.</code></pre>
```

This is separate from *syntax highlighting*, for which `span` is more appropriate. Combining both, one would get:

```
<p>The highlighted part below is where the error lies.</p>
<span>
  Too slow? Try reading the multipage copy of
  the spec instead: https://web.archive.org
  /web/20130103030411/http:
</span>
```

This
exam

ly not emphasized. In this
quotes to render in italics.

```
<article>
  <style scoped>
    blockquote mark, q mark {
      font: inherit; font-style: italic;
      text-decoration: none;
      background: transparent; color: inherit;
    }
    .bubble em {
      font: inherit; font-size: larger;
      text-decoration: underline;
    }
  </style>
  <h1>She knew</h1>
  <p>Did you notice the subtle joke in the joke on panel 4?</p>
  <blockquote>
    <p class="bubble">I didn't <em>want</em> to believe. <mark>Of course
    on some level I realized it was a known-plaintext attack.</mark> But I
    couldn't admit it until I saw for myself.</p>
  </blockquote>
  <p>(Emphasis mine.) I thought that was great. It's so pedantic, yet it
  explains everything neatly.</p>
</article>
```

Note, incidentally, the distinction between the `em` element in this example, which is part of the original text being quoted, and the `mark` element, which is highlighting a part for comment.

The following example shows the difference between denoting the *importance* of a span of text (`strong`) as opposed to denoting the *relevance* of a span of text (`mark`). It is an extract from a textbook, where the extract has had the parts relevant to the exam highlighted. The safety warnings, important though they may be, are apparently not relevant to the exam.

```
<h3>Wormhole Physics Introduction</h3>

<p><mark>A wormhole in normal conditions can be held open for a
maximum of just under 39 minutes.</mark> Conditions that can increase
the time include a powerful energy source coupled to one or both of
the gates connecting the wormhole, and a large gravity well (such as a
black hole).</p>

<p><mark>Momentum is preserved across the wormhole. Electromagnetic
radiation can travel in both directions through a wormhole,
but matter cannot.</mark></p>

<p>When a wormhole is created, a vortex normally forms.
<strong>Warning: The vortex caused by the wormhole opening will
annihilate anything in its path.</strong> Vortexes can be avoided when
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

4.6.21 The `ruby` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

W
Content: Too slow? Try reading the multipage copy of
S the spec instead: <https://web.archive.org>
Content: [/web/20130103030411/http://whatwg.org/html](https://web/20130103030411/http://whatwg.org/html)
DOM interface: [U](https://web/20130103030411/http://whatwg.org/html)



The `ruby` element allows one or more spans of phrasing content to be marked with ruby annotations. Ruby annotations are short runs of text presented alongside base text, primarily used in East Asian typography as a guide for pronunciation or to include other annotations. In Japanese, this form of typography is also known as *furigana*.

The content model of `ruby` elements consists of one or more of the following sequences:

1. One or the other of the following:
 - Phrasing content, but with no `ruby` elements and with no `ruby` element descendants
 - A single `ruby` element that itself has no `ruby` element descendants
2. One or the other of the following:
 - One or more `rt` elements
 - An `rp` element followed by one or more `rt` elements, each of which is itself followed by an `rp` element

The `ruby` and `rt` elements can be used for a variety of kinds of annotations, including in particular (though by no means limited to) those described below. For more details on Japanese Ruby in particular, and how to render Ruby for Japanese, see *Requirements for Japanese Text Layout*. [JLREQ]

Note: At the time of writing, CSS does not yet provide a way to fully control the rendering of the HTML `ruby` element. It is hoped that CSS will be extended to support the styles described below in due course.

Mono-ruby for individual base characters in Japanese

One or more hiragana or katakana characters (the ruby annotation) are placed with each ideographic character (the base text). This is used to provide readings of kanji characters.

```
<ruby>B<rt>annotation</ruby>
```

In this example, notice how each annotation corresponds to a single base character.

```
<ruby>君<rt>く</rt>ん</ruby><ruby>は<rt>は</rt>は</ruby><ruby>和<rt>わ</rt>和</ruby><ruby>し<rt>し</rt>し</ruby><ruby>テ<rt>テ</rt>テ</ruby><ruby>同<rt>同</rt>同</ruby>どう<ruby>ぜ<rt>ぜ</rt>ぜ</ruby>ぜず。
```

くん し わ どう
君子は和して同ぜず。

This example can also be written as follows, using one `ruby` element with two segments of base text and two annotations (one for each) rather than two back-to-back `ruby` elements each with one base text segment and annotation (as in the markup above):

```
<ruby>君<rt>く</rt>ん</ruby><ruby>は<rt>は</rt>は</ruby><ruby>和<rt>わ</rt>和</ruby><ruby>し<rt>し</rt>し</ruby><ruby>テ<rt>テ</rt>テ</ruby><ruby>同<rt>同</rt>同</ruby>どう<ruby>ぜ<rt>ぜ</rt>ぜ</ruby>ぜず。
```

Mono-ruby for compound words (jukugo)

This is similar to the previous case: each ideographic character in the compound word (the base text) has its reading given in hiragana or katakana characters (the ruby annotation). The difference is that the base text segments form a compound word

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

compound word (jukugo) corresponds to a single `ruby` element.

The rendering here is expected to be that each annotation be placed over (or next to, in vertical text) the corresponding base character, with the annotations not overhanging any of the adjacent characters.

```
<ruby>鬼<rt>き</rt>門<rt>もん</rt></ruby>の<ruby>方<rt>ほう</rt>角<rt>かく</rt>が<rt>が</rt></ruby>を
<ruby>凝<rt>ぎょう</rt>視<rt>し</rt></ruby>する
```

き もん ほ う が く ぎ ょ う し
鬼 門 の 方 角 を 凝 視 す る



Jukugo-ruby

This
read
renc

Too slow? Try reading the multipage copy of
the spec instead: [https://web.archive.org
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

se compound word has its
complicated Jukugo Ruby

ering is expected to be

が < /rt > </ruby > を

Note: For more details on Jukugo Ruby rendering, see Appendix F in the Requirements for Japanese Text Layout. [JLREQ]

Group ruby for describing meanings

The annotation describes the meaning of the base text, rather than (or in addition to) the pronunciation. As such, both the base text and the annotation can be multiple characters long.

```
<ruby>BASE<rt>annotation</ruby>
```

Here a compound ideographic word has its corresponding katakana given as an annotation.

```
<ruby>境界面<rt>インターフェース</ruby>
```

インターフェース
境 界 面

Here a compound ideographic word has its translation in English provided as an annotation.

```
<ruby lang="ja">編集者<rt lang="en">editor</ruby>
```

editor
編集者

Group ruby for Jukujiki readings

A phonetic reading that corresponds to multiple base characters, because a one-to-one mapping would be difficult. (In English, the words "Colonel" and "Lieutenant" are examples of words where a direct mapping of pronunciation to individual letters is, in some dialects, rather unclear.)

In this example, the name of a species of flowers has a phonetic reading provided using group ruby:

```
<ruby>紫陽花<rt>あじさい</ruby>
```

あじさい
紫陽花

Text with both phonetic and semantic annotations (double-sided ruby)

Sometimes, ruby styles described above are combined.

If this results in two annotations covering the same single base segment, then the annotations can just be placed back to back.

```
<ruby>BASE<rt>annotation 1<rt>annotation 2</ruby>
```

```
<ruby>B<rt>a<rt>a</ruby><ruby>A<rt>a<rt>a</ruby><ruby>S<rt>a<rt>a</ruby>
```

```
<ruby>E<rt>a<rt>a</ruby>
```

In this contrived example, some symbols are given names in English and French.

```
<ruby>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

In more complicated situations such as following examples, a nested `ruby` element is used to give the inner annotations, and then that whole `ruby` is then given an annotation at the "outer" level.

```
<ruby><ruby>B<rt>a</rt>A<rt>n</rt>S<rt>t</rt>E<rt>n</rt></ruby><rt>annotation</ruby>
```

Here both a phonetic reading and the meaning are given in ruby annotations. The annotation on the nested `ruby` element gives a mono-ruby phonetic annotation for each base character, while the annotation in the `rt` element that is a child of the outer `ruby` element gives the meaning using hiragana.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

東南の方角

Within a `ruby` element that does not have a `ruby` element ancestor, content is segmented and segments are placed into three categories: base text segments, annotation segments, and ignored segments. Ignored segments do not form part of the document's semantics (they consist of some inter-element whitespace and `rp` elements, the latter of which are used for legacy user agents that do not support ruby at all). Base text segments can overlap (with a limit of two segments overlapping any one position in the DOM, and with any segment having an earlier start point than an overlapping segment also having an equal or later end point, and any segment have a later end point than an overlapping segment also having an equal or earlier start point). Annotation segments correspond to `rt` elements. Each annotation segment can be associated with a base text segment, and each base text segment can have annotation segments associated with it. (In a conforming document, each base text segment is associated with at least one annotation segment, and each annotation segment is associated with one base text segment.) A `ruby` element represents the union of the segments of base text it contains, along with the mapping from those base text segments to annotation segments. Segments are described in terms of DOM ranges; annotation segment ranges always consist of exactly one element. [DOM]

At any particular time, the segmentation and categorisation of content of a `ruby` element is the result that would be obtained from running the following algorithm:

1. Let *base text segments* be an empty list of base text segments, each potentially with a list of base text subsegments.
2. Let *annotation segments* be an empty list of annotation segments, each potentially being associated with a base text segment or subsegment.
3. Let *root* be the `ruby` element for which the algorithm is being run.
4. If *root* has a `ruby` element ancestor, then jump to the step labeled *end*.
5. Let *current parent* be *root*.
6. Let *index* be 0.
7. Let *start index* be null.
8. Let *parent start index* be null.
9. Let *current base text* be null.
10. *Start mode*: If *index* is equal to or greater than the number of child nodes in *current parent*, then jump to the step labeled *end mode*.
 11. If the *index*th node in *current parent* is an `rt` or `rp` element, jump to the step labeled *annotation mode*.
 12. Set *start index* to the value of *index*.
 13. *Base mode*: If the *index*th node in *current parent* is a `ruby` element, and if *current parent* is the same element as *root*, then push a ruby level and then jump to the step labeled *start mode*.
 14. If the *index*th node in *current parent* is an `rt` or `rp` element, then set the current base text and then jump to the step labeled

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

step labeled *end mode*.

17. Jump back to the step labeled *base mode*.
18. *Annotation mode*: If the *indexth* node in *current parent* is an *rt* element, then push a ruby annotation and jump to the step labeled *annotation mode increment*.
19. If the *indexth* node in *current parent* is an *rp* element, jump to the step labeled *annotation mode increment*.
20. If the *indexth* node in *current parent* is not a *Text* node, or is a *Text* node that is not inter-element whitespace, then jump to the step labeled *base mode*.
21. *Annotate* the *indexth* node in *current parent* then jump to the step labeled *annotation mode increment*.
22. *Annotate* the *indexth* node in *current parent* then jump to the step labeled *annotation mode white-space skipper*.
23. If the *indexth* node in *current parent* is not a *Text* node, or is a *Text* node that is not inter-element whitespace, then jump to the step labeled *base mode*.
24. If the *lookahead index* node in *current parent* is not a *Text* node, or is a *Text* node that is not inter-element whitespace, then jump to the step labeled *base mode* (without further incrementing *index*, so the inter-element whitespace seen so far becomes part of the next base text segment).
25. Increment *lookahead index* by one.
26. Jump to the step labeled *annotation mode white-space skipper*.
27. *End mode*: If *current parent* is not the same element as *root*, then pop a ruby level and jump to the step labeled *base mode post-increment*.
28. *End*: Return *base text segments* and *annotation segments*. Any content of the *ruby* element not described by segments in either of those lists is implicitly in an *ignored segment*.

When the steps above say to **set the current base text**, it means to run the following steps at that point in the algorithm:

1. Let *text range* a DOM range whose start is the boundary point (*current parent*, *start index*) and whose end is the boundary point (*current parent*, *index*).
2. Let *new text segment* be a base text segment described by the range *annotation range*.
3. Add *new text segment* to *base text segments*.
4. Let *current base text* be *new text segment*.
5. Let *start index* be null.

When the steps above say to **push a ruby level**, it means to run the following steps at that point in the algorithm:

1. Let *current parent* be the *indexth* node in *current parent*.
2. Let *index* be 0.
3. Set *saved start index* to the value of *start index*.
4. Let *start index* be null.

When the steps above say to **pop a ruby level**, it means to run the following steps at that point in the algorithm:

1. Let *index* be the position of *current parent* in *root*.
2. Let *current parent* be *root*.
3. Increment *index* by one.
4. Set *start index* to the value of *saved start index*.
5. Let *saved start index* be null.

When the steps above say to **push a ruby annotation**, it means to run the following steps at that point in the algorithm:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. Let *new annotation segment* be an annotation segment described by the range *annotation range*.4. If *current base text* is not null, associate *new annotation segment* with *current base text*.5. Add *new annotation segment* to *annotation segments*.

In this example, each ideograph in the Japanese text 漢字 is annotated with its reading in hiragana.

<ruby>漢<rt>かん</rt>字<rt>じ</rt></ruby>



This Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

In this example, each ideograph in the traditional Chinese text 漢字 is annotated with its bopomofo reading.

<ruby>漢<rt>ㄏㄢˋ</rt>字<rt>ㄔˋ</rt></ruby>

This might be rendered as:

漢 ㄏ
字 ㄔ

In this example, each ideograph in the simplified Chinese text 汉字 is annotated with its pinyin reading.

...<ruby>汉<rt>hàn</rt>字<rt>zì</rt></ruby>...

This might be rendered as:

hàn zì
... 汉字 ...

In this more contrived example, the acronym "HTML" has four annotations: one for the whole acronym, briefly describing what it is, one for the letters "HT" expanding them to "Hypertext", one for the letter "M" expanding it to "Markup", and one for the letter "L" expanding it to "Language".

```
<ruby>
  <ruby>HT<rt>Hypertext</rt>M<rt>Markup</rt>L<rt>Language</rt></ruby>
  <rt>An abstract language for describing documents and applications
</ruby>
```

4.6.22 The `rt` element

Categories:

None.

Contexts in which this element can be used:

As a child of a `ruby` element.

Content model:

Phrasing content.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `rt` element marks the ruby text component of a ruby annotation. When it is the child of a `ruby` element, it doesn't represent anything itself, but the `ruby` element uses it as part of determining what *it* represents.

An `rt` element that is not a child of a `ruby` element represents the same thing as its children.

4.6.23 The `rp` element


Categories:

- ↳ Too slow? Try reading the multipage copy of

Contexts:
A the spec instead: <https://web.archive.org/web/20130103030411/http://www.whatwg.org/html/>

Content:
I /whatwg.org/html

Content a:

Global attributes

DOM interface:

Uses `HTMLElement`.

The `rp` element can be used to provide parentheses around a ruby text component of a ruby annotation, to be shown by user agents that don't support ruby annotations.

An `rp` element that is a child of a `ruby` element represents nothing. An `rp` element whose parent element is not a `ruby` element represents its children.

The example above, in which each ideograph in the text 漢字 is annotated with its phonetic reading, could be expanded to use `rp` so that in legacy user agents the readings are in parentheses:

```
... <ruby>漢<rp> (</rp><rt>かん</rt><rp>) </rp>字<rp> (</rp><rt>じ</rt><rp>) </rp></ruby>
...
```

In conforming user agents the rendering would be as above, but in user agents that do not support ruby, the rendering would be:

```
... 漢（かん）字（じ）...
```

When there are multiple annotations for a segment, `rp` elements can also be placed between the annotations. Here is another copy of an earlier contrived example showing some symbols with names given in English and French, but this time with `rp` elements as well:

```
<ruby>
  ♥<rp>: </rp><rt>Heart</rt><rp>, </rp><rt lang=fr>Cœur</rt><rp>. </rp>
  ♣<rp>: </rp><rt>Shamrock</rt><rp>, </rp><rt lang=fr>Trèfle</rt><rp>. </rp>
  *<rp>: </rp><rt>Star</rt><rp>, </rp><rt lang=fr>Étoile</rt><rp>. </rp>
</ruby>
```

This would make the example render as follows in non-ruby-capable user agents:

```
♥: Heart, Cœur. ♣: Shamrock, Trèfle. *: Star, Étoile.
```

4.6.24 The `bdi` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

Where phrasing content is expected.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4.6.26 The span element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

Contexts in which this element can be used:

- Content of [the spec instead: https://web.archive.org](#)
- Content of [/web/20130103030411/http://whatwg.org/html](#)
- DOM interface

IDL interface `HTMLSpanElement` : `HTMLElement` {};

The `span` element doesn't mean anything on its own, but can be useful when used together with the global attributes, e.g. `class`, `lang`, or `dir`. It represents its children.

In this example, a code fragment is marked up using `span` elements and `class` attributes so that its keywords and identifiers can be color-coded from CSS:

```
<pre><code class="lang-c"><span class="keyword">for</span> (<span class="ident">j</span> = 0; <span class="ident">j</span> &lt; 256; <span class="ident">j</span>++) { <span class="ident">i_t3</span> = (<span class="ident">i_t3</span> & 0xffff) | (<span class="ident">j</span> &lt;&lt; 17); <span class="ident">i_t6</span> = (((((<span class="ident">i_t3</span> >> 3) ^ <span class="ident">i_t3</span>) >> 1) ^ <span class="ident">i_t3</span>) >> 8) ^ <span class="ident">i_t3</span>) >> 5) & 0xff; <span class="keyword">if</span> (<span class="ident">i_t6</span> == <span class="ident">i_t1</span>) <span class="keyword">break</span>; }</code></pre>
```

4.6.27 The br element

Categories:

- Flow content.
- Phrasing content.

Contexts in which this element can be used:

- Where phrasing content is expected.

Content model:

- Empty.

Content attributes:

- Global attributes

DOM interface:

IDL interface `HTMLBRElement` : `HTMLElement` {};

The `br` element represents a line break.

Note: While line breaks are usually represented in visual media by physically moving subsequent text to a new line, a

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The following example is correct usage of the `br` element:

```
<p>P. Sherman<br>
42 Wallaby Way<br>
Sydney</p>
```

`br` elements must not be used for separating thematic groups in a paragraph.

The following examples are non-conforming, as they abuse the `br` element:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Here

```
<label>Name: <input name="name"></label></p>
<label>Address: <input name="address"></label></p>
```

If a paragraph consists of nothing but a single `br` element, it represents a placeholder blank line (e.g. as in a template). Such blank lines must not be used for presentation purposes.

Any content inside `br` elements must not be considered part of the surrounding text.

A `br` element should separate paragraphs for the purposes of the Unicode bidirectional algorithm. This requirement may be implemented indirectly through the style layer. For example, an HTML+CSS user agent could implement these requirements by implementing the CSS 'unicode-bidi' property. [BIDI] [CSS]

4.6.28 The `wbr` element

Categories:

Flow content.
Phrasing content.

Contexts in which this element can be used:

Where phrasing content is expected.

Content model:

Empty.

Content attributes:

Global attributes

DOM interface:

Uses `HTMLElement`.

The `wbr` element represents a line break opportunity.

For the purposes of applying the bidirectional algorithm to the paragraph-level container that a `wbr` element finds itself within, the `wbr` element must be treated like a U+200B ZERO WIDTH SPACE (i.e. it has no effect).

The requirements on handling the `bdi` element for the bidirectional algorithm may be implemented indirectly through the style layer, e.g. by implementing the suggestions in the rendering section.

In the following example, someone is quoted as saying something which, for effect, is written as one long word. However, to ensure that the text can be wrapped in a readable fashion, the individual words in the quote are separated using a `wbr` element.

```
<p>So then he pointed at the tiger and screamed
"there<wbr>is<wbr>no<wbr>way<wbr>you<wbr>are<wbr>ever<wbr>going<wbr>to<wbr>catch<wbr>me"!</p>
```

Here, especially long lines of code in a program listing have suggested wrapping points given using `wbr` elements.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

|| ...</pre>

Any content inside `wbr` elements must not be considered part of the surrounding text.

4.6.29 Usage summary

This section is non-normative.

Element		
a	H	Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html
em	S	
strong	In	
small	S	tive.</small>
s	Ir	
cite	Tl	
q	Q	:</q> but advised against
dfn	Defining instance	The term <dfn>organic food</dfn> refers to food produced without synthetic chemicals.
abbr	Abbreviations	Organic food in Ireland is certified by the <abbr title="Irish Organic Farmers and Growers Association">IOFGA</abbr>.
data	Machine-readable equivalent	Available starting today! <data value="UPC:022014640201">North Coast Organic Apple Cider</data>
time	Machine-readable equivalent of date- or time-related data	Available starting on <time datetime="2011-11-12">November 12th</time>!
code	Computer code	The <code>fruitdb</code> program can be used for tracking fruit production.
var	Variables	If there are <var>n</var> fruit in the bowl, at least <var>n</var>/2 will be ripe.
samp	Computer output	The computer said <samp>Unknown error -3</samp>.
kbd	User input	Hit <kbd>F1</kbd> to continue.
sub	Subscripts	Water is H₂O.
sup	Superscripts	The Hydrogen in heavy water is usually ²H.
i	Alternative voice	Lemonade consists primarily of <i>Citrus limon</i>.
b	Keywords	Take a lemon and squeeze it with a juicer.
u	Annotations	The mixture of apple juice and <u class="spelling">eldeflower</u> juice is very pleasant.
mark	Highlight	Elderflower cordial, with one <mark>part</mark> cordial to ten <mark>part</mark>s water, stands a<mark>part</mark> from the rest.
ruby, rt, rp	Ruby annotations	<ruby> OJ <rp>(<rt>Orange Juice</rt>)</rp></ruby>
bdi	Text directionality isolation	The recommended restaurant is <bdi lang="">My Juice Café (At The Beach)</bdi>.
bdo	Text directionality formatting	The proposal is to write English, but in reverse order. "Juice" would become "<bdo dir=rtl>Juice</bdo>"
span	Other	In French we call it sirop de sureau.
br	Line break	Simply Orange Juice Company Apopka, FL 32703 U.S.A.
wbr	Line breaking opportunity	www.simply<wbr>orange<wbr>juice.com

4.7 Edits

The `ins` and `del` elements represent edits to the document.

4.7.1 The `ins` element

Categories:

- Flow content.
- Phrasing content.
- Palpable content.

3,445 captures

5 Jun 2004 - 22 Aug 2020

Go

NOV JAN FEB

03

2012 2013 2014



▼ About this capture

Content attributes:

Global attributes

cite

datetime

DOM interface:

Uses the `HTMLModElement` interface



The ins element

The 1 Too slow? Try reading the multipage copy of
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

As does the following, because everything in the `aside` element here counts as phrasing content and therefore there is just one paragraph:

```
<aside>
  <ins>
    Apples are <em>tasty</em>
  </ins>
  <ins>
    So are pears.
  </ins>
</aside>
```

ins elements should not cross implied paragraph boundaries

The following example represents the addition of two paragraphs, the second of which was inserted in two parts. The first `ins` element in this example thus crosses a paragraph boundary, which is considered poor form.

```
<aside>
  <!-- don't do this -->
  <ins datetime="2005-03-16 00:00Z">
    <p> I like fruit. </p>
    Apples are <em>tasty</em>.
  </ins>
  <ins datetime="2007-12-19 00:00Z">
    So are pears.
  </ins>
</aside>
```

Here is a better way of marking this up. It uses more elements, but none of the elements cross implied paragraph boundaries.

```
<aside>
<ins datetime="2005-03-16 00:00Z">
  <p> I like fruit. </p>
</ins>
<ins datetime="2005-03-16 00:00Z">
  Apples are <em>tasty</em>.
</ins>
<ins datetime="2007-12-19 00:00Z">
  So are pears.
</ins>
</aside>
```

4.7.2 The `del` element

Categories:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Content model:

Transparent.

Content attributes:

Global attributes

cite

datetime

DOM interface:

U

The ~~del~~ element
del element

The i

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>


```
<li>Empty the dishwasher</li>
<li><del datetime="2009-10-11T01:25-07:00">Watch Walter Lewin's lectures</del></li>
<li><del datetime="2009-10-10T23:38-07:00">Download more tracks</del></li>
<li>Buy a printer</li>
</ul>
```

id time of their completion.

4.7.3 Attributes common to ins and del elements

The `cite` attribute may be used to specify the address of a document that explains the change. When that document is long, for instance the minutes of a meeting, authors are encouraged to include a fragment identifier pointing to the specific part of that document that discusses the change.

If the `cite` attribute is present, it must be a valid URL potentially surrounded by spaces that explains the change. To obtain the corresponding citation link, the value of the attribute must be resolved relative to the element. User agents may allow users to follow such citation links, but they are primarily intended for private use (e.g. by server-side scripts collecting statistics about a site's edits), not for readers.

The `datetime` attribute may be used to specify the time and date of the change.

If present, the `datetime` attribute's value must be a valid date string with optional time.

User agents must parse the `datetime` attribute according to the parse a date or time string algorithm. If that doesn't return a date or a global date and time, then the modification has no associated timestamp (the value is non-conforming; it is not a valid date string with optional time). Otherwise, the modification is marked as having been made at the given date or global date and time. If the given value is a global date and time then user agents should use the associated time-zone offset information to determine which time zone to present the given `datetime` in.

This value may be shown to the user, but it is primarily intended for private use.

The `ins` and `del` elements must implement the `HTMLModElement` interface:

```
interface HTMLModElement : HTMLElement {
    attribute DOMString cite;
    attribute DOMString dateTime;
};
```

The `cite` IDL attribute must reflect the element's `cite` content attribute. The `dateTime` IDL attribute must reflect the element's `datetime` content attribute.

4.7.4 Edits and paragraphs

This section is non-normative.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<ins>
<p>
  This is a paragraph that was inserted.
</p>
This is another paragraph whose first sentence was inserted
at the same time as the paragraph above.
</ins>
This is a second sentence, which was there all along.
</section>
```



By only wrapping the start of a thin section (not the end), it is possible to mark up the start of one paragraph and the end of the next. This is not considered good practice:

```
<section>
  This
  inser />
<p>Th
  This sentence was inserted too, but this is the
  third paragraph in this example.
  <!-- (don't do this) -->
</section>
```

cond paragraph, and the
d not considered good

However, due to the way implied paragraphs are defined, it is not possible to mark up the end of one paragraph and the start of the very next one using the same `ins` or `del` element. You instead have to use one (or two) `p` element(s) and two `ins` or `del` elements, as for example:

```
<section>
  <p>This is the first paragraph. <del>This sentence was
  deleted.</del></p>
  <p><del>This sentence was deleted too.</del> That
  sentence needed a separate &lt;del&gt; element.</p>
</section>
```

Partly because of the confusion described above, authors are strongly encouraged to always mark up all paragraphs with the `p` element, instead of having `ins` or `del` elements that cross implied paragraphs boundaries.

4.7.5 Edits and lists

This section is non-normative.

The content models of the `ol` and `ul` elements do not allow `ins` and `del` elements as children. Lists always represent all their items, including items that would otherwise have been marked as deleted.

To indicate that an item is inserted or deleted, an `ins` or `del` element can be wrapped around the contents of the `li` element. To indicate that an item has been replaced by another, a single `li` element can have one or more `del` elements followed by one or more `ins` elements.

In the following example, a list that started empty had items added and removed from it over time. The bits in the example that have been emphasized show the parts that are the "current" state of the list. The list item numbers don't take into account the edits, though.

```
<h1>Stop-ship bugs</h1>
<ol>
  <li><ins datetime="2008-02-12T15:20Z">Bug 225:
    Rain detector doesn't work in snow</ins></li>
  <li><del datetime="2008-03-01T20:22Z"><ins datetime="2008-02-14T12:02Z">Bug 228:
    Water buffer overflows in April</ins></del></li>
  <li><ins datetime="2008-02-16T13:50Z">Bug 230:
    Water heater doesn't use renewable fuels</ins></li>
  <li><del datetime="2008-02-20T21:15Z"><ins datetime="2008-02-16T14:25Z">Bug 232:
    Carbon dioxide emissions detected after startup</ins></del></li>
</ol>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<li><del>Apple</del></li>
<li>Orange</li>
<li><del>Pear</del></li>
<li><ins>Teal</ins></li>
<li><del>Lemon</del><ins>Yellow</ins></li>
<li>Olive</li>
<li><ins>Purple</ins></li>
</ul>
```



4.7.6 Edit Too slow? Try reading the multipage copy of

This section the spec instead: <https://web.archive.org>

The element elements, so /web/20130103030411/http:

To indicate that //whatwg.org/html can be wrapped

allow for the `ins` and `del`

cell in that row or column

Here, a table's row has been added:

```
<table>
  <thead>
    <tr> <th> Game name           <th> Game publisher   <th> Verdict
  <tbody>
    <tr> <td> Diablo 2          <td> Blizzard        <td> 8/10
    <tr> <td> Portal             <td> Valve            <td> 10/10
    <tr> <td> <ins>Portal 2</ins> <td> <ins>Valve</ins> <td> <ins>10/10</ins>
  </tbody>
</table>
```

Here, a column has been removed (the time at which it was removed is given also, as is a link to the page explaining why):

```
<table>
  <thead>
    <tr> <th> Game name           <th> Game publisher   <th> <del cite="/edits/r192">Verdict</del>
  <tbody>
    <tr> <td> Diablo 2          <td> Blizzard        <td> <del cite="/edits/r192">8/10</del>
    <tr> <td> Portal             <td> Valve            <td> <del cite="/edits/r192">10/10</del>
    <tr> <td> Portal 2            <td> Valve            <td> <del cite="/edits/r192">10/10</del>
  </tbody>
</table>
```

Generally speaking, there is no good way to indicate more complicated edits (e.g. that a cell was removed, moving all subsequent cells up or to the left).

4.8 Embedded content

IDL

4.8.1 The `img` element

Categories:

- Flow content.
- Phrasing content.
- Embedded content.
- If the element has a `usemap` attribute: Interactive content.
- Palpable content.

Contexts in which this element can be used:

Where embedded content is expected.

Content model:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

src
srcset
crossorigin
usemap
ismap
width
height

DOM interface:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
attribute DOMString useMap;
attribute boolean isMap;
attribute unsigned long width;
attribute unsigned long height;
readonly attribute unsigned long naturalWidth;
readonly attribute unsigned long naturalHeight;
readonly attribute boolean complete;
};
```

An `img` element represents an image.

The image given by the `src` and `srcset` attributes is the embedded content; the value of the `alt` attribute provides equivalent content for those who cannot process images or who have image loading disabled (i.e. it is the `img` element's fallback content).

The requirements on the `alt` attribute's value are described in the next section.

The `src` attribute must be present, and must contain a valid non-empty URL potentially surrounded by spaces referencing a non-interactive, optionally animated, image resource that is neither paged nor scripted.

The `srcset` attribute may also be present. If present, its value must consist of one or more image candidate strings, each separated from the next by a U+002C COMMA character (,). This attribute allows authors to provide alternative images for environments with smaller screens or screens with higher pixel densities.

The `srcset` attribute allows authors to provide a set of images to handle graphical displays of varying dimensions and pixel densities.

The attribute essentially takes a comma-separated list of URLs each with one or more descriptors giving the maximum viewport dimensions and pixel density allowed to use the image. From the available options, the user agent then picks the most appropriate image. If the viewport dimensions or pixel density changes, the user agent can replace the image data with a new image on the fly.

To specify an image, give first a URL, then one or more descriptors of the form 100w, 100h, or 2x, where "100w" means "maximum viewport width of 100 CSS pixels", "100h" is the same but for height, and "2x" means "maximum pixel density of 2 device pixels per CSS pixel".

An **image candidate string** consists of the following components, in order, with the further restrictions described below this list:

1. Zero or more space characters.
2. A valid non-empty URL referencing a non-interactive, optionally animated, image resource that is neither paged nor scripted.
3. Zero or more space characters.
4. Optionally a *width descriptor*, consisting of: a space character, a valid non-negative integer representing the *width descriptor value*, and a U+0077 LATIN SMALL LETTER W character.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

7. Zero or more space characters.

8. Optionally a *pixel density descriptor*, consisting of: a space character, a valid floating-point number giving a number greater than zero representing the *pixel density descriptor value*, and a U+0078 LATIN SMALL LETTER X character.

9. Zero or more space characters.

Each image candidate string must have at least one of the three optional descriptors. There must not be two image candidate strings in a `srcset` attribute whose *width descriptor value*, *height descriptor value*, and *pixel density descriptor value* are each identical to their counterpart in the other image candidate string; for the purposes of this requirement, omitted `width descriptors` and `height descriptors` are considered to have a value 1.

In this screen shot, the `srcset` attribute contains the following values:

```
Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
/web/20130103030411/http://whatwg.org/html
```

The image is a single-HD.jpeg 100w

Note: The requirements above imply that images can be static bitmaps (e.g. PNGs, GIFs, JPEGs), single-page vector documents (single-page PDFs, XML files with an SVG root element), animated bitmaps (APNGs, animated GIFs), animated vector graphics (XML files with an SVG root element that use declarative SMIL animation), and so forth. However, these definitions preclude SVG files with script, multipage PDF files, interactive MNG files, HTML documents, plain text documents, and so forth. [PNG] [GIF] [JPEG] [PDF] [XML] [APNG] [SVG] [MNG]

The `img` element must not be used as a layout tool. In particular, `img` elements should not be used to display transparent images, as they rarely convey meaning and rarely add anything useful to the document.

The `crossorigin` attribute is a CORS settings attribute. Its purpose is to allow images from third-party sites that allow cross-origin access to be used with `canvas`.

An `img` is always in one of the following states:

Unavailable

The user agent hasn't obtained any image data.

Partially available

The user agent has obtained some of the image data.

Completely available

The user agent has obtained all of the image data and at least the image dimensions are available.

Broken

The user agent has obtained all of the image data that it can, but it cannot even decode the image enough to get the image dimensions (e.g. the image is corrupted, or the format is not supported, or no data could be obtained).

When an `img` element is either in the partially available state or in the completely available state, it is said to be **available**.

An `img` element is initially unavailable.

When an `img` element is available, it provides a paint source whose width is the image's intrinsic width, whose height is the image's intrinsic height, and whose appearance is the intrinsic appearance of the image.

In a browsing context where scripting is disabled, user agents may obtain images immediately or on demand. In a browsing context where scripting is enabled, user agents must obtain images immediately.

A user agent that obtains images immediately must synchronously update the image data of an `img` element whenever that element is created with a `src` attribute, a `srcset` attribute, or both. A user agent that obtains images immediately must also synchronously update the image data of an `img` element whenever that element has its `src`, `srcset`, or `crossorigin` attribute set, changed, or removed.

A user agent that obtains images on demand must update the image data of an `img` element whenever it needs the image data (i.e. on demand), but only if the `img` element has a `src` or `srcset` attribute, and only if the `img` element is in the unavailable state. When an `img` element's `src`, `srcset`, or `crossorigin` attribute set, changed, or removed, if the user agent only obtains images on demand,

Go

NOV JAN FEB

03

2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

When an `img` element has a current pixel density that is not 1.0, the element's image data must be treated as if its resolution, in device pixels per CSS pixels, was the current pixel density.

For example, if the current pixel density is 3.125, that means that there are 300 device pixels per CSS inch, and thus if the image data is 300x600, it has an intrinsic dimension of 96 CSS pixels by 192 CSS pixels.

Each `Document` object must have a **list of available images**. Each image in this list is identified by a tuple consisting of an absolute URL, a CORS settings attribute mode, and, if the mode is not No CORS, an origin. User agents may copy entries from one `Document` object's list of available images to another at any time (e.g. when the `Document` is created, user agents can add to it all the images that are loaded in it so. User agents may also

remove images from the list of available images.

When the user agent is reading the multipage copy of the spec instead:

1. Reset the list of available images, discarding any pending tasks.
 2. If an `img` element is being generated, then abort these steps.

3. Forget the `img` element's current image data, if any.

4. If the user agent cannot support images, or its support for images has been disabled, then abort these steps.

5. If the element has a `srcset` attribute specified, then let `selected source` and `selected pixel density` be the URL and pixel density that results from processing the image candidates, respectively. Otherwise, if the element has a `src` attribute specified and its value is not the empty string, let `selected source` be the value of the element's `src` attribute, and `selected pixel density` be 1.0. Otherwise, let `selected source` be null and `selected pixel density` be undefined.

6. Let the `img` element's last selected source be `selected source` and the `img` element's current pixel density be `selected pixel density`.

7. Resolve `selected source`, relative to the element. If that is not successful, abort these steps.

8. Let `key` be a tuple consisting of the resulting absolute URL, the `img` element's `crossorigin` attribute's mode, and, if that mode is not No CORS, the `Document` object's origin.

9. If the list of available images contains an entry for `key`, then set the `img` element to the completely available state, update the presentation of the image appropriately, queue a task to fire a simple event named `load` at the `img` element, and abort these steps.

10. Asynchronously await a stable state, allowing the task that invoked this algorithm to continue. The synchronous section consists of all the remaining steps of this algorithm until the algorithm says the synchronous section has ended. (Steps in synchronous sections are marked with .)

11. If another instance of this algorithm for this `img` element was started after this instance (even if it aborted and is no longer running), then abort these steps.

Note: Only the last instance takes effect, to avoid multiple requests when, for example, the `src`, `srcset`, and `crossorigin` attributes are all set in succession.

12. If `selected source` is null, then set the element to the broken state, queue a task to fire a simple event named `error` at the `img` element, and abort these steps.

13. Do a potentially CORS-enabled fetch of the absolute URL that resulted from the earlier step, with the `mode` being the state of the element's `crossorigin` content attribute, the `origin` being the origin of the `img` element's `Document`, and the `default origin behaviour` set to `taint`.

The resource obtained in this fashion, if any, is the `img` element's image data. It can be either CORS-same-origin or CORS-cross-origin; this affects the origin of the image itself (e.g. when used on a `canvas`).

Fetching the image must delay the `load` event of the element's document until the task that is queued by the networking task source once the resource has been fetched (defined below) has been run.

⚠Warning! This, unfortunately, can be used to perform a rudimentary port scan of the user's local network (especially in conjunction with scripting, though scripting isn't actually necessary to carry out such an attack). User agents may implement cross-origin access control policies that are stricter than those described above to mitigate this attack, but

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the resource is in a supported image format, then each task that is queued by the networking task source while the image is being fetched must update the presentation of the image appropriately (e.g. if the image is a progressive JPEG, each packet can improve the resolution of the image); furthermore, the last task that is queued by the networking task source once the resource has been fetched must act as appropriate given the following alternatives:

↪ If the download was successful and the user agent was able to determine the image's width and height

Set the `img` element to the completely available state, update the presentation of the image appropriately, add the image to the list of available images using the key `key`, and fire a simple event named `load` at the `img` element.

↪ Otherwise

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

On task completion, if the `img` element cannot be obtained, or the `error` state, abort the fetching algorithm, discarding any pending tasks generated by that algorithm, and then queue a task to fire a simple event named `error` at the `img` element.

: `img` element.

: queued by the networking task source as each new body part arrives, the user agent must set the `state` at the `img` element.

cannot be obtained, or the `error` state, abort the fetching

While a user agent is running the above algorithm for an element `x`, there must be a strong reference from the element's `Document` to the element `x`, even if that element is not in its `Document`.

When an `img` element is in the completely available state *and* the user agent can decode the media data without errors, then the `img` element is said to be **fully decodable**.

Whether the image is fetched successfully or not (e.g. whether the response code was a 2xx code or equivalent) must be ignored when determining the image's type and whether it is a valid image.

Note: This allows servers to return images with error responses, and have them displayed.

The user agents should apply the image sniffing rules to determine the type of the image, with the image's associated Content-Type headers giving the *official type*. If these rules are not applied, then the type of the image must be the type given by the image's associated Content-Type headers.

User agents must not support non-image resources with the `img` element (e.g. XML files whose root element is an HTML element). User agents must not run executable code (e.g. scripts) embedded in the image resource. User agents must only display the first page of a multipage resource (e.g. a PDF file). User agents must not allow the resource to act in an interactive fashion, but should honor any animation in the resource.

This specification does not specify which image types are to be supported.

When the user agent is required to **process the image candidates** of an `img` element's `srcset` attribute, the user agent must run the following steps, which return a URL and pixel density (null and undefined respectively if no selection can be made):

1. Let `input` be the value of the `img` element's `srcset` attribute.
2. Let `position` be a pointer into `input`, initially pointing at the start of the string.
3. Let `raw candidates` be an initially empty ordered list of URLs with associated unparsed descriptors. The order of entries in the list is the order in which entries are added to the list.
4. *Splitting loop*: Skip whitespace.
5. Collect a sequence of characters that are not space characters, and let that be `url`.
6. If `url` is empty, then jump to the step labeled *descriptor parser*.
7. Collect a sequence of characters that are not U+002C COMMA characters (,), and let that be `descriptors`.
8. Add `url` to `raw candidates`, associated with `descriptors`.
9. If `position` is past the end of `input`, then jump to the step labeled *descriptor parser*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

12. *Descriptor parser*: Let *candidates* be an initially empty ordered list of URLs each with an associated pixel density, and optionally an associated width, height, or both. The order of entries in the list is the order in which entries are added to the list.
13. For each entry in *raw candidates* with URL *url* associated with the unparsed descriptors *unparsed descriptors*, in the order they were originally added to the list, run these substeps:
1. Let *descriptor list* be the result of splitting *unparsed descriptors* on spaces.
 2. Let *error* be *no*.



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

SMALL LETTER W

1. If *width* is not *absent*, then let *error* be *yes*.
 2. Apply the rules for parsing non-negative integers to the token. Let *width* be the result.
- ↳ If the token consists of a valid non-negative integer followed by a U+0068 LATIN SMALL LETTER H character
1. If *height* is not *absent*, then let *error* be *yes*.
 2. Apply the rules for parsing non-negative integers to the token. Let *height* be the result.
- ↳ If the token consists of a valid floating-point number followed by a U+0078 LATIN SMALL LETTER X character
1. If *density* is not *absent*, then let *error* be *yes*.
 2. Apply the rules for parsing floating-point number values to the token. Let *density* be the result.
7. If *width* is still *absent*, set it to Infinity.
 8. If *height* is still *absent*, set it to Infinity.
 9. If *density* is still *absent*, set it to 1.0.
 10. If *error* is still *no*, then add an entry to *candidates* whose URL is *url*, associated with a width *width*, a height *height*, and a pixel density *density*.
14. If the `img` element has a `src` attribute whose value is not the empty string, then run the following substeps:
1. Let *url* be the value of the element's `src` attribute.
 2. Add an entry to *candidates* whose URL is *url*, associated with a width Infinity, a height Infinity, and a pixel density 1.0.
15. If *candidates* is empty, return null as the URL and undefined as the pixel density and abort these steps.
16. If an entry *b* in *candidates* has the same associated width, height, and pixel density as an earlier entry *a* in *candidates*, then remove entry *b*. Repeat this step until none of the entries in *candidates* have the same associated width, height, and pixel density as an earlier entry.
17. Optionally, return the URL of an entry in *candidates* chosen by the user agent, and that entry's associated pixel density, and then abort these steps. The user agent may apply any algorithm or heuristic in its selection of an entry for the purposes of this step.

Note: This allows a user agent to override the default algorithm (as described in subsequent steps) in case the user agent has a reason to do so. For example, it would allow the user agent in highly bandwidth-constrained conditions to intentionally opt to use an image intended for a smaller screen size, on the assumption that it'll probably be good enough. Implementors are urged to avoid doing this if at all possible, to let authors have predictable results. The results of using an image intended for a different viewport size

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

needed to allow user agents to pick images intended for viewports with other dimensions.

18. Let *max width* be the width of the viewport, and let *max height* be the height of the viewport. [CSS]
19. If there are any entries in *candidates* that have an associated width that is less than *max width*, then remove them, unless that would remove all the entries, in which case remove only the entries whose associated width is less than the greatest such width.
20. If there are any entries in *candidates* that have an associated height that is less than *max height*, then remove them, unless that would remove all the entries, in which case remove only the entries whose associated height is less than the greatest such height.
21. If the the the agent-defined value giving s, in which case remove only
22. Rer st such width.
23. Rer st such height.
24. Remove all the entries in *candidates* that have an associated pixel density that is greater than the smallest such pixel density.
25. Return the URL of the sole remaining entry in *candidates*, and that entry's associated pixel density.

The user agent may at any time run the following algorithm to update an `img` element's image in order to react to changes in the environment. (User agents are *not required* to ever run this algorithm.)

1. Asynchronously await a stable state. The synchronous section consists of all the remaining steps of this algorithm until the algorithm says the synchronous section has ended. (Steps in synchronous sections are marked with .)
2. If the `img` element does not have a `srcset` attribute specified, is not in the completely available state, has image data whose resource type is `multipart/x-mixed-replace`, or if its update the image data algorithm is running, then abort this algorithm.
3. Let *selected source* and *selected pixel density* be the URL and pixel density that results from processing the image candidates, respectively.
4. If *selected source* is null, then abort these steps.
5. If *selected source* and *selected pixel density* are the same as the element's last selected source and current pixel density, then abort these steps.
6. Resolve *selected source*, relative to the element.
7. Let *CORS mode* be the state of the element's `crossorigin` content attribute.
8. If the resolve a URL algorithm is not successful, abort these steps.
9. End the synchronous section, continuing the remaining steps asynchronously.

10. Do a potentially CORS-enabled fetch of the resulting absolute URL, with the *mode* being *CORS mode*, the *origin* being the origin of the `img` element's `Document`, and the *default origin behaviour* set to *taint*.

If this download fails in any way (other than the response code not being a 2xx code, as mentioned earlier), or if the image format is unsupported (as determined by applying the image sniffing rules, again as mentioned earlier), or if the resource type is `multipart/x-mixed-replace`, then abort these steps.

Otherwise, wait for the fetch algorithm to queue its last task, and then continue with these steps. The data obtained in this way is used in the steps below.

11. Queue a task to run the following substeps:

1. If the `img` element's `src`, `srcset`, or `crossorigin` attributes have been set, changed, or removed since this algorithm started, then abort these steps.
2. Let the `img` element's last selected source be *selected source* and the `img` element's current pixel density be *selected pixel density*.
3. Replace the `img` element's image data with the resource obtained by the earlier step of this algorithm. It can be either

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The task source for the tasks queued by algorithms in this section is the DOM manipulation task source.

What an `img` element represents depends on the `src` attribute and the `alt` attribute.

↳ If the `src` attribute is set and the `alt` attribute is set to the empty string

The image is either decorative or supplemental to the rest of the content, redundant with some other information in the document.

If the image is available and the user agent is configured to display that image, the element represents the element's

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

User agents may provide

↳ If the `src` attribute is set and the `alt` attribute is not set

ment for the image.

ent represents the element's

Otherwise, the element represents the text given by the `alt` attribute. User agents may provide the user with a notification that an image is present but has been omitted from the rendering.

↳ If the `src` attribute is set and the `alt` attribute is not set

The image might be a key part of the content, and there is no textual equivalent of the image available.

Note: In a conforming document, the absence of the `alt` attribute indicates that the image is a key part of the content but that a textual replacement for the image was not available when the image was generated.

If the image is available and the user agent is configured to display that image, then the element represents the element's image data.

Otherwise, the user agent should display some sort of indicator that there is an image that is not being rendered, and may, if requested by the user, or if so configured, or when required to provide contextual information in response to navigation, provide caption information for the image, derived as follows:

1. If the image has a `title` attribute whose value is not the empty string, then the value of that attribute is the caption information; abort these steps.
2. If the image is a descendant of a `figure` element that has a child `figcaption` element, and, ignoring the `figcaption` element and its descendants, the `figure` element has no `Text` node descendants other than inter-element whitespace, and no embedded content descendant other than the `img` element, then the contents of the first such `figcaption` element are the caption information; abort these steps.
3. There is no caption information.

↳ If the `src` attribute is not set and either the `alt` attribute is set to the empty string or the `alt` attribute is not set at all

The element represents nothing.

↳ Otherwise

The element represents the text given by the `alt` attribute.

The `alt` attribute does not represent advisory information. User agents must not present the contents of the `alt` attribute in the same way as content of the `title` attribute.

User agents may always provide the user with the option to display any image, or to prevent any image from being displayed. User agents may also apply heuristics to help the user make use of the image when the user is unable to see it, e.g. due to a visual disability or because they are using a text terminal with no graphics capabilities. Such heuristics could include, for instance, optical character recognition (OCR) of text found within the image.

⚠Warning! While user agents are encouraged to repair cases of missing `alt` attributes, authors must not rely on such behavior. Requirements for providing text to act as an alternative for images are described in detail below.

The `contents` of `img` elements, if any, are ignored for the purposes of rendering.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

The `isMap` attribute is a boolean attribute. The attribute must not be specified on an element that does not have an ancestor `a` element with an `href` attribute.

The `img` element supports dimension attributes.

The `alt`, `src`, and `srcset` IDL attributes must reflect the respective content attributes of the same name.

The `crossOrigin` IDL attribute must reflect the `crossorigin` content attribute.

The `useMap` IDL attribute must reflect the `usemap` content attribute.



The `isMap` | **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

(Links are given below this box.)

`image`. /web/20130103030411/http:
`image`. //whatwg.org/html

The

not known.

They can be set, to change the corresponding content attributes.

`image`. `naturalWidth`

`image`. `naturalHeight`

These attributes return the intrinsic dimensions of the image, or zero if the dimensions are not known.

`image`. `complete`

Returns true if the image has been completely downloaded or if no image is specified; otherwise, returns false.

`image` = new `Image`([`width` [, `height`]])

Returns a new `img` element, with the `width` and `height` attributes set to the values passed in the relevant arguments, if applicable.

The IDL attributes `width` and `height` must return the rendered width and height of the image, in CSS pixels, if the image is being rendered, and is being rendered to a visual medium; or else the intrinsic width and height of the image, in CSS pixels, if the image is *available* but not being rendered to a visual medium; or else 0, if the image is not *available*. [CSS]

On setting, they must act as if they reflected the respective content attributes of the same name.

The IDL attributes `naturalWidth` and `naturalHeight` must return the intrinsic width and height of the image, in CSS pixels, if the image is *available*, or else 0. [CSS]

The IDL attribute `complete` must return true if any of the following conditions is true:

- Both the `src` attribute and the `srcset` attributes are omitted.
- The the `srcset` attribute is omitted and the `src` attribute's value is the empty string.
- The final task that is queued by the networking task source once the resource has been fetched has been queued, but has not yet been run, and the `img` element is not in the broken state.
- The `img` element is completely available.

Otherwise, the attribute must return false.

Note: The value of `complete` can thus change while a script is executing.

Three constructors are provided for creating `HTMLImageElement` objects (in addition to the factory methods from DOM such as `createElement()`: `Image()`, `Image(width)`, and `Image(width, height)`). When invoked as constructors, these must return a new `HTMLImageElement` object (a new `img` element). If the `width` argument is present, the new object's `width` content attribute must be set to `width`. If the `height` argument is also present, the new object's `height` content attribute must be set to `height`. The element's document must be the active document of the browsing context of the `Window` object on which the interface object of the invoked constructor is found.

A single image can have different appropriate alternative text depending on the context.

In each of the following cases, the same image is used, yet the `alt` text is different each time. The image is the coat of arms of

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Here it is used as an icon representing the town:

```
<p>Home town: </p>
```

Here it is used as part of a text on the town:

```
<p>Carouge has a coat of arms.</p>
<p></p>
<r>
```



Here **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

· /web/20130103030411/http://whatwg.org/html

Here it is used as part of a story.

```
<p>He picked up the folder and a piece of paper fell out.</p>
<p></p>
<p>He stared at the folder. S! The answer he had been looking for all this time was simply the letter S! How had he not seen that before? It all came together now. The phone call where Hector had referred to a lion's tail, the time Marco had stuck his tongue out....</p>
```

Here it is not known at the time of publication what the image will be, only that it will be a coat of arms of some kind, and thus no replacement text can be provided, and instead only a brief caption for the image is provided, in the title attribute:

```
<p>The last user to have uploaded a coat of arms uploaded this one:</p>
<p></p>
```

Ideally, the author would find a way to provide real replacement text even in this case, e.g. by asking the previous user. Not providing replacement text makes the document more difficult to use for people who are unable to view images, e.g. blind users, or users on very low-bandwidth connections or who pay by the byte, or users who are forced to use a text-only Web browser.

Here are some more examples showing the same picture used in different contexts, with different appropriate alternate texts each time.

```
<article>
  <h1>My cats</h1>
  <h2>Fluffy</h2>
  <p>Fluffy is my favorite.</p>
  
  <p>She's just too cute.</p>
  <h2>Miles</h2>
  <p>My other cat, Miles just eats and sleeps.</p>
</article>

<article>
  <h1>Photography</h1>
  <h2>Shooting moving targets indoors</h2>
  <p>The trick here is to know how to anticipate; to know at what speed and what distance the subject will pass by.</p>
  
  <h2>Nature by night</h2>
  <p>To achieve this, you'll need either an extremely sensitive film, or immense flash lights.</p>
</article>

<article>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<h2>music</h2>
<p>After our walks, having emptied my mind, I like listening to Bach.</p>
</article>

<article>
<h1>Fluffy and the Yarn</h1>
<p>Fluffy was a cat who liked to play with yarn. He also liked to jump.</p>
<aside></aside>
<p>He would play in the morning, he would play in the evening. </p>
```

Too slow? Try reading the multipage copy of

4.8.1.1 Re the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4.8.1.1.1 Ge

Except where appropriate to represent, as

: value must be an image is intended to

The most general rule to consider when writing alternative text is the following: **the intent is that replacing every image with the text of its alt attribute not change the meaning of the page.**

So, in general, alternative text can be written by considering what one would have written had one not been able to include the image.

A corollary to this is that the alt attribute's value should never contain text that could be considered the image's *caption*, *title*, or *legend*. It is supposed to contain replacement text that could be used by users *instead* of the image; it is not meant to supplement the image. The title attribute can be used for supplemental information.

Another corollary is that the alt attribute's value should not repeat information that is already provided in the prose next to the image.

Note: One way to think of alternative text is to think about how you would read the page containing the image to someone over the phone, without mentioning that there is an image present. Whatever you say instead of the image is typically a good start for writing the alternative text.

4.8.1.1.2 A link or button containing nothing but the image

When an a element that creates a hyperlink, or a button element, has no textual content but contains one or more images, the alt attributes must contain text that together convey the purpose of the link or button.

In this example, a user is asked to pick his preferred color from a list of three. Each color is given by an image, but for users who have configured their user agent not to display images, the color names are used instead:

```
<h1>Pick your color</h1>
<ul>
<li><a href="green.html"></a></li>
<li><a href="blue.html"></a></li>
<li><a href="red.html"></a></li>
</ul>
```

In this example, each button has a set of images to indicate the kind of color output desired by the user. The first image is used in each case to give the alternative text.

```
<button name="rgb">
</button>
<button name="cmyk"></button>
```

Since each image represents one part of the text, it could also be written like this:

```
<button name="rgb">
</button>
<button name="cmyk"></button>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
<button name="cmyk"></button>
```

4.8.1.1.3 A phrase or paragraph with an alternative graphical representation: charts, diagrams, graphs, maps, illustrations

Sometimes something can be more clearly stated in graphical form, for example as a flowchart, a diagram, a graph, or a simple map showing directions. In such cases, an image can be given using the `img` element, but the lesser textual version must still be given, so that users who are unable to view the image (e.g. because they have a very slow connection, or because they are using a text-only browser, or simply because they are blind) can still understand the meaning.

Too slow? Try reading the multipage copy of
 the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<p></p>
```

Here's another example, showing a good solution and a bad solution to the problem of including an image in a description.

First, here's the good solution. This sample shows how the alternative text should just be what you would have put in the prose if the image had never existed.

```
<!-- This is the correct way to do things. -->
<p>
  You are standing in an open field west of a house.
  
  There is a small mailbox here.
</p>
```

Second, here's the bad solution. In this incorrect way of doing things, the alternative text is simply a description of the image, instead of a textual replacement for the image. It's bad because when the image isn't shown, the text doesn't flow as well as in the first example.

```
<!-- This is the wrong way to do things. -->
<p>
  You are standing in an open field west of a house.
  
  There is a small mailbox here.
</p>
```

Text such as "Photo of white house with boarded door" would be equally bad alternative text (though it could be suitable for the `title` attribute or in the `figcaption` element of a `figure` with this image).

4.8.1.1.4 A short phrase or label with an alternative graphical representation: icons, logos

A document can contain information in iconic form. The icon is intended to help users of visual browsers to recognize features at a glance.

In some cases, the icon is supplemental to a text label conveying the same meaning. In those cases, the `alt` attribute must be present but must be empty.

Here the icons are next to text that conveys the same meaning, so they have an empty `alt` attribute:

```
<nav>
<p><a href="/help/"> Help</a></p>
<p><a href="/configure/">
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Here, posts on a news site are labeled with an icon indicating their topic.

```
<body>
<article>
<header>
<h1>Ratatouille wins <i>Best Movie of the Year</i> award</h1>
<p></p>
</header>
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
iPhone. This week, the panelists compare how reflective their
iPhones' Apple logos are.</p>
</article>
</body>
```

Many pages include logos, insignia, flags, or emblems, which stand for a particular entity such as a company, organization, project, band, software package, country, or some such.

If the logo is being used to represent the entity, e.g. as a page heading, the `alt` attribute must contain the name of the entity being represented by the logo. The `alt` attribute must *not* contain text like the word "logo", as it is not the fact that it is a logo that is being conveyed, it's the entity itself.

If the logo is being used next to the name of the entity that it represents, then the logo is supplemental, and its `alt` attribute must instead be empty.

If the logo is merely used as decorative material (as branding, or, for example, as a side image in an article that mentions the entity to which the logo belongs), then the entry below on purely decorative images applies. If the logo is actually being discussed, then it is being used as a phrase or paragraph (the description of the logo) with an alternative graphical representation (the logo itself), and the first entry above applies.

In the following snippets, all four of the above cases are present. First, we see a logo used to represent a company:

```
<h1></h1>
```

Next, we see a paragraph which uses a logo right next to the company name, and so doesn't have any alternative text:

```
<article>
<h2>News</h2>
<p>We have recently been looking at buying the  ABT company, a small Greek company
specializing in our type of product.</p>
```

In this third snippet, we have a logo being used in an aside, as part of the larger article discussing the acquisition:

```
<aside><p></p></aside>
<p>The ABT company has had a good quarter, and our
pie chart studies of their accounts suggest a much bigger blue slice
than its green and orange slices, which is always a good sign.</p>
</article>
```

Finally, we have an opinion piece talking about a logo, and the logo is therefore described in detail in the alternative text.

```
<p>Consider for a moment their logo:</p>
```

```
<p></p>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

at least that would look good on the cover of a blue book.</p>

This example shows how the alternative text should be written such that if the image isn't *available*, and the text is used instead, the text flows seamlessly into the surrounding text, as if the image had never been there in the first place.

4.8.1.1.5 Text that has been rendered to a graphic for typographical effect

Sometimes, an image just consists of text, and the purpose of the image is not to highlight the actual typographic effects used to render the text, but just to convey the text itself.

In such case Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

d plants. If the text is merely just the same text "Earth

4.8.1.1.6 A graphical representation of some of the surrounding text

In many cases, the image is actually just supplementary, and its presence merely reinforces the surrounding text. In these cases, the `alt` attribute must be present but its value must be the empty string.

In general, an image falls into this category if removing the image doesn't make the page any less useful, but including the image makes it a lot easier for users of visual browsers to understand the concept.

A flowchart that repeats the previous paragraph in graphical form:

```
<p>The Network passes data to the Input Stream Preprocessor, which
passes it to the Tokenizer, which passes it to the Tree Construction
stage. From there, data goes to both the DOM and to Script Execution.
Script Execution is linked to the DOM, and, using document.write(),
passes data to the Tokenizer.</p>
<p></p>
```

In these cases, it would be wrong to include alternative text that consists of just a caption. If a caption is to be included, then either the `title` attribute can be used, or the `figure` and `figcaption` elements can be used. In the latter case, the image would in fact be a phrase or paragraph with an alternative graphical representation, and would thus require alternative text.

```
<!-- Using the title="" attribute -->
<p>The Network passes data to the Input Stream Preprocessor, which
passes it to the Tokenizer, which passes it to the Tree Construction
stage. From there, data goes to both the DOM and to Script Execution.
Script Execution is linked to the DOM, and, using document.write(),
passes data to the Tokenizer.</p>
<p></p>

<!-- Using <figure> and <figcaption> -->
<p>The Network passes data to the Input Stream Preprocessor, which
passes it to the Tokenizer, which passes it to the Tree Construction
stage. From there, data goes to both the DOM and to Script Execution.
Script Execution is linked to the DOM, and, using document.write(),
passes data to the Tokenizer.</p>
<figure>

<figcaption>Flowchart representation of the parsing model.</figcaption>
</figure>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
passes data to the Tokenizer.</p>
<p></p>
<!-- Never put the image's caption in the alt="" attribute! -->
```

A graph that repeats the previous paragraph in graphical form:

```
<p>According to a study covering several billion pages,
about 62% of documents on the Web in 2007 triggered the Quirks
mode. This is because the Web is a wide design scheme — the
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4.8.1.1.7 A1 /web/20130103030411/http:

If an image is included in a page, it should point to the <http://whatwg.org/html>

-wide design scheme — the

However, a decorative image that isn't discussed by the surrounding text but still has some relevance can be included in a page using the `img` element. Such images are decorative, but still form part of the content. In these cases, the `alt` attribute must be present but its value must be the empty string.

Examples where the image is purely decorative despite being relevant would include things like a photo of the Black Rock City landscape in a blog post about an event at Burning Man, or an image of a painting inspired by a poem, on a page reciting that poem. The following snippet shows an example of the latter case (only the first verse is included in this snippet):

```
<h1>The Lady of Shalott</h1>
<p></p>
<p>On either side the river lie<br>
Long fields of barley and of rye,<br>
That clothe the wold and meet the sky;<br>
And through the field the road run by<br>
To many-tower'd Camelot;<br>
And up and down the people go,<br>
Gazing where the lilies blow<br>
Round an island there below,<br>
The island of Shalott.</p>
```

4.8.1.1.8 A group of images that form a single larger picture with no links

When a picture has been sliced into smaller image files that are then displayed together to form the complete picture again, one of the images must have its `alt` attribute set as per the relevant rules that would be appropriate for the picture as a whole, and then all the remaining images must have their `alt` attribute set to the empty string.

In the following example, a picture representing a company logo for XYZ Corp has been split into two pieces, the first containing the letters "XYZ" and the second with the word "Corp". The alternative text ("XYZ Corp") is all in the first image.

```
<h1></h1>
```

In the following example, a rating is shown as three filled stars and two empty stars. While the alternative text could have been "★★★☆☆", the author has instead decided to more helpfully give the rating in the form "3 out of 5". That is the alternative text of the first image, and the rest have blank alternative text.

```
<p>Rating: <meter max=5 value=3>
  
  </meter></p>
```

4.8.1.1.9 A group of images that form a single larger picture with links

Generally, image maps should be used instead of slicing an image for links.

However, if an image is indeed sliced and any of the components of the sliced picture are the sole contents of links, then one image per link must have alternative text in its `alt` attribute representing the purpose of the link.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
Noodliness do you wish to reach out for?</p>
<p><a href="?go=left" ></a>
 >
 ><a href="?go=right"></a></p>
```

4.8.1.1.10 A key part of the content

In some cases, the image is a critical part of the content. This could be the case, for instance, on a page that is part of a photo gallery. The image is the whole point of the page containing it.

How to prove it: Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

The general rule is to use the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

```
<figure>
 
 <figcaption>Screenshot of a KDE desktop.</figcaption>
</figure>
```

A graph in a financial report:

```

```

Note that "sales graph" would be inadequate alternative text for a sales graph. Text that would be a good *caption* is not generally suitable as replacement text.

Images that defy a complete description

In certain cases, the nature of the image might be such that providing thorough alternative text is impractical. For example, the image could be indistinct, or could be a complex fractal, or could be a detailed topographical map.

In these cases, the `alt` attribute must contain some suitable alternative text, but it may be somewhat brief.

Sometimes there simply is no text that can do justice to an image. For example, there is little that can be said to usefully describe a Rorschach inkblot test. However, a description, even if brief, is still better than nothing:

```
<figure>
 
 <figcaption>A black outline of the first of the ten cards in the Rorschach inkblot test.</figcaption>
</figure>
```

Note that the following would be a very bad use of alternative text:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
in the Rorschach inkblot test.</figcaption>
</figure>
```

Including the caption in the alternative text like this isn't useful because it effectively duplicates the caption for users who don't have images, taunting them twice yet not helping them any more than if they had only read or heard the caption once.

Another example of an image that defies full description is a fractal, which, by definition, is infinite in detail.



View of an image of the

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

id with
ller
ative
r bulbs

Images whc

In some unfortunate cases, there might be no alternative text available at all, either because the image is obtained in some automated fashion without any associated alternative text (e.g. a Webcam), or because the page is being generated by a script using user-provided images where the user did not provide suitable or usable alternative text (e.g. photograph sharing sites), or because the author does not himself know what the images represent (e.g. a blind photographer sharing an image on his blog).

In such cases, the `alt` attribute may be omitted, but one of the following conditions must be met as well:

- The `img` element is in a `figure` element that contains a `figcaption` element that contains content other than inter-element whitespace, and, ignoring the `figcaption` element and its descendants, the `figure` element has no `Text` node descendants other than inter-element whitespace, and no embedded content descendant other than the `img` element.
- The `title` attribute is present and has a non-empty value.

Note: *Relying on the `title` attribute is currently discouraged as many user agents do not expose the attribute in an accessible manner as required by this specification (e.g. requiring a pointing device such as a mouse to cause a tooltip to appear, which excludes keyboard-only users and touch-only users, such as anyone with a modern phone or tablet).*

Note: *Such cases are to be kept to an absolute minimum. If there is even the slightest possibility of the author having the ability to provide real alternative text, then it would not be acceptable to omit the `alt` attribute.*

A photo on a photo-sharing site, if the site received the image with no metadata other than the caption, could be marked up as follows:

```
<figure>

<figcaption>Bubbles traveled everywhere with us.</figcaption>
</figure>
```

It would be better, however, if a detailed description of the important parts of the image obtained from the user and included on the page.

A blind user's blog in which a photo taken by the user is shown. Initially, the user might not have any idea what the photo he took shows:

```
<article>
<h1>I took a photo</h1>
<p>I went out today and took a photo!</p>
<figure>

<figcaption>A photograph taken blindly from my front porch.</figcaption>
</figure>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<article>
  <h1>I took a photo</h1>
  <p>I went out today and took a photo!</p>
  <figure>
    

```

<input type=text name=captcha></label>
(If you cannot see the image, you can use an audio test instead.)</p>
```

Another example would be software that displays images and asks for alternative text precisely for the purpose of then writing a page with correct alternative text. Such a page could have a table of images, like this:

```
<table>
 <thead>
 <tr> <th> Image <th> Description
 <tbody>
 <tr>
 <td>
 <td> <input name="alt2421">
 <tr>
 <td>
 <td> <input name="alt2422">
 </tbody>
```

Notice that even in this example, as much useful information as possible is still included in the `title` attribute.

**Note:** Since some users cannot use images at all (e.g. because they have a very slow connection, or because they are using a text-only browser, or because they are listening to the page being read out by a hands-free automobile voice Web browser, or simply because they are blind), the `alt` attribute is only allowed to be omitted rather than being provided with replacement text when no alternative text is available and none can be made available, as in the above examples. Lack of effort from the part of the author is not an acceptable reason for omitting the `alt` attribute.

#### 4.8.1.11 An image not intended for the user

Generally authors should avoid using `img` elements for purposes other than showing images.

If an `img` element is being used for purposes other than showing an image, e.g. as part of a service to count page views, then the `alt` attribute must be the empty string.

In such cases, the `width` and `height` attributes should both be set to zero.

#### 4.8.1.12 An image in an e-mail or private document intended for a specific person who is known to be able to view images

This section does not apply to documents that are publicly accessible, or whose target audience is not necessarily personally known to the author, such as documents on a Web site, e-mails sent to public mailing lists, or software documentation.

When an image is included in a private communication (such as an HTML e-mail) aimed at a specific person who is known to be able

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

#### 4.8.1.1.13 Guidance for markup generators

Markup generators (such as WYSIWYG authoring tools) should, wherever possible, obtain alternative text from their users. However, it is recognized that in many cases, this will not be possible.

For images that are the sole contents of links, markup generators should examine the link target to determine the title of the target, or the URL of the target, and use information obtained in this manner as the alternative text.

For images that have captions, markup generators should use the `figcaption` elements, or the `title` attribute, to provide the image's alternative text.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

As a last resort, if the image is purely decorative and there is no caption, the generator may ignore the `alt` attribute altogether, as it is not required for decorative images.

Markup generators should not be expected to handle all cases. If they are unable to determine the alternative text for an image, they should silently ignore this error.

**Note:** This is intended to avoid markup generators from being pressured into replacing the error of omitting the `alt` attribute with the even more egregious error of providing phony alternative text, because state-of-the-art automated conformance checkers cannot distinguish phony alternative text from correct alternative text.

Markup generators should generally avoid using the image's own file name as the alternative text. Similarly, markup generators should avoid generating alternative text from any content that will be equally available to presentation user agents (e.g. Web browsers).

**Note:** This is because once a page is generated, it will typically not be updated, whereas the browsers that later read the page can be updated by the user, therefore the browser is likely to have more up-to-date and finely-tuned heuristics than the markup generator did when generating the page.

#### 4.8.1.1.14 Guidance for conformance checkers

A conformance checker must report the lack of an `alt` attribute as an error unless one of the conditions listed below applies:

- The `img` element is in a `figure` element that satisfies the conditions described above.
- The `img` element has a `title` attribute with a value that is not the empty string (also as described above).
- The conformance checker has been configured to assume that the document is an e-mail or document intended for a specific person who is known to be able to view images.
- The `img` element has a (non-conforming) `generator-unable-to-provide-required-alt` attribute whose value is the empty string. A conformance checker that is not reporting the lack of an `alt` attribute as an error must also not report the presence of the empty `generator-unable-to-provide-required-alt` attribute as an error. (This case does not represent a case where the document is conforming, only that the generator could not determine appropriate alternative text — validators are not required to show an error in this case, because such an error might encourage markup generators to include bogus alternative text purely in an attempt to silence validators. Naturally, conformance checkers *may* report the lack of an `alt` attribute as an error even in the presence of the `generator-unable-to-provide-required-alt` attribute; for example, there could be a user option to report *all* conformance errors even those that might be the more or less inevitable result of using a markup generator.)

#### 4.8.1.2 Adaptive images

This section is non-normative.

CSS and media queries can be used to construct graphical page layouts that adapt dynamically to the user's environment, in particular to different viewport dimensions and pixel densities. For content, however, CSS does not help; instead, we have the `img` element's `srcset` attribute. This section walks through a sample case showing how to use this attribute.

Consider a situation where on wide screens (wider than 600 CSS pixels) a 300×150 image named `a-rectangle.png` is to be used, but on smaller screens (600 CSS pixels and less), a smaller 100×100 image called `a-square.png` is to be used. The markup for this would look like this:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Note: For details on what to put in the `alt` attribute, see the earlier sections.**

The problem with this is that the user agent does not necessarily know what dimensions to use for the image when the image is loading. To avoid the layout having to be reflowed multiple times as the page is loading, CSS and CSS media queries can be used to provide the dimensions:

```
<figure>
<style scoped>
#a {
 @media
</style>
<img alt="Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
/web/20130103030411/http://whatwg.org/html
Alternatively,
agents that support srcset:>
 <figure>
 <style scoped media="all and (max-width: 600px)">
 #a { width: 100px; height: 100px; }
 </style>

 <figcaption>Barney Frank, 2011</figcaption>
 </figure>
```

Alternatively,  
agents that support `srcset`:

```
<figure>
<style scoped media="all and (max-width: 600px)">
 #a { width: 100px; height: 100px; }
</style>

<figcaption>Barney Frank, 2011</figcaption>
</figure>
```

The `srcset` attribute is used with the `src` attribute, which gives the URL of the image to use for legacy user agents that do not support the `srcset` attribute. This leads to a question of which image to provide in the `src` attribute.

The answer that results in the least duplication is to provide the image suitable for an infinite width and infinite height viewport with a pixel density of 1 CSS pixel per device pixel:

```

```

However, if legacy mobile user agents are more important, one can list all three images in the `srcset` attribute, overriding the `src` attribute entirely. To do this, the widest image has to have the pixel density descriptor instead of the width, since there is no way to specify an infinite width explicitly:

```

```

Since at this point the `src` attribute is actually being ignored entirely by `srcset`-supporting user agents, the `src` attribute can default to any image, including one that is neither the smallest nor biggest:

```

```

The dimensions in the `srcset` attribute are the maximum (viewport) dimensions that an image is intended for. It is possible to think of the numbers as minimums, though... if the images are listed in order, then the minimum for an image is the dimension given for the previous image. This example attempts to demonstrate this by using the file names to show the minimum values for each image:

```

```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Embedded content.  
Interactive content.  
Palpable content.

**Contexts in which this element can be used:**

Where embedded content is expected.

**Content model:**

Text that conforms to the requirements given in the prose.

**Content ↴**

c Too slow? Try reading the multipage copy of  
s the spec instead: <https://web.archive.org>  
n /web/20130103030411/http:  
s //whatwg.org/html  
a  
widtn  
height

**DOM interface:**

```
IDL interface HTMLIFrameElement : HTMLElement {
 attribute DOMString src;
 attribute DOMString srcdoc;
 attribute DOMString name;
 [PutForwards=value] readonly attribute DOMSettableTokenList sandbox;
 attribute boolean seamless;
 attribute boolean allowFullscreen;
 attribute DOMString width;
 attribute DOMString height;
 readonly attribute Document? contentDocument;
 readonly attribute WindowProxy? contentWindow;
};
```

The `iframe` element represents a nested browsing context.

The `src` attribute gives the address of a page that the nested browsing context is to contain. The attribute, if present, must be a valid non-empty URL potentially surrounded by spaces. If the `itemprop` is specified on an `iframe` element, then the `src` attribute must also be specified.

The `srcdoc` attribute gives the content of the page that the nested browsing context is to contain. The value of the attribute is the source of an `iframe srcdoc` document.

For `iframe` elements in HTML documents, the `srcdoc` attribute, if present, must have a value using the HTML syntax that consists of the following syntactic components, in the given order:

1. Any number of comments and space characters.
2. Optionally, a DOCTYPE.
3. Any number of comments and space characters.
4. The root element, in the form of an `html` element.
5. Any number of comments and space characters.

For `iframe` elements in XML documents, the `srcdoc` attribute, if present, must have a value that matches the production labeled `document` in the XML specification. [XML]

Here a blog uses the `srcdoc` attribute in conjunction with the `sandbox` and `seamless` attributes described below to provide users of user agents that support this feature with an extra layer of protection from script injection in the blog post comments:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
getting in boxes, it's going to be great!</p>
<footer>
 <p>Written by cap, 1 hour ago.</p>
</footer>
<article>
 <footer> Thirteen minutes ago, ch wrote: </footer>
 <iframe seamless sandbox srcdoc=<p>did you get a cover picture yet?"</p>></iframe>
</article>
<article>
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Notice the way that quotes have to be escaped (otherwise the `srcdoc` attribute would end prematurely), and the way raw ampersands (e.g. in URLs or in prose) mentioned in the sandboxed content have to be *doubly* escaped — once so that the ampersand is preserved when originally parsing the `srcdoc` attribute, and once more to prevent the ampersand from being misinterpreted when parsing the sandboxed content.

**Note:** In the HTML syntax, authors need only remember to use U+0022 QUOTATION MARK characters ("') to wrap the attribute contents and then to escape all U+0022 QUOTATION MARK ("') and U+0026 AMPERSAND (&) characters, and to specify the `sandbox` attribute, to ensure safe embedding of content.

**Note:** Due to restrictions of the XHTML syntax, in XML the U+003C LESS-THAN SIGN character (<) needs to be escaped as well. In order to prevent attribute-value normalization, some of XML's whitespace characters — specifically U+0009 CHARACTER TABULATION (tab), U+000A LINE FEED (LF), and U+000D CARRIAGE RETURN (CR) — also need to be escaped. [XML]

**Note:** If the `src` attribute and the `srcdoc` attribute are both specified together, the `srcdoc` attribute takes priority. This allows authors to provide a fallback URL for legacy user agents that do not support the `srcdoc` attribute.

When an `iframe` element is inserted into a document, the user agent must create a nested browsing context, and then process the `iframe` attributes for the first time.

When an `iframe` element is removed from a document, the user agent must discard the nested browsing context.

**Note:** This happens without any `unload` events firing (the nested browsing context and its Document are discarded, not unloaded).

Whenever an `iframe` element with a nested browsing context has its `srcdoc` attribute set, changed, or removed, the user agent must process the `iframe` attributes.

Similarly, whenever an `iframe` element with a nested browsing context but with no `srcdoc` attribute specified has its `src` attribute set, changed, or removed, the user agent must process the `iframe` attributes.

When the user agent is to **process the `iframe` attributes**, it must run the first appropriate steps from the following list:

↳ If the `srcdoc` attribute is specified

Navigate the element's child browsing context to a resource whose Content-Type is `text/html`, whose URL is `about:srcdoc`, and whose data consists of the value of the attribute. The resulting Document must be considered an `iframe` `srcdoc` document.

↳ Otherwise, if the element has no `src` attribute specified, and the user agent is processing the `iframe`'s attributes for the first time

Queue a task to fire a simple event named `load` at the `iframe` element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If that is not successful, then let *url* be the string "about:blank". Otherwise, let *url* be the resulting absolute URL.

2. If there exists an ancestor browsing context whose active document's address, ignoring fragment identifiers, is equal to *url*, then abort these steps.
3. Navigate the element's child browsing context to *url*.

Any navigation required of the user agent in the process the `iframe` attributes algorithm must be completed as an explicit self-navigation over time of the navigation over.

Furthermore time of the navigation over.  
**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**  
 Similarly, if the algorithm was navigation over.

When a Document is created, a simple event named `load`

at the `iframe` element. When content whose URL has the same origin as the `iframe` element's Document fails to load (e.g. due to a DNS error, network error, or if the server returned a 4xx or 5xx status code or equivalent), then the user agent must queue a task to fire a simple event named `error` at the element instead. (This event does not fire for parse errors, script errors, or any errors for cross-origin resources.)

The task source for these tasks is the DOM manipulation task source.

**Note: A `load` event is also fired at the `iframe` element when it is created if no other data is loaded in it.**

When the `iframe`'s browsing context's active document is not ready for post-load tasks, and when anything in the `iframe` is delaying the load event of the `iframe`'s browsing context's active document, and when the `iframe`'s browsing context is in the delaying load events mode, the `iframe` must delay the load event of its document.

**Note: If, during the handling of the `load` event, the browsing context in the `iframe` is again navigated, that will further delay the `load` event.**

**Note: If, when the element is created, the `srcdoc` attribute is not set, and the `src` attribute is either also not set or set but its value cannot be resolved, the browsing context will remain at the initial `about:blank` page.**

**Note: If the user navigates away from this page, the `iframe`'s corresponding `WindowProxy` object will proxy new `Window` objects for new `Document` objects, but the `src` attribute will not change.**

The `name` attribute, if present, must be a valid browsing context name. The given value is used to name the nested browsing context. When the browsing context is created, if the attribute is present, the browsing context name must be set to the value of this attribute; otherwise, the browsing context name must be set to the empty string.

Whenever the `name` attribute is set, the nested browsing context's name must be changed to the new value. If the attribute is removed, the browsing context name must be set to the empty string.

The `sandbox` attribute, when specified, enables a set of extra restrictions on any content hosted by the `iframe`. Its value must be an unordered set of unique space-separated tokens that are ASCII case-insensitive. The allowed values are `allow-forms`, `allow-pointer-lock`, `allow-popups`, `allow-same-origin`, `allow-scripts`, and `allow-top-navigation`.

When the attribute is set, the content is treated as being from a unique origin, forms, scripts, and various potentially annoying APIs are disabled, links are prevented from targeting other browsing contexts, and plugins are secured. The `allow-same-origin` keyword allows the content to be treated as being from the same origin instead of forcing it into a unique origin; the `allow-top-navigation` keyword allows the content to navigate its top-level browsing context; and the `allow-forms`, `allow-pointer-lock`, `allow-popups` and `allow-scripts` keywords re-enable forms, the pointer lock API, popups, and scripts respectively. [POINTERLOCK]

**⚠Warning! Setting both the `allow-scripts` and `allow-same-origin` keywords together when the embedded page has the same origin as the page containing the `iframe` allows the embedded page to simply remove the `sandbox` attribute and then**

Go

NOV JAN FEB

03

2012 2013 2014

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

**⚠ Warning!** Potentially hostile files should not be served from the same server as the file containing the `iframe` element. Sandboxing hostile content is of minimal help if an attacker can convince the user to just visit the hostile content directly, rather than in the `iframe`. To limit the damage that can be caused by hostile HTML content, it should be served from a separate dedicated domain. Using a different domain ensures that scripts in the files are unable to attack the site, even if the user is tricked into visiting those pages directly, without the protection of the `sandbox` attribute.

When an `iframe` element with a `sandbox` attribute has its nested browsing context created (before the initial `about:blank` Document is created), and when an `iframe` element's `sandbox` attribute is set or changed while it has a nested browsing context, the user agent must parse the sandboxing directive using the attribute's value as the `input`, the `iframe` element's nested browsing context's `if...` context's `if...` attribute, the `allow fullscreen` flag. The user agent must empty the

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

In this example, we're embedding a page from another origin. The `script` and `form` flags are disabled in the `sandbox` attribute. The `allow` attribute is empty, so the embedded page has no additional permissions.

```
<p>We're not scared of you! Here is your content, unedited:</p>
<iframe sandbox src="http://usercontent.example.net/getusercontent.cgi?id=12193"></iframe>
```

**⚠ Warning!** It is important to use a separate domain so that if the attacker convinces the user to visit that page directly, the page doesn't run in the context of the site's origin, which would make the user vulnerable to any attack found in the page.

In this example, a gadget from another site is embedded. The gadget has scripting and forms enabled, and the origin sandbox restrictions are lifted, allowing the gadget to communicate with its originating server. The sandbox is still useful, however, as it disables plugins and popups, thus reducing the risk of the user being exposed to malware and other annoyances.

```
<iframe sandbox="allow-same-origin allow-forms allow-scripts"
src="http://maps.example.com/embedded.html"></iframe>
```

Suppose a file A contained the following fragment:

```
<iframe sandbox="allow-same-origin allow-forms" src=B></iframe>
```

Suppose that file B contained an `iframe` also:

```
<iframe sandbox="allow-scripts" src=C></iframe>
```

Further, suppose that file C contained a link:

```
Link
```

For this example, suppose all the files were served as `text/html`.

Page C in this scenario has all the sandboxing flags set. Scripts are disabled, because the `iframe` in A has scripts disabled, and this overrides the `allow-scripts` keyword set on the `iframe` in B. Forms are also disabled, because the inner `iframe` (in B) does not have the `allow-forms` keyword set.

Suppose now that a script in A removes all the `sandbox` attributes in A and B. This would change nothing immediately. If the user clicked the link in C, loading page D into the `iframe` in B, page D would now act as if the `iframe` in B had the `allow-same-origin` and `allow-forms` keywords set, because that was the state of the nested browsing context in the `iframe` in A when page B was loaded.

Generally speaking, dynamically removing or changing the `sandbox` attribute is ill-advised, because it can make it quite hard to reason about what will be allowed and what will not.

The `seamless` attribute is a boolean attribute. When specified, it indicates that the `iframe` element's browsing context is to be rendered in a manner that makes it appear to be part of the containing document (seamlessly included in the parent document).

An HTML inclusion is effected using this attribute as in the following example. In this case, the inclusion is of a site-wide navigation bar.

```
<!DOCTYPE HTML>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

```
<iframe seamless src="nav.inc"></iframe>
</nav>
</header>
...
```

An `iframe` element is said to be **in seamless mode** when all of the following conditions are met:

- The `seamless` attribute is set on the `iframe` element, and

and

**Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

When an `if`

- The user agent must set the **seamless browsing context flag** to true for that browsing context. This will cause links to open in the parent browsing context unless an explicit self-navigation override is used (`target="_self"`).
- Media queries in the context of the `iframe`'s browsing context (e.g. on `media` attributes of `style` elements in Documents in that `iframe`) must be evaluated with respect to the nearest ancestor browsing context that is not itself being nested through an `iframe` that is in seamless mode. [MQ]
- In a CSS-supporting user agent: the user agent must add all the style sheets that apply to the `iframe` element to the cascade of the active document of the `iframe` element's nested browsing context, at the appropriate cascade levels, before any style sheets specified by the document itself.
- In a CSS-supporting user agent: the user agent must, for the purpose of CSS property inheritance only, treat the root element of the active document of the `iframe` element's nested browsing context as being a child of the `iframe` element. (Thus inherited properties on the root element of the document in the `iframe` will inherit the computed values of those properties on the `iframe` element instead of taking their initial values.)
- In visual media, in a CSS-supporting user agent: the user agent should set the intrinsic width of the `iframe` to the width that the element would have if it was a non-replaced block-level element with 'width: auto', unless that width would be zero (e.g. if the element is floating or absolutely positioned), in which case the user agent should set the intrinsic width of the `iframe` to the shrink-to-fit width of the root element (if any) of the content rendered in the `iframe`.
- In visual media, in a CSS-supporting user agent: the user agent should set the intrinsic height of the `iframe` to the shortest height that would make the content rendered in the `iframe` at its current width (as given in the previous bullet point) have no scrollable overflow at its bottom edge. Scrollable overflow is any overflow that would increase the range to which a scrollbar or other scrolling mechanism can scroll.
- In visual media, in a CSS-supporting user agent: the user agent must force the height of the initial containing block of the active document of the nested browsing context of the `iframe` to zero.

**Note: This is intended to get around the otherwise circular dependency of percentage dimensions that depend on the height of the containing block, thus affecting the height of the document's bounding box, thus affecting the height of the viewport, thus affecting the size of the initial containing block.**

- In speech media, the user agent should render the nested browsing context without announcing that it is a separate document.
- User agents should, in general, act as if the active document of the `iframe`'s nested browsing context was part of the document that the `iframe` is in, if any.

For example if the user agent supports listing all the links in a document, links in "seamlessly" nested documents would be included in that list without being significantly distinguished from links in the document itself.

- The nested browsing context's `Window` object's cross-boundary event parent is the browsing context container. [DOM]

If the attribute is not specified, or if the origin conditions listed above are not met, then the user agent should render the nested browsing context in a manner that is clearly distinguishable as a separate browsing context, and the seamless browsing context flag must be set to false for that browsing context.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*Note: The attribute can be set or removed dynamically, with the rendering updating in tandem.*

In this example, the site's navigation is embedded using a client-side include using an `iframe`. Any links in the `iframe` will, in new user agents, be automatically opened in the `iframe`'s parent browsing context; for legacy user agents, the site could also include a `base` element with a `target` attribute with the value `_parent`. Similarly, in new user agents the styles of the parent page will be automatically applied to the contents of the frame, but to support legacy user agents authors might wish to include the styles explicitly.

```
<nav><iframe seamless src="nav.include.html"></iframe></nav>
```



**Note:** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `allowfullscreen` attribute controls whether the browser allows the `iframe` to be displayed in a full-screen mode.This is in the `iframe` element's context, e.g. there is another ancestor element.The `iframe` element has well-defined dimensions.

Dimensions (e.g. ad units have well-defined dimensions).

An `iframe` element never has fallback content, as it will always create a nested browsing context, regardless of whether the specified initial contents are successfully used.

Descendants of `iframe` elements represent nothing. (In legacy user agents that do not support `iframe` elements, the contents would be parsed as markup that could act as fallback content.)

When used in HTML documents, the allowed content model of `iframe` elements is text, except that invoking the HTML fragment parsing algorithm with the `iframe` element as the `context` element and the text contents as the `input` must result in a list of nodes that are all phrasing content, with no parse errors having occurred, with no `script` elements being anywhere in the list or as descendants of elements in the list, and with all the elements in the list (including their descendants) being themselves conforming.

The `iframe` element must be empty in XML documents.

**Note:** The HTML parser treats markup inside `iframe` elements as text.

The IDL attributes `src`, `srcdoc`, `name`, `sandbox`, and `seamless` must reflect the respective content attributes of the same name.

The `allowFullscreen` IDL attribute must reflect the `allowfullscreen` content attribute.

The `contentDocument` IDL attribute must return the `Document` object of the active document of the `iframe` element's nested browsing context, if any, or null otherwise.

The `contentWindow` IDL attribute must return the `WindowProxy` object of the `iframe` element's nested browsing context, if any, or null otherwise.

Here is an example of a page using an `iframe` to include advertising from an advertising broker:

```
<iframe src="http://ads.example.com/?customerid=923513721&format=banner"
 width="468" height="60"></iframe>
```

IDL

### 4.8.3 The `embed` element

#### Categories:

- Flow content.
- Phrasing content.
- Embedded content.
- Interactive content.
- Palpable content.

#### Contexts in which this element can be used:

- Where embedded content is expected.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

src  
type  
width  
height  
Any other attribute that has no namespace (see prose).

#### DOM interface:

```
interface HTMLEmbedElement : HTMLElement {
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

at other interfaces.

The `embed` element represents an integration point for an external (typically non-HTML) application or interactive content.

The `src` attribute gives the address of the resource being embedded. The attribute, if present, must contain a valid non-empty URL potentially surrounded by spaces.

If the `itemprop` is specified on an `embed` element, then the `src` attribute must also be specified.

The `type` attribute, if present, gives the MIME type by which the plugin to instantiate is selected. The value must be a valid MIME type. If both the `type` attribute and the `src` attribute are present, then the `type` attribute must specify the same type as the explicit Content-Type metadata of the resource given by the `src` attribute.

When the element is created with neither a `src` attribute nor a `type` attribute, and when attributes are removed such that neither attribute is present on the element anymore, and when the element has a media element ancestor, and when the element has an ancestor `object` element that is *not* showing its fallback content, any plugins instantiated for the element must be removed, and the `embed` element represents nothing.

An `embed` element is said to be **potentially active** when the following conditions are all met simultaneously:

- The element is in a `Document` or was in a `Document` the last time the event loop reached step 1.
- The element's `Document` is fully active.
- The element has either a `src` attribute set or a `type` attribute set (or both).
- The element's `src` attribute is either absent or its value is not the empty string.
- The element is not a descendant of a media element.
- The element is not a descendant of an `object` element that is not showing its fallback content.
- The element is being rendered, or was being rendered the last time the event loop reached step 1.

Whenever an `embed` element that was not potentially active becomes potentially active, and whenever a potentially active `embed` element that is remaining potentially active and has its `src` attribute set, changed, or removed or its `type` attribute set, changed, or removed, the appropriate set of steps from the following list applies:

#### ↳ If the element has a `src` attribute set

The user agent must resolve the value of the element's `src` attribute, relative to the element. If that is successful, the user agent should fetch the resulting absolute URL, from the element's browsing context scope origin if it has one. The task that is queued by the networking task source once the resource has been fetched must find and instantiate an appropriate plugin based on the content's type, and hand that plugin the content of the resource, replacing any previously instantiated plugin for the element.

Fetching the resource must delay the load event of the element's document.

#### ↳ If the element has no `src` attribute set

The user agent should find and instantiate an appropriate plugin based on the value of the `type` attribute.

Whenever an `embed` element that was potentially active stops being potentially active, any plugin that had been instantiated for that element must be unloaded.

When a plugin is to be instantiated but it cannot be secured and the sandboxed plugins browsing context flag is set on the `embed`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**⚠ Warning! Plugins that cannot be secured are disabled in sandboxed browsing contexts because they might not honor the restrictions imposed by the sandbox (e.g. they might allow scripting even when scripting in the sandbox is disabled). User agents should convey the danger of overriding the sandbox to the user if an option to do so is provided.**

The **type** of the content being embedded is defined as follows:

1. If the element has a `type` attribute, and that attribute's value is a type that a plugin supports, then the value of the `type` attribute is the content's type.
2. Otherwise, if the `src` component of the URL of the specified resource (after any query string) matches a pattern that a plugin supports, then the content's type is the four character end with the four character type.
3. Otherwise, `/web/20130103030411/http://whatwg.org/html`.
4. Otherwise, `//whatwg.org/html`.

The `embed` element must use a default plugin.  
(This default could be as simple as saying "Unsupported Format".)

Whether the resource is fetched successfully or not (e.g. whether the response code was a 2xx code or equivalent) must be ignored when determining the resource's type and when handing the resource to the plugin.

**Note: This allows servers to return data for plugins even with error responses (e.g. HTTP 500 Internal Server Error codes can still contain plugin data).**

Any namespace-less attribute other than `name`, `align`, `hspace`, and `vspace` may be specified on the `embed` element, so long as its name is XML-compatible and contains no uppercase ASCII letters. These attributes are then passed as parameters to the plugin.

**Note: All attributes in HTML documents get lowercased automatically, so the restriction on uppercase letters doesn't affect such documents.**

**Note: The four exceptions are to exclude legacy attributes that have side-effects beyond just sending parameters to the plugin.**

The user agent should pass the names and values of all the attributes of the `embed` element that have no namespace to the plugin used, when it is instantiated.

The `HTMLEmbedElement` object representing the element must expose the scriptable interface of the plugin instantiated for the `embed` element. At a minimum, this interface must implement the **legacy caller operation**. (It is suggested that the default behavior of this legacy caller operation, e.g. the behavior of the default plugin's legacy caller operation, be to throw a `NotSupportedError` exception.)

The `embed` element supports dimension attributes.

The IDL attributes `src` and `type` each must reflect the respective content attributes of the same name.

Here's a way to embed a resource that requires a proprietary plugin, like Flash:

```
<embed src="catgame.swf">
```

If the user does not have the plugin (for example if the plugin vendor doesn't support the user's platform), then the user will be unable to use the resource.

To pass the plugin a parameter "quality" with the value "high", an attribute can be specified:

```
<embed src="catgame.swf" quality="high">
```

This would be equivalent to the following, when using an `object` element instead:

```
<object data="catgame.swf">
 <param name="quality" value="high">
</object>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Embedded content.

If the element has a `usemap` attribute: Interactive content.

Listed, submittable, form-associated element.

Palpable content.

**Contexts in which this element can be used:**

Where embedded content is expected.

**Content model:****Content:** Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

form  
width  
height

**DOM interface:**

```
IDL interface HTMLObjectElement : HTMLElement {
 attribute DOMString data;
 attribute DOMString type;
 attribute boolean typeMustMatch;
 attribute DOMString name;
 attribute DOMString useMap;
 readonly attribute HTMLFormElement? form;
 attribute DOMString width;
 attribute DOMString height;
 readonly attribute Document? contentDocument;
 readonly attribute WindowProxy? contentWindow;

 readonly attribute boolean willValidate;
 readonly attribute ValidityState validity;
 readonly attribute DOMString validationMessage;
 boolean checkValidity();
 void setCustomValidity(DOMString error);

 legacycaller any (any... arguments);
};
```

Depending on the type of content instantiated by the `object` element, the node also supports other interfaces.

The `object` element can represent an external resource, which, depending on the type of the resource, will either be treated as an image, as a nested browsing context, or as an external resource to be processed by a plugin.

The `data` attribute, if present, specifies the address of the resource. If present, the attribute must be a valid non-empty URL potentially surrounded by spaces.

**⚠Warning!** Authors who reference resources from other origins that they do not trust are urged to use the `typeMustMatch` attribute defined below. Without that attribute, it is possible in certain cases for an attacker on the remote host to use the plugin mechanism to run arbitrary scripts, even if the author has used features such as the Flash "allowScriptAccess" parameter.

The `type` attribute, if present, specifies the type of the resource. If present, the attribute must be a valid MIME type.

At least one of either the `data` attribute or the `type` attribute must be present.

If the `itemprop` is specified on an `object` element, then the `data` attribute must also be specified.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `name` attribute, if present, must be a valid browsing context name. The given value is used to name the nested browsing context, if applicable.

Whenever one of the following conditions occur:

- the element is created,
- the element is popped off the stack of open elements of an HTML parser or XML parser,
- the element is not on the stack of open elements of an HTML parser or XML parser, and either inserted into a document or removed
- the **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**,
- neither the element's `classid` attribute nor its `data` attribute are present, and its `type` attribute is set, changed, or removed,
- the element changes from being rendered to not being rendered, or vice versa,

...the user agent must queue a task to run the following steps to (re)determine what the `object` element represents. The task source for this task is the DOM manipulation task source. This task being queued or actively running must delay the load event of the element's document.

1. If the user has indicated a preference that this `object` element's fallback content be shown instead of the element's usual behavior, then jump to the last step in the overall set of steps (fallback).

**Note: For example, a user could ask for the element's fallback content to be shown because that content uses a format that the user finds more accessible.**

2. If the element has an ancestor media element, or has an ancestor `object` element that is *not* showing its fallback content, or if the element is not in a `Document` with a browsing context, or if the element's `Document` is not fully active, or if the element is still in the stack of open elements of an HTML parser or XML parser, or if the element is not being rendered, then jump to the last step in the overall set of steps (fallback).
3. If the `classid` attribute is present, and has a value that isn't the empty string, then: if the user agent can find a plugin suitable according to the value of the `classid` attribute, and either plugins aren't being sandboxed or that plugin can be secured, then that plugin should be used, and the value of the `data` attribute, if any, should be passed to the plugin. If no suitable plugin can be found, or if the plugin reports an error, jump to the last step in the overall set of steps (fallback).

4. If the `data` attribute is present and its value is not the empty string, then:

1. If the `type` attribute is present and its value is not a type that the user agent supports, and is not a type that the user agent can find a plugin for, then the user agent may jump to the last step in the overall set of steps (fallback) without fetching the content to examine its real type.
2. Resolve the URL specified by the `data` attribute, relative to the element.
3. If that failed, fire a simple event named `error` at the element, then jump to the last step in the overall set of steps (fallback).
4. Fetch the resulting absolute URL, from the element's browsing context scope origin if it has one.

Fetching the resource must delay the load event of the element's document until the task that is queued by the networking task source once the resource has been fetched (defined next) has been run.

For the purposes of the application cache networking model, this fetch operation is not for a child browsing context (though it might end up being used for one after all, as defined below).

5. If the resource is not yet available (e.g. because the resource was not available in the cache, so that loading the resource required making a request over the network), then jump to the last step in the overall set of steps (fallback). The task that is queued by the networking task source once the resource is available must restart this algorithm from this step. Resources can load incrementally; user agents may opt to consider a resource "available" whenever enough

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

7. Determine the *resource type*, as follows:

1. Let the *resource type* be unknown.
2. If the `object` element has a `type` attribute and a `typemustmatch` attribute, and the resource has associated Content-Type metadata, and the type specified in the resource's Content-Type metadata is an ASCII case-insensitive match for the value of the element's `type` attribute, then let *resource type* be that type and jump to the step below labeled *handler*.
3. If the `object` element has a `typemustmatch` attribute, jump to the step below labeled *handler*.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

source, and the resource has ed in the resource's Content-

bed a resource that uses a  
the user agent with a  
cs.

value is not a type that the user agent supports, but it *is* a type that a plugin supports, then let the *resource type* be the type specified in that `type` attribute, and jump to the step below labeled *handler*.

6. Run the appropriate set of steps from the following list:

#### ↳ The resource has associated Content-Type metadata

1. Let `binary` be false.
2. If the type specified in the resource's Content-Type metadata is "text/plain", and the result of applying the rules for distinguishing if a resource is text or binary to the resource is that the resource is not `text/plain`, then set `binary` to true.
3. If the type specified in the resource's Content-Type metadata is "application/octet-stream", then set `binary` to true.
4. If `binary` is false, then let the *resource type* be the type specified in the resource's Content-Type metadata, and jump to the step below labeled *handler*.
5. If there is a `type` attribute present on the `object` element, and its value is not `application/octet-stream`, then run the following steps:
  1. If the attribute's value is a type that a plugin supports, or the attribute's value is a type that starts with "image/" that is not also an XML MIME type, then let the *resource type* be the type specified in that `type` attribute.
  2. Jump to the step below labeled *handler*.

#### ↳ The resource does not have associated Content-Type metadata

1. If there is a `type` attribute present on the `object` element, then let the *tentative type* be the type specified in that `type` attribute.  
Otherwise, let *tentative type* be the sniffed type of the resource.
2. If *tentative type* is *not* `application/octet-stream`, then let *resource type* be *tentative type* and jump to the step below labeled *handler*.

7. If the `<path>` component of the URL of the specified resource (after any redirects) matches a pattern that a plugin supports, then let *resource type* be the type that that plugin can handle.

For example, a plugin might say that it can handle resources with `<path>` components that end with the four character string ".swf".

**Note: It is possible for this step to finish, or for one of the substeps above to jump straight to the next step, with *resource type* still being unknown. In both cases, the next step will trigger fallback.**

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

Otherwise, the user agent should use the plugin that supports `resource type` and pass the content of the resource to that plugin. If the plugin reports an error, then jump to the last step in the overall set of steps (fallback).

↳ **If the `resource type` is an XML MIME type, or if the `resource type` does not start with "`image/`"**

The `object` element must be associated with a newly created nested browsing context, if it does not already have one.



↑ browsing context must then  
set element's document's  
object element doesn't get  
ent must queue a task to fire  
**fired at the**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If the `name` attribute is present, the browsing context name must be set to the value of this attribute; otherwise, the browsing context name must be set to the empty string.

**Note:** *In certain situations, e.g. if the resource was fetched from an application cache but it is an HTML file with a `manifest` attribute that points to a different application cache manifest, the navigation of the browsing context will be restarted so as to load the resource afresh from the network or a different application cache. Even if the resource is then found to have a different type, it is still used as part of a nested browsing context: only the `navigate` algorithm is restarted, not this `object` algorithm.*

↳ **If the `resource type` starts with "`image/`", and support for images has not been disabled**

Apply the image sniffing rules to determine the type of the image.

The `object` element represents the specified image. The image is not a nested browsing context.

If the image cannot be rendered, e.g. because it is malformed or in an unsupported format, jump to the last step in the overall set of steps (fallback).

↳ **Otherwise**

The given `resource type` is not supported. Jump to the last step in the overall set of steps (fallback).

**Note:** *If the previous step ended with the resource type being unknown, this is the case that is triggered.*

9. The element's contents are not part of what the `object` element represents.

10. Once the resource is completely loaded, queue a task to fire a simple event named `load` at the element.

The task source for this task is the DOM manipulation task source.

5. If the `data` attribute is absent but the `type` attribute is present, and the user agent can find a plugin suitable according to the value of the `type` attribute, and either plugins aren't being sandboxed or the plugin can be secured, then that plugin should be used. If these conditions cannot be met, or if the plugin reports an error, jump to the next step (fallback).

6. (Fallback.) The `object` element represents the element's children, ignoring any leading `param` element children. This is the element's fallback content. If the element has an instantiated plugin, then unload it.

When the algorithm above instantiates a plugin, the user agent should pass to the plugin used the names and values of all the attributes on the element, in the order they were added to the element, with the attributes added by the parser being ordered in source order, followed by a parameter named "PARAM" whose value is null, followed by all the names and values of parameters given by `param` elements that are children of the `object` element, in tree order. If the plugin supports a scriptable interface, the `HTMLObjectElement` object representing the element should expose that interface. The `object` element represents the plugin. The plugin is not a nested browsing context.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

user agents with different capabilities, with the user agent picking the first one it supports.

Whenever the `name` attribute is set, if the `object` element has a nested browsing context, its name must be changed to the new value. If the attribute is removed, if the `object` element has a browsing context, the browsing context name must be set to the empty string.

The `usemap` attribute, if present while the `object` element represents an image, can indicate that the object has an associated image map. The attribute must be ignored if the `object` element doesn't represent an image.

The `form` attribute is used to explicitly associate the `object` element with its form owner.

**Constraint**

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html/browsing-context/**

The `object`

The IDL attri

IDL attribute

The content

//whatwg.org/html/browsing-context/

ame. The `typeMustMatch` usemap content attribute.

ect element's nested

The `contentWindow` IDL attribute must return the `WindowProxy` object of the `object` element's nested browsing context, if it has one; otherwise, it must return null.

The `willValidate`, `validity`, and `validationMessage` attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `form` IDL attribute is part of the element's forms API.

All `object` elements have a **legacy caller operation**. If the `object` element has an instantiated plugin that supports a scriptable interface that defines a legacy caller operation, then that must be the behavior of the object's legacy caller operation. Otherwise, the object's legacy caller operation must be to throw a `NotSupportedError` exception.

In the following example, a Java applet is embedded in a page using the `object` element. (Generally speaking, it is better to avoid using applets like these and instead use native JavaScript and HTML to provide the functionality, since that way the application will work on all Web browsers without requiring a third-party plugin. Many devices, especially embedded devices, do not support third-party technologies like Java.)

```
<figure>
<object type="application/x-java-applet">
 <param name="code" value="MyJavaClass">
 <p>You do not have Java available, or it is disabled.</p>
</object>
<figcaption>My Java Clock</figcaption>
</figure>
```

In this example, an HTML page is embedded in another using the `object` element.

```
<figure>
<object data="clock.html"></object>
<figcaption>My HTML Clock</figcaption>
</figure>
```

The following example shows how a plugin can be used in HTML (in this case the Flash plugin, to show a video file). Fallback is provided for users who do not have Flash enabled, in this case using the `video` element to show the video for those using user agents that support `video`, and finally providing a link to the video for those who have neither Flash nor a `video`-capable browser.

```
<p>Look at my video:

<object type="application/x-shockwave-flash">
 <param name=movie value="http://video.example.com/library/watch.swf">
 <param name=allowfullscreen value=true>
 <param name=flashvars value="http://video.example.com/vids/315981">
 <video controls src="http://video.example.com/vids/315981">
 View video.
 </video>
</object>
</p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

**Contexts in which this element can be used:**

As a child of an `object` element, before any flow content.

**Content model:**

Empty.

**Content attributes:**

Global attributes

`name`

`value`



**DOM intro** Too slow? Try reading the multipage copy of  
( the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

The `param` element defines parameters for plugins invoked by `object` elements. It does not represent anything on its own.

The `name` attribute gives the name of the parameter.

The `value` attribute gives the value of the parameter.

Both attributes must be present. They may have any value.

If both attributes are present, and if the parent element of the `param` is an `object` element, then the element defines a **parameter** with the given name-value pair.

If either the name or value of a parameter defined by a `param` element that is the child of an `object` element that represents an instantiated plugin changes, and if that plugin is communicating with the user agent using an API that features the ability to update the plugin when the name or value of a parameter so changes, then the user agent must appropriately exercise that ability to notify the plugin of the change.

The IDL attributes `name` and `value` must both reflect the respective content attributes of the same name.

The following example shows how the `param` element can be used to pass a parameter to a plugin, in this case the O3D plugin.

```
<!DOCTYPE HTML>
<html lang="en">
 <head>
 <title>O3D Utah Teapot</title>
 </head>
 <body>
 <p>
 <object type="application/vnd.o3d.auto">
 <param name="o3d_features" value="FloatingPointTextures">

 <p>To see the teapot actually rendered by O3D on your computer, please download and install the O3D plugin.</p>
 </object>
 <script src="o3d-teapot.js"></script>
 </p>
 </body>
</html>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Embedded content.

If the element has a `controls` attribute: Interactive content.

Palpable content.

### Contexts in which this element can be used:

Where embedded content is expected.

### Content model:

If the element has a `src` attribute: zero or more `track` elements, then transparent, ~~or~~ with no media element

de

If `src` Too slow? Try reading the multipage copy of ~~or~~ `track` elements, then  
tr

Content: the spec instead: <https://web.archive.org>

C /web/20130103030411/http:

S //whatwg.org/html

P

preload

autoplay

mediagroup

loop

muted

controls

width

height

### DOM interface:

```
IDL interface HTMLVideoElement : HTMLMediaElement {
 attribute unsigned long width;
 attribute unsigned long height;
 readonly attribute unsigned long videoWidth;
 readonly attribute unsigned long videoHeight;
 attribute DOMString poster;
};
```

A `video` element is used for playing videos or movies, and audio files with captions.

Content may be provided inside the `video` element. User agents should not show this content to the user; it is intended for older Web browsers which do not support `video`, so that legacy video plugins can be tried, or to show text to the users of these older browsers informing them of how to access the video contents.

*Note: In particular, this content is not intended to address accessibility concerns. To make video content accessible to the partially sighted, the blind, the hard-of-hearing, the deaf, and those with other physical or cognitive disabilities, a variety of features are available. Captions can be provided, either embedded in the video stream or as external files using the `track` element. Sign-language tracks can be provided, again either embedded in the video stream or by synchronizing multiple video elements using the `mediagroup` attribute or a `MediaController` object. Audio descriptions can be provided, either as a separate track embedded in the video stream, or a separate audio track in an `audio` element slaved to the same controller as the `video` element(s), or in text form using a WebVTT file referenced using the `track` element and synthesized into speech by the user agent. WebVTT can also be used to provide chapter titles. For users who would rather not use a media element at all, transcripts or other textual alternatives can be provided by simply linking to them in the prose near the `video` element. [WEBVTT]*

The `video` element is a media element whose media data is ostensibly video data, possibly with associated audio data.

The `src`, `preload`, `autoplay`, `mediagroup`, `loop`, `muted`, and `controls` attributes are the attributes common to all media elements.

The `poster` attribute gives the address of an image file that the user agent can show while no video data is available. The attribute, if present, must contain a valid non-empty URL potentially surrounded by spaces.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

changing the poster frame.

2. If the `poster` attribute's value is the empty string or if the attribute is absent, then there is no poster frame; abort these steps.
3. Resolve the `poster` attribute's value relative to the element. If this fails, then there is no poster frame; abort these steps.
4. Fetch the resulting absolute URL, from the element's `Document`'s origin. This must delay the load event of the element's `document`.
5. If an image is thus obtained, the poster frame is that image. Otherwise there is no poster frame.

**Note:** *Typically*, Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

A `video` element represents its *effective frame of the video*:

- ↳ When the `video` element is not `playing` (or `HAVE_METADATA` but no `current` value but the media resource does not have a video channel)
- ↳ When the `video` element is paused, the current playback position is the first frame of video, and the element's `showPosterFlag` is set
 

The `video` element represents its poster frame.
- ↳ When the `video` element is paused, and the frame of video corresponding to the current playback position is not available (e.g. because the video is seeking or buffering)
- ↳ When the `video` element is neither potentially playing nor paused (e.g. when seeking or stalled)
 

The `video` element represents the last frame of the video to have been rendered.
- ↳ When the `video` element is paused
 

The `video` element represents the frame of video corresponding to the current playback position.
- ↳ Otherwise (the `video` element has a video channel and is potentially playing)
 

The `video` element represents the frame of video at the continuously increasing "current" position. When the current playback position changes such that the last frame rendered is no longer the frame corresponding to the current playback position in the video, the new frame must be rendered.

**Note:** Which frame in a video stream corresponds to a particular playback position is defined by the video stream's format.

The `video` element also represents any text track cues whose text track cue active flag is set and whose text track is in the showing mode, and any audio from the media resource, at the current playback position.

Any audio associated with the media resource must, if played, be played synchronized with the current playback position, at the element's effective media volume.

In addition to the above, the user agent may provide messages to the user (such as "buffering", "no video loaded", "error", or more detailed information) by overlaying text or icons on the video or other areas of the element's playback area, or in another appropriate manner.

User agents that cannot render the video may instead make the element represent a link to an external video playback utility or to the video data itself.

When a `video` element's media resource has a video channel, the element provides a paint source whose width is the media resource's intrinsic width, whose height is the media resource's intrinsic height, and whose appearance is the frame of video corresponding to the current playback position, if that is available, or else (e.g. when the video is seeking or buffering) its previous appearance, if any, or else (e.g. because the video is still loading the first frame) blackness.

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

▼ About this capture

The **intrinsic width** and **intrinsic height** of the media resource are the dimensions of the resource in CSS pixels after taking into account the resource's dimensions, aspect ratio, clean aperture, resolution, and so forth, as defined for the format used by the resource. If an anamorphic format does not define how to apply the aspect ratio to the video data's dimensions to obtain the "correct" dimensions, then the user agent must apply the ratio by increasing one dimension and leaving the other unchanged.

The `videoWidth` IDL attribute must return the intrinsic width of the video in CSS pixels. The `videoHeight` IDL attribute must return the intrinsic height of the video in CSS pixels. If the element's `readyState` attribute is HAVE NOTHING, then the attributes must return 0.

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

sk area such that the video  
n the video content's aspect  
video, the video will be  
sent nothing.

**Note:** In user agents that implement CSS, the above requirement can be implemented by using the style rule suggested in the rendering section.

The intrinsic width of a `video` element's playback area is the intrinsic width of the poster frame, if that is available and the element currently represents its poster frame; otherwise, it is the intrinsic width of the video resource, if that is available; otherwise it is 300 CSS pixels.

The intrinsic height of a `video` element's playback area is the intrinsic height of the poster frame, if that is available and the element currently represents its poster frame; otherwise it is the intrinsic height of the video resource, if that is available; otherwise it is 150 CSS pixels.

User agents should provide controls to enable or disable the display of closed captions, audio description tracks, and other additional data associated with the video stream, though such features should, again, not interfere with the page's normal rendering.

User agents may allow users to view the video content in manners more suitable to the user (e.g. full-screen or in an independent resizable window). As for the other user interface features, controls to enable this should not interfere with the page's normal rendering unless the user agent is exposing a user interface. In such an independent context, however, user agents may make full user interfaces visible, with, e.g., play, pause, seeking, and volume controls, even if the `controls` attribute is absent.

User agents may allow video playback to affect system features that could interfere with the user's experience; for example, user agents could disable screensavers while video playback is in progress.

The `poster` IDL attribute must reflect the `poster` content attribute.

This example shows how to detect when a video has failed to play correctly.

```
<script>
function failed(e) {
 // video playback failed - show a message saying why
 switch (e.target.error.code) {
 case e.target.error.MEDIA_ERR_ABORTED:
 alert('You aborted the video playback.');
 break;
 case e.target.error.MEDIA_ERR_NETWORK:
 alert('A network error caused the video download to fail part-way.');
 break;
 case e.target.error.MEDIA_ERR_DECODE:
 alert('The video playback was aborted due to a corruption problem or because the
video used features your browser did not support.');
 break;
 case e.target.error.MEDIA_ERR_SRC_NOT_SUPPORTED:
 alert('The video could not be loaded, either because the server or network failed or
because the format is not supported.');
 break;
 default:
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<p><video src="tgif.vid" autoplay controls onerror="failed(event)"></video></p>
<p>Download the video file.</p>
```

## 4.8.7 The audio element

### Categories:

Flow content.  
Phrasing content



E

If the element does not have a `src` attribute: zero or more `source` elements, then zero or more `track` elements, then transparent, but with no media element descendants.

### Contexts

If the element does not have a `src` attribute: zero or more `source` elements, then zero or more `track` elements, then transparent, but with no media element descendants.

### Content

If the element does not have a `src` attribute: zero or more `source` elements, then zero or more `track` elements, then transparent, but with no media element descendants.

media element

If the element does not have a `src` attribute: zero or more `source` elements, then zero or more `track` elements, then transparent, but with no media element descendants.

### Content attributes:

Global attributes  
`src`  
`crossorigin`  
`preload`  
`autoplay`  
`mediagroup`  
`loop`  
`muted`  
`controls`

### DOM interface:

```
IDL [NamedConstructor=Audio(),
 NamedConstructor=Audio(DOMString src)]
interface HTMLAudioElement : HTMLMediaElement {};
```

An `audio` element represents a sound or audio stream.

Content may be provided inside the `audio` element. User agents should not show this content to the user; it is intended for older Web browsers which do not support `audio`, so that legacy audio plugins can be tried, or to show text to the users of these older browsers informing them of how to access the audio contents.

**Note:** In particular, this content is not intended to address accessibility concerns. To make audio content accessible to the deaf or to those with other physical or cognitive disabilities, a variety of features are available. If captions or a sign language video are available, the `video` element can be used instead of the `audio` element to play the audio, allowing users to enable the visual alternatives. Chapter titles can be provided to aid navigation, using the `track` element and a WebVTT file. And, naturally, transcripts or other textual alternatives can be provided by simply linking to them in the prose near the `audio` element. [WEBVTT]

The `audio` element is a media element whose media data is ostensibly audio data.

The `src`, `preload`, `autoplay`, `mediagroup`, `loop`, `muted`, and `controls` attributes are the attributes common to all media elements.

When an `audio` element is potentially playing, it must have its audio data played synchronized with the current playback position, at the element's effective media volume.

When an `audio` element is not potentially playing, audio must not play for the element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Two constructors are provided for creating `HTMLAudioElement` objects (in addition to the factory methods from DOM such as `createElement()`): `Audio()` and `Audio(src)`. When invoked as constructors, these must return a new `HTMLAudioElement` object (a new `audio` element). The element must have its `preload` attribute set to the literal value "auto". If the `src` argument is present, the object created must have its `src` content attribute set to the provided value, and the user agent must invoke the object's resource selection algorithm before returning. The element's document must be the active document of the browsing context of the Window object on which the interface object of the invoked constructor is found.



#### 4.8.8 The

Category

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

Contexts

/web/20130103030411/http:  
/whatwg.org/html

Content

Empty.

#### Content attributes:

Global attributes

`src`

`type`

`media`

#### DOM interface:

```
IDL interface HTMLSourceElement : HTMLElement {
 attribute DOMString src;
 attribute DOMString type;
 attribute DOMString media;
};
```

The `source` element allows authors to specify multiple alternative media resources for media elements. It does not represent anything on its own.

The `src` attribute gives the address of the media resource. The value must be a valid non-empty URL potentially surrounded by spaces. This attribute must be present.

**Note:** Dynamically modifying a `source` element and its attribute when the element is already inserted in a `video` or `audio` element will have no effect. To change what is playing, just use the `src` attribute on the `media` element directly, possibly making use of the `canPlayType()` method to pick from amongst available resources. Generally, manipulating `source` elements manually after the document has been parsed is an unnecessarily complicated approach.

The `type` attribute gives the type of the media resource, to help the user agent determine if it can play this media resource before fetching it. If specified, its value must be a valid MIME type. The `codecs` parameter, which certain MIME types define, might be necessary to specify exactly how the resource is encoded. [RFC4281]

The following list shows some examples of how to use the `codecs=MIME` parameter in the `type` attribute.

**H.264 Constrained baseline profile video (main and extended video compatible) level 3 and Low-Complexity AAC audio in MP4 container**

```
<source src='video.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
```

**H.264 Extended profile video (baseline-compatible) level 3 and Low-Complexity AAC audio in MP4 container**

```
<source src='video.mp4' type='video/mp4; codecs="avc1.58A01E, mp4a.40.2"'>
```

**H.264 Main profile video level 3 and Low-Complexity AAC audio in MP4 container**

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

&lt;source src='video.mp4' type='video/mp4; codecs="avc1.64001E, mp4a.40.2"'&gt;

**MPEG-4 Visual Simple Profile Level 0 video and Low-Complexity AAC audio in MP4 container**

&lt;source src='video.mp4' type='video/mp4; codecs="mp4v.20.8, mp4a.40.2"'&gt;

**MPEG-4 Advanced Simple Profile Level 0 video and Low-Complexity AAC audio in MP4 container**

&lt;source src='video.mp4' type='video/mp4; codecs="mp4v.20.240, mp4a.40.2"'&gt;

**MPEG-4 Visual Simple Profile Level 0 video and AAC audio in MP4 container**Too slow? Try reading the multipage copy of [nr](#)The spec instead: [https://web.archive.org](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

/web/20130103030411/http:

The <http://whatwg.org/html>

'&gt;

'&gt;

'&gt;

**Vorbis audio alone in Ogg container**

&lt;source src='audio.ogg' type='audio/ogg; codecs=vorbis'&gt;

**Speex audio alone in Ogg container**

&lt;source src='audio.spx' type='audio/ogg; codecs=speex'&gt;

**FLAC audio alone in Ogg container**

&lt;source src='audio.oga' type='audio/ogg; codecs=flac'&gt;

**Dirac video and Vorbis audio in Ogg container**

&lt;source src='video.ogv' type='video/ogg; codecs="dirac, vorbis"'&gt;

The **media** attribute gives the intended media type of the media resource, to help the user agent determine if this media resource is useful to the user before fetching it. Its value must be a valid media query.

**Note:** The resource selection algorithm is defined in such a way that when the **media** attribute is omitted the user agent acts the same as if the value was "all", i.e. by default the media resource is suitable for all media.

If a **source** element is inserted as a child of a **media** element that has no **src** attribute and whose **networkState** has the value **NETWORK\_EMPTY**, the user agent must invoke the **media** element's resource selection algorithm.

The IDL attributes **src**, **type**, and **media** must reflect the respective content attributes of the same name.

If the author isn't sure if the user agents will all be able to render the media resources provided, the author can listen to the **error** event on the last **source** element and trigger fallback behavior:

```
<script>
function fallback(video) {
 // replace <video> with its contents
 while (video.hasChildNodes()) {
 if (video.firstChild instanceof HTMLSourceElement)
 video.removeChild(video.firstChild);
 else
 video.parentNode.insertBefore(video.firstChild, video);
 }
 video.parentNode.removeChild(video);
}
</script>
<video controls autoplay>
<source src='video.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src='video.ogv' type='video/ogg; codecs="theora, vorbis"'>
 onerror="fallback(parentNode)">
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**Categories:**

None.

**Contexts in which this element can be used:**

As a child of a media element, before any flow content.

**Content model:**

Empty.

**Content ↴**

c Too slow? Try reading the multipage copy of  
 k the spec instead: [https://web.archive.org  
 s /web/20130103030411/http:  
 l //whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

**DOM inter.**

```
IDL interface HTMLTrackElement : HTMLElement {
 attribute DOMString kind;
 attribute DOMString src;
 attribute DOMString srclang;
 attribute DOMString label;
 attribute boolean default;

 const unsigned short NONE = 0;
 const unsigned short LOADING = 1;
 const unsigned short LOADED = 2;
 const unsigned short ERROR = 3;
 readonly attribute unsigned short readyState;

 readonly attribute TextTrack track;
};
```

The `track` element allows authors to specify explicit external timed text tracks for media elements. It does not represent anything on its own.

The `kind` attribute is an enumerated attribute. The following table lists the keywords defined for this attribute. The keyword given in the first cell of each row maps to the state given in the second cell.

Keyword	State	Brief description
<code>subtitles</code>	<b>Subtitles</b>	Transcription or translation of the dialogue, suitable for when the sound is available but not understood (e.g. because the user does not understand the language of the media resource's audio track). Overlaid on the video.
<code>captions</code>	<b>Captions</b>	Transcription or translation of the dialogue, sound effects, relevant musical cues, and other relevant audio information, suitable for when sound is unavailable or not clearly audible (e.g. because it is muted, drowned-out by ambient noise, or because the user is deaf). Overlaid on the video; labeled as appropriate for the hard-of-hearing.
<code>descriptions</code>	<b>Descriptions</b>	Textual descriptions of the video component of the media resource, intended for audio synthesis when the visual component is obscured, unavailable, or not usable (e.g. because the user is interacting with the application without a screen while driving, or because the user is blind). Synthesized as audio.
<code>chapters</code>	<b>Chapters</b>	Chapter titles, intended to be used for navigating the media resource. Displayed as an interactive (potentially nested) list in the user agent's interface.
<code>metadata</code>	<b>Metadata</b>	Tracks intended for use from script. Not displayed by the user agent.

The attribute may be omitted. The *missing value* `default` is the `subtitles` state.

The `src` attribute gives the address of the text track data. The value must be a valid non-empty URL potentially surrounded by spaces. This attribute must be present.

If the element has a `src` attribute whose value is not the empty string and whose value, when the attribute was set, could be successfully resolved relative to the element, then the element's **track URL** is the resulting absolute URL. Otherwise, the element's track URL is the empty string.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `srclang` attribute gives the language of the text track data. The value must be a valid BCP 47 language tag. This attribute must be present if the element's `kind` attribute is in the subtitles state. [BCP47]

If the element has a `srclang` attribute whose value is not the empty string, then the element's **track language** is the value of the attribute. Otherwise, the element has no track language.

The `label` attribute gives a user-readable title for the track. This title is used by user agents when listing subtitle, caption, and audio description tracks in their user interface.



The value of element child or have value

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

re must not be two track g attributes are both missing oth have the same value.

If the elem Othe

Otherwise, t

The `default` attribute indicates t

the value of the attribute.

the user's preferences do

Each media element must have no more than one `track` element child whose `kind` attribute is in the subtitles or captions state and whose `default` attribute is specified.

Each media element must have no more than one `track` element child whose `kind` attribute is in the description state and whose `default` attribute is specified.

Each media element must have no more than one `track` element child whose `kind` attribute is in the chapters state and whose `default` attribute is specified.

**Note: There is no limit on the number of `track` elements whose `kind` attribute is in the metadata state and whose `default` attribute is specified.**

This box is non-normative. Implementation requirements are given below this box.

#### `track.readyState`

Returns the text track readiness state, represented by a number from the following list:

##### `track.NONE (0)`

The text track not loaded state.

##### `track.LOADING (1)`

The text track loading state.

##### `track.LOADED (2)`

The text track loaded state.

##### `track.ERROR (3)`

The text track failed to load state.

#### `track.track`

Returns the `TextTrack` object corresponding to the text track of the `track` element.

The `readyState` attribute must return the numeric value corresponding to the text track readiness state of the `track` element's text track, as defined by the following list:

##### `NONE (numeric value 0)`

The text track not loaded state.

##### `LOADING (numeric value 1)`

The text track loading state.

##### `LOADED (numeric value 2)`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

The `track` IDL attribute must, on getting, return the `track` element's text track's corresponding `TextTrack` object.

The `src`, `srclang`, `label`, and `default` IDL attributes must reflect the respective content attributes of the same name. The `kind` IDL attribute must reflect the content attribute of the same name, limited to only known values.

This video has subtitles in several languages:

```
<video src="brave.webm">
<track kind=subtitles src="brave.en.vtt" srclang=en label="English">
<track kind=captions src="brave.en.hoh.vtt" srclang=en label="English for the Hard of
```

F Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### 4.8.10 Media elements

```
interface HTMLMediaElement : HTMLElement {
 // error state
 readonly attribute MediaError? error;

 // network state
 attribute DOMString src;
 readonly attribute DOMString currentSrc;
 attribute DOMString crossOrigin;
 const unsigned short NETWORK_EMPTY = 0;
 const unsigned short NETWORK_IDLE = 1;
 const unsigned short NETWORK_LOADING = 2;
 const unsigned short NETWORK_NO_SOURCE = 3;
 readonly attribute unsigned short networkState;
 attribute DOMString preload;
 readonly attribute TimeRanges buffered;
 void load();
 DOMString canPlayType(DOMString type);

 // ready state
 const unsigned short HAVE NOTHING = 0;
 const unsigned short HAVE_METADATA = 1;
 const unsigned short HAVE_CURRENT_DATA = 2;
 const unsigned short HAVE_FUTURE_DATA = 3;
 const unsigned short HAVE_ENOUGH_DATA = 4;
 readonly attribute unsigned short readyState;
 readonly attribute boolean seeking;

 // playback state
 attribute double currentTime;
 void fastSeek(double time);
 readonly attribute unrestricted double duration;
 readonly attribute Date startDate;
 readonly attribute boolean paused;
 attribute double defaultPlaybackRate;
 attribute double playbackRate;
 readonly attribute TimeRanges played;
 readonly attribute TimeRanges seekable;
 readonly attribute boolean ended;
 attribute boolean autoplay;
 attribute boolean loop;
 void play();
 void pause();
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
// controls
 attribute boolean controls;
 attribute double volume;
 attribute boolean muted;
 attribute boolean defaultMuted;

// tracks
readonly attribute AudioTrackList audioTracks;
readonly attribute VideoTrackList videoTracks;
re
Te
lang
};

Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
al DOMString
```

The `media` element is defined in the [HTML specification](#). It represents a media resource, such as an audio or video file.

Media elements have several attributes:

`src`: The URL of the media resource. `srcset`: A list of URLs for different media resources, each with its own `width` and `height` values. `type`: The MIME type of the media resource. `preload`: How much of the media resource should be preloaded. `controls`: Whether the media element has controls like play/pause, volume, and seek. `loop`: Whether the media should loop. `autoplay`: Whether the media should start playing automatically. `volume`: The volume of the media. `muted`: Whether the media is muted. `defaultMuted`: Whether the media is muted by default. `audioTracks`: An `AudioTrackList` object containing the audio tracks of the media. `videoTracks`: A `VideoTrackList` object containing the video tracks of the media.

A media resource can have multiple audio and video tracks. For the purposes of a media element, the video data of the media resource is only that of the currently selected track (if any) given by the element's `videoTracks` attribute, and the audio data of the media resource is the result of mixing all the currently enabled tracks (if any) given by the element's `audioTracks` attribute.

**Note:** Both `audio` and `video` elements can be used for both audio and video. The main difference between the two is simply that the `audio` element has no playback area for visual content (such as video or captions), whereas the `video` element does.

Except where otherwise specified, the task source for all the tasks queued in this section and its subsections is the **media element event task source**.

#### 4.8.10.1 Error codes

This box is non-normative. Implementation requirements are given below this box.

`media.error`

Returns a `MediaError` object representing the current error state of the element.

Returns null if there is no error.

All media elements have an associated error status, which records the last error the element encountered since its resource selection algorithm was last invoked. The `error` attribute, on getting, must return the `MediaError` object created for this last error, or null if there has not been an error.

```
interface MediaError {
 const unsigned short MEDIA_ERR_ABORTED = 1;
 const unsigned short MEDIA_ERR_NETWORK = 2;
 const unsigned short MEDIA_ERR_DECODE = 3;
 const unsigned short MEDIA_ERR_SRC_NOT_SUPPORTED = 4;
 readonly attribute unsigned short code;
};
```

This box is non-normative. Implementation requirements are given below this box.

`media.error.code`

Returns the current error's error code, from the list below.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**MEDIA\_ERR\_NETWORK (numeric value 2)**

A network error of some description caused the user agent to stop fetching the media resource, after the resource was established to be usable.

**MEDIA\_ERR\_DECODE (numeric value 3)**

An error of some description occurred while decoding the media resource, after the resource was established to be usable.

**MEDIA\_ERR\_SRC\_NOT\_SUPPORTED (numeric value 4)**

The media resource indicated by the `src` attribute was not suitable.

**4.8.10.2 L Too slow? Try reading the multipage copy of**

**the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

ow. The attribute, if present,

must contain

If the `itemprop`The `crosso`

If a `src` attribute of a media element is set or changed, the user agent must invoke the media element's media element load algorithm. (Removing the `src` attribute does not do this, even if there are `source` elements present.)

The `src` IDL attribute on media elements must reflect the content attribute of the same name.

The `crossOrigin` IDL attribute must reflect the `crossorigin` content attribute.

This box is non-normative. Implementation requirements are given below this box.

**media . currentSrc**

Returns the address of the current media resource.

Returns the empty string when there is no media resource.

The `currentSrc` IDL attribute is initially the empty string. Its value is changed by the resource selection algorithm defined below.

**Note:** There are two ways to specify a media resource, the `src` attribute, or `source` elements. The attribute overrides the elements.

**4.8.10.3 MIME types**

A media resource can be described in terms of its `type`, specifically a MIME type, in some cases with a `codecs` parameter. (Whether the `codecs` parameter is allowed or not depends on the MIME type.) [RFC4281]

Types are usually somewhat incomplete descriptions; for example "video/mpeg" doesn't say anything except what the container type is, and even a type like "video/mp4; codecs="avc1.42E01E, mp4a.40.2"" doesn't include information like the actual bitrate (only the maximum bitrate). Thus, given a type, a user agent can often only know whether it *might* be able to play media of that type (with varying levels of confidence), or whether it definitely *cannot* play media of that type.

**A type that the user agent knows it cannot render** is one that describes a resource that the user agent definitely does not support, for example because it doesn't recognize the container type, or it doesn't support the listed codecs.

The MIME type "application/octet-stream" with no parameters is never a type that the user agent knows it cannot render. User agents must treat that type as equivalent to the lack of any explicit Content-Type metadata when it is used to label a potential media resource.

**Note:** Only the MIME type "application/octet-stream" with no parameters is special-cased here; if any parameter appears with it, it will be treated just like any other MIME type. This is a deviation from the rule that unknown MIME type parameters should be ignored.

This box is non-normative. Implementation requirements are given below this box.

**media . canPlayType(type)**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

The `canPlayType(type)` method must return the empty string if `type` is a type that the user agent knows it cannot render or is the type "application/octet-stream"; it must return "probably" if the user agent is confident that the type represents a media resource that it can render if used in with this `audio` or `video` element; and it must return "maybe" otherwise. Implementors are encouraged to return "maybe" unless the type can be confidently established as being supported or not. Generally, a user agent should never return "probably" for a type that allows the `codecs` parameter if that parameter is not present.

This script tests to see if the user agent supports a (fictional) new format to dynamically decide whether to use a `video` element or a plugin:

&lt;



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```

 // no configuration of browser support
 // but we have a plugin
 // so use plugin instead
 videoElement = document.createElement("embed");
} else if (support == "") {
 // no support from browser and no plugin
 // do nothing
 videoElement = null;
}
if (videoElement) {
 while (videoSection.hasChildNodes())
 videoSection.removeChild(videoSection.firstChild);
 videoElement.setAttribute("src", "playing-cats.nfv");
 videoSection.appendChild(videoElement);
}
</script>

```

or.plugins) {

**Note:** The `type` attribute of the `source` element allows the user agent to avoid downloading resources that use formats it cannot render.

### 4.8.10.4 Network states

This box is non-normative. Implementation requirements are given below this box.

**media . networkState**

Returns the current state of network activity for the element, from the codes in the list below.

As media elements interact with the network, their current network activity is represented by the `networkState` attribute. On getting, it must return the current network state of the element, which must be one of the following values:

**NETWORK\_EMPTY (numeric value 0)**

The element has not yet been initialized. All attributes are in their initial states.

**NETWORK\_IDLE (numeric value 1)**

The element's resource selection algorithm is active and has selected a resource, but it is not actually using the network at this time.

**NETWORK\_LOADING (numeric value 2)**

The user agent is actively trying to download data.

**NETWORK\_NO\_SOURCE (numeric value 3)**

The element's resource selection algorithm is active, but it has not yet found a resource to use.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

This box is non-normative. Implementation requirements are given below this box.

**`media . load()`**

Causes the element to reset and start selecting and loading a new media resource from scratch.

All media elements have an **autoplaying flag**, which must begin in the true state, and a **delaying-the-load-event flag**, which must begin in the false state. When the `load()` method is called, if the media element's `networkState` is set to `NETWORK_LOADING` or `NETWORK_IDLE`, queue a task to fire a simple event named `abort` at the media element.

When the `load()` method is called, if the media element's `networkState` is set to `NETWORK_EMPTY`, queue a task to fire a simple event named `emptied` at the media element. Then invoke the media element's resource selection algorithm.

1. If the `load()` method is called, queue a task to fire a simple event named `emptied` at the media element. Then invoke the media element's resource selection algorithm.
2. If the `load()` method is called, queue a task to fire a simple event named `emptied` at the media element. Then invoke the media element's resource selection algorithm.

If there are any tasks that were queued by the resource selection algorithm for this same media element from the DOM manipulation task source in one of the task queues, then remove those tasks.

**Note:** *Basically, pending events and callbacks for the media element are discarded when the media element starts loading a new resource.*

3. If the media element's `networkState` is set to `NETWORK_LOADING` or `NETWORK_IDLE`, queue a task to fire a simple event named `abort` at the media element.

4. If the media element's `networkState` is not set to `NETWORK_EMPTY`, then run these substeps:

1. Queue a task to fire a simple event named `emptied` at the media element.
2. If a fetching process is in progress for the media element, the user agent should stop it.
3. Forget the media element's media-resource-specific text tracks.
4. If `readyState` is not set to `HAVE NOTHING`, then set it to that state.
5. If the `paused` attribute is false, then set it to true.
6. If `seeking` is true, set it to false.
7. Set the current playback position to 0.

Set the official playback position to 0.

If this changed the official playback position, then queue a task to fire a simple event named `timeupdate` at the media element.

8. Set the initial playback position to 0.
9. Set the timeline offset to Not-a-Number (NaN).
10. Update the `duration` attribute to Not-a-Number (NaN).

**Note:** *The user agent will not fire a `durationchange` event for this particular change of the duration.*

5. Set the `playbackRate` attribute to the value of the `defaultPlaybackRate` attribute.

6. Set the `error` attribute to null and the `autoplaying` flag to true.

7. Invoke the media element's resource selection algorithm.

8. **Note:** *Playback of any previously playing media resource for this element stops.*

The **resource selection algorithm** for a media element is as follows. This algorithm is always invoked synchronously, but one of the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

2. Set the element's `showPoster` flag to true.
  3. Asynchronously await a stable state, allowing the task that invoked this algorithm to continue. The synchronous section consists of all the remaining steps of this algorithm until the algorithm says the synchronous section has ended. (Steps in synchronous sections are marked with .)
  4. If the media element's `blockedOnParser` flag is false, then populate the list of pending text tracks.
  5. If the media element has a `src` attribute, then let `mode` be `attribute`.
 

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
  6. `NET` /`whatwg.org/html`
  7. Queue a task to fire a simple event named `loadstart` at the media element.
  8. If `mode` is `attribute`, then run these substeps:
    1. *Process candidate:* If the `src` attribute's value is the empty string, then end the synchronous section, and jump down to the *failed* step below.
    2. Let `absolute URL` be the absolute URL that would have resulted from resolving the URL specified by the `src` attribute's value relative to the media element when the `src` attribute was last changed.
    3. If `absolute URL` was obtained successfully, set the `currentSrc` attribute to `absolute URL`.
    4. End the synchronous section, continuing the remaining steps asynchronously.
    5. If `absolute URL` was obtained successfully, run the resource fetch algorithm with `absolute URL`. If that algorithm returns without aborting *this* one, then the load failed.
    6. *Failed:* Reaching this step indicates that the media resource failed to load or that the given URL could not be resolved. In one atomic operation, run the following steps:
      1. Set the `error` attribute to a new `MediaError` object whose `code` attribute is set to `MEDIA_ERR_SRC_NOT_SUPPORTED`.
      2. Forget the media element's media-resource-specific text tracks.
      3. Set the element's `networkState` attribute to the `NETWORK_NO_SOURCE` value.
      4. Set the element's `showPoster` flag to true.
    7. Queue a task to fire a simple event named `error` at the media element, using the DOM manipulation task source.
    8. Set the element's `delayingTheLoadEvent` flag to false. This stops delaying the load event.
    9. Abort these steps. Until the `load()` method is invoked or the `src` attribute is changed, the element won't attempt to load another resource.
- Otherwise, the `source` elements will be used; run these substeps:
1. Let `pointer` be a position defined by two adjacent nodes in the media element's child list, treating the start of the list (before the first child in the list, if any) and end of the list (after the last child in the list, if any) as nodes in their own right. One node is the node before `pointer`, and the other node is the node after `pointer`. Initially, let `pointer` be the position between the `candidate` node and the next node, if there are any, or the end of the list, if it is the last node.
- As nodes are inserted and removed into the media element, `pointer` must be updated as follows:
- If a new node is inserted between the two nodes that define `pointer`**
- Let `pointer` be the point between the node before `pointer` and the new node. In other words, insertions at `pointer` go after `pointer`.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Let *pointer* be the point between the node before *pointer* and the node after the node before *pointer*. Just as with the previous case, *pointer* doesn't move relative to the remaining nodes.

Other changes don't affect *pointer*.

2. *Process candidate*: If *candidate* does not have a `src` attribute, or if its `src` attribute's value is the empty string, then end the synchronous section, and jump down to the *failed* step below.
3. Let *absolute URL* be the absolute URL that would have resulted from resolving the URL specified by *candidate*'s

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

jump down to the *failed* step

any codecs described by user agent knows it cannot

end the synchronous

section, and jump down to the *raisea* step below.

7. Set the `currentSrc` attribute to *absolute URL*.
8. End the synchronous section, continuing the remaining steps asynchronously.
9. Run the resource fetch algorithm with *absolute URL*. If that algorithm returns without aborting *this* one, then the load failed.
10. *Failed*: Queue a task to fire a simple event named `error` at the *candidate* element, in the context of the fetching process that was used to try to obtain *candidate*'s corresponding media resource in the resource fetch algorithm.
11. Asynchronously await a stable state. The synchronous section consists of all the remaining steps of this algorithm until the algorithm says the synchronous section has ended. (Steps in synchronous sections are marked with 
- 12. Forget the media element's media-resource-specific text tracks.
- 13. *Find next candidate*: Let *candidate* be null.
- 14. *Search loop*: If the node after *pointer* is the end of the list, then jump to the *waiting* step below.
- 15. If the node after *pointer* is a `source` element, let *candidate* be that element.
- 16. Advance *pointer* so that the node before *pointer* is now the node that was after *pointer*, and the node after *pointer* is the node after the node that used to be after *pointer*, if any.
- 17. If *candidate* is null, jump back to the *search loop* step. Otherwise, jump back to the *process candidate* step.
- 18. *Waiting*: Set the element's `networkState` attribute to the `NETWORK_NO_SOURCE` value.
- 19. Set the element's `show poster` flag to true.
- 20. Set the element's delaying-the-load-event flag to false. This stops delaying the load event.
- 21. End the synchronous section, continuing the remaining steps asynchronously.
- 22. Wait until the node after *pointer* is a node other than the end of the list. (This step might wait forever.)
- 23. Asynchronously await a stable state. The synchronous section consists of all the remaining steps of this algorithm until the algorithm says the synchronous section has ended. (Steps in synchronous sections are marked with 
- 24. Set the element's delaying-the-load-event flag back to true (this delays the load event again, in case it hasn't been fired yet).
- 25. Set the `networkState` back to `NETWORK_LOADING`.
- 26. Jump back to the *find next candidate* step above.

The **resource fetch algorithm** for a media element and a given absolute URL is as follows:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

3. Optionally, run the following substeps. This is the expected behavior if the user agent intends to not attempt to fetch the resource until the user requests it explicitly (e.g. as a way to implement the `preload` attribute's `none` keyword).

1. Set the `networkState` to `NETWORK_IDLE`.
2. Queue a task to fire a simple event named `suspend` at the element.
3. Wait for the task to be run.
4. Wait for an implementation-defined event (e.g. the user requesting that the media element begin playback).

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

mode being the state of the Document, and the default mode or CORS-cross-origin; this affects, whether a canvas gets .

While the load is not suspended (see below), every 350ms ( $\pm 200\text{ms}$ ) or for every byte received, whichever is least frequent, queue a task to fire a simple event named `progress` at the element.

The **stall timeout** is a user-agent defined length of time, which should be about three seconds. When a media element that is actively attempting to obtain media data has failed to receive any data for a duration equal to the stall timeout, the user agent must queue a task to fire a simple event named `stalled` at the element.

User agents may allow users to selectively block or slow media data downloads. When a media element's download has been blocked altogether, the user agent must act as if it was stalled (as opposed to acting as if the connection was closed). The rate of the download may also be throttled automatically by the user agent, e.g. to balance the download with other connections sharing the same bandwidth.

User agents may decide to not download more content at any time, e.g. after buffering five minutes of a one hour media resource, while waiting for the user to decide whether to play the resource or not, or while waiting for user input in an interactive resource. When a media element's download has been suspended, the user agent must queue a task to set the `networkState` to `NETWORK_IDLE` and fire a simple event named `suspend` at the element. If and when downloading of the resource resumes, the user agent must queue a task to set the `networkState` to `NETWORK_LOADING`. Between the queuing of these tasks, the load is suspended (so `progress` events don't fire, as described above).

**Note:** The `preload` attribute provides a hint regarding how much buffering the author thinks is advisable, even in the absence of the `autoplay` attribute.

When a user agent decides to completely stall a download, e.g. if it is waiting until the user starts playback before downloading any further content, the element's delaying-the-load-event flag must be set to false. This stops delaying the load event.

The user agent may use whatever means necessary to fetch the resource (within the constraints put forward by this and other specifications); for example, reconnecting to the server in the face of network errors, using HTTP range retrieval requests, or switching to a streaming protocol. The user agent must consider a resource erroneous only if it has given up trying to fetch it.

This specification does not currently say whether or how to check the MIME types of the media resources, or whether or how to perform file type sniffing using the actual file data. Implementors differ in their intentions on this matter and it is therefore unclear what the right solution is. In the absence of any requirement here, the HTTP specification's strict requirement to follow the Content-Type header prevails ("Content-Type specifies the media type of the underlying data." ... "If and only if the media type is not given by a Content-Type field, the recipient MAY attempt to guess the media type via inspection of its content and/or the name extension(s) of the URI used to identify the resource.").

The networking task source tasks to process the data as it is being fetched must, when appropriate, include the relevant substeps from the following list:

- ↪ If the media data cannot be fetched at all, due to network errors, causing the user agent to give up trying to fetch the resource
- ↪ If the media data can be fetched but is found by inspection to be in an unsupported format, or can otherwise not

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

to execute the following steps:

1. The user agent should cancel the fetching process.
2. Abort this subalgorithm, returning to the resource selection algorithm.

↳ If the media resource is found to have an audio track

1. Create an `AudioTrack` object to represent the audio track.

2. Update the media element's `audioTracks` attribute's `AudioTrackList` object with the new

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

object cancelable, and that uses `AudioTrack` object, at this

↳ `/web/20130103030411/http://whatwg.org/html`

2. Update the media element's `videoTracks` attribute's `VideoTrackList` object with the new `VideoTrack` object.
3. Fire a trusted event with the name `addtrack`, that does not bubble and is not cancelable, and that uses the `TrackEvent` interface, with the `track` attribute initialized to the new `VideoTrack` object, at this `VideoTrackList` object.

↳ Once enough of the media data has been fetched to determine the duration of the media resource, its dimensions, and other metadata

This indicates that the resource is usable. The user agent must follow these substeps:

1. Establish the media timeline for the purposes of the current playback position, the earliest possible position, and the initial playback position, based on the media data.
2. Update the timeline offset to the date and time that corresponds to the zero time in the media timeline established in the previous step, if any. If no explicit time and date is given by the media resource, the timeline offset must be set to Not-a-Number (NaN).
3. Set the current playback position and the official playback position to the earliest possible position.
4. Update the `duration` attribute with the time of the last frame of the resource, if known, on the media timeline established above. If it is not known (e.g. a stream that is in principle infinite), update the `duration` attribute to the value positive Infinity.

**Note: The user agent will queue a task to fire a simple event named `durationchange` at the element at this point.**

5. For `video` elements, set the `videoWidth` and `videoHeight` attributes.

6. Set the `readyState` attribute to `HAVE_METADATA`.

**Note: A `loadedmetadata` DOM event will be fired as part of setting the `readyState` attribute to a new value.**

7. Let `jumped` be false.

8. If the media element's default playback start position is greater than zero, then seek to that time, and let `jumped` be true.

9. Let the media element's default playback start position be zero.

10. If either the media resource or the address of the *current media resource* indicate a particular start time, then set the initial playback position to that time and, if `jumped` is still false, seek to that time and let `jumped` be true.

|| For example, with media formats that support the *Media Fragments URI* fragment identifier

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

must be selected.

12. If the media element has a current media controller, then: if *jumped* is true and the initial playback position, relative to the current media controller's timeline, is greater than the current media controller's media controller position, then seek the media controller to the media element's initial playback position, relative to the current media controller's timeline; otherwise, seek the media element to the media controller position, relative to the media element's timeline.

Once the `readyState` attribute reaches `HAVE_CURRENT_DATA`, after the ~~readystatechange~~ event has been fired, set `networkState` to `NETWORK_IDLE`.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

*tching the metadata  
he rules described  
'WORK\_IDLE value and  
source and go through*

↳ Once the entire media resource has been fetched (but potentially before any of it has been decoded)

Fire a simple event named `progress` at the media element.

Set the `networkState` to `NETWORK_IDLE` and fire a simple event named `suspend` at the media element.

If the user agent ever discards any media data and then needs to resume the network activity to obtain it again, then it must queue a task to set the `networkState` to `NETWORK_LOADING`.

*Note: If the user agent can keep the media resource loaded, then the algorithm will continue to its final step below, which aborts the algorithm.*

↳ If the connection is interrupted after some media data has been received, causing the user agent to give up trying to fetch the resource

Fatal network errors that occur after the user agent has established whether the *current media resource* is usable (i.e. once the media element's `readyState` attribute is no longer `HAVE NOTHING`) must cause the user agent to execute the following steps:

1. The user agent should cancel the fetching process.
2. Set the `error` attribute to a new `MediaError` object whose `code` attribute is set to `MEDIA_ERR_NETWORK`.
3. Fire a simple event named `error` at the media element.
4. Set the element's `networkState` attribute to the `NETWORK_IDLE` value.
5. Set the element's delaying-the-load-event flag to false. This stops delaying the load event.
6. Abort the overall resource selection algorithm.

↳ If the media data is corrupted

Fatal errors in decoding the media data that occur after the user agent has established whether the *current media resource* is usable must cause the user agent to execute the following steps:

1. The user agent should cancel the fetching process.
2. Set the `error` attribute to a new `MediaError` object whose `code` attribute is set to `MEDIA_ERR_DECODE`.
3. Fire a simple event named `error` at the media element.
4. If the media element's `readyState` attribute has a value equal to `HAVE NOTHING`, set the element's `networkState` attribute to the `NETWORK_EMPTY` value, set the element's `show poster` flag to true, and fire a simple event named `emptied` at the element.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

#### ↳ If the media data fetching process is aborted by the user

The fetching process is aborted by the user, e.g. because the user navigated the browsing context to another page, the user agent must execute the following steps. These steps are not followed if the `load()` method itself is invoked while these steps are running, as the steps above handle that particular kind of abort.

1. The user agent should cancel the fetching process.
2. Set the `error` attribute to a new `MediaError` object whose `code` attribute is set to 

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

THING, set the element's `now poster` flag to true, and `value`.

3. Set the element's `delaying the load event flag` to `false`. This stops delaying the load event.
6. Abort the overall resource selection algorithm.

#### ↳ If the media data can be fetched but has non-fatal errors or uses, in part, codecs that are unsupported, preventing the user agent from rendering the content completely correctly but not preventing playback altogether

The server returning data that is partially usable but cannot be optimally rendered must cause the user agent to render just the bits it can handle, and ignore the rest.

#### ↳ If the media resource is found to declare a media-resource-specific text track that the user agent supports

If the media data is CORS-same-origin, run the steps to expose a media-resource-specific text track with the relevant data.

**Note:** Cross-origin videos do not expose their subtitles, since that would allow attacks such as hostile sites reading subtitles from confidential videos on a user's intranet.

When the networking task source has queued the last task as part of fetching the media resource (i.e. once the download has completed), if the fetching process completes without errors, including decoding the media data, and if all of the data is available to the user agent without network access, then, the user agent must move on to the next step. This might never happen, e.g. when streaming an infinite resource such as Web radio, or if the resource is longer than the user agent's ability to cache data.

While the user agent might still need network access to obtain parts of the media resource, the user agent must remain on this step.

For example, if the user agent has discarded the first half of a video, the user agent will remain at this step even once the playback has ended, because there is always the chance the user will seek back to the start. In fact, in this situation, once playback has ended, the user agent will end up firing a `suspend` event, as described earlier.

5. If the user agent ever reaches this step (which can only happen if the entire resource gets loaded and kept available): abort the overall resource selection algorithm.

The `preload` attribute is an enumerated attribute. The following table lists the keywords and states for the attribute — the keywords in the left column map to the states in the cell in the second column on the same row as the keyword. The attribute can be changed even once the media resource is being buffered or played; the descriptions in the table below are to be interpreted with that in mind.

Keyword	State	Brief description
<code>none</code>	<code>None</code>	Hints to the user agent that either the author does not expect the user to need the media resource, or that the server wants to minimize unnecessary traffic. This state does not provide a hint regarding how aggressively to actually download the media resource if buffering starts anyway (e.g. once the user hits "play").
<code>metadata</code>	<code>Metadata</code>	Hints to the user agent that the author does not expect the user to need the media resource, but that fetching the resource metadata (dimensions, track list, duration, etc), and maybe even the first few frames, is reasonable. If the user agent precisely fetches no more than the metadata, then the media element will end up with its <code>readyState</code> attribute set to <code>HAVE_METADATA</code> ; typically though, some frames will be obtained as well and it will probably be <code>HAVE_CURRENT_DATA</code> or <code>HAVE_FUTURE_DATA</code> . When the media resource is playing, hints to the user agent that bandwidth is to be considered scarce, e.g. suggesting throttling the download so that the media data is obtained at the slowest possible rate that still maintains consistent playback.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

though the Metadata state is suggested as a compromise between reducing server load and providing an optimal user experience.

**Note:** Authors might switch the attribute from "none" or "metadata" to "auto" dynamically once the user begins playback. For example, on a page with many videos this might be used to indicate that the many videos are not to be downloaded unless requested, but that once one is requested it is to be downloaded aggressively.

The `preload` attribute is intended to provide a hint to the user agent about what the author thinks will lead to the best user experience. The attribute may be ignored altogether, for example based on explicit user preferences or based on the available connectivity.

The `preload`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/media>.

es.

naturally has to buffer  
ever.

ts are given below this box.

Returns a `TimeRanges` object that represents the ranges of the media resource that the user agent has buffered.

The `buffered` attribute must return a new static normalized `TimeRanges` object that represents the ranges of the media resource, if any, that the user agent has buffered, at the time the attribute is evaluated. User agents must accurately determine the ranges available, even for media streams where this can only be determined by tedious inspection.

**Note:** Typically this will be a single range anchored at the zero point, but if, e.g. the user agent uses HTTP range requests in response to seeking, then there could be multiple ranges.

User agents may discard previously buffered data.

**Note:** Thus, a time position included within a range of the objects returned by the `buffered` attribute at one time can end up being not included in the range(s) of objects returned by the same attribute at later times.

#### 4.8.10.6 Offsets into the media resource

This box is non-normative. Implementation requirements are given below this box.

`media.duration`

Returns the length of the media resource, in seconds, assuming that the start of the media resource is at time zero.

Returns NaN if the duration isn't available.

Returns Infinity for unbounded streams.

`media.currentTime [= value]`

Returns the official playback position, in seconds.

Can be set, to seek to the given time.

Will throw an `InvalidStateError` exception if there is no selected media resource or if there is a current media controller.

A media resource has a **media timeline** that maps times (in seconds) to positions in the media resource. The origin of a timeline is its earliest defined position. The duration of a timeline is its last defined position.

**Establishing the media timeline:** If the media resource somehow specifies an explicit timeline whose origin is not negative (i.e. gives each frame a specific time offset and gives the first frame a zero or positive offset), then the media timeline should be that timeline. (Whether the media resource can specify a timeline or not depends on the media resource's format.) If the media resource specifies an explicit start time and date, then that time and date should be considered the zero point in the media timeline; the timeline offset will be the time and date, exposed using the `startDate` attribute.

If the media resource has a discontinuous timeline, the user agent must extend the timeline used at the start of the resource across the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

not expose those times; it would instead expose the times as 00:15..00:29 and 00:29..01:02, as a single video.

In the rare case of a media resource that does not have an explicit timeline, the zero time on the media timeline should correspond to the first frame of the media resource. In the even rarer case of a media resource with no explicit timings of any kind, not even frame durations, the user agent must itself determine the time for each frame in a user-agent-defined manner.

**Note:** An example of a file format with no explicit timeline but with explicit frame durations is the Animated GIF format. An example of a file format with no explicit timings at all is the JPEG-push format (`multipart/x-mixed-replace` with JPEG frames, often used as the format for MJPEG streams).



If, in the case of frame origin: otherwise, it began receiving

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**Note: A to a frame**

n earlier point than the first of the media resource; at which the user agent

*still supports seeking*

Consider a stream from a TV broadcaster, which begins streaming on a sunny Friday afternoon in October, and always sends connecting user agents the media data on the same media timeline, with its zero time set to the start of this stream. Months later, user agents connecting to this stream will find that the first frame they receive has a time with millions of seconds. The `startDate` attribute would always return the date that the broadcast started; this would allow controllers to display real times in their scrubber (e.g. "2:30pm") rather than a time relative to when the broadcast began ("8 months, 4 hours, 12 minutes, and 23 seconds").

Consider a stream that carries a video with several concatenated fragments, broadcast by a server that does not allow user agents to request specific times but instead just streams the video data in a predetermined order, with the first frame delivered always being identified as the frame with time zero. If a user agent connects to this stream and receives fragments defined as covering timestamps 2010-03-20 23:15:00 UTC to 2010-03-21 00:05:00 UTC and 2010-02-12 14:25:00 UTC to 2010-02-12 14:35:00 UTC, it would expose this with a media timeline starting at 0s and extending to 3,600s (one hour). Assuming the streaming server disconnected at the end of the second clip, the `duration` attribute would then return 3,600. The `startDate` attribute would return a `Date` object with a time corresponding to 2010-03-20 23:15:00 UTC. However, if a different user agent connected five minutes later, it would (presumably) receive fragments covering timestamps 2010-03-20 23:20:00 UTC to 2010-03-21 00:05:00 UTC and 2010-02-12 14:25:00 UTC to 2010-02-12 14:35:00 UTC, and would expose this with a media timeline starting at 0s and extending to 3,300s (fifty five minutes). In this case, the `startDate` attribute would return a `Date` object with a time corresponding to 2010-03-20 23:20:00 UTC.

In both of these examples, the `seekable` attribute would give the ranges that the controller would want to actually display in its UI; typically, if the servers don't support seeking to arbitrary times, this would be the range of time from the moment the user agent connected to the stream up to the latest frame that the user agent has obtained; however, if the user agent starts discarding earlier information, the actual range might be shorter.

In any case, the user agent must ensure that the earliest possible position (as defined below) using the established media timeline, is greater than or equal to zero.

The media timeline also has an associated clock. Which clock is used is user-agent defined, and may be media resource-dependent, but it should approximate the user's wall clock.

**Note:** All the media elements that share current media controller use the same clock for their media timeline.

Media elements have a **current playback position**, which must initially (i.e. in the absence of media data) be zero seconds. The current playback position is a time on the media timeline.

Media elements also have an **official playback position**, which must initially be set to zero seconds. The official playback position is an approximation of the current playback position that is kept stable while scripts are running.

Media elements also have a **default playback start position**, which must initially be set to zero seconds. This time is used to allow the element to be seeked even before the media is loaded.

Each media element has a **show poster flag**. When a media element is created, this flag must be set to true. This flag is used to control when the user agent is to show a poster frame for a `video` element instead of showing the video contents.

The `currentTime` attribute must, on getting, return the media element's default playback start position, unless that is zero, in which case it must return the element's official playback position. The returned value must be expressed in seconds. On setting, if the media

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Media elements have an **initial playback position**, which must initially (i.e. in the absence of media data) be zero seconds. The initial playback position is updated when a media resource is loaded. The initial playback position is a time on the media timeline.

If the media resource is a streaming resource, then the user agent might be unable to obtain certain parts of the resource after it has expired from its buffer. Similarly, some media resources might have a media timeline that doesn't start at zero. The **earliest possible position** is the earliest position in the stream or resource that the user agent can ever obtain again. It is also a time on the media timeline.

**Note:** The earliest possible position is not explicitly exposed in the API; it corresponds to the start time of the first range in

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**Note:** E  
metadata.  
position.

ossible position, the user event at the element in the a task to fire a simple event

at kicks in when the .  
e earliest possible

If at any time the user agent learns that an audio or video track has ended and all media data relating to that track corresponds to parts of the media timeline that are *before* the earliest possible position, the user agent may queue a task to remove the track from the `audioTracks` attribute's `AudioTrackList` object or the `videoTracks` attribute's `VideoTrackList` object as appropriate and then fire a trusted event with the name `removetrack`, that does not bubble and is not cancelable, and that uses the `TrackEvent` interface, with the `track` attribute initialized to the `AudioTrack` or `VideoTrack` object representing the track, at the media element's aforementioned `AudioTrackList` or `VideoTrackList` object.

The `duration` attribute must return the time of the end of the media resource, in seconds, on the media timeline. If no media data is available, then the attributes must return the Not-a-Number (NaN) value. If the media resource is not known to be bounded (e.g. streaming radio, or a live event with no announced end time), then the attribute must return the positive Infinity value.

The user agent must determine the duration of the media resource before playing any part of the media data and before setting `readyState` to a value equal to or greater than `HAVE_METADATA`, even if doing so requires fetching multiple parts of the resource.

When the length of the media resource changes to a known value (e.g. from being unknown to known, or from a previously established length to a new length) the user agent must queue a task to fire a simple event named `durationchange` at the media element. (The event is not fired when the duration is reset as part of loading a new media resource.) If the duration is changed such that the current playback position ends up being greater than the time of the end of the media resource, then the user agent must also seek to the time of the end of the media resource.

If an "infinite" stream ends for some reason, then the duration would change from positive Infinity to the time of the last frame or sample in the stream, and the `durationchange` event would be fired. Similarly, if the user agent initially estimated the media resource's duration instead of determining it precisely, and later revises the estimate based on new information, then the duration would change and the `durationchange` event would be fired.

Some video files also have an explicit date and time corresponding to the zero time in the media timeline, known as the **timeline offset**. Initially, the timeline offset must be set to Not-a-Number (NaN).

The `startDate` attribute must return a new `Date` object representing the current timeline offset.

The `loop` attribute is a boolean attribute that, if specified, indicates that the media element is to seek back to the start of the media resource upon reaching the end.

The `loop` attribute has no effect while the element has a current media controller.

The `loop` IDL attribute must reflect the content attribute of the same name.

#### 4.8.10.7 Ready states

This box is non-normative. Implementation requirements are given below this box.

`media . readyState`

Go

NOV JAN FEB

03

2012 2013 2014

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

Media elements have a *ready state*, which describes to what degree they are ready to be rendered at the current playback position. The possible values are as follows; the ready state of a media element at any particular time is the greatest value describing the state of the element:

**HAVE NOTHING (numeric value 0)**

No information regarding the media resource is available. No data for the current playback position is available. Media elements whose `networkState` attribute are set to `NETWORK_EMPTY` are always in the `HAVE NOTHING` state.

**HAVE\_METADATA (numeric value 1)**

Eno  
dim  
ava

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

**HAVE\_CURR**

Dat  
suc  
HAV  
to th

/web/20130103030411/http:  
//whatwg.org/html

of the current frame; and to when playback has ended.

use of a `video` element, the g. No media data is

le that the user agent could directly reverting to the e, in video this corresponds playback position is at the end

**HAVE\_FUTURE\_DATA (numeric value 3)**

Data for the immediate current playback position is available, as well as enough data for the user agent to advance the current playback position in the direction of playback at least a little without immediately reverting to the `HAVE_METADATA` state, and the text tracks are ready. For example, in video this corresponds to the user agent having data for at least the current frame and the next frame when the current playback position is at the instant in time between the two frames, or to the user agent having the video data for the current frame and audio data to keep playing at least a little when the current playback position is in the middle of a frame. The user agent cannot be in this state if playback has ended, as the current playback position can never advance in this case.

**HAVE\_ENOUGH\_DATA (numeric value 4)**

All the conditions described for the `HAVE_FUTURE_DATA` state are met, and, in addition, either of the following conditions is also true:

- The user agent estimates that data is being fetched at a rate where the current playback position, if it were to advance at the effective playback rate, would not overtake the available data before playback reaches the end of the media resource.
- The user agent has entered a state where waiting longer will not result in further data being obtained, and therefore nothing would be gained by delaying playback any further. (For example, the buffer might be full.)

**Note:** In practice, the difference between `HAVE_METADATA` and `HAVE_CURRENT_DATA` is negligible. Really the only time the difference is relevant is when painting a `video` element onto a canvas, where it distinguishes the case where something will be drawn (`HAVE_CURRENT_DATA` or greater) from the case where nothing is drawn (`HAVE_METADATA` or less). Similarly, the difference between `HAVE_CURRENT_DATA` (only the current frame) and `HAVE_FUTURE_DATA` (at least this frame and the next) can be negligible (in the extreme, only one frame). The only time that distinction really matters is when a page provides an interface for "frame-by-frame" navigation.

When the ready state of a media element whose `networkState` is not `NETWORK_EMPTY` changes, the user agent must follow the steps given below:

1. Apply the first applicable set of substeps from the following list:

↳ If the previous ready state was `HAVE NOTHING`, and the new ready state is `HAVE_METADATA`

Queue a task to fire a simple event named `loadedmetadata` at the element.

**Note:** Before this task is run, as part of the event loop mechanism, the rendering will have been updated to resize the `video` element if appropriate.

↳ If the previous ready state was `HAVE_METADATA` and the new ready state is `HAVE_CURRENT_DATA` or greater

If this is the first time this occurs for this media element since the `load()` algorithm was last invoked, the user agent must queue a task to fire a simple event named `loadeddata` at the element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the media element was potentially playing before its `readyState` attribute changed to a value lower than `HAVE_FUTURE_DATA`, and the element has not ended playback, and playback has not stopped due to errors, paused for user interaction, or paused for in-band content, the user agent must queue a task to fire a simple event named `timeupdate` at the element, and queue a task to fire a simple event named `waiting` at the element.

↳ If the previous ready state was `HAVE_CURRENT_DATA` or less, and the new ready state is `HAVE_FUTURE_DATA`

The user agent must queue a task to fire a simple event named `canplay`.

If the element's `paused` attribute is false, the user agent must queue a task to fire a simple event named `canplaythrough`.

↳ Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

queue a task to fire a simple  
to fire a simple event named

it has an `autoplay` attribute  
it have the `sandboxed`

automatic features browsing context flag set, then the user agent may also set the `paused` attribute to false, set the element's `show poster` flag to false, queue a task to fire a simple event named `play`, and queue a task to fire a simple event named `playing`.

**Note:** User agents do not need to support `autoplay`, and it is suggested that user agents honor user preferences on the matter. Authors are urged to use the `autoplay` attribute rather than using script to force the video to play, so as to allow the user to override the behavior if so desired.

In any case, the user agent must finally queue a task to fire a simple event named `canplaythrough`.

2. If the media element has a current media controller, then report the controller state for the media element's current media controller.

**Note:** It is possible for the ready state of a media element to jump between these states discontinuously. For example, the state of a media element can jump straight from `HAVE_METADATA` to `HAVE_ENOUGH_DATA` without passing through the `HAVE_CURRENT_DATA` and `HAVE_FUTURE_DATA` states.

The `readyState` IDL attribute must, on getting, return the value described above that describes the current ready state of the media element.

The `autoplay` attribute is a boolean attribute. When present, the user agent (as described in the algorithm described herein) will automatically begin playback of the media resource as soon as it can do so without stopping.

**Note:** Authors are urged to use the `autoplay` attribute rather than using script to trigger automatic playback, as this allows the user to override the automatic playback when it is not desired, e.g. when using a screen reader. Authors are also encouraged to consider not using the automatic playback behavior at all, and instead to let the user agent wait for the user to start playback explicitly.

The `autoplay` IDL attribute must reflect the content attribute of the same name.

#### 4.8.10.8 Playing the media resource

This box is non-normative. Implementation requirements are given below this box.

`media . paused`

Returns true if playback is paused; false otherwise.

`media . ended`

Returns true if playback has reached the end of the media resource.

`media . defaultPlaybackRate [= value]`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the element has a current media controller, the `defaultPlaybackRate` attribute is ignored and the current media controller's `defaultPlaybackRate` is used instead.

### `media . playbackRate [= value]`

Returns the current rate playback, where 1.0 is normal speed.

Can be set, to change the rate of playback.

When the element has a current media controller, the `playbackRate` attribute is ignored and the current media controller's `playbackRate` is used instead.

### `media . Too slow? Try reading the multipage copy of`

`Ref` the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### `media . /web/20130103030411/http:`

`Sel`

`enc` //whatwg.org/html

### `media . pause()`

Sets the `paused` attribute to true, loading the media resource if necessary.

The `paused` attribute represents whether the media element is paused or not. The attribute must initially be true.

A media element is a **blocked media element** if its `readyState` attribute is in the `HAVE NOTHING` state, the `HAVE_METADATA` state, or the `HAVE_CURRENT_DATA` state, or if the element has paused for user interaction or paused for in-band content.

A media element is said to be **potentially playing** when its `paused` attribute is false, the element has not ended playback, playback has not stopped due to errors, the element either has no current media controller or has a current media controller but is not blocked on its media controller, and the element is not a blocked media element.

**Note:** A waiting DOM event can be fired as a result of an element that is potentially playing stopping playback due to its `readyState` attribute changing to a value lower than `HAVE_FUTURE_DATA`.

A media element is said to have **ended playback** when:

- The element's `readyState` attribute is `HAVE_METADATA` or greater, and
- Either:
  - The current playback position is the end of the media resource, and
  - The direction of playback is forwards, and
  - Either the media element does not have a `loop` attribute specified, or the media element has a current media controller.

Or:

- The current playback position is the earliest possible position, and
- The direction of playback is backwards.

The `ended` attribute must return true if, the last time the event loop reached step 1, the media element had ended playback and the direction of playback was forwards, and false otherwise.

A media element is said to have **stopped due to errors** when the element's `readyState` attribute is `HAVE_METADATA` or greater, and the user agent encounters a non-fatal error during the processing of the media data, and due to that error, is not able to play the content at the current playback position.

A media element is said to have **paused for user interaction** when its `paused` attribute is false, the `readyState` attribute is either `HAVE_FUTURE_DATA` or `HAVE_ENOUGH_DATA` and the user agent has reached a point in the media resource where the user has to make a selection for the resource to continue. If the media element has a current media controller when this happens, then the user agent must report the controller state for the media element's current media controller. If the media element has a current media

Go

NOV JAN FEB

03

2012 2013 2014

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

When a media element that is potentially playing stops playing because it has paused for user interaction, the user agent must queue a task to fire a simple event named `timeupdate` at the element.

A media element is said to have **paused for in-band content** when its `paused` attribute is false, the `readyState` attribute is either `HAVE_FUTURE_DATA` or `HAVE_ENOUGH_DATA` and the user agent has suspended playback of the media resource in order to play content that is temporally anchored to the media resource and has a non-zero length, or to play content that is temporally anchored to a segment of the media resource but has a length longer than that segment. If the media element has a current media controller when this happens, then the user agent must report the controller state for the media element's current media controller. If the media element has a current media controller when the user agent unsuspends playback, the user agent must ~~similarly~~ report the controller state for the media eler

One desc text t **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

it is playing audio  
er than the time between the

When the cl agent must ~~1~~ is forwards, then the user

1. If the ~~1~~ direction of playback is still backwards, then seek to the earliest possible position of the media resource and abort these steps.
2. As defined above, the `ended` IDL attribute starts returning true once the event loop's current task ends.
3. Queue a task to fire a simple event named `timeupdate` at the media element.
4. Queue a task that, if the media element does not have a current media controller, and the media element has still ended playback, and the direction of playback is still forwards, and `paused` is false, changes `paused` to true and fires a simple event named `pause` at the media element.
5. Queue a task to fire a simple event named `ended` at the media element.
6. If the media element has a current media controller, then report the controller state for the media element's current media controller.

When the current playback position reaches the earliest possible position of the media resource when the direction of playback is backwards, then the user agent must only queue a task to fire a simple event named `timeupdate` at the element.

The `defaultPlaybackRate` attribute gives the desired speed at which the media resource is to play, as a multiple of its intrinsic speed. The attribute is mutable: on getting it must return the last value it was set to, or 1.0 if it hasn't yet been set; on setting the attribute must be set to the new value.

**Note:** The `defaultPlaybackRate` is used by the user agent when it exposes a user interface to the user.

The `playbackRate` attribute gives the effective playback rate (assuming there is no current media controller overriding it), which is the speed at which the media resource plays, as a multiple of its intrinsic speed. If it is not equal to the `defaultPlaybackRate`, then the implication is that the user is using a feature such as fast forward or slow motion playback. The attribute is mutable: on getting it must return the last value it was set to, or 1.0 if it hasn't yet been set; on setting the attribute must be set to the new value, and the playback will change speed (if the element is potentially playing and there is no current media controller).

When the `defaultPlaybackRate` or `playbackRate` attributes change value (either by being set by script or by being changed directly by the user agent, e.g. in response to user control) the user agent must queue a task to fire a simple event named `ratechange` at the media element.

**Note:** The `defaultPlaybackRate` and `playbackRate` attributes have no effect when the media element has a current media controller; the namesake attributes on the `MediaController` object are used instead in that situation.

The `played` attribute must return a new static normalized `TimeRanges` object that represents the ranges of points on the media timeline of the media resource reached through the usual monotonic increase of the current playback position during normal playback, if any, at the time the attribute is evaluated.

When the `play()` method on a media element is invoked, the user agent must run the following steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Note:** This will cause the user agent to queue a task to fire a simple event named `timeupdate` at the media element.

3. If the media element has a current media controller, then bring the media element up to speed with its new media controller.

4. If the media element's `paused` attribute is true, run the following substeps:

1. Change the value of `paused` to false.



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

.DATA, or  
nt.

'A or HAVE\_ENOUGH\_DATA:

5. Set the media element's `autoplaying` flag to false.

6. If the media element has a current media controller, then report the controller state for the media element's current media controller.

When the `pause()` method is invoked, and when the user agent is required to pause the media element, the user agent must run the following steps:

1. If the media element's `networkState` attribute has the value `NETWORK_EMPTY`, invoke the media element's resource selection algorithm.

2. Set the media element's `autoplaying` flag to false.

3. If the media element's `paused` attribute is false, run the following steps:

1. Change the value of `paused` to true.

2. Queue a task to fire a simple event named `timeupdate` at the element.

3. Queue a task to fire a simple event named `pause` at the element.

4. Set the official playback position to the current playback position.

4. If the media element has a current media controller, then report the controller state for the media element's current media controller.

The **effective playback rate** is not necessarily the element's `playbackRate`. When a media element has a current media controller, its effective playback rate is the `MediaController`'s media controller playback rate. Otherwise, the effective playback rate is just the element's `playbackRate`. Thus, the current media controller overrides the media element.

If the effective playback rate is positive or zero, then the **direction of playback** is forwards. Otherwise, it is backwards.

When a media element is potentially playing and its `Document` is a fully active `Document`, its current playback position must increase monotonically at effective playback rate units of media time per unit time of the media timeline's clock. (This specification always refers to this as an *increase*, but that increase could actually be a *decrease* if the effective playback rate is negative.)

**Note:** The effective playback rate can be 0.0, in which case the current playback position doesn't move, despite playback not being paused (paused doesn't become true, and the pause event doesn't fire).

**Note:** This specification doesn't define how the user agent achieves the appropriate playback rate — depending on the protocol and media available, it is plausible that the user agent could negotiate with the server to have the server provide the media data at the appropriate rate, so that (except for the period between when the rate is changed and when the server updates the stream's playback rate) the client doesn't actually have to drop or interpolate any frames.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Media elements that are potentially playing while not in a Document must not play any video, but should play any audio component. Media elements must not stop playing just because all references to them have been removed; only once a media element is in a state where no further audio could ever be played by that element may the element be garbage collected.

**Note:** It is possible for an element to which no explicit references exist to play audio, even if such an element is not still actively playing: for instance, it could have a current media controller that still has references and can still be unpause, or it could be unpause but stalled waiting for content to buffer.



Each media  
of cues of a  
introduced c  
cues must b

When the cl  
following ste  
complete, ai  
a long time, th..

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

track cue is added to the list  
edia element's list of newly  
s in that text track's list of

user agent must run the  
ust wait for the steps to  
eeded — if one iteration takes  
(.)

1. Let *current cues* be a list of cues, initialized to contain all the cues of all the hidden or showing text tracks of the media element (not the disabled ones) whose start times are less than or equal to the current playback position and whose end times are greater than the current playback position.
2. Let *other cues* be a list of cues, initialized to contain all the cues of hidden and showing text tracks of the media element that are not present in *current cues*.
3. Let *last time* be the current playback position at the time this algorithm was last run for this media element, if this is not the first time it has run.
4. If the current playback position has, since the last time this algorithm was run, only changed through its usual monotonic increase during normal playback, then let *missed cues* be the list of cues in *other cues* whose start times are greater than or equal to *last time* and whose end times are less than or equal to the current playback position. Otherwise, let *missed cues* be an empty list.
5. Remove all the cues in *missed cues* that are also in the media element's list of newly introduced cues, and then empty the element's list of newly introduced cues.
6. If the time was reached through the usual monotonic increase of the current playback position during normal playback, and if the user agent has not fired a `timeupdate` event at the element in the past 15 to 250ms and is not still running event handlers for such an event, then the user agent must queue a task to fire a simple event named `timeupdate` at the element. (In the other cases, such as explicit seeks, relevant events get fired as part of the overall process of changing the current playback position.)

**Note:** The event thus is not to be fired faster than about 66Hz or slower than 4Hz (assuming the event handlers don't take longer than 250ms to run). User agents are encouraged to vary the frequency of the event based on the system load and the average cost of processing the event each time, so that the UI updates are not any more frequent than the user agent can comfortably handle while decoding the video.

7. If all of the cues in *current cues* have their text track cue active flag set, none of the cues in *other cues* have their text track cue active flag set, and *missed cues* is empty, then abort these steps.
8. If the time was reached through the usual monotonic increase of the current playback position during normal playback, and there are cues in *other cues* that have their text track cue pause-on-exit flag set and that either have their text track cue active flag set or are also in *missed cues*, then immediately pause the media element.

**Note:** In the other cases, such as explicit seeks, playback is not paused by going past the end time of a cue, even if that cue has its text track cue pause-on-exit flag set.

9. Let *events* be a list of tasks, initially empty. Each task in this list will be associated with a text track, a text track cue, and a time, which are used to sort the list before the tasks are queued.

Let *affected tracks* be a list of text tracks, initially empty.

When the steps below say to **prepare an event** named *event* for a text track cue *target* with a time *time*, the user agent must

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. Add to the newly created task to *events*, associated with the time *time*, the text track *track*, and the text track cue *target*.
4. Add *track* to *affected tracks*.
10. For each text track cue in *missed cues*, prepare an event named `enter` for the `TextTrackCue` object with the text track cue start time.
11. For each text track cue in *other cues* that either has its text track cue active flag set or is in *missed cues*, prepare an event named `exit` for the `TextTrackCue` object with the later of the text track cue end time the text track cue start time.
12. For the **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html** an event named `enter` for
13. Sorry, before those that fire `exit` events.
14. Queue each task in *events*, in list order.
15. Sort *affected tracks* in the same order as the text tracks appear in the media element's list of text tracks, and remove duplicates.
16. For each text track in *affected tracks*, in the list order, queue a task to fire a simple event named `cuechange` at the `TextTrack` object, and, if the text track has a corresponding `track` element, to then fire a simple event named `cuechange` at the `track` element as well.
17. Set the text track cue active flag of all the cues in the *current cues*, and unset the text track cue active flag of all the cues in the *other cues*.
18. Run the rules for updating the text track rendering of each of the text tracks in *affected tracks* that are showing. For example, for text tracks based on WebVTT, the rules for updating the display of WebVTT text tracks. [WEBVTT]

For the purposes of the algorithm above, a text track cue is considered to be part of a text track only if it is listed in the text track list of cues, not merely if it is associated with the text track.

**Note:** If the media element's Document stops being a fully active document, then the playback will stop until the document is active again.

When a media element is removed from a `Document`, the user agent must run the following steps:

1. Asynchronously await a stable state, allowing the task that removed the media element from the `Document` to continue. The synchronous section consists of all the remaining steps of this algorithm. (Steps in the synchronous section are marked with .
2. If the media element is in a `Document`, abort these steps.
3. If the media element's `networkState` attribute has the value `NETWORK_EMPTY`, abort these steps.
4. Pause the media element.

#### 4.8.10.9 Seeking

This box is non-normative. Implementation requirements are given below this box.

`media . seeking`

Returns true if the user agent is currently seeking.

`media . seekable`

Returns a `TimeRanges` object that represents the ranges of the media resource to which it is possible for the user agent to seek.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The `seeking` attribute must initially have the value `false`.

The `fastSeek()` method must seek to the time given by the method's argument, with the `approximate-for-speed` flag set.

When the user agent is required to `seek` to a particular *new playback position* in the media resource, optionally with the `approximate-for-speed` flag set, it means that the user agent must run the following steps. This algorithm interacts closely with the event loop mechanism; in particular, it has a synchronous section (which is triggered as part of the event loop algorithm). Steps in that section are marked with

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. Set

2. If th

3. If th

inst

4. Set ti.

5. If the seek was in response to a DOM method call or setting of an IDL attribute, then continue the script. The remainder of these steps must be run asynchronously. With the exception of the steps marked with (which could be aborted at any time by another instance of this algorithm being invoked).

6. If the *new playback position* is later than the end of the media resource, then let it be the end of the media resource instead.

7. If the *new playback position* is less than the earliest possible position, let it be that position instead.

8. If the (possibly now changed) *new playback position* is not in one of the ranges given in the `seekable` attribute, then let it be the position in one of the ranges given in the `seekable` attribute that is the nearest to the *new playback position*. If two positions both satisfy that constraint (i.e. the *new playback position* is exactly in the middle between two ranges in the `seekable` attribute) then use the position that is closest to the current playback position. If there are no ranges given in the `seekable` attribute then set the `seeking` IDL attribute to `false` and abort these steps.

9. If the `approximate-for-speed` flag is set, adjust the *new playback position* to a value that will allow for playback to resume promptly. If *new playback position* before this step is before current playback position, then the adjusted *new playback position* must also be before the current playback position. Similarly, if the *new playback position* before this step is after current playback position, then the adjusted *new playback position* must also be after the current playback position.

For example, the user agent could snap to the nearest key frame, so that it doesn't have to spend time decoding then discarding intermediate frames before resuming playback.

10. Queue a task to fire a simple event named `seeking` at the element.

11. Set the current playback position to the given *new playback position*.

**Note:** If the media element was potentially playing immediately before it started seeking, but seeking caused its `readyState` attribute to change to a value lower than `HAVE_FUTURE_DATA`, then a `waiting` will be fired at the element.

**Note:** The `currentTime` attribute does not get updated asynchronously, as it returns the official playback position, not the current playback position.

12. Wait until the user agent has established whether or not the media data for the *new playback position* is available, and, if it is, until it has decoded enough data to play back that position.

13. Await a stable state. The synchronous section consists of all the remaining steps of this algorithm. (Steps in the synchronous section are marked with .

14. Set the `seeking` IDL attribute to `false`.

15. Queue a task to fire a simple event named `timeupdate` at the element.

16. Queue a task to fire a simple event named `seeked` at the element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

agent and the server support `Range` requests, then the attribute would return an object with one range, whose start is the time of the first frame (the earliest possible position, typically zero), and whose end is the same as the time of the first frame plus the `duration` attribute's value (which would equal the time of the last frame, and might be positive infinity).

**Note:** The range might be continuously changing, e.g. if the user agent is buffering a sliding window on an infinite stream. This is the behavior seen with DVRs viewing live TV, for instance.

Media resources might be internally scripted or interactive. Thus, a media element could play in a non-linear fashion. If this happens, the user agent must seek the media in a discontinuous fashion (so to speak). The controller application must handle this.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### 4.8.10.10 /[web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

A media resource has a media resource sign-language

nary video and audio tracks, options, alternative angles, or

This box is non-normative. Implementation requirements are given below this box.

#### `media.audioTracks`

Returns an `AudioTrackList` object representing the audio tracks available in the media resource.

#### `media.videoTracks`

Returns a `VideoTrackList` object representing the video tracks available in the media resource.

The `audioTracks` attribute of a media element must return a live `AudioTrackList` object representing the audio tracks available in the media element's media resource. The same object must be returned each time.

The `videoTracks` attribute of a media element must return a live `VideoTrackList` object representing the video tracks available in the media element's media resource. The same object must be returned each time.

**Note:** There are only ever one `AudioTrackList` object and one `VideoTrackList` object per media element, even if another media resource is loaded into the element: the objects are reused. (The `AudioTrack` and `VideoTrack` objects are not, though.)

In this example, a script defines a function that takes a URL to a video and a reference to an element where the video is to be placed. That function then tries to load the video, and, once it is loaded, checks to see if there is a sign-language track available. If there is, it also displays that track. Both tracks are just placed in the given container; it's assumed that styles have been applied to make this work in a pretty way!

```
<script>
 function loadVideo(url, container) {
 var controller = new MediaController();
 var video = document.createElement('video');
 video.src = url;
 video.autoplay = true;
 video.controls = true;
 video.controller = controller;
 container.appendChild(video);
 video.onloadedmetadata = function (event) {
 for (var i = 0; i < video.videoTracks.length; i += 1) {
 if (video.videoTracks[i].kind == 'sign') {
 var sign = document.createElement('video');
 sign.src = url + '#track=' + video.videoTracks[i].id;
 sign.autoplay = true;
 sign.controller = controller;
 container.appendChild(sign);
 }
 }
 };
 }
</script>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

#### 4.8.10.10.1 `AudioTrackList` and `VideoTrackList` objects

The `AudioTrackList` and `VideoTrackList` interfaces are used by attributes defined in the previous section.

```
interface AudioTrackList : EventTarget {
 readonly attribute unsigned long length;
 getter AudioTrack (unsigned long index);
 Aur
 Too slow? Try reading the multipage copy of
 the spec instead: https://web.archive.org
}; /web/20130103030411/http:
interface VideoTrackList : EventTarget {
 readonly attribute unsigned long length;
 getter VideoTrack (unsigned long index);
 VideoTrack? getTrackById(DOMString id);
 readonly attribute long selectedIndex;

 attribute EventHandler onchange;
 attribute EventHandler onaddtrack;
 attribute EventHandler onremovetrack;
};

interface VideoTrack {
 readonly attribute DOMString id;
 readonly attribute DOMString kind;
 readonly attribute DOMString label;
 readonly attribute DOMString language;
 attribute boolean selected;
};
```



This box is non-normative. Implementation requirements are given below this box.

`media.audioTracks.length`  
`media.videoTracks.length`

Returns the number of tracks in the list.

`audioTrack = media.audioTracks[index]`  
`videoTrack = media.videoTracks[index]`

Returns the specified `AudioTrack` or `VideoTrack` object.

`audioTrack = media.audioTracks.getTrackById(id)`  
`videoTrack = media.videoTracks.getTrackById(id)`

Returns the `AudioTrack` or `VideoTrack` object with the given identifier, or null if no track has that identifier.

`audioTrack.id`  
`videoTrack.id`

Returns the ID of the given track. This is the ID that can be used with a fragment identifier if the format supports the `Media Fragments URI` syntax, and that can be used with the `getTrackById()` method. [MEDIAFRAG]

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

`audioTrack . label``videoTrack . label`

Returns the label of the given track, if known, or the empty string otherwise.

`audioTrack . language``videoTrack . language`

Returns the language of the given track, if known, or the empty string otherwise.

`audioTrack . active`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/media.html#track>

Multaneously, they are

`media . tracks`

`Re[active]` `Ca[ctive]` `mix` `//whatwg.org/html`

`videoTrack . active`

Returns true if the given track is active, and false otherwise.

Can be set, to change whether the track is selected or not. Either zero or one video track is selected; selecting a new track while a previous one is selected will unselect the previous one.

An `AudioTrackList` object represents a dynamic list of zero or more audio tracks, of which zero or more can be enabled at a time. Each audio track is represented by an `AudioTrack` object.

A `VideoTrackList` object represents a dynamic list of zero or more video tracks, of which zero or one can be selected at a time. Each video track is represented by a `VideoTrack` object.

Tracks in `AudioTrackList` and `VideoTrackList` objects must be consistently ordered. If the media resource is in a format that defines an order, then that order must be used; otherwise, the order must be the relative order in which the tracks are declared in the media resource. The order used is called the *natural order* of the list.

**Note:** Each track in a `TrackList` thus has an index; the first has the index 0, and each subsequent track is numbered one higher than the previous one. If a media resource dynamically adds or removes audio or video tracks, then the indices of the tracks will change dynamically. If the media resource changes entirely, then all the previous tracks will be removed and replaced with new tracks.

The `AudioTrackList.length` and `VideoTrackList.length` attributes must return the number of tracks represented by their objects at the time of getting.

The supported property indices of `AudioTrackList` and `VideoTrackList` objects at any instant are the numbers from zero to the number of tracks represented by the respective object minus one, if any tracks are represented. If an `AudioTrackList` or `VideoTrackList` object represents no tracks, it has no supported property indices.

To determine the value of an indexed property for a given index `index` in an `AudioTrackList` or `VideoTrackList` object `list`, the user agent must return the `AudioTrack` or `VideoTrack` object that represents the `index`th track in `list`.

The `AudioTrackList.getTrackById(id)` and `VideoTrackList.getTrackById(id)` methods must return the first `AudioTrack` or `VideoTrack` object (respectively) in the `AudioTrack` or `VideoTrack` object (respectively) whose identifier is equal to the value of the `id` argument (in the natural order of the list, as defined above). When no tracks match the given argument, the methods must return null.

The `AudioTrack` and `VideoTrack` objects represent specific tracks of a media resource. Each track can have an identifier, category, label, and language. These aspects of a track are permanent for the lifetime of the track; even if a track is removed from a media resource's `AudioTrackList` or `VideoTrackList` objects, those aspects do not change.

In addition, `AudioTrack` objects can each be enabled or disabled; this is the audio track's *enabled state*. When an `AudioTrack` is created, its *enabled state* must be set to false (disabled). The resource fetch algorithm can override this.

Similarly, a single `VideoTrack` object per `VideoTrackList` object can be selected, this is the video track's *selection state*. When a `VideoTrack` is created, its *selection state* must be set to false (not selected). The resource fetch algorithm can override this.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

For example, in Ogg files, this would be the Name header field of the track. [OGGSKELETONHEADERS]

The `AudioTrack.kind` and `VideoTrack.kind` attributes must return the category of the track, if it has one, or the empty string otherwise.

The category of a track is the string given in the first column of the table below that is the most appropriate for the track based on the definitions in the table's second and third columns, as determined by the metadata included in the track in the media resource. The cell in the third column of a row says what the category given in the cell in the first column of that row applies to; a category is only appropriate for an audio track if it applies to audio tracks, and a category is only appropriate for video tracks if it applies to video tracks. Categories must be returned for

VideoTrack

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

he `Role` element conveys  
[OGGSKELETONHEADERS] [DASH]

For Ogg files:  
the information  
[WEBMCG]

Category

Category	Definition	Type	Spec
"alternative"	A possible alternative to the main track, e.g. a different take of a song (audio), or a different angle (video).	Audio and video.	Ogg: "audio/alternate" or "video/alternate"; DASH: "alternate" without "main" and "commentary" roles, and, for audio, without the "dub" role (other roles ignored).
"captions"	A version of the main video track with captions burnt in. (For legacy content; new content would use text tracks.)	Video only.	DASH: "caption" and "main" roles together (other roles ignored).
"description"	An audio description of a video track.	Audio only.	Ogg: "audio/audiodesc".
"main"	The primary audio or video track.	Audio and video.	Ogg: "audio/main" or "video/main"; WebM: the "FlagDefault" element is set; DASH: "main" role without "caption", "subtitle", and "dub" roles (other roles ignored).
"main-desc"	The primary audio track, mixed with audio descriptions.	Audio only.	AC3 audio in MPEG-2 TS: bsmod=2 and full_svc=1.
"sign"	A sign-language interpretation of an audio track.	Video only.	Ogg: "video/sign".
"subtitles"	A version of the main video track with subtitles burnt in. (For legacy content; new content would use text tracks.)	Video only.	DASH: "subtitle" and "main" roles together (other roles ignored).
"translation"	A translated version of the main audio track.	Audio only.	Ogg: "audio/dub". DASH: "dub" and "main" roles together (other roles ignored).
"commentary"	Commentary on the primary audio or video track, e.g. a director's commentary.	Audio and video.	DASH: "commentary" role without "main" role (other roles ignored).
"" (empty string)	No explicit kind, or the kind given by the track's metadata is not recognised by the user agent.	Audio and video.	Any other track type, track role, or combination of track roles not described above.

The `AudioTrack.label` and `VideoTrack.label` attributes must return the label of the track, if it has one, or the empty string otherwise.

The `AudioTrack.language` and `VideoTrack.language` attributes must return the BCP 47 language tag of the language of the track, if it has one, or the empty string otherwise. If the user agent is not able to express that language as a BCP 47 language tag (for example because the language information in the media resource's format is a free-form string without a defined interpretation), then the method must return the empty string, as if the track had no language.

The `AudioTrack.enabled` attribute, on getting, must return true if the track is currently enabled, and false otherwise. On setting, it must enable the track if the new value is true, and disable it otherwise. (If the track is no longer in an `AudioTrackList` object, then the track being enabled or disabled has no effect beyond changing the value of the attribute on the `AudioTrack` object.)

Whenever an audio track in an `AudioTrackList` is enabled or disabled, the user agent must queue a task to fire a simple event named `change` at the `AudioTrackList` object.

The `VideoTrackList.selectedIndex` attribute must return the index of the currently selected track, if any. If the `VideoTrackList` object does not currently represent any tracks, or if none of the tracks are selected, it must instead return -1.

The `VideoTrack.selected` attribute, on getting, must return true if the track is currently selected, and false otherwise. On setting, it must select the track if the new value is true, and unselect it otherwise. If the track is in a `VideoTrackList`, then all the other `VideoTrack` objects in that list must be unselected. (If the track is no longer in a `VideoTrackList` object, then the track being selected or unselected has no effect beyond changing the value of the attribute on the `VideoTrack` object.)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

all objects implementing the `AudioTrackList` and `VideoTrackList` interfaces:

Event handler	Event handler event type
<code>onchange</code>	<code>change</code>
<code>onaddtrack</code>	<code>addtrack</code>
<code>onremovetrack</code>	<code>removetrack</code>

The task source for the tasks listed in this section is the DOM manipulation task source.



### 4.8.10.10.2 Too slow? Try reading the multipage copy of

The `audio` tracks declaration fragment identifier is so possible to select specific tracks. The format of the

In this //whatwg.org/html such as

ier syntax is embedded in a track. [MEDIAFRAG]

```
<video src="myvideo#track=Alternative"></video>
```

### 4.8.10.11 Synchronising multiple media elements

#### 4.8.10.11.1 Introduction

Each media element can have a `MediaController`. A `MediaController` is an object that coordinates the playback of multiple media elements, for instance so that a sign-language interpreter track can be overlaid on a video track, with the two being kept in sync.

By default, a media element has no `MediaController`. An implicit `MediaController` can be assigned using the `mediagroup` content attribute. An explicit `MediaController` can be assigned directly using the `controller` IDL attribute.

Media elements with a `MediaController` are said to be *slaved* to their controller. The `MediaController` modifies the playback rate and the playback volume of each of the media elements slaved to it, and ensures that when any of its slaved media elements unexpectedly stall, the others are stopped at the same time.

When a media element is slaved to a `MediaController`, its playback rate is fixed to that of the other tracks in the same `MediaController`, and any looping is disabled.

#### 4.8.10.11.2 Media controllers

```
enum MediaControllerPlaybackState { "waiting", "playing", "ended" };
[Constructor]
interface MediaController : EventTarget {
 readonly attribute unsigned short readyState; // uses HTMLMediaElement.readyState's values

 readonly attribute TimeRanges buffered;
 readonly attribute TimeRanges seekable;
 readonly attribute unrestricted double duration;
 attribute double currentTime;

 readonly attribute boolean paused;
 readonly attribute MediaControllerPlaybackState playbackState;
 readonly attribute TimeRanges played;
 void pause();
 void unpause();
 void play(); // calls play() on all media elements as well

 attribute double defaultPlaybackRate;
 attribute double playbackRate;

 attribute double volume;
 attribute boolean muted;
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
attribute EventHandler oncanplaythrough;
attribute EventHandler onplaying;
attribute EventHandler onended;
attribute EventHandler onwaiting;

attribute EventHandler ondurationchange;
attribute EventHandler ontimeupdate;
attribute EventHandler onplay;
attribute EventHandler onpause;
```



Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Links are given below this box.

**control**

Rel

**`media . controller [= controller]`**Returns the current `MediaController` for the `media` element, if any; returns null otherwise.Can be set, to set an explicit `MediaController`. Doing so removes the `mediagroup` attribute, if any.**`controller . readyState`**Returns the state that the `MediaController` was in the last time it fired events as a result of reporting the controller state. The values of this attribute are the same as for the `readyState` attribute of `media` elements.**`controller . buffered`**Returns a `TimeRanges` object that represents the intersection of the time ranges for which the user agent has all relevant media data for all the slaved media elements.**`controller . seekable`**Returns a `TimeRanges` object that represents the intersection of the time ranges into which the user agent can seek for all the slaved media elements.**`controller . duration`**

Returns the difference between the earliest playable moment and the latest playable moment (not considering whether the data in question is actually buffered or directly seekable, but not including time in the future for infinite streams). Will return zero if there is no media.

**`controller . currentTime [= value]`**Returns the current playback position, in seconds, as a position between zero time and the current `duration`.

Can be set, to seek to the given time.

**`controller . paused`**Returns true if playback is paused; false otherwise. When this attribute is true, any `media` element slaved to this controller will be stopped.**`controller . playbackState`**Returns the state that the `MediaController` was in the last time it fired events as a result of reporting the controller state. The value of this attribute is either "playing", indicating that the media is actively playing, "ended", indicating that the media is not playing because playback has reached the end of all the slaved media elements, or "waiting", indicating that the media is not playing for some other reason (e.g. the `MediaController` is paused).**`controller . pause()`**Sets the `paused` attribute to true.**`controller . unpause()`**Sets the `paused` attribute to false.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Returns a `TimeRanges` object that represents the union of the time ranges in all the slaved media elements that have been played.

#### `controller.defaultPlaybackRate [= value]`

Returns the default rate of playback.

Can be set, to change the default rate of playback.

This default rate has no direct effect on playback, but if the user switches to a fast-forward mode, when they return to the normal playback mode, it is expected that rate of playback (`playbackRate`) will be restored to this default rate.

`control` Too slow? Try reading the multipage copy of  
Re: Cal the spec instead: <https://web.archive.org>

`control` /web/20130103030411/http:

Ref: lou //whatwg.org/html

:the quietest and 1.0 the

Can be set, to change the volume multiplier.

Throws an `IndexSizeError` if the new value is not in the range 0.0 .. 1.0.

#### `controller.muted [= value]`

Returns true if all audio is muted (regardless of other attributes either on the controller or on any media elements slaved to this controller), and false otherwise.

Can be set, to change whether the audio is muted or not.

A media element can have a **current media controller**, which is a `MediaController` object. When a media element is created without a `mediagroup` attribute, it does not have a current media controller. (If it is created *with* such an attribute, then that attribute initializes the current media controller, as defined below.)

The **slaved media elements** of a `MediaController` are the media elements whose current media controller is that `MediaController`. All the slaved media elements of a `MediaController` must use the same clock for their definition of their media timeline's unit time. When the user agent is required to act on each slaved media element in turn, they must be processed in the order that they were last associated with the `MediaController`.

The `controller` attribute on a media element, on getting, must return the element's current media controller, if any, or null otherwise. On setting, the user agent must run the following steps:

1. Let *m* be the media element in question.
2. Let *old controller* be *m*'s current media controller, if it currently has one, and null otherwise.
3. Let *new controller* be null.
4. Let *m* have no current media controller, if it currently has one.
5. Remove the element's `mediagroup` content attribute, if any.
6. If the new value is null, then jump to the *update controllers* step below.
7. Let *m*'s current media controller be the new value.
8. Let *new controller* be *m*'s current media controller.
9. Bring the media element up to speed with its new media controller.
10. *Update controllers*: If *old controller* and *new controller* are the same (whether both null or both the same controller) then abort these steps.
11. If *old controller* is not null and still has one or more slaved media elements, then report the controller state for *old controller*.
12. If *new controller* is not null, then report the controller state for *new controller*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The **seekable** attribute must return a new static normalized `TimeRanges` object that represents the intersection of the ranges of the media resources of the slaved media elements that the user agent is able to seek to, at the time the attribute is evaluated.

The **buffered** attribute must return a new static normalized `TimeRanges` object that represents the intersection of the ranges of the media resources of the slaved media elements that the user agent has buffered, at the time the attribute is evaluated. User agents must accurately determine the ranges available, even for media streams where this can only be determined by tedious inspection.

The **duration** attribute must return the media controller duration.



Every 15 to 7  
agent must  
media contr  
must immed

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

happens least often, the user  
MediaController's  
oller duration, the user agent

The **current** attribute must return the new value.  
[/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

the media controller to the

Every 15 to 1  
agent must queue a task to fire a simple event named `timeupdate` at the MediaController.

happens least often, the user

When a `MediaController` is created it is a **playing media controller**. It can be changed into a **paused media controller** and back either via the user agent's user interface (when the element is exposing a user interface to the user) or by script using the APIs defined in this section (see below).

The **paused** attribute must return true if the `MediaController` object is a paused media controller, and false otherwise.

When the **pause()** method is invoked, if the `MediaController` is a playing media controller then the user agent must change the `MediaController` into a paused media controller, queue a task to fire a simple event named `pause` at the `MediaController`, and then report the controller state of the `MediaController`.

When the **unpause()** method is invoked, if the `MediaController` is a paused media controller, the user agent must change the `MediaController` into a playing media controller, queue a task to fire a simple event named `play` at the `MediaController`, and then report the controller state of the `MediaController`.

When the **play()** method is invoked, the user agent must invoke the `play` method of each slaved media element in turn, and then invoke the `unpause` method of the `MediaController`.

The **playbackState** attribute must return the value to which it was most recently set. When the `MediaController` object is created, the attribute must be set to the value "waiting". The value is updated by the report the controller state algorithm below.

The **played** attribute must return a new static normalized `TimeRanges` object that represents the union of the ranges of points on the media timelines of the media resources of the slaved media elements that the user agent has so far reached through the usual monotonic increase of their current playback positions during normal playback, at the time the attribute is evaluated.

A `MediaController` has a **media controller default playback rate** and a **media controller playback rate**, which must both be set to 1.0 when the `MediaController` object is created.

The **defaultPlaybackRate** attribute, on getting, must return the `MediaController`'s media controller default playback rate, and on setting, must set the `MediaController`'s media controller default playback rate to the new value, then queue a task to fire a simple event named `ratechange` at the `MediaController`.

The **playbackRate** attribute, on getting, must return the `MediaController`'s media controller playback rate, and on setting, must set the `MediaController`'s media controller playback rate to the new value, then queue a task to fire a simple event named `ratechange` at the `MediaController`.

A `MediaController` has a **media controller volume multiplier**, which must be set to 1.0 when the `MediaController` object is created, and a **media controller mute override**, which must initially be false.

The **volume** attribute, on getting, must return the `MediaController`'s media controller volume multiplier, and on setting, if the new value is in the range 0.0 to 1.0 inclusive, must set the `MediaController`'s media controller volume multiplier to the new value and queue a task to fire a simple event named `volumechange` at the `MediaController`. If the new value is outside the range 0.0 to 1.0 inclusive, then, on setting, an `IndexSizeError` exception must be thrown instead.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The media resources of all the slaved media elements of a `MediaController` have a defined temporal relationship which provides relative offsets between the zero time of each such media resource: for media resources with a timeline offset, their relative offsets are the difference between their timeline offset; the zero times of all the media resources without a timeline offset are not offset from each other (i.e. the origins of their timelines are cotemporal); and finally, the zero time of the media resource with the earliest timeline offset (if any) is not offset from the zero times of the media resources without a timeline offset (i.e. the origins of media resources without a timeline offset are further cotemporal with the earliest defined point on the timeline of the media resource with the earliest timeline offset).

The **media resource end position** of a media resource in a media element is defined as follows: the media resource has a finite and known duration; off frame of me

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the slaved media elements duration of that .er timeline, of any of the esource end position of the ntroller timeline.

Each `MediaController` has a **media controller position**. This is the time on the `MediaController`'s timeline at which the user agent is trying to play the slaved media elements. When a `MediaController` is created, its media controller position is initially zero.

When the user agent is to **bring a media element up to speed with its new media controller**, it must seek that media element to the `MediaController`'s media controller position relative to the media element's timeline.

When the user agent is to **seek the media controller** to a particular *new playback position*, it must follow these steps:

1. If the *new playback position* is less than zero, then set it to zero.
2. If the *new playback position* is greater than the media controller duration, then set it to the media controller duration.
3. Set the media controller position to the *new playback position*.
4. Seek each slaved media element to the *new playback position* relative to the media element timeline.

A `MediaController` is a **blocked media controller** if the `MediaController` is a paused media controller, or if any of its slaved media elements are blocked media elements, or if any of its slaved media elements whose autoplaying flag is true still have their `paused` attribute set to true, or if all of its slaved media elements have their `paused` attribute set to true.

A media element is **blocked on its media controller** if the `MediaController` is a blocked media controller, or if its media controller position is either before the media resource's earliest possible position relative to the `MediaController`'s timeline or after the end of the media resource relative to the `MediaController`'s timeline.

When a `MediaController` is not a blocked media controller and it has at least one slaved media element whose `Document` is a fully active `Document`, the `MediaController`'s media controller position must increase monotonically at media controller playback rate units of time on the `MediaController`'s timeline per unit time of the clock used by its slaved media elements.

When the zero point on the timeline of a `MediaController` moves relative to the timelines of the slaved media elements by a time difference  $\Delta T$ , the `MediaController`'s media controller position must be decremented by  $\Delta T$ .

**Note:** In some situations, e.g. when playing back a live stream without buffering anything, the media controller position would increase monotonically as described above at the same rate as the  $\Delta T$  described in the previous paragraph decreases it, with the end result that for all intents and purposes, the media controller position would appear to remain constant (probably with the value 0).

A `MediaController` has a **most recently reported readiness state**, which is a number from 0 to 4 derived from the numbers used for the media element `readyState` attribute, and a **most recently reported playback state**, which is either `playing`, `waiting`, or `ended`.

When a `MediaController` is created, its most recently reported readiness state must be set to 0, and its most recently reported playback state must be set to `waiting`.

When a user agent is required to **report the controller state** for a `MediaController`, the user agent must run the following steps:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

substeps:

1. Let *next state* be the MediaController's most recently reported readiness state.
2. *Loop*: Increment *next state* by one.
3. Queue a task to run the following steps:

1. Set the MediaController's readyState attribute to the value *next state*.



' name corresponding to the

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

Oth  
a ta  
new

' readiness state then queue responding to the value of

Value

Value	
0	emptied
1	loadedmetadata
2	loadeddata
3	canplay
4	canplaythrough

3. Let the MediaController's most recently reported readiness state be *new readiness state*.

4. Initialize *new playback state* by setting it to the state given for the first matching condition from the following list:

↳ If the MediaController has no slaved media elements

Let *new playback state* be *waiting*.

↳ If all of the MediaController's slaved media elements have ended playback and the media controller playback rate is positive or zero

Let *new playback state* be *ended*.

↳ If the MediaController is a blocked media controller

Let *new playback state* be *waiting*.

↳ Otherwise

Let *new playback state* be *playing*.

5. If the MediaController's most recently reported playback state is not equal to *new playback state* and the *new playback state* is *ended*, then queue a task that, if the MediaController object is a playing media controller, and all of the MediaController's slaved media elements have still ended playback, and the media controller playback rate is still positive or zero, changes the MediaController object to a paused media controller and then fires a simple event named *pause* at the MediaController object.

6. If the MediaController's most recently reported playback state is not equal to *new playback state* then queue a task to run the following steps:

1. Set the MediaController's playbackState attribute to the value given in the second column of the row of the following table whose first column contains the *new playback state*.

2. Fire a simple event at the MediaController object, whose name is the value given in the third column of the row of the following table whose first column contains the *new playback state*.

New playback state	New value for playbackState	Event name
playing	"playing"	playing
waiting	"waiting"	waiting
ended	"ended"	ended

7. Let the MediaController's most recently reported playback state be *new playback state*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Event	Impacted
onloadedmetadata	loadedmetadata
onloadeddata	loadeddata
oncanplay	canplay
oncanplaythrough	canplaythrough
onplaying	playing
onended	ended
onwaiting	waiting

ondurationchange

ontimeupdate

onplay

onpause

onratechange

onvolumechange

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The task source

#### 4.8.10.11.3 Assigning a media controller declaratively

The `mediagroup` content attribute on media elements can be used to link multiple media elements together by implicitly creating a `MediaController`. The value is `text`; media elements with the same value are automatically linked by the user agent.

When a media element is created with a `mediagroup` attribute, and when a media element's `mediagroup` attribute is set, changed, or removed, the user agent must run the following steps:

1. Let  $m$  be the media element in question.
2. Let  $old\ controller$  be  $m$ 's current media controller, if it currently has one, and null otherwise.
3. Let  $new\ controller$  be null.
4. Let  $m$  have no current media controller, if it currently has one.
5. If  $m$ 's `mediagroup` attribute is being removed, then jump to the *update controllers* step below.
6. If there is another media element whose `Document` is the same as  $m$ 's `Document` (even if one or both of these elements are not actually *in* the `Document`), and which also has a `mediagroup` attribute, and whose `mediagroup` attribute has the same value as the new value of  $m$ 's `mediagroup` attribute, then let  $controller$  be that media element's current media controller.  
Otherwise, let  $controller$  be a newly created `MediaController`.
7. Let  $m$ 's current media controller be  $controller$ .
8. Let  $new\ controller$  be  $m$ 's current media controller.
9. Bring the media element up to speed with its new media controller.
10. *Update controllers*: If  $old\ controller$  and  $new\ controller$  are the same (whether both null or both the same controller) then abort these steps.
11. If  $old\ controller$  is not null and still has one or more slaved media elements, then report the controller state for  $old\ controller$ .
12. If  $new\ controller$  is not null, then report the controller state for  $new\ controller$ .

The `mediaGroup` IDL attribute on media elements must reflect the `mediagroup` content attribute.

Multiple media elements referencing the same media resource will share a single network request. This can be used to efficiently play two (video) tracks from the same media resource in two different places on the screen. Used with the `mediagroup` attribute, these elements can also be kept synchronised.

In this example, a sign-language interpreter track from a movie file is overlaid on the primary video track of that same video file using two `video` elements, some CSS, and an implicit `MediaController`:

```
<article>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

```
</style>
<div>
 <video src="movie.vid#track=Video&track=English" autoplay controls mediagroup=movie>
</video>
 <video src="movie.vid#track=sign" autoplay mediagroup=movie></video>
</div>
</article>
```

**4.8.10.12****4.8.10.12.1 Too slow? Try reading the multipage copy of****the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>**

A media element sorted as follows.

1. The first track in the media element.
2. Any other tracks in the media element.
3. Any tracks in the media element's parent element.

The text tracks are

in the order defined by the

A text track consists of:

***The kind of text track***

This decides how the track is handled by the user agent. The kind is represented by a string. The possible strings are:

- subtitles
- captions
- descriptions
- chapters
- metadata

The kind of track can change dynamically, in the case of a text track corresponding to a `track` element.***A label***

This is a human-readable string intended to identify the track for the user.

The label of a track can change dynamically, in the case of a text track corresponding to a `track` element.

When a text track label is the empty string, the user agent should automatically generate an appropriate label from the text track's other properties (e.g. the kind of text track and the text track's language) for use in its user interface. This automatically-generated label is not exposed in the API.

***An in-band metadata track dispatch type***

This is a string extracted from the media resource specifically for in-band metadata tracks to enable such tracks to be dispatched to different scripts in the document.

For example, a traditional TV station broadcast streamed on the Web and augmented with Web-specific interactive features could include text tracks with metadata for ad targeting, trivia game data during game shows, player states during sports games, recipe information during food programs, and so forth. As each program starts and ends, new tracks might be added or removed from the stream, and as each one is added, the user agent could bind them to dedicated script modules using the value of this attribute.

Other than for in-band metadata text tracks, the in-band metadata track dispatch type is the empty string. How this value is populated for different media formats is described in steps to expose a media-resource-specific text track.

***A language***

This is a string (a BCP 47 language tag) representing the language of the text track's cues. [BCP47]

The language of a text track can change dynamically, in the case of a text track corresponding to a `track` element.***A readiness state***

One of the following:

***Not loaded***

Indicates that the text track's cues have not been obtained.

***Loading***

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

### **Failed to load**

Indicates that the text track was enabled, but when the user agent attempted to obtain it, this failed in some way (e.g. URL could not be resolved, network error, unknown text track format). Some or all of the cues are likely missing and will not be obtained.

The readiness state of a text track changes dynamically as the track is obtained.

#### **A mode**

One of the following:



*Dis* **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

in the DOM, the user agent did not attempt to obtain the

*Hid* **/web/20130103030411/http://whatwg.org/html**

cues. If no attempt has yet been made to obtain the cues, the user agent is

maintaining a list of which cues are active, and events are being fired accordingly.

#### **Showing**

Indicates that the text track is active. If no attempt has yet been made to obtain the track's cues, the user agent will perform such an attempt momentarily. The user agent is maintaining a list of which cues are active, and events are being fired accordingly. In addition, for text tracks whose kind is subtitles or captions, the cues are being overlaid on the video as appropriate; for text tracks whose kind is descriptions, the user agent is making the cues available to the user in a non-visual fashion; and for text tracks whose kind is chapters, the user agent is making available to the user a mechanism by which the user can navigate to any point in the media resource by selecting a cue.

#### **A list of zero or more cues**

A list of text track cues, along with **rules for updating the text track rendering**. For example, for WebVTT, the rules for updating the display of WebVTT text tracks. [WEBVTT]

The list of cues of a text track can change dynamically, either because the text track has not yet been loaded or is still loading, or due to DOM manipulation.

Each text track has a corresponding `TextTrack` object.

Each media element has a **list of pending text tracks**, which must initially be empty, a **blocked-on-parser** flag, which must initially be false, and a **did-perform-automatic-track-selection** flag, which must also initially be false.

When the user agent is required to **populate the list of pending text tracks** of a media element, the user agent must add to the element's list of pending text tracks each text track in the element's list of text tracks whose text track mode is not disabled and whose text track readiness state is loading.

Whenever a `track` element's parent node changes, the user agent must remove the corresponding text track from any list of pending text tracks that it is in.

Whenever a text track's text track readiness state changes to either loaded or failed to load, the user agent must remove it from any list of pending text tracks that it is in.

When a media element is created by an HTML parser or XML parser, the user agent must set the element's blocked-on-parser flag to true. When a media element is popped off the stack of open elements of an HTML parser or XML parser, the user agent must honor user preferences for automatic text track selection, populate the list of pending text tracks, and set the element's blocked-on-parser flag to false.

The text tracks of a media element are **ready** when both the element's list of pending text tracks is empty and the element's blocked-on-parser flag is false.

A **text track cue** is the unit of time-sensitive data in a text track, corresponding for instance for subtitles and captions to the text that appears at a particular time and disappears at another time.

Each text track cue consists of:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The time, in seconds and fractions of a second, that describes the beginning of the range of the media data to which the cue applies.

### **An end time**

The time, in seconds and fractions of a second, that describes the end of the range of the media data to which the cue applies.

### **A pause-on-exit flag**

A boolean indicating whether playback of the media resource is to pause when the end of the range to which the cue applies is reached.



### **A writing dir**

A writing direction, which defines the direction in which consecutive lines of text are displayed. It is a string consisting of one or more of the following characters:

- w: writing direction is horizontal to the right.
- b: writing direction is horizontal to the left.
- d: writing direction is vertical, oriented downwards.
- c: writing direction is vertical, oriented upwards.
- If there are multiple characters, they are separated by spaces.
- size: indicates the size of the lines, as a percentage of the height of the video.

Other values are invalid.

consecutive lines displayed  
ith consecutive lines  
ioned horizontally, with

ideo, and text position and

size percentages are relative to

the height of the video.

### **A snap-to-lines flag**

A boolean indicating whether the line's position is a line position (positioned to a multiple of the line dimensions of the first line of the cue), or whether it is a percentage of the dimension of the video.

Cues whose text track cue snap-to-lines flag is set will be placed within the title-safe area on user agents that use overscan. Cues with the flag unset will be positioned as requested (modulo overlap avoidance if multiple cues are in the same place).

### **A line position**

Either a number giving the position of the lines of the cue, to be interpreted as defined by the writing direction and snap-to-lines flag of the cue, or the special value **auto**, which means the position is to depend on the other active tracks.

A text track cue has a **text track cue computed line position** whose value is that returned by the following algorithm, which is defined in terms of the other aspects of the cue:

1. If the text track cue line position is numeric, the text track cue snap-to-lines flag of the text track cue is not set, and the text track cue line position is negative or greater than 100, then return 100 and abort these steps.
2. If the text track cue line position is numeric, return the value of the text track cue line position and abort these steps. (Either the text track cue snap-to-lines flag is set, so any value, not just those in the range 0..100, is valid, or the value is in the range 0..100 and is thus valid regardless of the value of that flag.)
3. If the text track cue snap-to-lines flag of the text track cue is not set, return the value 100 and abort these steps. (The text track cue line position is the special value **auto**.)
4. Let *cue* be the text track cue.
5. If *cue* is not in a list of cues of a text track, or if that text track is not in the list of text tracks of a media element, return -1 and abort these steps.
6. Let *track* be the text track whose list of cues the *cue* is in.
7. Let *n* be the number of text tracks whose text track mode is showing and that are in the media element's list of text tracks before *track*.
8. Increment *n* by one.
9. Negate *n*.
10. Return *n*.

### **A text position**

A number giving the position of the text of the cue within each line, to be interpreted as a percentage of the video, as defined by the writing direction.

### **A size**

A number giving the size of the box within which the text of each line of the cue is to be aligned, to be interpreted as a

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

***Start alignment***

The text is aligned towards its start side.

***Middle alignment***

The text is aligned centered between its start and end sides.

***End alignment***

The text is aligned towards its end side.

***Left alignment***

The text is aligned to the left.

***Right\****

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

*The text of*

The

Each text track  
changed thro

riting direction. [BIDI]

ted to a DOM fragment.

on can be dynamically

In addition, each text track cue has two pieces of dynamic information:

***The active flag***

This flag must be initially unset. The flag is used to ensure events are fired appropriately when the cue becomes active or inactive, and to make sure the right cues are rendered.

The user agent must synchronously unset this flag whenever the text track cue is removed from its text track's text track list of cues; whenever the text track itself is removed from its media element's list of text tracks or has its text track mode changed to disabled; and whenever the media element's `readyState` is changed back to `HAVE NOTHING`. When the flag is unset in this way for one or more cues in text tracks that were showing prior to the relevant incident, the user agent must, after having unset the flag for all the affected cues, apply the rules for updating the text track rendering of those text tracks. For example, for text tracks based on WebVTT, the rules for updating the display of WebVTT text tracks. [WEBVTT]

***The display state***

This is used as part of the rendering model, to keep cues in a consistent position. It must initially be empty. Whenever the text track cue active flag is unset, the user agent must empty the text track cue display state.

When a text track cue whose active flag is set has its writing direction, snap-to-lines flag, line position, text position, size, alignment, or text change value, then the user agent must empty the text track cue display state, and then immediately run the text track's rules for updating the display of WebVTT text tracks.

The text track cues of a media element's text tracks are ordered relative to each other in the **text track cue order**, which is determined as follows: first group the cues by their text track, with the groups being sorted in the same order as their text tracks appear in the media element's list of text tracks; then, within each group, cues must be sorted by their start time, earliest first; then, any cues with the same start time must be sorted by their end time, latest first; and finally, any cues with identical end times must be sorted in the order they were last added to their respective text track list of cues, oldest first (so e.g. for cues from a WebVTT file, that would initially be the order in which the cues were listed in the file). [WEBVTT]

**4.8.10.12.2 Sourcing in-band text tracks**

A **media-resource-specific text track** is a text track that corresponds to data found in the media resource.

Rules for processing and rendering such data are defined by the relevant specifications, e.g. the specification of the video format if the media resource is a video.

When a media resource contains data that the user agent recognises and supports as being equivalent to a text track, the user agent runs the **steps to expose a media-resource-specific text track** with the relevant data, as follows.

1. Associate the relevant data with a new text track and its corresponding new `TextTrack` object. The text track is a media-resource-specific text track.
2. Set the new text track's kind, label, and language based on the semantics of the relevant data, as defined by the relevant specification. If there is no label in that data, then the label must be set to the empty string.
3. Associate the text track list of cues with the rules for updating the text track rendering appropriate for the format in question.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[OGGSKELETONHEADERS]

↳ If the media resource is a WebM file

The text track in-band metadata track dispatch type must be set to the value of the `CodecID` element.  
[WEBMCG]

↳ If the media resource is an MPEG-2 file

Let `stream type` be the value of the "stream\_type" field describing the text track's type in the file's program map section, interpreted as an 8-bit unsigned integer. Let `length` be the value of the "ES\_info\_length" field for the track  
[MPEG2]

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

the MPEG-2 specification.  
track in-band metadata track  
or more descriptor bytes

↳ /web/20130103030411/http://whatwg.org/html

of the text track's `trak` box  
the `stsd` box has neither a  
must be set to the empty

string. Otherwise, if the `stsd` box has a `mett` box then the text track in-band metadata track dispatch type must be set to the concatenation of the string "mett", a U+0020 SPACE character, and the value of the first `mime_format` field of the first `mett` box of the `stsd` box, or the empty string if that field is absent in that box. Otherwise, if the `stsd` box has no `mett` box but has a `metx` box then the text track in-band metadata track dispatch type must be set to the concatenation of the string "metx", a U+0020 SPACE character, and the value of the first `namespace` field of the first `metx` box of the `stsd` box, or the empty string if that field is absent in that box.  
[MPEG4]

5. Populate the new text track's list of cues with the cues parsed so far, following the guidelines for exposing cues, and begin updating it dynamically as necessary.
6. Set the new text track's readiness state to loaded.
7. Set the new text track's mode to the mode consistent with the user's preferences and the requirements of the relevant specification for the data.
8. Add the new text track to the media element's list of text tracks.
9. Fire a trusted event with the name `addtrack`, that does not bubble and is not cancelable, and that uses the `TrackEvent` interface, with the `track` attribute initialized to the text track's `TextTrack` object, at the media element's `textTracks` attribute's `TextTrackList` object.

When a media element is to **forget the media element's media-resource-specific text tracks**, the user agent must remove from the media element's list of text tracks all the media-resource-specific text tracks.

#### 4.8.10.12.3 Sourcing out-of-band text tracks

When a `track` element is created, it must be associated with a new text track (with its value set as defined below) and its corresponding new `TextTrack` object.

The text track kind is determined from the state of the element's `kind` attribute according to the following table; for a state given in a cell of the first column, the kind is the string given in the second column:

State	String
Subtitles	subtitles
Captions	captions
Descriptions	descriptions
Chapters	chapters
Metadata	metadata

The text track label is the element's track label.

The text track language is the element's track language, if any, or the empty string otherwise.

As the `kind`, `label`, and `srclang` attributes are set, changed, or removed, the text track must update accordingly, as per the



The text track readiness state is initially not loaded, and the text track mode is initially disabled.

5 Jun 2004 - 22 Aug

The Tex

2012 2013 2014

03

2012 2013 2014

▼ About this capture

The text track list of cues is initially empty. It is dynamically modified when the referenced file is parsed. Associated with the list are the rules for updating the text track rendering appropriate for the format in question; for WebVTT, this is the rules for updating the display of WebVTT text tracks. [WEBVTT]

When a `track` element's parent element changes and the new parent is a media element, then the user agent must add the `track` element's corresponding text track to the media element's list of text tracks, and then queue a task to fire a trusted event with the name `addtrack`, that does not bubble and is not cancelable, and that uses the `TrackEvent` interface, with the `track` attribute initialized to the text track's `TextTrack` object, at the media element's `textTracks` attribute's `TextTrack` object.

When a `track` element has the `name` attribute initialized, try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

agent must remove the task to fire a trusted event with face, with the track extTrackList object.

When a text  
to run the fo [//whatwg.org/html](http://whatwg.org/html)

User agent must queue a task

1. If the element's blocked-on-parser flag is true, abort these steps.
  2. If the element's did-perform-automatic-track-selection flag is true, abort these steps.
  3. Honor user preferences for automatic text track selection for this element.

When the user agent is required to **honor user preferences for automatic text track selection** for a media element, the user agent must run the following steps:

1. Perform automatic text track selection for subtitles and captions.
  2. Perform automatic text track selection for descriptions.
  3. Perform automatic text track selection for chapters.
  4. If there are any text tracks in the media element's list of text tracks whose text track kind is metadata that correspond to track elements with a default attribute set whose text track mode is set to disabled, then set the text track mode of all such tracks to hidden
  5. Set the element's did-perform-automatic-track-selection flag to true.

When the steps above say to **perform automatic text track selection** for one or more text track kinds, it means to run the following steps:

1. Let *candidates* be a list consisting of the text tracks in the media element's list of text tracks whose text track kind is one of the kinds that were passed to the algorithm, if any, in the order given in the list of text tracks.
  2. If *candidates* is empty, then abort these steps.
  3. If any of the text tracks in *candidates* have a text track mode set to showing, abort these steps.
  4. If the user has expressed an interest in having a track from *candidates* enabled based on its text track kind, text track language, and text track label, then set its text track mode to showing.

**Note:** For example, the user could have set a browser preference to the effect of "I want French captions whenever possible", or "If there is a subtitle track with 'Commentary' in the title, enable it", or "If there are audio description tracks available, enable one, ideally in Swiss German, but failing that in Standard Swiss German or Standard German".

Otherwise, if there are any text tracks in `candidates` that correspond to track elements with a `default` attribute set whose text track mode is set to disabled, then set the text track mode of the first such track to showing.

When a text track corresponding to a `track` element experiences any of the following circumstances, the user agent must start the `track` processing model for that text track and its `track` element:

- The `track` element is created.
  - The text track has its text track mode changed

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

1. If another occurrence of this algorithm is already running for this text track and its `track` element, abort these steps, letting that other algorithm take care of this element.

2. If the text track's text track mode is not set to one of hidden or showing, abort these steps.

3. If the text track's `track` element does not have a media element as a parent, abort these steps.

4. Run the remainder of these steps asynchronously, allowing whatever caused these steps to run to continue.

5. *Top:*

`mar` Too slow? Try reading the multipage copy of

6. `⌚ S` the spec instead: <https://web.archive.org>

7. `⌚ L` /web/20130103030411/http:

8. `⌚ I` //whatwg.org/html

The synchronous section are

nt media element's

9. End the synchronous section, continuing the remaining steps asynchronously.

10. If `URL` is not the empty string, perform a potentially CORS-enabled fetch of `URL`, with the `mode` being `CORS mode`, the `origin` being the origin of the `track` element's `Document`, and the `default origin behaviour` set to `fail`.

The resource obtained in this fashion, if any, contains the text track data. If any data is obtained, it is by definition CORS-same-origin (cross-origin resources that are not suitably CORS-enabled do not get this far).

The tasks queued by the fetching algorithm on the networking task source to process the data as it is being fetched must determine the type of the resource. If the type of the resource is not a supported text track format, the load will fail, as described below. Otherwise, the resource's data must be passed to the appropriate parser (e.g. the WebVTT parser) as it is received, with the text track list of cues being used for that parser's output. [WEBVTT]

**Note:** The appropriate parser will synchronously (during these networking task source tasks) and incrementally (as each such task is run with whatever data has been received from the network) update the text track list of cues.

This specification does not currently say whether or how to check the MIME types of text tracks, or whether or how to perform file type sniffing using the actual file data. Implementors differ in their intentions on this matter and it is therefore unclear what the right solution is. In the absence of any requirement here, the HTTP specification's strict requirement to follow the Content-Type header prevails ("Content-Type specifies the media type of the underlying data." ... "If and only if the media type is not given by a Content-Type field, the recipient MAY attempt to guess the media type via inspection of its content and/or the name extension(s) of the URL used to identify the resource.").

If the fetching algorithm fails for any reason (network error, the server returns an error code, a cross-origin check fails, etc), if `URL` is the empty string, or if the type of the resource is not a supported text track format, then run these steps:

1. Queue a task to first change the text track readiness state to failed to load and then fire a simple event named `error` at the `track` element.
2. Wait until the text track readiness state is no longer set to loading.
3. Wait until the track URL is no longer equal to `URL`, at the same time as the text track mode is set to hidden or showing.
4. Jump to the step labeled `top`.

If the fetching algorithm does not fail, then the final task that is queued by the networking task source must run the following steps after it has tried to parse the data:

1. Change the text track readiness state to loaded.
2. If the file was successfully processed, fire a simple event named `load` at the `track` element.

Otherwise, the file was not successfully processed (e.g. the format in question is an XML format and the file contained

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4. Jump back to the step labeled *top*.

If, while the fetching algorithm is active, either:

- the track URL changes so that it is no longer equal to *URL*, while the text track mode is set to hidden or showing; or
- the text track mode changes to hidden or showing, while the track URL is not equal to *URL*

...then the user agent must run the following steps:



nd in particular, not adding

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

Unt /web/20130103030411/http:

Whenever a text track's *text* previously given *URL*, in any,

ously empty the element's source being obtained using the

### 4.8.10.12.4 Guidelines for exposing cues in various formats as text track cues

How a specific format's text track cues are to be interpreted for the purposes of processing by an HTML user agent is defined by that format. In the absence of such a specification, this section provides some constraints within which implementations can attempt to consistently expose such formats.

To support the text track model of HTML, each unit of timed data is converted to a text track cue. Where the mapping of the format's features to the aspects of a text track cue as defined in this specification are not defined, implementations must ensure that the mapping is consistent with the definitions of the aspects of a text track cue as defined above, as well as with the following constraints:

#### The text track cue identifier

Should be set to the empty string if the format has no obvious analogue to a per-cue identifier.

#### The text track cue pause-on-exit flag

Should be set to false.

#### The text track cue writing direction

Should be set to horizontal if the concept of writing direction doesn't really apply (e.g. the cue consists of a bitmap image).

#### The text track cue snap-to-lines flag

Should be set to false unless the format uses a rendering and positioning model for cues that is largely consistent with the WebVTT cue text rendering rules.

#### The text track cue line position

#### The text track cue text position

#### The text track cue size

#### The text track cue alignment

If the format uses a rendering and positioning model for cues that can be largely simulated using the WebVTT cue text rendering rules, then these should be set to the values that would give the same effect for WebVTT cues. Otherwise, they should be set to zero.

### 4.8.10.12.5 Text track API

```
interface TextTrackList : EventTarget {
 readonly attribute unsigned long length;
 getter TextTrack (unsigned long index);

 attribute EventHandler onaddtrack;
 attribute EventHandler onremovetrack;
};
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**media.textTracks[n]**

Returns the `TextTrack` object representing the *n*th text track in the `media` element's list of text tracks.

**track.track**

Returns the `TextTrack` object representing the `track` element's text track.

A `TextTrac`The `textTr`  
tracks in the  
time the attriThe `length`  
`TextTrack`

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

`TextTrack` objects of the text  
list must be returned each

nted by the

The supported property indices of a `TextTrackList` object at any instant are the numbers from zero to the number of text tracks in the list represented by the `TextTrackList` object minus one, if any. If there are no text tracks in the list, there are no supported property indices.

To determine the value of an indexed property of a `TextTrackList` object for a given index *index*, the user agent must return the *index*th text track in the list represented by the `TextTrackList` object.

```
enum TextTrackMode { "disabled", "hidden", "showing" };
interface TextTrack : EventTarget {
 readonly attribute DOMString kind;
 readonly attribute DOMString label;
 readonly attribute DOMString language;
 readonly attribute DOMString inBandMetadataTrackDispatchType;

 attribute TextTrackMode mode;

 readonly attribute TextTrackCueList? cues;
 readonly attribute TextTrackCueList? activeCues;

 void addCue(TextTrackCue cue);
 void removeCue(TextTrackCue cue);

 attribute EventHandler oncuechange;
};
```

This box is non-normative. Implementation requirements are given below this box.

**textTrack = media.addTextTrack(*kind* [, *label* [, *language*]])**

Creates and returns a new `TextTrack` object, which is also added to the `media` element's list of text tracks.

**textTrack.kind**

Returns the text track kind string.

**textTrack.label**

Returns the text track label, if there is one, or the empty string otherwise (indicating that a custom label probably needs to be generated from the other attributes of the object if the object is exposed to the user).

**textTrack.language**

Returns the text track language string.

**textTrack.inBandMetadataTrackDispatchType**

Returns the text track in-band metadata track dispatch type string.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The text track disabled mode.

**"hidden"**

The text track hidden mode.

**"showing"**

The text track showing mode.

Can be set, to change the mode.

***textTrack***

Ref Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

***textTrack***

Ref pos /web/20130103030411/http://whatwg.org/html

before the current playback***textTrack***Adds the given cue to *textTrack*'s text track list of cues.***textTrack*.removeCue(*cue*)**Removes the given cue from *textTrack*'s text track list of cues.

The `addTextTrack(kind, label, language)` method of media elements, when invoked, must run the following steps:

1. If *kind* is not one of the following strings, then throw a `SyntaxError` exception and abort these steps:
  - o subtitles
  - o captions
  - o descriptions
  - o chapters
  - o metadata
2. If the *label* argument was omitted, let *label* be the empty string.
3. If the *language* argument was omitted, let *language* be the empty string.
4. Create a new `TextTrack` object.
5. Create a new text track corresponding to the new object, and set its text track kind to *kind*, its text track label to *label*, its text track language to *language*, its text track readiness state to the text track loaded state, its text track mode to the text track hidden mode, and its text track list of cues to an empty list. Associate the text track list of cues with the rules for updating the display of WebVTT text tracks as its rules for updating the text track rendering. [WEBVTT]
6. Add the new text track to the media element's list of text tracks.
7. Queue a task to fire a trusted event with the name `addtrack`, that does not bubble and is not cancelable, and that uses the `TrackEvent` interface, with the `track` attribute initialized to the new text track's `TextTrack` object, at the media element's `textTracks` attribute's `TextTrackList` object.
8. Return the new `TextTrack` object.

The `kind` attribute must return the text track kind of the text track that the `TextTrack` object represents.

The `label` attribute must return the text track label of the text track that the `TextTrack` object represents.

The `language` attribute must return the text track language of the text track that the `TextTrack` object represents.

The `inBandMetadataTrackDispatchType` attribute must return the text track in-band metadata track dispatch type of the text track that the `TextTrack` object represents.

The `mode` attribute, on getting, must return the string corresponding to the text track mode of the text track that the `TextTrack` object represents, as defined by the following list:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**"showing"**

The text track showing mode.

On setting, if the new value isn't equal to what the attribute would currently return, the new value must be processed as follows:

↳ If the new value is "disabled"

Set the text track mode of the text track that the `TextTrack` object represents to the text track disabled mode.

↳ If the new

**Too slow? Try reading the multipage copy of**

k hidden mode.

↳ If the new

**the spec instead: https://web.archive.org**

k showing mode.

If the text track mode of the text track that the `TextTrack` object represents is not the text track disabled mode, then the `activeCues` attribute must return a live `TextTrackCueList` object that represents the subset of the text track list of cues of the text track that the `TextTrack` object represents whose active flag was set when the script started, in text track cue order. Otherwise, it must return null.

If the text track mode of the text track that the `TextTrack` object represents is the text track disabled mode, then the `activeCues` attribute must return null.

If the text track mode of the text track that the `TextTrack` object represents is the text track hidden mode, then the `activeCues` attribute must return a live `TextTrackCueList` object that represents the subset of the text track list of cues of the text track that the `TextTrack` object represents whose active flag was set when the script started, in text track cue order. Otherwise, it must return null.

When an object is returned, the same object must be returned each time.

The **earliest possible position when the script started** is whatever the earliest possible position was the last time the event loop reached step 1.

If the text track mode of the text track that the `TextTrack` object represents is not the text track disabled mode, then the `activeCues` attribute must return a live `TextTrackCueList` object that represents the subset of the text track list of cues of the text track that the `TextTrack` object represents whose active flag was set when the script started, in text track cue order. Otherwise, it must return null. When an object is returned, the same object must be returned each time.

A text track cue's **active flag was set when the script started** if its text track cue active flag was set the last time the event loop reached step 1.

The `addCue(cue)` method of `TextTrack` objects, when invoked, must run the following steps:

1. If the given `cue` is in a text track list of cues, then remove `cue` from that text track list of cues.
2. Add `cue` to the method's `TextTrack` object's text track's text track list of cues.

The `removeCue(cue)` method of `TextTrack` objects, when invoked, must run the following steps:

1. If the given `cue` is not currently listed in the method's `TextTrack` object's text track's text track list of cues, then throw a `NotFoundError` exception.
2. Remove `cue` from the method's `TextTrack` object's text track's text track list of cues.

In this example, an `audio` element is used to play a specific sound-effect from a sound file containing many sound effects. A cue is used to pause the audio, so that it ends exactly at the end of the clip, even if the browser is busy running some script. If the page had relied on script to pause the audio, then the start of the next clip might be heard if the browser was not able to run the script at the exact time specified.

```

var sfx = new Audio('sfx.wav');
var sounds = sfx.addTextTrack('metadata');

// add sounds we care about
function addFX(start, end, name) {
 var cue = new TextTrackCue(start, end, '');
 cue.id = name;
 cue.pauseOnExit = true;
 sounds.addCue(cue);
}
addFX(12.783, 13.612, 'dog bark');
addFX(13.612, 15.091, 'kitten mew')

function playSound(id) {
 sfx.currentTime = sounds.getCueById(id).startTime;
}

```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

 playSound('dog bark');
 }
 // meow when the user tries to leave
 window.onbeforeunload = function () {
 playSound('kitten mew');
 return 'Are you sure you want to leave this awesome page?';
 }
}

```

inten:  
re ge  
Te };  
Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/cuelist>.

Returns the number of cues in the list.

#### `cuelist[index]`

Returns the text track cue with index *index* in the list. The cues are sorted in text track cue order.

#### `cuelist.getcueById(id)`

Returns the first text track cue (in text track cue order) with text track cue identifier *id*.

Returns null if none of the cues have the given identifier or if the argument is the empty string.

A `TextTrackCueList` object represents a dynamically updating list of text track cues in a given order.

The `length` attribute must return the number of cues in the list represented by the `TextTrackCueList` object.

The supported property indices of a `TextTrackCueList` object at any instant are the numbers from zero to the number of cues in the list represented by the `TextTrackCueList` object minus one, if any. If there are no cues in the list, there are no supported property indices.

To determine the value of an indexed property for a given index *index*, the user agent must return the *index*th text track cue in the list represented by the `TextTrackCueList` object.

The `getcueById(id)` method, when called with an argument other than the empty string, must return the first text track cue in the list represented by the `TextTrackCueList` object whose text track cue identifier is *id*, if any, or null otherwise. If the argument is the empty string, then the method must return null.

```

enum AutoKeyword { "auto" };
[Constructor(double startTime, double endTime, DOMString text)]
interface TextTrackCue : EventTarget {
 readonly attribute TextTrack? track;

 attribute DOMString id;
 attribute double startTime;
 attribute double endTime;
 attribute boolean pauseOnExit;
 attribute DOMString vertical;
 attribute boolean snapToLines;
 attribute (long or AutoKeyword) line;
 attribute long position;
 attribute long size;
 attribute DOMString align;
 attribute DOMString text;
 DocumentFragment getCueAsHTML();
}

```

Go

NOV

JAN

FEB

◀

03

▶

2012

2013

2014

▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020



This box is non-normative. Implementation requirements are given below this box.

**`cue = new TextTrackCue( startTime, endTime, text )`**

Returns a new `TextTrackCue` object, for use with the `addCue()` method.

The `startTime` argument sets the text track cue start time.

The `endTime` argument sets the text track cue end time.

The `text` argument sets the text track cue text.



`cue . tr`: Too slow? Try reading the multipage copy of

`cue . id`: the spec instead: <https://web.archive.org>

`cue . id`: [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

`cue . sta`:

Returns the text track cue start time, in seconds.

Can be set.

**`cue . endTime [ = value ]`**

Returns the text track cue end time, in seconds.

Can be set.

**`cue . pauseOnExit [ = value ]`**

Returns true if the text track cue pause-on-exit flag is set, false otherwise.

Can be set.

**`cue . vertical [ = value ]`**

Returns a string representing the text track cue writing direction, as follows:

**↳ If it is horizontal**

The empty string.

**↳ If it is vertical growing left**

The string "rl".

**↳ If it is vertical growing right**

The string "lr".

Can be set.

**`cue . snapToLines [ = value ]`**

Returns true if the text track cue snap-to-lines flag is set, false otherwise.

Can be set.

**`cue . line [ = value ]`**

Returns the text track cue line position. In the case of the value being auto, the string "auto" is returned.

Can be set.

**`cue . position [ = value ]`**

Returns the text track cue text position.

Can be set.

**`cue . size [ = value ]`**

Returns the text track cue size.

Can be set.

Go NOV JAN FEB  
 ← 03 →  
 2012 2013 2014 ▾ About this capture

[3,445 captures](#)  
 5 Jun 2004 - 22 Aug 2020

↳ If it is middle alignment

The string "middle".

↳ If it is end alignment

The string "end".

↳ If it is left alignment



↳ Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

Call /web/20130103030411/http:  
`cue.te` //whatwg.org/html  
`Ret.`

Can be set.

`fragment = cue.getcueAsHTML()`

Returns the text track cue text as a DocumentFragment of HTML elements and other DOM nodes.

The `TextTrackCue(startTime, endTime, text)` constructor, when invoked, must run the following steps:

1. Create a new text track cue. Let `cue` be that text track cue.
2. Let `cue`'s text track cue start time be the value of the `startTime` argument, interpreted as a time in seconds.
3. Let `cue`'s text track cue end time be the value of the `endTime` argument, interpreted as a time in seconds.
4. Let `cue`'s text track cue text be the value of the `text` argument, and let the rules for its interpretation be the WebVTT cue text parsing rules, the WebVTT cue text rendering rules, and the WebVTT cue text DOM construction rules. [WEBVTT]
5. Let `cue`'s text track cue identifier be the empty string.
6. Let `cue`'s text track cue pause-on-exit flag be false.
7. Let `cue`'s text track cue writing direction be horizontal.
8. Let `cue`'s text track cue snap-to-lines flag be true.
9. Let `cue`'s text track cue line position be auto.
10. Let `cue`'s text track cue text position be 50.
11. Let `cue`'s text track cue size be 100.
12. Let `cue`'s text track cue alignment be middle alignment.
13. Return the `TextTrackCue` object representing `cue`.

The `track` attribute, on getting, must return the `TextTrack` object of the text track in whose list of cues the text track cue that the `TextTrackCue` object represents finds itself, if any; or null otherwise.

The `id` attribute, on getting, must return the text track cue identifier of the text track cue that the `TextTrackCue` object represents. On setting, the text track cue identifier must be set to the new value.

The `startTime` attribute, on getting, must return the text track cue start time of the text track cue that the `TextTrackCue` object represents, in seconds. On setting, the text track cue start time must be set to the new value, interpreted in seconds.

The `endTime` attribute, on getting, must return the text track cue end time of the text track cue that the `TextTrackCue` object represents, in seconds. On setting, the text track cue end time must be set to the new value, interpreted in seconds.

The `pauseOnExit` attribute, on getting, must return true if the text track cue pause-on-exit flag of the text track cue that the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Text track cue writing direction	direction value
Horizontal	"" (the empty string)
Vertical growing left	"rl"
Vertical growing right	"lr"

On setting, the text track cue writing direction must be set to the value given in the first cell of the row in the table above whose second cell is a case-sensitive match for the new value, if any. If none of the values match, then the user agent must instead throw a `SyntaxError`.

The `snapshotTC` TextTrack value is true  
 The `line` attribute represents the new value  
 Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

track cue that the direction value must be set if the new

TextTrackCue object's line position must be set to

The `position` attribute, on getting, must return the text track cue text position of the text track cue that the `TextTrackCue` object represents. On setting, if the new value is negative or greater than 100, then an `IndexSizeError` exception must be thrown. Otherwise, the text track cue text position must be set to the new value.

The `size` attribute, on getting, must return the text track cue size of the text track cue that the `TextTrackCue` object represents. On setting, if the new value is negative or greater than 100, then an `IndexSizeError` exception must be thrown. Otherwise, the text track cue size must be set to the new value.

The `align` attribute, on getting, must return the string from the second cell of the row in the table below whose first cell is the text track cue alignment of the text track cue that the `TextTrackCue` object represents:

Text track cue alignment	align value
Start alignment	"start"
Middle alignment	"middle"
End alignment	"end"
Left alignment	"left"
Right alignment	"right"

On setting, the text track cue alignment must be set to the value given in the first cell of the row in the table above whose second cell is a case-sensitive match for the new value, if any. If none of the values match, then the user agent must instead throw a `SyntaxError` exception.

The `text` attribute, on getting, must return the raw text track cue text of the text track cue that the `TextTrackCue` object represents. On setting, the text track cue text must be set to the new value.

The `getcueAsHTML()` method must convert the text track cue text to a `DocumentFragment` for the script's document of the entry script, using the appropriate rules for doing so. For example, for WebVTT, those rules are the WebVTT cue text parsing rules and the WebVTT cue text DOM construction rules. [WEBVTT]

#### 4.8.10.12.6 Text tracks describing chapters

Chapters are segments of a media resource with a given title. Chapters can be nested, in the same way that sections in a document outline can have subsections.

Each text track cue in a text track being used for describing chapters has three key features: the text track cue start time, giving the start time of the chapter, the text track cue end time, giving the end time of the chapter, and the text track cue text giving the chapter title.

The **rules for constructing the chapter tree from a text track** are as follows. They produce a potentially nested list of chapters, each of which have a start time, end time, title, and a list of nested chapters. This algorithm discards cues that do not correctly nest within each other, or that are out of order.

1. Let `list` be a copy of the list of cues of the text track being processed.
2. Remove from `list` any text track cue whose text track cue end time is before its text track cue start time.

Go

NOV JAN FEB

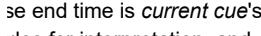
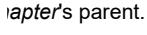
03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

5. *Loop*: If *list* is empty, jump to the step labeled *end*.
6. Let *current cue* be the first cue in *list*, and then remove it from *list*.
7. If *current cue*'s text track cue start time is less than the start time of *current chapter*, then return to the step labeled *loop*.
8. While *current cue*'s text track cue start time is greater than or equal to *current chapter*'s end time, let *current chapter* be *current chapter*'s parent chapter.
9. If *curr*  to the step labeled *loop*.
10. Create **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org** 
11. Append [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html) 
12. Let [//whatwg.org/html](https://whatwg.org/html) 
13. Return to the step labeled *loop*.
14. *End*: Return *output*.

The following snippet of a WebVTT file shows how nested chapters can be marked up. The file describes three 50-minute chapters, "Astrophysics", "Computational Physics", and "General Relativity". The first has three subchapters, the second has four, and the third has two. [WEBVTT]

```
WEBVTT

00:00:00.000 --> 00:50:00.000
Astrophysics

00:00:00.000 --> 00:10:00.000
Introduction to Astrophysics

00:10:00.000 --> 00:45:00.000
The Solar System

00:00:00.000 --> 00:10:00.000
Coursework Description

00:50:00.000 --> 01:40:00.000
Computational Physics

00:50:00.000 --> 00:55:00.000
Introduction to Programming

00:55:00.000 --> 01:30:00.000
Data Structures

01:30:00.000 --> 01:35:00.000
Answers to Last Exam

01:35:00.000 --> 01:40:00.000
Coursework Description

01:40:00.000 --> 02:30:00.000
General Relativity

01:40:00.000 --> 02:00:00.000
Tensor Algebra

02:00:00.000 --> 02:30:00.000
The General Relativistic Field Equations
```

Go

NOV JAN FEB

03

2012 2013 2014



## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Event handler	Event handler event type
<code>onaddtrack</code>	<code>addtrack</code>
<code>onremovetrack</code>	<code>removetrack</code>

The following are the event handlers that (and their corresponding event handler event types) must be supported, as IDL attributes, by all objects implementing the `TextTrack` interface:

Event handler	Event handler event type
<code>oncuechange</code>	<code>cuechange</code>

The following are the event handlers that (and their corresponding event handler event types) must be supported, as IDL attributes, by all objects implementing the `TextTrack` interface:

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

### 4.8.10.13 User interface

The `controls` attribute is a boolean attribute. If present, it indicates that the author has not provided a scripted controller and would like the user agent to provide its own set of controls.

If the attribute is present, or if scripting is disabled for the media element, then the user agent should expose a user interface to the user. This user interface should include features to begin playback, pause playback, seek to an arbitrary position in the content (if the content supports arbitrary seeking), change the volume, change the display of closed captions or embedded sign-language tracks, select different audio tracks or turn on audio descriptions, and show the media content in manners more suitable to the user (e.g. full-screen video or in an independent resizable window). Other controls may also be made available.

If the media element has a current media controller, then the user agent should expose audio tracks from all the slaved media elements (although avoiding duplicates if the same media resource is being used several times). If a media resource's audio track exposed in this way has no known name, and it is the only audio track for a particular media element, the user agent should use the element's `title` attribute, if any, as the name (or as part of the name) of that track.

Even when the attribute is absent, however, user agents may provide controls to affect playback of the media resource (e.g. play, pause, seeking, and volume controls), but such features should not interfere with the page's normal rendering. For example, such features could be exposed in the media element's context menu. The user agent may implement this simply by exposing a user interface to the user as described above (as if the `controls` attribute was present).

Where possible (specifically, for starting, stopping, pausing, and unpauseing playback, for seeking, for changing the rate of playback, for fast-forwarding or rewinding, for listing, enabling, and disabling text tracks, and for muting or changing the volume of the audio), user interface features exposed by the user agent must be implemented in terms of the DOM API described above, so that, e.g., all the same events fire.

When a media element has a current media controller, the user agent's user interface for pausing and unpauseing playback, for seeking, for changing the rate of playback, for fast-forwarding or rewinding, and for muting or changing the volume of audio of the entire group must be implemented in terms of the `MediaController` API exposed on that current media controller. When a media element has a current media controller, and all the slaved media elements of that `MediaController` are paused, the user agent should also unpause all the slaved media elements when the user invokes a user agent interface control for beginning playback.

The "play" function in the user agent's interface must set the `playbackRate` attribute to the value of the `defaultPlaybackRate` attribute before invoking the `play()` method. When a media element has a current media controller, the attributes and method with those names on that `MediaController` object must be used. Otherwise, the attributes and method with those names on the media element itself must be used.

Features such as fast-forward or rewind must be implemented by only changing the `playbackRate` attribute (and not the `defaultPlaybackRate` attribute). Again, when a media element has a current media controller, the attributes with those names on that `MediaController` object must be used; otherwise, the attributes with those names on the media element itself must be used.

When a media element has a current media controller, seeking must be implemented in terms of the `currentTime` attribute on that `MediaController` object. Otherwise, the user agent must directly seek to the requested position in the media element's media timeline. For media resources where seeking to an arbitrary position would be slow, user agents are encouraged to use the `approximate-for-speed` flag when seeking in response to the user manipulating an approximate position interface such as a seek bar.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

and whose text track kind is `chapters` should be used. Such tracks must be interpreted according to the rules for constructing the chapter tree from a text track. When seeking in response to a user manipulating a chapter selection interface, user agents should not use the `approximate-for-speed` flag.

The `controls` IDL attribute must reflect the content attribute of the same name.

This box is non-normative. Implementation requirements are given below this box.

`media . volume [= value]`



last and 1.0 the loudest.

`Ret` Too slow? Try reading the multipage copy of  
`Cal` the spec instead: <https://web.archive.org>

`media . /web/20130103030411/http:`  
`Ret` `//whatwg.org/html` ↗ being honored.  
`Cal`

The `volume` attribute must return the playback volume of any audio portions of the media element, in the range 0.0 (silent) to 1.0 (loudest). Initially, the volume should be 1.0, but user agents may remember the last set value across sessions, on a per-site basis or otherwise, so the volume may start at other values. On setting, if the new value is in the range 0.0 to 1.0 inclusive, the playback volume of any audio portions of the media element must be set to the new value. If the new value is outside the range 0.0 to 1.0 inclusive, then, on setting, an `IndexSizeError` exception must be thrown instead.

The `muted` attribute must return true if the audio output is muted and false otherwise. Initially, the audio output should not be muted (false), but user agents may remember the last set value across sessions, on a per-site basis or otherwise, so the muted state may start as muted (true). On setting, if the new value is true then the audio output should be muted and if the new value is false it should be unmuted.

Whenever either of the values that would be returned by the `volume` and `muted` attributes change, the user agent must queue a task to fire a simple event named `volumechange` at the media element.

An element's **effective media volume** is determined as follows:

- If the user has indicated that the user agent is to override the volume of the element, then the element's effective media volume is the volume desired by the user. Abort these steps.
- If the element's audio output is muted, the element's effective media volume is zero. Abort these steps.
- If the element has a current media controller and that `MediaController` object's media controller mute override is true, the element's effective media volume is zero. Abort these steps.
- Let `volume` be the playback volume of the audio portions of the media element, in range 0.0 (silent) to 1.0 (loudest).
- If the element has a current media controller, multiply `volume` by that `MediaController` object's media controller volume multiplier.
- The element's effective media volume is `volume`, interpreted relative to the range 0.0 to 1.0, with 0.0 being silent, and 1.0 being the loudest setting, values in between increasing in loudness. The range need not be linear. The loudest setting may be lower than the system's loudest possible setting, for example the user could have set a maximum volume.

The `muted` attribute on media elements is a boolean attribute that controls the default state of the audio output of the media resource, potentially overriding user preferences.

When a media element is created, if it has a `muted` attribute specified, the user agent must mute the media element's audio output, overriding any user preference.

The `defaultMuted` IDL attribute must reflect the `muted` content attribute.

**Note: This attribute has no dynamic effect (it only controls the default state of the element).**

This video (an advertisement) autoplays, but to avoid annoying users, it does so without sound, and allows the user to turn the sound on.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Objects implementing the `TimeRanges` interface represent a list of ranges (periods) of time.

```
interface TimeRanges {
 readonly attribute unsigned long length;
 double start(unsigned long index);
 double end(unsigned long index);
};
```

Comments are given below this box.

`media`. Too slow? Try reading the multipage copy of  
<sub>Ref</sub> the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`time = i`  
<sub>Ref</sub> /  
<sub>Ref</sub> `Thr` <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`time = media . . . . .`

Returns the time for the end of the range with the given index.

Throws an `IndexSizeError` if the index is out of range.

The `length` IDL attribute must return the number of ranges represented by the object.

The `start(index)` method must return the position of the start of the `index`th range represented by the object, in seconds measured from the start of the timeline that the object covers.

The `end(index)` method must return the position of the end of the `index`th range represented by the object, in seconds measured from the start of the timeline that the object covers.

These methods must throw `IndexSizeError` exceptions if called with an `index` argument greater than or equal to the number of ranges represented by the object.

When a `TimeRanges` object is said to be a **normalized `TimeRanges` object**, the ranges it represents must obey the following criteria:

- The start of a range must be greater than the end of all earlier ranges.
- The start of a range must be less than the end of that same range.

In other words, the ranges in such an object are ordered, don't overlap, aren't empty, and don't touch (adjacent ranges are folded into one bigger range).

Ranges in a `TimeRanges` object must be inclusive.

Thus, the end of a range would be equal to the start of a following adjacent (touching but not overlapping) range. Similarly, a range covering a whole timeline anchored at zero would have a start equal to zero and an end equal to the duration of the timeline.

The timelines used by the objects returned by the `buffered`, `seekable` and `played` IDL attributes of media elements must be that element's media timeline.

#### 4.8.10.15 Event definitions

```
[Constructor(DOMString type, optional TrackEventInit eventInitDict)]
interface TrackEvent : Event {
 readonly attribute object? track;
};

dictionary TrackEventInit : EventInit {
 object? track;
};
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `track` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the context information for the event.

#### 4.8.10.16 Event summary

*This section is non-normative.*

The following



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ons

NG

NG

ERR\_ABORTED, networkState

<code>loadstart</code>	Event	An error occurs while fetching the media data.	readyState equals either <code>NETWORK_EMPTY</code> or <code>NETWORK_IDLE</code> , depending on when the download was aborted.
<code>error</code>	Event	An error occurs while fetching the media data.	error is an object with the code <code>MEDIA_ERR_NETWORK</code> or higher. networkState equals either <code>NETWORK_EMPTY</code> or <code>NETWORK_IDLE</code> , depending on when the download was aborted.
<code>emptied</code>	Event	A media element whose <code>networkState</code> was previously not in the <code>NETWORK_EMPTY</code> state has just switched to that state (either because of a fatal error during load that's about to be reported, or because the <code>load()</code> method was invoked while the resource selection algorithm was already running).	networkState is <code>NETWORK_EMPTY</code> ; all the IDL attributes are in their initial states.
<code>stalled</code>	Event	The user agent is trying to fetch media data, but data is unexpectedly not forthcoming.	networkState is <code>NETWORK_LOADING</code> .
<code>loadedmetadata</code>	Event	The user agent has just determined the duration and dimensions of the media resource and the text tracks are ready.	readyState is newly equal to <code>HAVE_METADATA</code> or greater for the first time.
<code>loadeddata</code>	Event	The user agent can render the media data at the current playback position for the first time.	readyState newly increased to <code>HAVE_CURRENT_DATA</code> or greater for the first time.
<code>canplay</code>	Event	The user agent can resume playback of the media data, but estimates that if playback were to be started now, the media resource could not be rendered at the current playback rate up to its end without having to stop for further buffering of content.	readyState newly increased to <code>HAVE_FUTURE_DATA</code> or greater.
<code>canplaythrough</code>	Event	The user agent estimates that if playback were to be started now, the media resource could be rendered at the current playback rate all the way to its end without having to stop for further buffering.	readyState is newly equal to <code>HAVE_ENOUGH_DATA</code> .
<code>playing</code>	Event	Playback is ready to start after having been paused or delayed due to lack of media data.	readyState is newly equal to or greater than <code>HAVE_FUTURE_DATA</code> and <code>paused</code> is <code>false</code> , or <code>paused</code> is newly <code>false</code> and <code>readyState</code> is equal to or greater than <code>HAVE_FUTURE_DATA</code> . Even if this event fires, the element might still not be potentially playing, e.g. if the element is blocked on its media controller (e.g. because the current media controller is paused, or another slaved media element is stalled somehow, or because the media resource has no data corresponding to the media controller position), or the element is paused for user interaction or paused for in-band content.
<code>waiting</code>	Event	Playback has stopped because the next frame is not available, but the user agent expects that frame to become available in due course.	readyState is equal to or less than <code>HAVE_CURRENT_DATA</code> , and <code>paused</code> is <code>false</code> . Either <code>seeking</code> is <code>true</code> , or the current playback position is not contained in any of the ranges in <code>buffered</code> . It is possible for playback to stop for other reasons without <code>paused</code> being <code>false</code> , but those reasons do not fire this event (and when those situations resolve, a separate <code>playing</code> event is not fired either); e.g. the element is newly blocked on its media controller, or playback ended, or playback stopped due to errors, or the element has paused for user interaction or paused for in-band content.
<code>seeking</code>	Event	The <code>seeking</code> IDL attribute changed to true.	
<code>seeked</code>	Event	The <code>seeking</code> IDL attribute changed to false.	
<code>ended</code>	Event	Playback has stopped because the end of the media resource was reached.	currentTime equals the end of the media resource; <code>ended</code> is <code>true</code> .
<code>durationchange</code>	Event	The <code>duration</code> attribute has just been updated.	

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

		play() method has returned, or when the autoplay attribute has caused playback to begin.	
<b>pause</b>	Event	The element has been paused. Fired after the pause() method has returned.	paused is newly true.
<b>ratechange</b>	Event	Either the defaultPlaybackRate or the playbackRate attribute has just been updated.	
<b>volumechange</b>	Event	Either the volume attribute or the muted attribute has changed. Fired after the relevant attribute's setter has returned.	

The followin

Too slow? Try reading the multipage copy of  
**Event name** the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/canplay>

<b>emptied</b>	Event	There are no longer any slaved media
<b>loadedmetadata</b>	Event	
<b>loadeddata</b>	Event	
<b>canplay</b>	Event	
<b>canplaythrough</b>	Event	All the slaved media elements newly have readyState set to HAVE_ENOUGH_DATA or greater.
<b>playing</b>	Event	The MediaController is no longer a blocked media controller.
<b>ended</b>	Event	The MediaController has reached the end of all the slaved media elements.
<b>waiting</b>	Event	The MediaController is now a blocked media controller.
<b>ended</b>	Event	All the slaved media elements have newly ended playback.
<b>durationchange</b>	Event	The duration attribute has just been updated.
<b>timeupdate</b>	Event	The media controller position changed.
<b>play</b>	Event	The paused attribute is newly false.
<b>pause</b>	Event	The paused attribute is newly true.
<b>ratechange</b>	Event	Either the defaultPlaybackRate attribute or the playbackRate attribute has just been updated.
<b>volumechange</b>	Event	Either the volume attribute or the muted attribute has just been updated.

#### 4.8.10.17 Security and privacy considerations

The main security and privacy implications of the video and audio elements come from the ability to embed media cross-origin. There are two directions that threats can flow: from hostile content to a victim page, and from a hostile page to victim content.

If a victim page embeds hostile content, the threat is that the content might contain scripted code that attempts to interact with the Document that embeds the content. To avoid this, user agents must ensure that there is no access from the content to the embedding page. In the case of media content that uses DOM concepts, the embedded content must be treated as if it was in its own unrelated top-level browsing context.

For instance, if an SVG animation was embedded in a video element, the user agent would not give it access to the DOM of the outer page. From the perspective of scripts in the SVG resource, the SVG file would appear to be in a lone top-level browsing context with no parent.

If a hostile page embeds victim content, the threat is that the embedding page could obtain information from the content that it would not otherwise have access to. The API does expose some information: the existence of the media, its type, its duration, its size, and the performance characteristics of its host. Such information is already potentially problematic, but in practice the same information can more or less be obtained using the img element, and so it has been deemed acceptable.

However, significantly more sensitive information could be obtained if the user agent further exposes metadata within the content such as subtitles or chapter titles. Such information is therefore only exposed if the video resource passes a CORS resource sharing check. The crossorigin attribute allows authors to control how this check is performed. [CORS]

Without this restriction, an attacker could trick a user running within a corporate network into visiting a site that attempts to load a video from a previously leaked location on the corporation's intranet. If such a video included confidential plans for a new product, then being able to read the subtitles would present a serious confidentiality breach.

#### 4.8.10.18 Best practices for authors using media elements

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

the element and allowing it to be garbage collected, or, even better, by removing the element's `src` attribute and any `source` element descendants, and invoking the element's `load()` method.

Similarly, when the playback rate is not exactly 1.0, hardware, software, or format limitations can cause video frames to be dropped and audio to be choppy or muted.

#### 4.8.10.19 Best practices for implementors of media elements

*This section is non-normative*



How accurate is the spec? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Implementation issue.

For example, depends on streams, a user agent gets rebuffered when it has to seek through the file, which gets rebuffered when it has to seek through the file again. This is a quality-of-implementation issue.

es that have been buffered data is obtained in byte order, actually decoding all of the average bit rate seen so far)

As a general rule, it is better to report that everything had been buffered when it had not.

Another quality-of-implementation issue would be playing a video backwards when the codec is designed only for forward playback (e.g. there aren't many key frames, and they are far apart, and the intervening frames only have deltas from the previous frame). User agents could do a poor job, e.g. only showing key frames; however, better implementations would do more work and thus do a better job, e.g. actually decoding parts of the video forwards, storing the complete frames, and then playing the frames backwards.

Similarly, while implementations are allowed to drop buffered data at any time (there is no requirement that a user agent keep all the media data obtained for the lifetime of the media element), it is again a quality of implementation issue: user agents with sufficient resources to keep all the data around are encouraged to do so, as this allows for a better user experience. For example, if the user is watching a live stream, a user agent could allow the user only to view the live video; however, a better user agent would buffer everything and allow the user to seek through the earlier material, pause it, play it forwards and backwards, etc.

When multiple tracks are synchronised with a `MediaController`, it is possible for scripts to add and remove media elements from the `MediaController`'s list of slaved media elements, even while these tracks are playing. How smoothly the media plays back in such situations is another quality-of-implementation issue.

When a media element that is paused is removed from a document and not reinserted before the next time the event loop spins, implementations that are resource constrained are encouraged to take that opportunity to release all hardware resources (like video planes, networking resources, and data buffers) used by the media element. (User agents still have to keep track of the playback position and so forth, though, in case playback is later restarted.)

#### 4.8.11 The canvas element

##### Categories:

- Flow content.
- Phrasing content.
- Embedded content.
- Palpable content.

##### Contexts in which this element can be used:

Where embedded content is expected.

##### Content model:

Transparent, but with no interactive content descendants except for `a` elements, `button` elements, `input` elements whose `type` attribute are in the Checkbox or Radio Button states, and `input` elements that are buttons.

##### Content attributes:

- Global attributes
- `width`
- `height`

##### DOM interface:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

```
attribute unsigned long height;

RenderingContext? getContext(DOMString contextId, any... arguments);
boolean supportsContext(DOMString contextId, any... arguments);

void setContext(RenderingContext context);
CanvasProxy transferControlToProxy();

DOMString toDataURL(optional DOMString type, any... arguments);
any... arguments);
any... arguments);
any... arguments);
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

The `canvas` element is used for rendering graphics, game art.

rendering graphs, game

Authors should not use the `canvas` element in a document when a more suitable element is available. For example, it is inappropriate to use a `canvas` element to render a page heading: if the desired presentation of the heading is graphically intense, it should be marked up using appropriate elements (typically `h1`) and then styled using CSS and supporting technologies such as XBL.

When authors use the `canvas` element, they must also provide content that, when presented to the user, conveys essentially the same function or purpose as the `canvas`' bitmap. This content may be placed as content of the `canvas` element. The contents of the `canvas` element, if any, are the element's fallback content.

In interactive visual media, if scripting is enabled for the `canvas` element, and if support for `canvas` elements has been enabled, the `canvas` element represents embedded content consisting of a dynamically created image, the element's bitmap.

In non-interactive, static, visual media, if the `canvas` element has been previously associated with a rendering context (e.g. if the page was viewed in an interactive visual medium and is now being printed, or if some script that ran during the page layout process painted on the element), then the `canvas` element represents embedded content with the element's current bitmap and size. Otherwise, the `canvas` element represents its fallback content instead.

In non-visual media, and in visual media if scripting is disabled for the `canvas` element or if support for `canvas` elements has been disabled, the `canvas` element represents its fallback content instead.

When a `canvas` element represents embedded content, the user can still focus descendants of the `canvas` element (in the fallback content). When an element is focused, it is the target of keyboard interaction events (even though the element itself is not visible). This allows authors to make an interactive canvas keyboard-accessible: authors should have a one-to-one mapping of interactive regions to focusable elements in the fallback content. (Focus has no effect on mouse interaction events.) [DOMEVENTS]

The `canvas` element has two attributes to control the size of the coordinate space: `width` and `height`. These attributes, when specified, must have values that are valid non-negative integers. The rules for parsing non-negative integers must be used to obtain their numeric values. If an attribute is missing, or if parsing its value returns an error, then the default value must be used instead. The `width` attribute defaults to 300, and the `height` attribute defaults to 150.

The intrinsic dimensions of the `canvas` element when it represents embedded content are equal to the dimensions of the element's bitmap.

**Note:** A `canvas` element can be sized arbitrarily by a style sheet, its bitmap is then subject to the 'object-fit' CSS property. [CSSIMAGES]

The bitmaps of `canvas` elements, as well as some of the bitmaps of rendering contexts, such as those described in the section on the `CanvasRenderingContext2D` object below, have an `origin-clean` flag, which can be set to true or false. Initially, when the `canvas` element is created, its bitmap's origin-clean flag must be set to true.

A `canvas` bitmap can also have a hit region list, as described in the `CanvasRenderingContext2D` section below.

A `canvas` element can have a rendering context bound to it. Initially, it does not have a bound rendering context. To keep track of whether it has a rendering context or not, and what kind of rendering context it is, a `canvas` also has a `canvas context mode`, which

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

removed.

When a `canvas` element represents embedded content, it provides a paint source whose width is the element's intrinsic width, whose height is the element's intrinsic height, and whose appearance is the element's bitmap.

Whenever the `width` and `height` content attributes are set, removed, changed, or redundantly set to the value they already have, if the canvas context mode is direct-2d, the user agent must set bitmap dimensions to the numeric values of the `width` and `height` content attributes.

The `width` ar

with the same defaults.

**Note: 1** **that of** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/context>

**density will match**

ts are given below this box.

Returns `null` if the given context ID is not supported, or the canvas has already been initialized with the other context type (e.g. trying to get a "2d" context after getting a "webgl" context).

This specification defines the "2d" context below. There is also a specification that defines a "webgl" context. [WEBGL]

Returns `null` if the given context ID is not supported, if the canvas has already been initialized with the other context type (e.g. trying to get a "2d" context after getting a "webgl" context).

Throws an `InvalidStateError` if the `getContext()` or `transferControlToProxy()` methods have been used.

**supported = canvas . supportsContext(contextId [, ... ])**

Returns `false` if calling `getContext()` with the same arguments would definitely return `null`, and `true` otherwise.

This return value is not a guarantee that `getContext()` will or will not return an object, as conditions (e.g. availability of system resources) can vary over time.

Throws an `InvalidStateError` if the `getContext()` or `transferControlToProxy()` methods have been used.

**canvas . setContext(context)**

Sets the `canvas`' rendering context to the given object.

Throws an `InvalidStateError` exception if the `getContext()` or `transferControlToProxy()` methods have been used.

There are two ways for a `canvas` element to acquire a rendering context: the `canvas` element can provide one via the `getContext()` method, and one can be assigned to it via the `setContext()` method. In addition, the whole issue of a rendering context can be taken out of the `canvas` element's hands and passed to a `CanvasProxy` object, which itself can then be assigned a rendering context using its `setContext()` method.

These three methods are mutually exclusive; calling any of the three makes the other two start throwing `InvalidStateError` exceptions when called.

Each rendering context has a **context bitmap mode**, which is one of **fixed**, **unbound**, or **bound**. Initially, rendering contexts must be in the unbound mode.

The `getContext(contextId, arguments...)` method of the `canvas` element, when invoked, must run the steps in the cell of the following table whose column header describes the `canvas` element's canvas context mode and whose row header describes the method's first argument.

	none	direct-2d	direct-webgl	indirect	proxied
"2d"	Set the <code>canvas</code> element's context mode to direct-2d; follow the 2D context creation algorithm defined in the section below, passing it the <code>canvas</code> element, to obtain a <code>CanvasRenderingContext2D</code> object; set that object's context bitmap mode to fixed, and return the <code>CanvasRenderingContext2D</code> object	Return the same object as was returned the last time the method was invoked with this same	Return null.	Throw an <code>InvalidStateError</code> exception.	Throw an <code>InvalidStateError</code> exception.

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

<b>WebGL feature in its current configuration</b>	value is null, then return null and abort these steps, otherwise, set the <code>canvas</code> element's context mode to <code>direct-webgl</code> , set the new <code>WebGLRenderingContext</code> object's context bitmap mode to <code>fixed</code> , and return the <code>WebGLRenderingContext</code> object† [WEBGL]		last time the method was invoked with this same argument.		
<b>A vendor-specific extension*</b>	Behave as defined for the extension.	Behave as defined for the extension.	Behave as defined for the extension.	Throw an <code>InvalidStateError</code> exception.	Throw an <code>InvalidStateError</code> exception.
<b>An unsupported value†</b>	Return null.	Return null.	Return null.	Throw an <code>InvalidStateError</code> exception.	Throw an <code>InvalidStateError</code> exception.

\* Vendors may Too slow? Try reading the multipage copy of

† For example implementation the spec instead: <https://web.archive.org>

<http://web/20130103030411/>

The support for //whatwg.org/html

`getContext`

ving no software fallback

bGL specification for details.

must return false if calling

must return false if calling

The `setContext(context)` method of the `canvas` element, when invoked, must run following the steps:

1. If the `canvas` element's canvas context mode is neither `none` nor `indirect`, throw an `InvalidStateError` and abort these steps.
  2. If `context`'s context bitmap mode is `fixed`, then throw an `InvalidStateError` and abort these steps.
  3. If `context`'s context bitmap mode is `bound`, then run `context`'s unbinding steps and set its `context`'s context bitmap mode to `unbound`.
  4. Run `context`'s binding steps to bind it to this `canvas` element.
  5. Set the `canvas` element's context mode to `indirect` and the `context`'s context bitmap mode to `bound`.

*This box is non-normative. Implementation requirements are given below this box.*

```
url = canvas . toDataURL([type, ...])
url = canvas . toDataURLHD([type, ...])
```

Returns a data: URI for the image in the canvas

The first argument, if provided, controls the type of the image to be returned (e.g. PNG or JPEG). The default is image/png; that type is also used if the given type isn't supported. The other arguments are specific to the type, and control the way that the image is generated, as given in the table below.

When trying to use types other than "image/png", authors can check if the image was really returned in the requested format by checking to see if the returned string starts with one of the exact strings "data:image/png," or "data:image/png;". If it does, the image is PNG, and thus the requested type was not supported. (The one exception to this is if the canvas has either no height or no width, in which case the result might simply be "data:;".)

The `toDataURL()` method returns the data at a resolution of 96dpi. The `toDataURLHD()` method returns it at the native canvas bitmap resolution.

```
canvas.toBlob(callback [, type, ...])
canvas.toBlobAs(callback [, type, ...])
```

Creates a `Blob` object representing a file containing the image in the canvas, and invokes a callback with a handle to that object.

The second argument, if provided, controls the type of the image to be returned (e.g. PNG or JPEG). The default is image/png; that type is also used if the given type isn't supported. The other arguments are specific to the type, and control the way that the image is generated, as given in the table below.

The `toBlob()` method provides the data at a resolution of 96dpi. The `toBlobHD()` method provides it at the native canvas bitmap resolution.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

the string "`data:`," and abort these steps. (This is the shortest `data:` URL; it represents the empty string in a `text/plain` resource.)

3. Let `file` be a serialization of the `canvas` element's bitmap as a file, using the method's arguments (if any) as the `arguments`, and with the `native` flag set if the method is the `toDataURLHD()` method, and not otherwise.
4. Return a `data:` URL representing `file`. [RFC2397]

The `toBlob()` and `toBlobHD()` methods must run the following steps:

1. If the `length` of `arguments` is zero, then let `result` be a `Blob` object with a single byte of data (whose `size` is one). Otherwise, if the `length` is greater than one, then throw a `RangeError` exception and abort these steps.
2. Let `blob` be a `Blob` object with a single byte of data (whose `size` is one).
3. Let `options` be `{}.`
4. If the `length` of `arguments` is zero, then let `result` be a `Blob` object with a single byte of data (whose `size` is one). Otherwise, if the `length` is greater than one, then throw a `RangeError` exception and abort these steps.
5. If `callback` is null, abort these steps.
6. If `callback` is not null, queue a task to invoke the `FileCallback` `callback` with `result` as its argument. The task source for this task is the `canvas blob serialization task source`. [FILESYSTEMAPI]

#### 4.8.11.1 Proxying canvases to workers

Since DOM nodes cannot be accessed across worker boundaries, a proxy object is needed to enable workers to render to `canvas` elements in `Documents`.

```
interface CanvasProxy {
 void setContext(RenderingContext context);
};

CanvasProxy implements Transferable;
```

*This box is non-normative. Implementation requirements are given below this box.*

`canvasProxy = canvas . transferControlToProxy()`

Returns a `CanvasProxy` object that can be used to transfer control for this `canvas` over to another document (e.g. an `iframe` from another origin) or to a worker.

Throws an `InvalidStateError` exception if the `getContext()` or `setContext()` methods have been used.

`canvasProxy . setContext(context)`

Sets the `CanvasProxy` object's `canvas` element's rendering context to the given object.

Throws an `InvalidStateError` exception if the `CanvasProxy` has been transferred.

The `transferControlToProxy()` method of the `canvas` element, when invoked, must run following the steps:

1. If the `canvas` element's canvas context mode is not `none`, throw an `InvalidStateError` and abort these steps.
2. Set the `canvas` element's context mode to proxied.
3. Return a `CanvasProxy` object bound to this `canvas` element.

A `CanvasProxy` object can be neutered (like any `Transferable` object), meaning it can no longer be transferred, and can be disabled, meaning it can no longer be bound to rendering contexts. When first created, a `CanvasProxy` object must be neither.

A `CanvasProxy` is created with a link to a `canvas` element. A `CanvasProxy` object that has not been disabled must have a strong

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

2. If the `CanvasProxy` object has not been neutered, then neuter it.
3. If `context`'s context bitmap mode is fixed, then throw an `InvalidStateError` and abort these steps.
4. If `context`'s context bitmap mode is bound, then run `context`'s unbinding steps and set its `context`'s context bitmap mode to unbound.
5. Run `context`'s binding steps to bind it to this `CanvasProxy` object's `canvas` element.

6. Set `t†`

To transfer a `canvas` element to another `CanvasProxy` object linked to the same `Document`, do the following steps:

Here is some sample code:

```
Here
To transfer a canvas element to another CanvasProxy object linked to the same Document, do the following steps:
1. Create a new Worker object.
2. Post a message to the worker with the canvas element as the argument.
3. In the worker's onmessage event handler, get the canvas element from the event data.
4. Create a new CanvasProxy object and transfer control to it.
5. Set the canvas element's rendering context to the new CanvasProxy object.
6. Set the canvas element's width and height to match the new CanvasProxy object's width and height.
7. Commit the canvas element to the new CanvasProxy object.
```

```
<
<script>
 var canvas = document.getElementsByTagName('canvas')[0];
 var proxy = canvas.transferControlToProxy();
 var worker = new Worker('clock.js');
 worker.postMessage(proxy, [proxy]);
</script>
```

Second, the worker:

```
onmessage = function (event) {
 var context = new CanvasRenderingContext2d();
 event.data.setContext(context); // event.data is the CanvasProxy object
 setInterval(function () {
 context.clearRect(0, 0, context.width, context.height);
 context.fillText(0, 100, new Date());
 context.commit();
 }, 1000);
};
```

#### 4.8.11.2 The 2D rendering context

```
typedef (HTMLImageElement or
 HTMLVideoElement or
 HTMLCanvasElement or
 CanvasRenderingContext2D or
 ImageBitmap) CanvasImageSource;

[Constructor(optional unsigned long width, optional unsigned long height)]
interface CanvasRenderingContext2D {

 // back-reference to the canvas
 readonly attribute HTMLCanvasElement canvas;

 // canvas dimensions
 attribute unsigned long width;
 attribute unsigned long height;

 // for contexts that aren't directly fixed to a specific canvas
 void commit(); // push the image to the output bitmap

 // state
 void save(); // push state on state stack
 void restore(); // pop state stack and restore state
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
void transform(unrestricted double a, unrestricted double b, unrestricted double c,
unrestricted double d, unrestricted double e, unrestricted double f);
void setTransform(unrestricted double a, unrestricted double b, unrestricted double c,
unrestricted double d, unrestricted double e, unrestricted double f);
void resetTransform();

// compositing
attribute unrestricted double globalAlpha; // (default 1.0)
attribute DOMString globalCompositeOperation; // (default source-over)

// Too slow? Try reading the multipage copy of
// the spec instead: https://web.archive.org
// /web/20130103030411/http:
blac eStyle; // (default
black tyle; // (default
CanvasGradient createLinearGradient(double x0, double y0, double x1, double y1);
CanvasGradient createRadialGradient(double x0, double y0, double r0, double x1, double y1,
double r1);
CanvasPattern createPattern(CanvasImageSource image, DOMString repetition);

// shadows
attribute unrestricted double shadowOffsetX; // (default 0)
attribute unrestricted double shadowOffsetY; // (default 0)
attribute unrestricted double shadowBlur; // (default 0)
attribute DOMString shadowColor; // (default transparent black)

// rects
void clearRect(unrestricted double x, unrestricted double y, unrestricted double w,
unrestricted double h);
void fillRect(unrestricted double x, unrestricted double y, unrestricted double w,
unrestricted double h);
void strokeRect(unrestricted double x, unrestricted double y, unrestricted double w,
unrestricted double h);

// path API (see also CanvasPathMethods)
void beginPath();
void fill();
void fill(Path path);
void stroke();
void stroke(Path path);
void drawSystemFocusRing(Element element);
void drawSystemFocusRing(Path path, Element element);
boolean drawCustomFocusRing(Element element);
boolean drawCustomFocusRing(Path path, Element element);
void scrollPathIntoView();
void scrollPathIntoView(Path path);
void clip();
void clip(Path path);
void resetClip();
boolean isPointInPath(unrestricted double x, unrestricted double y);
boolean isPointInPath(Path path, unrestricted double x, unrestricted double y);

// text (see also the CanvasDrawingStyles interface)
void fillText(DOMString text, unrestricted double x, unrestricted double y, optional
unrestricted double maxWidth);
void strokeText(DOMString text, unrestricted double x, unrestricted double y, optional
unrestricted double maxWidth);
TextMetrics measureText(DOMString text);
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

unrestricted double sw, unrestricted double sh, unrestricted double dx, unrestricted double dy, unrestricted double dw, unrestricted double dh);

// hit regions  
void addHitRegion(HitRegionOptions options);  
void removeHitRegion(HitRegionOptions options);

// pixel manipulation  
ImageData createImageData(double sw, double sh);

Im  
Im  
Im  
vo  
vo  
doub  
vo  
vo  
dirtyY, double dirtyX,  
dirtyX, double  
dirtyWidth, double dirtyHeight);  
};  
CanvasRenderingContext2D implements CanvasDrawingStyles;  
CanvasRenderingContext2D implements CanvasPathMethods;

[NoInterfaceObject]  
interface **CanvasDrawingStyles** {  
 // line caps/joins  
 attribute unrestricted double lineWidth; // (default 1)  
 attribute DOMString lineCap; // "butt", "round", "square" (default "butt")  
 attribute DOMString lineJoin; // "round", "bevel", "miter" (default "miter")  
 attribute unrestricted double miterLimit; // (default 10)  
  
 // dashed lines  
 void setLineDash(sequence<unrestricted double> segments); // default empty  
 sequence<unrestricted double> getLineDash();  
 attribute unrestricted double lineDashOffset;  
  
 // text  
 attribute DOMString font; // (default 10px sans-serif)  
 attribute DOMString textAlign; // "start", "end", "left", "right", "center"  
(default: "start")  
 attribute DOMString textBaseline; // "top", "hanging", "middle", "alphabetic",  
 "ideographic", "bottom" (default: "alphabetic")  
 attribute DOMString direction; // "ltr", "rtl", "inherit" (default: "inherit")  
};  
  
[NoInterfaceObject]  
interface **CanvasPathMethods** {  
 // shared path API methods  
 void closePath();  
 void moveTo(unrestricted double x, unrestricted double y);  
 void lineTo(unrestricted double x, unrestricted double y);  
 void quadraticCurveTo(unrestricted double cpx, unrestricted double cpy, unrestricted  
 double x, unrestricted double y);  
 void bezierCurveTo(unrestricted double cp1x, unrestricted double cp1y, unrestricted double  
 cp2x, unrestricted double cp2y, unrestricted double x, unrestricted double y);  
 void arcTo(unrestricted double x1, unrestricted double y1, unrestricted double x2,  
 unrestricted double y2, unrestricted double radius);  
 void arcTo(unrestricted double x1, unrestricted double y1, unrestricted double x2,  
 unrestricted double y2, unrestricted double radiusX, unrestricted double radiusY,  
 unrestricted double rotation);  
 void rect(unrestricted double x, unrestricted double y, unrestricted double w,  
 unrestricted double h);

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
unrestricted double endAngle, boolean anticlockwise);
};

interface CanvasGradient {
 // opaque object
 void addColorStop(double offset, DOMString color);
};

interface CanvasPattern {
 //
 vo Too slow? Try reading the multipage copy of
};
 the spec instead: https://web.archive.org
inte
 // /web/20130103030411/http:
re
re //whatwg.org/html
rea

 // y-direction
 readonly attribute double fontBoundingBoxAscent;
 readonly attribute double fontBoundingBoxDescent;
 readonly attribute double actualBoundingBoxAscent;
 readonly attribute double actualBoundingBoxDescent;
 readonly attribute double emHeightAscent;
 readonly attribute double emHeightDescent;
 readonly attribute double hangingBaseline;
 readonly attribute double alphabeticBaseline;
 readonly attribute double ideographicBaseline;
};

dictionary HitRegionOptions {
 Path? path = null;
 DOMString id = "";
 DOMString? parentID = null;
 DOMString cursor = "inherit";
 // for control-backed regions:
 Element? control = null;
 // for unbacked regions:
 DOMString? label = null;
 DOMString? role = null;
};

interface ImageData {
 readonly attribute unsigned long width;
 readonly attribute unsigned long height;
 readonly attribute Uint8ClampedArray data;
};

[Constructor(optional Element scope)]
interface DrawingStyle { };
DrawingStyle implements CanvasDrawingStyles;

[Constructor,
 Constructor(Path path),
 Constructor(DOMString d)]
interface Path {
 void addPath(Path path, SVGMatrix? transformation);
 void addPathBy strokingPath(Path path, CanvasDrawingStyles styles, SVGMatrix?
transformation);
 void addText(DOMString text, CanvasDrawingStyles styles, SVGMatrix? transformation,
unrestricted double x, unrestricted double y, optional unrestricted double maxWidth);
```

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
void addPathBy strokingText(DOMString text, CanvasDrawingStyles styles, SVGMatrix? transformation, Path path, optional unrestricted double maxWidth);
};

Path implements CanvasPathMethods;
```

This box is non-normative. Implementation requirements are given below this box.

**context = canvas.getContext('2d')**

Return



as element.

**context**: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**context.width****context.height**

Return the dimensions of the bitmap, in CSS pixels.

Can be set, to update the bitmap's dimensions. If the rendering context is bound to a canvas, this will also update the canvas' intrinsic dimensions.

**context.commit()**If the rendering context is bound to a `canvas`, display the current frame.

A `CanvasRenderingContext2D` object can be obtained in two ways: the `getContext()` method on a `canvas` element (which invokes the 2D context creation algorithm), and the `CanvasRenderingContext2D()` constructor.

A `CanvasRenderingContext2D` object has a **scratch bitmap** and can be bound to an **output bitmap**. These are initialized when the object is created, and can be subsequently adjusted when the rendering context is bound or unbound. In some cases, these bitmaps are the same underlying bitmap. In general, the scratch bitmap is what scripts interact with, and the output bitmap is what is being displayed. These bitmaps always have the same dimensions.

Each such bitmap has an origin-clean flag, which can be set to true or false. Initially, when one of these bitmaps is created, its origin-clean flag must be set to true.

These bitmaps also have a hit region list, which is described in a later section. Initially, this list is empty. Scratch bitmaps also have a **list of pending interface actions**, which can contain instructions to draw the user's attention to a location on the bitmap, and instructions to scroll to a location on the bitmap. Initially, this list is also empty.

The **2D context creation algorithm**, which is passed a `target` (a `canvas` element), consists of running the following steps:

1. Create a new `CanvasRenderingContext2D` object.
2. Initialize its `canvas` attribute to point to `target`.
3. Let the new `CanvasRenderingContext2D` object's output bitmap and scratch bitmap both be the same bitmap as `target`'s bitmap (so that they are shared).
4. Set bitmap dimensions to the numeric values of `target`'s `width` and `height` content attributes.
5. Return the new `CanvasRenderingContext2D` object.

The `CanvasRenderingContext2D()` constructor, when invoked, must run the following steps:

1. Create a new `CanvasRenderingContext2D` object.
2. Initialize its `canvas` attribute to null.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

5. Set bitmap dimensions to *width* and *height*.
6. Let the new `CanvasRenderingContext2D` object have no output bitmap.
7. Return the new `CanvasRenderingContext2D` object.

When the user agent is required to **commit the scratch bitmap** for a rendering context, it must run the following steps:

1. Let *bitmap copy* be a copy of the rendering context's scratch bitmap.
2. Let *output bitmap* be the rendering context's output bitmap.
3. Let *list of pending interface actions* be an empty list.
4. Let *list of pending interface actions* be an empty list.
5. Empty *list of pending interface actions*.
6. If the `document` object has a `lastOutputBitmap` attribute, then set *output bitmap* to `lastOutputBitmap`. Otherwise, set *output bitmap* to the rendering context's output bitmap.
7. Let *output bitmap* be the rendering context's output bitmap.
8. Queue a task from the canvas updating task source of the output bitmap's `canvas` element's event loop to perform the following substeps:
  1. Overwrite *output bitmap* with *bitmap copy*.
  2. Overwrite *output bitmap*'s origin-clean flag with *origin-clean flag copy*.
  3. Overwrite *output bitmap*'s hit region list with *hit region list copy*.
  4. Follow the directions in the *list of pending interface actions copy*.

The **canvas updating task source** is just used for the algorithm above.

The `commit()` method must run the following steps:

1. If the rendering context's context bitmap mode is fixed, throw an `InvalidStateError` exception and abort these steps.
2. Commit the scratch bitmap for the rendering context.

**Note:** *The scratch bitmap is only committed when the `commit()` method is called. (This doesn't matter for canvas elements in direct-2d mode, since there the scratch bitmap is also the canvas element's bitmap so every drawing operation is immediately drawn.)*

When the user agent is to **set bitmap dimensions** to *width* and *height*, it must run the following steps:

1. Reset the rendering context to its default state.
2. Clear the scratch bitmap's hit region list and its list of pending interface actions.
3. Resize the scratch bitmap to the new *width* and *height* and clear it to fully transparent black.
4. If the rendering context has an output bitmap, and the scratch bitmap is a different bitmap than the output bitmap, then resize the output bitmap to the new *width* and *height* and clear it to fully transparent black.
5. If the rendering context's context bitmap mode is fixed, then run these substeps:
  1. Let *canvas* be the `canvas` element to which the rendering context's `canvas` attribute was initialized.
  2. If the rendering context's context bitmap mode is fixed and the numeric value of the *canvas'* `width` content attribute differs from *width*, then set *canvas'* `width` content attribute to the shortest possible string representing *width* as a valid non-negative integer.
  3. If the rendering context's context bitmap mode is fixed and the numeric value of the *canvas'* `height` content attribute differs from *height*, then set *canvas'* `height` content attribute to the shortest possible string representing *height* as a valid non-negative integer.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
canvas.setAttribute('width', '300'); // clears the canvas
context.fillRect(0,100,50,50);
canvas.width = canvas.width; // clears the canvas
context.fillRect(100,0,50,50); // only this square remains
```

When the user agent is to run the **unbinding steps** for a rendering context, it must run the following steps:

1. Reset the rendering context to its default state.

2. Clear

3. Clear
4. Set
5. Let

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

When the user agent runs the <https://web.archive.org/web/20130103030411/http://whatwg.org/html> steps:

it must run the following

1. Reset the rendering context to its default state.

2. Clear the scratch bitmap's hit region list and its list of pending interface actions.

3. Resize the `CanvasRenderingContext2D` object's scratch bitmap to the dimensions of `target`'s bitmap and clear it to fully transparent black.

4. Set the `CanvasRenderingContext2D` object's scratch bitmap's origin-clean flag to true.

5. Let the `CanvasRenderingContext2D` object's output bitmap be `target`'s bitmap.

The `CanvasRenderingContext2D` 2D rendering context represents a flat linear Cartesian surface whose origin (0,0) is at the top left corner, with the coordinate space having x values increasing when going right, and y values increasing when going down. The x-coordinate of the left-most edge is equal to the width of the rendering context's scratch bitmap in CSS pixels; similarly, the y-coordinate of the bottom-most edge is equal to the height of the rendering context's scratch bitmap in CSS pixels.

The size of the coordinate space does not necessarily represent the size of the actual bitmaps that the user agent will use internally or during rendering. On high-definition displays, for instance, the user agent may internally use bitmaps with two device pixels per unit in the coordinate space, so that the rendering remains at high quality throughout. Anti-aliasing can similarly be implemented using oversampling with bitmaps of a higher resolution than the final image on the display.

The `canvas` attribute must return the value it was initialized to when the object was created.

The `width` attribute, on getting, must return the width of the rendering context's scratch bitmap, in CSS pixels. On setting, it must set bitmap dimensions to the new value and the current height of the rendering context's scratch bitmap in CSS pixels, respectively.

The `height` attribute, on getting, must return the height of the rendering context's scratch bitmap, in CSS pixels. On setting, it must set bitmap dimensions to the current width of the rendering context's scratch bitmap in CSS pixels and the new value, respectively.

Except where otherwise specified, for the 2D context interface, **any method call with a numeric argument whose value is infinite or a NaN value must be ignored**.

Whenever the CSS value `currentColor` is used as a color in the `CanvasRenderingContext2D` API, the "computed value of the 'color' property" for the purposes of determining the computed value of the `currentColor` keyword is the value described by the appropriate entry in the following list:

#### If the rendering context's context bitmap mode is fixed and the canvas element is being rendered

The "computed value of the 'color' property" for the purposes of determining the computed value of the `currentColor` keyword is the computed value of the 'color' property on the `canvas` element at the time that the color is specified (e.g. when the appropriate attribute is set, or when the method is called; not when the color is rendered or otherwise used). [CSSCOLOR]

#### In all other cases

The "computed value of the 'color' property" for the purposes of determining the computed value of the `currentColor` keyword is fully opaque black. [CSSCOLOR]

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*canvases can be used by another, and there is therefore no way to know which is the element in question at the time that the color is specified.*

**Note:** Similar concerns exist with font-related properties; the rules for those are described in detail in the relevant section below.

#### 4.8.11.2.1 Implementation notes

This section i

Although the rendering cc the actually The scratch implementat it until such t ImageBitmap ..

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

bitmaps per canvas or canv as, and one bitmap for

ad, and therefore is that have been applied to Image(), or the

The bitmap of a canvas element is the one bitmap that's pretty much always going to be needed in practice. The output bitmap of a rendering context, when it has one, is always just an alias to a canvas element's bitmap.

Additional bitmaps are sometimes needed, e.g. to enable fast drawing when the canvas is being painted at a different size than its intrinsic size, or to enable double buffering so that the rendering commands from the scratch bitmap can be applied without the rendering being updated midway.

#### 4.8.11.2.2 The canvas state

Each CanvasRenderingContext2D rendering context maintains a stack of drawing states. **Drawing states** consist of:

- The current transformation matrix.
- The current clipping region.
- The current values of the following attributes: strokeStyle, fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, lineDashOffset, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation, font, textAlign, textBaseline, direction, imageSmoothingEnabled.
- The current dash list.

**Note:** The current default path and the rendering context's bitmaps are not part of the drawing state. The current default path is persistent, and can only be reset using the beginPath() method. The bitmaps depend on whether and how the rendering context is bound to a canvas element.

This box is non-normative. Implementation requirements are given below this box.

`context.save()`

Pushes the current state onto the stack.

`context.restore()`

Pops the top state on the stack, restoring the context to that state.

The `save()` method must push a copy of the current drawing state onto the drawing state stack.

The `restore()` method must pop the top entry in the drawing state stack, and reset the drawing state it describes. If there is no saved state, the method must do nothing.

When the user agent is to **reset the rendering context to its default state**, it must clear the drawing state stack and everything that drawing state consists of to initial values.

#### 4.8.11.2.3 DrawingStyle objects

All the line styles (line width, caps, joins, and dash patterns) and text styles (fonts) described in the next two sections apply to CanvasRenderingContext2D objects and to DrawingStyle objects. This section defines the constructor used to obtain a

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Creates a new `DrawingStyle` object, optionally using a specific element for resolving relative keywords and sizes in font specifications.

Each `DrawingStyle` object can have a **styles scope object**.

The `DrawingStyle()` constructor, when invoked, must return a newly created `DrawingStyle` object. If the constructor was passed an argument, then the `DrawingStyle` object's styles scope object is that element. Otherwise, in the JavaScript global environment is a document environment.

Window object  
Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

**4.8.11.2.4 L** [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Properties are given below this box.

`context`.

`styles.lineWidth [= value]`

Returns the current line width.

Can be set, to change the line width. Values that are not finite values greater than zero are ignored.

`context.lineCap [= value]`

`styles.lineCap [= value]`

Returns the current line cap style.

Can be set, to change the line cap style.

The possible line cap styles are `butt`, `round`, and `square`. Other values are ignored.

`context.lineJoin [= value]`

`styles.lineJoin [= value]`

Returns the current line join style.

Can be set, to change the line join style.

The possible line join styles are `bevel`, `round`, and `miter`. Other values are ignored.

`context.miterLimit [= value]`

`styles.miterLimit [= value]`

Returns the current miter limit ratio.

Can be set, to change the miter limit ratio. Values that are not finite values greater than zero are ignored.

`context.setLineDash(segments)`

`styles.setLineDash(segments)`

Sets the current line dash pattern (as used when stroking). The argument is a list of distances for which to alternately have the line on and the line off.

`segments = context.getLineDash()`

`segments = styles.getLineDash()`

Returns a copy of the current line dash pattern. The array returned will always have an even number of entries (i.e. the pattern is normalized).

`context.lineDashOffset`

`styles.lineDashOffset`

Returns the phase offset (in the same units as the line dash pattern).

Can be set, to change the phase offset. Values that are not finite values are ignored.

Objects that implement the `CanvasDrawingStyles` interface have attributes and methods (defined in this section) that control how

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the object implementing the `CanvasDrawingStyles` interface is created, the `lineWidth` attribute must initially have the value 1.0.

The `lineCap` attribute defines the type of endings that UAs will place on the end of lines. The three valid values are `butt`, `round`, and `square`.

On getting, it must return the current value. On setting, if the new value is one of the literal strings `butt`, `round`, and `square`, then the current value must be changed to the new value; other values must be ignored, leaving the value unchanged.

When the object implementing the `CanvasDrawingStyles` interface is created, the `lineCap` attribute must initially have the value `butt`.  
**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The `lineJoin` attribute defines the type of joins that UAs will make between segments of lines. The three valid values are `bevel`, `round`, and `miter`.  
 On getting, it must return the current value. On setting, if the new value is one of the literal strings `bevel`, `round`, and `miter`, then the current value must be changed to the new value; other values must be ignored, leaving the value unchanged.

When the object implementing the `CanvasDrawingStyles` interface is created, the `lineJoin` attribute must initially have the value `miter`.

When the `lineJoin` attribute has the value `miter`, strokes use the miter limit ratio to decide how to render joins. The miter limit ratio can be explicitly set using the `miterLimit` attribute. On getting, it must return the current value. On setting, zero, negative, infinite, and NaN values must be ignored, leaving the value unchanged; other values must change the current value to the new value.

When the object implementing the `CanvasDrawingStyles` interface is created, the `miterLimit` attribute must initially have the value 10.0.

Each `CanvasDrawingStyles` object has a **dash list**, which is either empty or consists of an even number of non-negative numbers. Initially, the dash list must be empty.

When the `setLineDash()` method is invoked, it must run the following steps:

1. Let `a` be the argument.
2. If any value in `a` is not finite (e.g. an Infinity or a NaN value), or if any value is negative (less than zero), then abort these steps (without throwing an exception; user agents could show a message on a developer console, though, as that would be helpful for debugging).
3. If the number of elements in `a` is odd, then let `a` be the concatenation of two copies of `a`.
4. Let the object's dash list be `a`.

When the `getLineDash()` method is invoked, it must return a sequence whose values are the values of the object's dash list, in the same order.

It is sometimes useful to change the "phase" of the dash pattern, e.g. to achieve a "marching ants" effect. The phase can be set using the `lineDashOffset` attribute. On getting, it must return the current value. On setting, infinite and NaN values must be ignored, leaving the value unchanged; other values must change the current value to the new value.

When the object implementing the `CanvasDrawingStyles` interface is created, the `lineDashOffset` attribute must initially have the value 0.0.

When a user agent is to **trace a path**, given an object `style` that implements the `CanvasDrawingStyles` interface, it must run the following algorithm. This algorithm returns a new path.

1. Let `path` be a copy of the path being traced.
2. Remove from `path` any subpaths containing no lines (i.e. empty subpaths with zero points, and subpaths with just one point).
3. Replace each point in each subpath of `path` other than the first point and the last point of each subpath by a *join* that joins the line leading to that point to the line leading out of that point, such that the subpaths all consist of two points (a starting point

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

the last point to a join (from the previously last line to the newly added closing line), and change the first point to a join (from the newly added closing line to the first line).

5. If the `styles dash` list is empty, jump to the step labeled `joins`.
6. Let `width` be the aggregate length of all the lines of all the subpaths in `path`, in coordinate space units.
7. Let `offset` be the value of the `styles lineDashOffset`, in coordinate space units.
8. While `offset` is greater than `width`, decrement it by `width`.



Whi  
9. Offs  
lenc  
offs  
10. Def  
first  
11. Let `position` be `u`.  
12. Let `index` be `0`.  
13. Let `current state` be `off` (the other states being `on` and `zero-on`).  
14. *Dash On:* Let `segment length` be the value of the `styles dash` list's `indexth` entry.  
15. Increment `position` by `segment length`.  
16. If `position` is greater than `width`, then jump to the step labeled `joins`.  
17. If `segment length` is non-zero, let `current state` be `on`.  
18. Increment `index` by one.  
19. *Dash Off:* Let `segment length` be the value of the `styles dash` list's `indexth` entry.  
20. Let `start` be the offset `position` on `L`.  
21. Increment `position` by `segment length`.  
22. If `position` is greater than `width`, then let `end` be the offset `width` on `L`. Otherwise, let `end` be the offset `position` on `L`.  
23. Jump to the first appropriate step:

↳ If `segment length` is zero and `current state` is `off`  
Do nothing, just continue to the next step.

#### ↳ If `current state` is `off`

Cut the line on which `end` finds itself short at `end` and place a point there, cutting the subpath that it was in two; remove all line segments, joins, points, and subpaths that are between `start` and `end`; and finally place a single point at `start` with no lines connecting to it.

The point has a *directionality* for the purposes of drawing line caps (see below). The directionality is the direction that the original line had at that point (i.e. when `L` was defined above).

#### ↳ Otherwise

Cut the line on which `start` finds itself into two at `start` and place a point there, cutting the subpath that it was in two, and similarly cut the line on which `end` finds itself short at `end` and place a point there, cutting the subpath that it was in two, and then remove all line segments, joins, points, and subpaths that are between `start` and `end`.

If `start` and `end` are the same point, then this results in just the line being cut in two and two points being inserted there, with nothing being removed, unless a join also happens to be at that point, in which case the join must be removed.

24. If `position` is greater than `width`, then jump to the step labeled `joins`.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

28. *Joins*: Remove from *path* any subpath that originally formed part of the subpath added in the *offset subpath* step above.
29. Create a new path that describes the result of inflating the paths in *path* perpendicular to the direction of the path to the *styles lineWidth* width, replacing each point with the end cap necessary to satisfy the *styles lineCap* attribute as described previously and elaborated below, and replacing each join with the join necessary to satisfy the *styles lineJoin* type, as defined below.

**Caps**: Each point has a flat edge perpendicular to the direction of the line coming out of it. This is then augmented according to the value of the *styles lineCap*. The *butt* value means that no additional line cap is added. The *round* value means that a

sem' eac styl add Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

Poi eac /web/20130103030411/http://whatwg.org/html

Join corners round half the line width away from the join point, one perpendicular to each line, each on the side furthest from the other line.

A filled triangle connecting these two opposite corners with a straight line, with the third point of the triangle being the join point, must be added at all joins. The *lineJoin* attribute controls whether anything else is rendered. The three aforementioned values have the following meanings:

The *bevel* value means that this is all that is rendered at joins.

The *round* value means that a filled arc connecting the two aforementioned corners of the join, abutting (and not overlapping) the aforementioned triangle, with the diameter equal to the line width and the origin at the point of the join, must be added at joins.

The *miter* value means that a second filled triangle must (if it can given the miter length) be added at the join, with one line being the line between the two aforementioned corners, abutting the first triangle, and the other two being continuations of the outside edges of the two joining lines, as long as required to intersect without going over the miter length.

The miter length is the distance from the point where the join occurs to the intersection of the line edges on the outside of the join. The miter limit ratio is the maximum allowed ratio of the miter length to half the line width. If the miter length would cause the miter limit ratio (as set by the *style miterLimit* attribute) to be exceeded, this second triangle must not be added.

Subpaths in the newly created path must wind clockwise, regardless of the direction of paths in *path*.

30. Return the newly created path.

#### 4.8.11.2.5 Text styles

This box is non-normative. Implementation requirements are given below this box.

**context . font [= value]**  
**styles . font [= value]**

Returns the current font settings.

Can be set, to change the font. The syntax is the same as for the CSS 'font' property; values that cannot be parsed as CSS font values are ignored.

Relative keywords and lengths are computed relative to the font of the `canvas` element.

**context . textAlign [= value]**  
**styles . textAlign [= value]**

Returns the current text alignment settings.

Can be set, to change the alignment. The possible values are and their meanings are given below. Other values are ignored. The default is `start`.

**context . textBaseline [= value]**  
**styles . textBaseline [= value]**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

`context.direction [= value]``styles.direction [= value]`

Returns the current directionality.

Can be set, to change the directionality. The possible values and their meanings are given below. Other values are ignored.

The default is `inherit`.

Objects that implement the `CanvasDrawingStyles` interface have attributes (defined in this section) that control how text is laid out

(rasterized or objects who: JavaScript code found; other DrawingSt

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

RenderingContext2D objects, if the context object of the active renderingContext2D object is not the worker. For

The `font` IDL

independent

forced to 'normal', with the 'font-size' component converted to CSS pixels, and with system fonts being computed to explicit values. If the new value is syntactically incorrect (including using property-independent style sheet syntax like 'inherit' or 'initial'), then it must be ignored, without assigning a new font value. [CSS]

Font family names must be interpreted in the context of the font style source object when the font is to be used; any fonts embedded using `@font-face` or loaded using the `FontLoader` that are visible to the font style source object must therefore be available once they are loaded. If a font is used before it is fully loaded, or if the font style source object does not have that font in scope at the time the font is to be used, then it must be treated as if it was an unknown font, falling back to another as described by the relevant CSS specifications. [CSSFONTS]

On getting, the `font` attribute must return the serialized form of the current font of the context (with no 'line-height' component). [CSSOM]

For example, after the following statement:

```
context.font = 'italic 400 12px/2 Unknown Font, sans-serif';
```

...the expression `context.font` would evaluate to the string "italic 12px "Unknown Font", sans-serif". The "400" font-weight doesn't appear because that is the default value. The line-height doesn't appear because it is forced to "normal", the default value.

When the object implementing the `CanvasDrawingStyles` interface is created, the font of the context must be set to 10px sans-serif. When the 'font-size' component is set to lengths using percentages, 'em' or 'ex' units, or the 'larger' or 'smaller' keywords, these must be interpreted relative to the computed value of the 'font-size' property of the font style source object at the time that the attribute is set, if it is an element. When the 'font-weight' component is set to the relative values 'bolder' and 'lighter', these must be interpreted relative to the computed value of the 'font-weight' property of the font style source object at the time that the attribute is set, if it is an element. If the computed values are undefined for a particular case (e.g. because the font style source object is not an element or is not being rendered), then the relative keywords must be interpreted relative to the normal-weight 10px sans-serif default.

The `textAlign` IDL attribute, on getting, must return the current value. On setting, if the value is one of `start`, `end`, `left`, `right`, or `center`, then the value must be changed to the new value. Otherwise, the new value must be ignored. When the object implementing the `CanvasDrawingStyles` interface is created, the `textAlign` attribute must initially have the value `start`.

The `textBaseline` IDL attribute, on getting, must return the current value. On setting, if the value is one of `top`, `hanging`, `middle`, `alphabetic`, `ideographic`, or `bottom`, then the value must be changed to the new value. Otherwise, the new value must be ignored. When the object implementing the `CanvasDrawingStyles` interface is created, the `textBaseline` attribute must initially have the value `alphabetic`.

The `direction` IDL attribute, on getting, must return the current value. On setting, if the value is one of `ltr`, `rtl`, or `inherit`, then the value must be changed to the new value. Otherwise, the new value must be ignored. When the object implementing the `CanvasDrawingStyles` interface is created, the `direction` attribute must initially have the value `inherit`.

The `textBaseline` attribute's allowed keywords are as follows:

`start`

Align to the start edge of the text (left side in left-to-right text, right side in right-to-left text).

[Go](#) NOV JAN FEB  
 ← 03 →  
 2012 2013 2014 ▾ About this capture

3,445 captures  
 5 Jun 2004 - 22 Aug 2020

Align to the left.

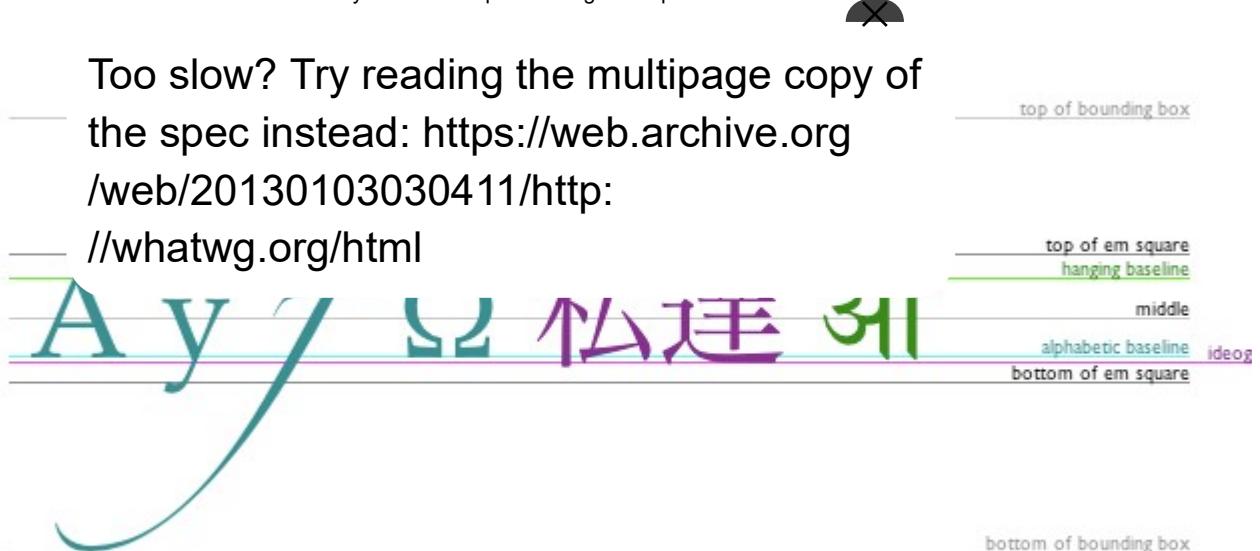
*right*

Align to the right.

*center*

Align to the center.

The `textBaseline` attribute's allowed keywords correspond to alignment points in the font:



The keywords map to these alignment points as follows:

*top*

The top of the em square

*hanging*

The hanging baseline

*middle*

The middle of the em square

*alphabetic*

The alphabetic baseline

*ideographic*

The ideographic baseline

*bottom*

The bottom of the em square

The `textBaseline` attribute's allowed keywords are as follows:

*ltr*

Treat input to the text preparation algorithm as left-to-right text.

*rtl*

Treat input to the text preparation algorithm as right-to-left text.

*inherit*

Default to the directionality of the `canvas` element or `Document` as appropriate.

The **text preparation algorithm** is as follows. It takes as input a string `text`, a `CanvasDrawingStyles` object `target`, and an optional length `maxWidth`. It returns an array of glyph shapes, each positioned on a common coordinate space, a *physical alignment* whose

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

3. Let *font* be the current font of *target*, as given by that object's *font* attribute.

4. Apply the appropriate step from the following list to determine the value of *direction*:

↳ If the *target* object's *direction* attribute has the value "ltr"

Let *direction* be 'ltr'.

↳ If the *target* object's *direction* attribute has the value "rtl"

Let *direction* be 'rtl'.



↳ Too slow? Try reading the multipage copy of

↳ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

not element child

element child.

↳ /web/20130103030411/http:

↳ //whatwg.org/html

5. Form a hypothetical infinitely-wide CSS line box containing a single inline box containing the text *text*, with all the properties at their initial values except the 'font' property of the inline box set to *font*, the 'direction' property of the inline box set to *direction*, and the 'white-space' property set to 'pre'. [CSS]

6. If *maxWidth* was provided and the hypothetical width of the inline box in the hypothetical line box is greater than *maxWidth* CSS pixels, then change *font* to have a more condensed font (if one is available or if a reasonably readable one can be synthesized by applying a horizontal scale factor to the font) or a smaller font, and return to the previous step.

7. The *anchor point* is a point on the inline box, and the *physical alignment* is one of the values *left*, *right*, and *center*. These variables are determined by the *textAlign* and *textBaseline* values as follows:

Horizontal position:

If *textAlign* is *left*

If *textAlign* is *start* and *direction* is 'ltr'

If *textAlign* is *end* and *direction* is 'rtl'

Let the *anchor point*'s horizontal position be the left edge of the inline box, and let *physical alignment* be *left*.

If *textAlign* is *right*

If *textAlign* is *end* and *direction* is 'ltr'

If *textAlign* is *start* and *direction* is 'rtl'

Let the *anchor point*'s horizontal position be the right edge of the inline box, and let *physical alignment* be *right*.

If *textAlign* is *center*

Let the *anchor point*'s horizontal position be half way between the left and right edges of the inline box, and let *physical alignment* be *center*.

Vertical position:

If *textBaseline* is *top*

Let the *anchor point*'s vertical position be the top of the em box of the first available font of the inline box.

If *textBaseline* is *hanging*

Let the *anchor point*'s vertical position be the hanging baseline of the first available font of the inline box.

If *textBaseline* is *middle*

Let the *anchor point*'s vertical position be half way between the bottom and the top of the em box of the first available font of the inline box.

If *textBaseline* is *alphabetic*

Let the *anchor point*'s vertical position be the alphabetic baseline of the first available font of the inline box.

If *textBaseline* is *ideographic*

Let the *anchor point*'s vertical position be the ideographic baseline of the first available font of the inline box.

If *textBaseline* is *bottom*

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

9. Return *result*, *physical alignment*, and the inline box.

#### 4.8.11.2.6 Building paths

Each object implementing the `CanvasPathMethods` interface has a path. A **path** has a list of zero or more subpaths. Each subpath consists of a list of one or more points, connected by straight or curved lines, and a flag indicating whether the subpath is closed or not. A closed subpath is one where the last point of the subpath is connected to the first point of the subpath by a straight line. Subpaths with fewer than two points are ignored when painting the path.



When an object is given below this box, it is given below this box.

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

`context.createPath()`  
`path.closePath()`  
`CreatePath()`

Marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.

`context.lineTo(x, y)`  
`path.lineTo(x, y)`

Adds the given point to the current subpath, connected to the previous one by a straight line.

`context.quadraticCurveTo(cpx, cpy, x, y)`  
`path.quadraticCurveTo(cpx, cpy, x, y)`

Adds the given point to the current subpath, connected to the previous one by a quadratic Bézier curve with the given control point.

`context.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)`  
`path.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)`

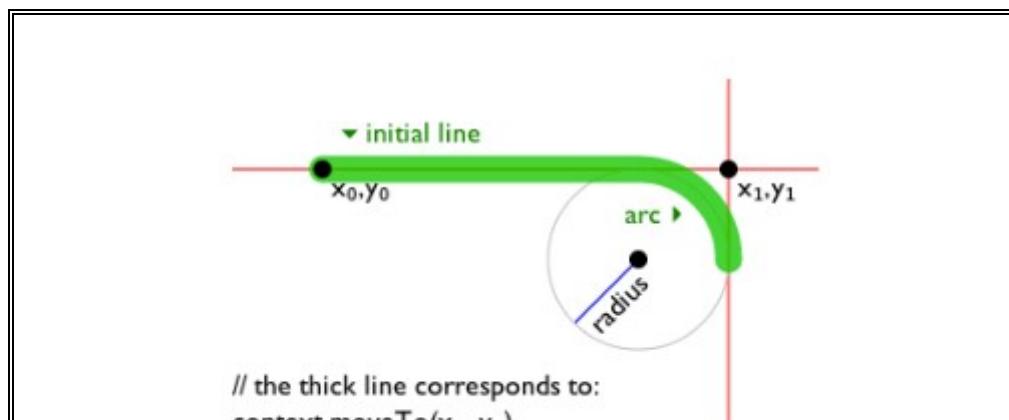
Adds the given point to the current subpath, connected to the previous one by a cubic Bézier curve with the given control points.

`context.arcTo(x1, y1, x2, y2, radiusX [, radiusY, rotation ])`  
`path.arcTo(x1, y1, x2, y2, radiusX [, radiusY, rotation ])`

Adds an arc with the given control points and radius to the current subpath, connected to the previous point by a straight line.

If two radii are provided, the first controls the width of the arc's ellipse, and the second controls the height. If only one is provided, or if they are the same, the arc is from a circle. In the case of an ellipse, the rotation argument controls the clockwise inclination of the ellipse relative to the x-axis.

Throws an `IndexSizeError` exception if the given radius is negative.



Go

NOV JAN FEB

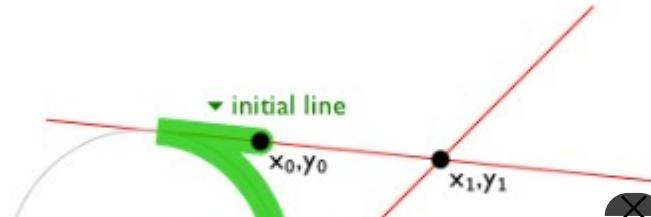
03

2012 2013 2014

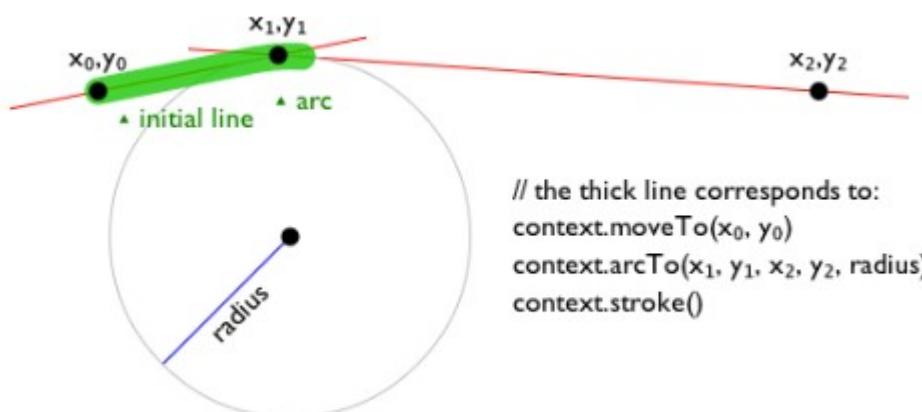
[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

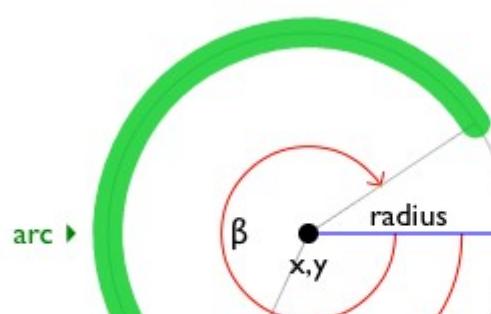


Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

to:  
( $x, y, radius$ )  
context.stroke()// the thick line corresponds to:  
context.moveTo( $x_0, y_0$ )  
context.arcTo( $x_1, y_1, x_2, y_2, radius$ )  
context.stroke()`context.arc(x, y, radius, startAngle, endAngle [, anticlockwise ])``path.arc(x, y, radius, startAngle, endAngle [, anticlockwise ])`

Adds points to the subpath such that the arc described by the circumference of the circle described by the arguments, starting at the given start angle and ending at the given end angle, going in the given direction (defaulting to clockwise), is added to the path, connected to the previous point by a straight line.

Throws an `IndexSizeError` exception if the given radius is negative.

// the thick line corresponds to:  
context.moveTo( $x_0, y_0$ )  
context.arc(x, y, radius,  $\alpha, \beta$ )  
context.stroke()

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

`context.ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise)`  
`path.ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise)`

Adds points to the subpath such that the arc described by the circumference of the ellipse described by the arguments, starting at the given start angle and ending at the given end angle, going in the given direction (defaulting to clockwise), is added to the path. connected to the previous point bv a straitght line.



**Thr** **Too slow? Try reading the multipage copy of**  
**context** **the spec instead: https://web.archive.org**  
**path.ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise)**  
**Add** **/web/20130103030411/http://whatwg.org/html**

The following ...

For `CanvasRenderingContext2D` objects, the points passed to the methods, and the resulting lines added to current default path by these methods, must be transformed according to the current transformation matrix before being added to the path.

The `moveTo(x, y)` method must create a new subpath with the specified point as its first (and only) point.

When the user agent is to ensure there is a subpath for a coordinate  $(x, y)$  on a path, the user agent must check to see if the path has any subpaths, and if it does not, then the user agent must create a new subpath with the point  $(x, y)$  as its first (and only) point, as if the `moveTo()` method had been called.

The `closePath()` method must do nothing if the object's path has no subpaths. Otherwise, it must mark the last subpath as closed, create a new subpath whose first point is the same as the previous subpath's first point, and finally add this new subpath to the path.

**Note:** If the last subpath had more than one point in its list of points, then this is equivalent to adding a straight line connecting the last point back to the first point, thus "closing" the shape, and then repeating the last (possibly implied) `moveTo()` call.

New points and the lines connecting them are added to subpaths using the methods described below. In all cases, the methods only modify the last subpath in the object's path.

The `lineTo(x, y)` method must ensure there is a subpath for  $(x, y)$  if the object's path has no subpaths. Otherwise, it must connect the last point in the subpath to the given point  $(x, y)$  using a straight line, and must then add the given point  $(x, y)$  to the subpath.

The `quadraticCurveTo(cpX, cpY, x, y)` method must ensure there is a subpath for  $(cpX, cpY)$ , and then must connect the last point in the subpath to the given point  $(x, y)$  using a quadratic Bézier curve with control point  $(cpX, cpY)$ , and must then add the given point  $(x, y)$  to the subpath. [BEZIER]

The `bezierCurveTo(cp1X, cp1Y, cp2X, cp2Y, x, y)` method must ensure there is a subpath for  $(cp1X, cp1Y)$ , and then must connect the last point in the subpath to the given point  $(x, y)$  using a cubic Bézier curve with control points  $(cp1X, cp1Y)$  and  $(cp2X, cp2Y)$ . Then, it must add the point  $(x, y)$  to the subpath. [BEZIER]

The `arcTo(x1, y1, x2, y2, radiusX, radiusY, rotation)` method must first ensure there is a subpath for  $(x1, y1)$ . Then, the behavior depends on the arguments and the last point in the subpath, as described below.

Negative values for `radiusX` or `radiusY` must cause the implementation to throw an `IndexSizeError` exception. If `radiusY` is omitted, user agents must act as if it had the same value as `radiusX`.

Let the point  $(x_0, y_0)$  be the last point in the subpath, transformed by the inverse of the current transformation matrix (so that it is in the same coordinate system as the points passed to the method).

If the point  $(x_0, y_0)$  is equal to the point  $(x_1, y_1)$ , or if the point  $(x_1, y_1)$  is equal to the point  $(x_2, y_2)$ , or if both `radiusX` and `radiusY` are zero, then the method must add the point  $(x_1, y_1)$  to the subpath, and connect that point to the previous point  $(x_0, y_0)$  by a straight line.

Otherwise, if the points  $(x_0, y_0)$ ,  $(x_1, y_1)$ , and  $(x_2, y_2)$  all lie on a single straight line, then the method must add the point  $(x_1, y_1)$  to the subpath, and connect that point to the previous point  $(x_0, y_0)$  by a straight line.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

straight line, adding the start tangent point to the subpath, and then must connect the start tangent point to the end tangent point by *The Arc*, adding the end tangent point to the subpath.

The `arc(x, y, radius, startAngle, endAngle, anticlockwise)` and `ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise)` methods draw arcs.

The `arc()` method is equivalent to the `ellipse()` method in the case where the two radii are equal. When the `arc()` method is invoked, it must act as if the `ellipse()` method had been invoked with the `radiusX` and `radiusY` arguments set to the value of the `radius` argument.

arguments c

**Too slow? Try reading the multipage copy of**

When the e: add a strai

the subpath, /web/20130103030411/http:

Consider an about its orig

`endAngle` alone points respectively.

paths, then the method must and end points of the arc to

: `radiusY`, and that is rotated at `startAngle` and axis, are the start and end

If the `anticlockwise` argument false and `endAngle-startAngle` is equal to or greater than  $2\pi$ , or, if the `anticlockwise` argument is `true` and `startAngle-endAngle` is equal to or greater than  $2\pi$ , then the arc is the whole circumference of this ellipse.

Otherwise, the arc is the path along the circumference of this ellipse from the start point to the end point, going anti-clockwise if the `anticlockwise` argument is true, and clockwise otherwise. Since the points are on the ellipse, as opposed to being simply angles from zero, the arc can never cover an angle greater than  $2\pi$  radians.

Negative values for `radiusX` or `radiusY` must cause the implementation to throw an `IndexSizeError` exception.

The `rect(x, y, w, h)` method must create a new subpath containing just the four points  $(x, y), (x+w, y), (x+w, y+h), (x, y+h)$ , with those four points connected by straight lines, and must then mark the subpath as closed. It must then create a new subpath with the point  $(x, y)$  as the only point in the subpath.

### 4.8.11.2.7 Path objects

Path objects can be used to declare paths that are then later used on `CanvasRenderingContext2D` objects. In addition to many of the APIs described in earlier sections, Path objects have methods to combine paths, and to add text to paths.

This box is non-normative. Implementation requirements are given below this box.

`path = new Path()`

Creates a new empty `Path` object.

`path = new Path(path)`

Creates a new `Path` object that is a copy of the argument.

`path = new Path(d)`

Creates a new path with the path described by the argument, interpreted as SVG path data. [SVG]

`path . addPath(path, transform)`

`path . addPathByStrokingPath(path, styles, transform)`

Adds to the path the path given by the argument.

In the case of the stroking variants, the line styles are taken from the `styles` argument, which can be either a `DrawingStyle` object or a `CanvasRenderingContext2D` object.

`path . addText(text, styles, transform, x, y [, maxWidth ])`

`path . addText(text, styles, transform, path [, maxWidth ])`

`path . addPathByStrokingText(text, styles, transform, x, y [, maxWidth ])`

`path . addPathByStrokingText(text, styles, transform, path [, maxWidth ])`

Adds to the path a series of subpaths corresponding to the given text. If the arguments give a coordinate, the text is drawn

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `Path()` constructor, when invoked, must return a newly created `Path` object.

The `Path(path)` constructor, when invoked, must return a newly created `Path` object, to which the subpaths of the argument are added. (In other words, it returns a copy of the argument.)

The `Path(d)` constructor must run the following steps:



1. **Par**: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html> obtaining an SVG path. [SVG]
2. **Let** `d` [//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)
3. **Crea**
4. Create a new subpath in the `Path` object with  $(x, y)$  as the only point in the subpath.
5. Return the `Path` object as the constructed object.

*and applying path*

The `addPath(b, transform)` method, when invoked on a `Path` object `a`, must run the following steps:

1. If the `Path` object `b` has no subpaths, abort these steps.
2. Create a copy of all the subpaths in `b`. Let this copy be known as `c`.
3. Transform all the coordinates and lines in `c` by the transform matrix `transform`, if it is not null.
4. Let  $(x, y)$  be the last point in the last subpath of `c`.
5. Add all the subpaths in `c` to `a`.
6. Create a new subpath in `a` with  $(x, y)$  as the only point in the subpath.

The `addPathByStrokingPath(b, styles, transform)` method, when invoked on a `Path` object `a`, must run the following steps:

1. If the `Path` object `b` has no subpaths, abort these steps.
2. Create a copy of all the subpaths in `b`. Let this copy be known as `c`.
3. Transform all the coordinates and lines in `c` by transformation matrix `transform`, if it is not null.
4. Let a new list of subpaths `d` be the result of tracing `c`, using the `styles` argument for the line styles.
5. Let  $(x, y)$  be the last point in the last subpath of `d`.
6. Add all the subpaths in `d` to `a`.
7. Create a new subpath in `a` with  $(x, y)$  as the only point in the subpath.

The `addText()` and `addPathByStrokingText()` methods each come in two variants: one rendering text at a given coordinate, and one rendering text along a given path. In both cases, the methods take a `CanvasDrawingStyles` object argument for the text and (if appropriate) line styles to use, an `SVGMatrix` object `transform` (which can be null), and a maximum width can optionally be provided.

When one of the `addText()` and `addPathByStrokingText()` variants that take as argument an  $(x, y)$  coordinate is invoked, the method must run the following algorithm:

1. Run the text preparation algorithm, passing it `text`, the `CanvasDrawingStyles` object argument, and, if the `maxWidth` argument was provided, that argument. Let `glyphs` be the result.
2. Move all the shapes in `glyphs` to the right by  $x$  CSS pixels and down by  $y$  CSS pixels.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

CanvasDrawingStyles object argument for the line styles.

5. Transform all the coordinates and lines in *glyph subpaths* by the transformation matrix *transform*, if it is not null.
6. Let  $(x_{final}, y_{final})$  be the last point in the last subpath of *glyph subpaths*.
7. Add all the subpaths in *glyph subpaths* to the *Path* object.
8. Create a new subpath in the *Path* object with  $(x_{final}, y_{final})$  as the only point in the subpath.

When one of

must run the

## Too slow? Try reading the multipage copy of

1. Let *path* be the *Path* object created in step 8.
2. Let *width* be the maximum width of all the subpaths in *path*.
3. Run *alignPath* on *path* with *width* as the *maxWidth* argument, and, if the *maxWidth* argument is not null, with the resulting alignment value.

4. Let *width* be the aggregate length of all the subpaths in *path*, including the distances from the last point of each closed subpath to the first point of that subpath.
5. Define *L* to be a linear coordinate line for all the subpaths in *path*, with additional lines drawn between the last point and the first point of each closed subpath, such that the first point of the first subpath is defined as point 0, and the last point of the last subpath, if the last subpath is not closed, or the second occurrence first point of that subpath, if it is closed, is defined as point *width*.

6. Let *offset* be determined according to the appropriate step below:

↳ If *physical alignment* is *left*

Let *offset* be zero.

↳ If *physical alignment* is *right*

Let *offset* be *width*.

↳ If *physical alignment* is *center*

Let *offset* be half of *width*.

7. Move all the shapes in *glyphs* to the right by *offset* CSS pixels.

8. For each glyph *glyph* in the *glyphs* array, run these substeps:

1. Let *dx* be the x-coordinate of the horizontal center of the bounding box of the shape described by *glyph*, in CSS pixels.

2. If *dx* is negative or greater than *width*, skip the remainder of these substeps for this glyph.

3. Recast *dx* to coordinate spaces units in *path*. (This just changes the dimensionality of *dx*, not its numeric value.)

4. Find the point *p* on *path* (or implied closing lines in *path*) that corresponds to the position *dx* on the coordinate line *L*.

5. Let *θ* be the clockwise angle from the positive x-axis to the side of the line that is tangential to *path* at the point *p* that is going in the same direction as the line at point *p*.

6. Rotate the shape described by *glyph* clockwise by *θ* about the point that is at the *dx* coordinate horizontally and the zero coordinate vertically.

7. Let  $(x, y)$  be the coordinate of the point *p*.

8. Move the shape described by *glyph* to the right by *x* and down by *y*.

9. Let *glyph subpaths* be a list of subpaths describing the shape given in *glyph*, with each CSS pixel in the coordinate space of *glyph* mapped to one coordinate space unit in *glyph subpaths*. Subpaths in *glyph subpaths* must wind clockwise, regardless of how the user agent's font subsystem renders fonts and regardless of how the fonts themselves are defined.

10. If the method is `addPathBy strokingText()`, replace *glyph subpaths* by the result of tracing *glyph subpaths*, using the *CanvasDrawingStyles* object argument for the line styles.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

13. Add all the subpaths in *glyph subpaths* to *target*.9. Create a new subpath in the `Path` object with  $(x_{final}, y_{final})$  as the only point in the subpath.

#### 4.8.11.2.8 Transformations

Each `CanvasRenderingContext2D` object has a *current transformation matrix*, as well as methods (described in this section) to manipulate it. When a `CanvasRenderingContext2D` object is created, its transformation matrix must be initialized to the identity transform.

The transform objects, on the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

**Note:**

**Note:** The transform objects, on the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

**Note:**

**Note:** Followed by a rotation transformation that rotates drawing operations by a quarter turn, and a rectangle twice as wide as it is tall is then drawn on the canvas, the actual result will be a square.

This box is non-normative. Implementation requirements are given below this box.

`context.currentTransform [ = value ]`

Returns the transformation matrix, as an `SVGMatrix` object.

Can be set, to change the transformation matrix.

`context.scale(x, y)`

Changes the transformation matrix to apply a scaling transformation with the given characteristics.

`context.rotate(angle)`

Changes the transformation matrix to apply a rotation transformation with the given characteristics. The angle is in radians.

`context.translate(x, y)`

Changes the transformation matrix to apply a translation transformation with the given characteristics.

`context.transform(a, b, c, d, e, f)`

Changes the transformation matrix to apply the matrix given by the arguments as described below.

`context.setTransform(a, b, c, d, e, f)`

Changes the transformation matrix to the matrix given by the arguments as described below.

`context.resetTransform()`

Changes the transformation matrix to the identity transform.

The `currentTransform`, on getting, must return the last object that it was set to. On setting, its value must be changed to the new value, and the transformation matrix must be updated to match the matrix described by the new value. When the `CanvasRenderingContext2D` object is created, the `currentTransform` attribute must be set a newly created `SVGMatrix` object. When the transformation matrix is mutated by the methods described in this section, the last `SVGMatrix` object to which the attribute has been set must be mutated in a corresponding fashion.

The `scale(x, y)` method must add the scaling transformation described by the arguments to the transformation matrix. The `x` argument represents the scale factor in the horizontal direction and the `y` argument represents the scale factor in the vertical direction. The factors are multiples.

The `rotate(angle)` method must add the rotation transformation described by the argument to the transformation matrix. The `angle` argument represents a clockwise rotation angle expressed in radians.

The `translate(x, y)` method must add the translation transformation described by the arguments to the transformation matrix. The

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
a c e
b d f
0 0 1
```

**Note:** The arguments `a`, `b`, `c`, `d`, `e`, and `f` are sometimes called `m11`, `m12`, `m21`, `m22`, `dx`, and `dy` or `m11`, `m21`, `m12`, `m22`, `dx`, and `dy`. Care should be taken in particular with the order of the second and third arguments (`b` and `c`) as their order varies from API to API and APIs sometimes use the notation `m12/m21` and sometimes `m21/m12` for those positions.



The `setTransform` method takes a `Matrix` object as its argument, and then invoke the `transform` method on the element with that matrix.

The `resetTransform` method takes no arguments and removes any transformation applied by the `setTransform` method.

#### 4.8.11.2.9 In <http://whatwg.org/html>

Several methods take a `CanvasImageSource` object as an argument.

This union type allows objects implementing any of the following interfaces to be used as image sources:

- `HTMLImageElement` (`img` elements)
- `HTMLVideoElement` (`video` elements)
- `HTMLCanvasElement` (`canvas` elements)
- `CanvasRenderingContext2D`
- `ImageBitmap`

When a user agent is required to check the usability of the `image` argument, where `image` is a `CanvasImageSource` object, the user agent must run these steps, which return either `good`, `bad`, or `aborted`:

1. If the `image` argument is an `HTMLImageElement` object that is not fully decodable, or if the `image` argument is an `HTMLVideoElement` object whose `readyState` attribute is either `HAVE NOTHING` or `HAVE_METADATA`, then return `bad` and abort these steps.
2. If the `image` argument is an `HTMLCanvasElement` object with either a horizontal dimension or a vertical dimension equal to zero, then the implementation throw an `InvalidStateError` exception and return `aborted`.

When a `CanvasImageSource` object represents an `HTMLImageElement`, the element's image must be used as the source image.

Specifically, when a `CanvasImageSource` object represents an animated image in an `HTMLImageElement`, the user agent must use the poster frame of the animation, or, if there is no poster frame, the first frame of the animation, when rendering the image for `CanvasRenderingContext2D` APIs.

When a `CanvasImageSource` object represents an `HTMLVideoElement`, then the frame at the current playback position when the `getFrameForImage` method with the `image` argument is invoked must be used as the source image when rendering the image for `CanvasRenderingContext2D` APIs, and the source image's dimensions must be the intrinsic width and intrinsic height of the media resource (i.e. after any aspect-ratio correction has been applied).

When a `CanvasImageSource` object represents an `HTMLCanvasElement`, the element's bitmap must be used as the source image.

When a `CanvasImageSource` object represents a `CanvasRenderingContext2D`, the object's scratch bitmap must be used as the source image.

When a `CanvasImageSource` object represents an element that is being rendered and that element has been resized, the original image data of the source image must be used, not the image as it is rendered (e.g. `width` and `height` attributes on the source element have no effect on how the object is interpreted when rendering the image for `CanvasRenderingContext2D` APIs).

When a `CanvasImageSource` object represents an `ImageBitmap`, the object's bitmap image data must be used as the source image.

The `image` argument is not origin-clean if it is an `HTMLImageElement` or `HTMLVideoElement` whose origin is not the same as the origin of the document.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

This box is non-normative. Implementation requirements are given below this box.

**context.fillStyle [ = value ]**

Returns the current style used for filling shapes.

Can be set, to change the fill style.

The style can be either a string containing a CSS color, or a `CanvasGradient` or `CanvasPattern` object. Invalid values are ignored.



**context.fillStyle** Too slow? Try reading the multipage copy of  
 Re: the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

rn object. Invalid values

The `fillStyle` attribute represents the color or style to use inside shapes, and the `strokeStyle` attribute represents the color or style to use for the lines around the shapes.

Both attributes can be either strings, `CanvasGradients`, or `CanvasPatterns`. On setting, strings must be parsed as CSS `<color>` values and the color assigned, and `CanvasGradient` and `CanvasPattern` objects must be assigned themselves. [CSSCOLOR] If the value is a string but cannot be parsed as a CSS `<color>` value, then it must be ignored, and the attribute must retain its previous value. If the new value is a `CanvasPattern` object that is marked as not origin-clean, then the scratch bitmap's origin-clean flag must be set to false.

When set to a `CanvasPattern` or `CanvasGradient` object, the assignment is live, meaning that changes made to the object after the assignment do affect subsequent stroking or filling of shapes.

On getting, if the value is a color, then the serialization of the color must be returned. Otherwise, if it is not a color but a `CanvasGradient` or `CanvasPattern`, then the respective object must be returned. (Such objects are opaque and therefore only useful for assigning to other attributes or for comparison to other gradients or patterns.)

The **serialization of a color** for a color value is a string, computed as follows: if it has alpha equal to 1.0, then the string is a lowercase six-digit hex value, prefixed with a "#" character (U+0023 NUMBER SIGN), with the first two digits representing the red component, the next two digits representing the green component, and the last two digits representing the blue component, the digits being lowercase ASCII hex digits. Otherwise, the color value has alpha less than 1.0, and the string is the color value in the CSS `rgba()` functional-notation format: the literal string `rgba` (U+0072 U+0067 U+0062 U+0061) followed by a U+0028 LEFT PARENTHESIS, a base-ten integer in the range 0-255 representing the red component (using ASCII digits in the shortest form possible), a literal U+002C COMMA and U+0020 SPACE, an integer for the green component, a comma and a space, an integer for the blue component, another comma and space, a U+0030 DIGIT ZERO, if the alpha value is greater than zero then a U+002E FULL STOP (representing the decimal point), if the alpha value is greater than zero then one or more ASCII digits representing the fractional part of the alpha, and finally a U+0029 RIGHT PARENTHESIS. User agents must express the fractional part of the alpha value, if any, with the level of precision necessary for the alpha value, when reparsed, to be interpreted as the same alpha value.

When the context is created, the `fillStyle` and `strokeStyle` attributes must initially have the string value `#000000`.

When the value is a color, it must not be affected by the transformation matrix when used to draw on bitmaps.

There are two types of gradients, linear gradients and radial gradients, both represented by objects implementing the opaque `CanvasGradient` interface.

Once a gradient has been created (see below), stops are placed along it to define how the colors are distributed along the gradient. The color of the gradient at each stop is the color specified for that stop. Between each such stop, the colors and the alpha component must be linearly interpolated over the RGBA space without premultiplying the alpha value to find the color to use at that offset. Before the first stop, the color must be the color of the first stop. After the last stop, the color must be the color of the last stop. When there are no stops, the gradient is transparent black.

This box is non-normative. Implementation requirements are given below this box.

**gradient.addColorStop(offset, color)**

Adds a color stop with the given color to the gradient at the given offset. 0.0 is the offset at one end of the gradient, 1.0 is

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

**2012** **2013** **2014**

▼ About this capture

gradient-context : create linear gradient(x, y, z, t);

Returns a `CanvasGradient` object that represents a linear gradient that paints along the line given by the coordinates represented by the arguments.

```
gradient = context.createRadialGradient(x0, y0, r0, x1, y1, r1)
```

Returns a `CanvasGradient` object that represents a radial gradient that paints along the cone given by the circles represented by the arguments.

If either of the radii are negative, throws an `IndexSizeError` exception.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

| gradient. If the *offset* is less  
d as a CSS <color> value,  
offset *offset* relative to the  
added at the same offset on a  
each subsequent one  
ach point to be ignored).

The `createLinearGradient(x0, y0, x1, y1)` method takes four arguments that represent the start point ( $x0, y0$ ) and end point ( $x1, y1$ ) of the gradient. The method must return a linear `CanvasGradient` initialized with the specified line.

Linear gradients must be rendered such that all points on a line perpendicular to the line that crosses the start and end points have the color at the point where those two lines cross (with the colors coming from the interpolation and extrapolation described above). The points in the linear gradient must be transformed as described by the current transformation matrix when rendering.

If  $x0 = x1$  and  $y0 = y1$ , then the linear gradient must paint nothing.

The `createRadialGradient(x0, y0, r0, x1, y1, r1)` method takes six arguments, the first three representing the start circle with origin  $(x0, y0)$  and radius  $r0$ , and the last three representing the end circle with origin  $(x1, y1)$  and radius  $r1$ . The values are in coordinate space units. If either of  $r0$  or  $r1$  are negative, an `IndexSizeError` exception must be thrown. Otherwise, the method must return a radial `CanvasGradient` initialized with the two specified circles.

Radial gradients must be rendered by following these steps:

1. If  $x_0 = x_1$  and  $y_0 = y_1$  and  $r_0 = r_1$ , then the radial gradient must paint nothing. Abort these steps.

2. Let  $x(\omega) = (x_1 - x_0)\omega + x_0$

$$\text{Let } y(\omega) = (y_1 - y_0)\omega + y_0$$

$$\text{Let } r(\omega) = (r_1 - r_0)\omega + r_0$$

Let the color at  $\omega$  be the color at that position on the gradient (with the colors coming from the interpolation and extrapolation described above).

3. For all values of  $\omega$  where  $r(\omega) > 0$ , starting with the value of  $\omega$  nearest to positive infinity and ending with the value of  $\omega$  nearest to negative infinity, draw the circumference of the circle with radius  $r(\omega)$  at position  $(x(\omega), y(\omega))$ , with the color at  $\omega$ , but only painting on the parts of the bitmap that have not yet been painted on by earlier circles in this step for this rendering of the gradient.

**Note:** This effectively creates a cone, touched by the two circles defined in the creation of the gradient, with the part of the cone before the start circle (0.0) using the color of the first offset, the part of the cone after the end circle (1.0) using the color of the last offset, and areas outside the cone untouched by the gradient (transparent black).

The resulting radial gradient must then be transformed as described by the current transformation matrix when rendering.

Gradients must be painted only where the relevant stroking or filling effects require that they be drawn.

Patterns are represented by objects implementing the opaque `CanvasPattern` interface.

*This box is non-normative. Implementation requirements are given below this box.*

```
pattern = context.createPattern(image, repetition)
```

Returns a `CanvasPattern` object that uses the given image and repeats in the direction(s) given by the `repetition`.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

values, throws a `SyntaxError` exception. If the image isn't yet fully decoded, then the method returns null.

#### `pattern.setTransform(transform)`

Sets the transformation matrix that will be used when rendering the pattern during a fill or stroke painting operation.

To create objects of this type, the `createPattern(image, repetition)` method is used. When the method is invoked, the user agent must run the following steps:

1. Let **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**
2. Check if `repetition` is `repeat`, `repeat-x`, or `repeat-y`. If so, continue. Otherwise, set `repetition` to `repeat`.
3. If `repetition` is `repeat`, then
  - a. If `image` is `null`, then throw a `SyntaxError` exception and abort these steps.
  - b. Create a new `CanvasPattern` object with the image `image` and the repetition behavior given by `repetition`.
  - c. If the `image` argument is not origin-clean, then mark the `CanvasPattern` object as **not origin-clean**.
  - d. Return the `CanvasPattern` object.

own and the method doesn't see it returns *good*; continue

-repeat", throw a

Modifying the `image` used when creating a `CanvasPattern` object after calling the `createPattern()` method must not affect the pattern(s) rendered by the `CanvasPattern` object.

Patterns have a transformation matrix, which controls how the pattern is used when it is painted. Initially, a pattern's transformation matrix must be the identity transform.

When the `setTransform()` method is invoked on the pattern, the user agent must replace the pattern's transformation matrix with the one described by the `SVGMatrix` object provided as an argument to the method.

When a pattern is to be rendered within an area, the user agent must run the following steps to determine what is rendered:

1. Create an infinite transparent black bitmap.
2. Place a copy of the image on the bitmap, anchored such that its top left corner is at the origin of the coordinate space, with one coordinate space unit per CSS pixel of the image, then place repeated copies of this image horizontally to the left and right, if the repetition behavior is "`repeat-x`", or vertically up and down, if the repetition behavior is "`repeat-y`", or in all four directions all over the bitmap, if the repetition behavior is "`repeat`".

If the original image data is a bitmap image, the value painted at a point in the area of the repetitions is computed by filtering the original image data. If the `imageSmoothingEnabled` attribute is set to true, then the user agent may use any filtering algorithm (for example bilinear interpolation or nearest-neighbor). If the `imageSmoothingEnabled` attribute is set to false, the image must be rendered using nearest-neighbor interpolation. When such a filtering algorithm requires a pixel value from outside the original image data, it must instead use the value from wrapping the pixel's coordinates to the original image's dimensions. (That is, the filter uses 'repeat' behavior, regardless of the value of the pattern's repetition behavior.)

3. Transform the resulting bitmap according to the pattern's transformation matrix.
4. Transform the resulting bitmap again, this time according to the current transformation matrix.
5. Replace any part of the image outside the area in which the pattern is to be rendered with transparent black.
6. The resulting bitmap is what is to be rendered, with the same origin and same scale.

If a radial gradient or repeated pattern is used when the transformation matrix is singular, the resulting style must be transparent black (otherwise the gradient or pattern would be collapsed to a point or line, leaving the other pixels undefined). Linear gradients and solid colors always define all points even with singular transformation matrices.

#### 4.8.11.2.11 Drawing rectangles to the bitmap

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Shapes are painted without affecting the current default path, and are subject to the clipping region, and, with the exception of `clearRect()`, also shadow effects, global alpha, and global composition operators.

This box is non-normative. Implementation requirements are given below this box.

`context.clearRect(x, y, w, h)`

Clears all pixels on the bitmap in the given rectangle to transparent black.



`context`

Pai Too slow? Try reading the multipage copy of

context the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Pai /web/20130103030411/http:

The `clearRect`

1. Let `pixels` be the set of pixels in the specified rectangle that also intersect the current clipping region.
2. Clear the pixels in `pixels` to a fully transparent black, erasing any previous image.
3. Clear regions that cover the pixels in `pixels` on the scratch bitmap.

**Note:** If either height or width are zero, this method has no effect, since the set of pixels would be empty.

The `fillRect(x, y, w, h)` method must paint the specified rectangular area using the `fillStyle`. If either height or width are zero, this method has no effect.

The `strokeRect(x, y, w, h)` method must take the result of tracing the path described below, using the `CanvasRenderingContext2D` object's line styles, and fill it with the `strokeStyle`.

If both `w` and `h` are zero, the path has a single subpath with just one point (`x, y`), and no lines, and this method thus has no effect (the trace a path algorithm returns an empty path in that case).

If just one of either `w` or `h` is zero, then the path has a single subpath consisting of two points, with coordinates (`x, y`) and (`x+w, y+h`), in that order, connected by a single straight line.

Otherwise, the path has a single subpath consisting of four points, with coordinates (`x, y`), (`x+w, y`), (`x+w, y+h`), and (`x, y+h`), connected to each other in that order by straight lines.

#### 4.8.11.2.12 Drawing text to the bitmap

This box is non-normative. Implementation requirements are given below this box.

`context.fillText(text, x, y [, maxWidth] )`

`context.strokeText(text, x, y [, maxWidth] )`

Fills or strokes (respectively) the given text at the given position. If a maximum width is provided, the text will be scaled to fit that width if necessary.

`metrics = context.measureText(text)`

Returns a `TextMetrics` object with the metrics of the given text in the current font.

`metrics.width`

`metrics.actualBoundingBoxLeft`

`metrics.actualBoundingBoxRight`

`metrics.fontBoundingBoxAscent`

`metrics.fontBoundingBoxDescent`

`metrics.actualBoundingBoxAscent`

`metrics.actualBoundingBoxDescent`

`metrics.emHeightAscent`

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

---

Go

**NOV JAN FEB**

◀ 03 ▶

2012 2013 2014



▼ About this capture

Returns the measurement described below

The `CanvasRenderingContext2D` interface provides the following methods for rendering text:

The `fillText()` and `strokeText()` methods take three or four arguments, `text`, `x`, `y`, and optionally `maxWidth`, and render the given `text` at the given `(x, y)` coordinates ensuring that the text isn't wider than `maxWidth` if specified, using the current `font`, `textAlign`, and `textBaseline` values. Specifically, when the methods are called, the user agent must run the following steps:

1. Run was **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>**  
1, if the *maxWidth* argument
  2. Move **/web/20130103030411/http://whatwg.org/html**  
:SS pixel in the coordinate
  3. Pair spa  
For **For**  
reverse holds: `strokeStyle` must be applied to the result of tracing the shapes using the `CanvasRenderingContext2D` object for the line styles, and `fillStyle` must be ignored.  
d. For `strokeText()`, the
  - These shapes are painted without affecting the current path, and are subject to shadow effects, global alpha, the clipping region, and global composition operators.
  4. If the text preparation algorithm used a font that has an origin that is not the same as the entry script's origin (even if "using a font" means just checking if that font has a particular glyph in it before falling back to another font), then set the scratch bitmap's origin-clean flag to false.

The `measureText()` method takes one argument, `text`. When the method is invoked, the user agent must run the text preparation algorithm, passing it `text` and the `CanvasRenderingContext2D` object, and then using the returned inline box must create a new `TextMetrics` object with its attributes set as described in the following list. If doing these measurements requires using a font that has an origin that is not the same as that of the `Document` object that owns the `canvas` element (even if "using a font" means just checking if that font has a particular glyph in it before falling back to another font), then the method must throw a `SecurityError` exception. Otherwise, it must return the new `TextMetrics` object. [CSS]

*width* attribute

The width of that inline box, in CSS pixels. (The text's advance width.)



#### *actualBoundingBoxLeft* attribute

The distance parallel to the baseline from the alignment point given by the `textAlign` attribute to the left side of the bounding rectangle of the given text, in CSS pixels; positive numbers indicating a distance going left from the given alignment point.

**Note:** The sum of this value and the next (`actualBoundingBoxRight`) can be wider than the width of the inline box (`width`), in particular with slanted fonts where characters overhang their advance width.

#### *actualBoundingBoxRight* attribute

The distance parallel to the baseline from the alignment point given by the `textAlign` attribute to the right side of the bounding rectangle of the given text, in CSS pixels; positive numbers indicating a distance going right from the given alignment point.

#### *fontBoundingBoxAscent* attribute

The distance from the horizontal line indicated by the `textBaseline` attribute to the top of the highest bounding rectangle of all the fonts used to render the text, in CSS pixels; positive numbers indicating a distance going up from the given baseline.

**Note:** This value and the next are useful when rendering a background that must have a consistent height even if the exact text being rendered changes. The `actualBoundingBoxAscent` attribute (and its corresponding attribute for the descent) are useful when drawing a bounding box around specific text.

#### *fontBoundingBoxDescent* attribute

The distance from the horizontal line indicated by the `textBaseline` attribute to the bottom of the lowest bounding rectangle of all the fonts used to render the text, in CSS pixels; positive numbers indicating a distance going down from the given

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*Note: This number can vary greatly based on the input text, even if the first font specified covers all the characters in the input. For example, the `actualBoundingBoxAscent` of a lowercase "o" from an alphabetic baseline would be less than that of an uppercase "F". The value can easily be negative; for example, the distance from the top of the em box (`textBaseline` value "top") to the top of the bounding rectangle when the given text is just a single comma "," would likely (unless the font is quite unusual) be negative.*

#### `actualBoundingBoxDescent` attribute

The `actualBoundingBoxDescent` attribute gives the bounding rectangle of the line box, in CSS pixels; positive numbers indicating that the given baseline is below the bottom of the em square in the line box, in CSS pixels; negative numbers indicating that the given baseline is above the bottom of the em square (so this value will usually be negative). (Zero if the given baseline is the bottom of the em square.)

#### `emHeight` attribute

The distance from the horizontal line indicated by the `textBaseline` attribute to the bottom of the em square in the line box, in CSS pixels; positive numbers indicating that the given baseline is below the bottom of the em square (so this value will usually be negative). (Zero if the given baseline is the top of the em square.)

#### `hangingBaseline` attribute

The distance from the horizontal line indicated by the `textBaseline` attribute to the hanging baseline of the line box, in CSS pixels; positive numbers indicating that the given baseline is below the hanging baseline. (Zero if the given baseline is the hanging baseline.)

#### `alphabeticBaseline` attribute

The distance from the horizontal line indicated by the `textBaseline` attribute to the alphabetic baseline of the line box, in CSS pixels; positive numbers indicating that the given baseline is below the alphabetic baseline. (Zero if the given baseline is the alphabetic baseline.)

#### `ideographicBaseline` attribute

The distance from the horizontal line indicated by the `textBaseline` attribute to the ideographic baseline of the line box, in CSS pixels; positive numbers indicating that the given baseline is below the ideographic baseline. (Zero if the given baseline is the ideographic baseline.)

*Note: Glyphs rendered using `fillText()` and `strokeText()` can spill out of the box given by the font size (the em square size) and the width returned by `measureText()` (the text width). Authors are encouraged to use the bounding box values described above if this is an issue.*

*Note: A future version of the 2D context API may provide a way to render fragments of documents, rendered using CSS, straight to the canvas. This would be provided in preference to a dedicated way of doing multiline layout.*

#### 4.8.11.2.13 Drawing paths to the canvas

The context always has a **current default path**. There is only one current default path, it is not part of the drawing state. The current default path is a path, as described above.

*This box is non-normative. Implementation requirements are given below this box.*

`context.beginPath()`

Resets the current default path.

`context.fill()`

`context.fill(path)`

Fills the subpaths of the current default path or the given path with the current fill style.

`context.stroke()`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

`context.drawSystemFocusRing(path, element)`

If the given element is focused, draws a focus ring around the current default path or the given path, following the platform conventions for focus rings.

`shouldDraw = context.drawCustomFocusRing(element)``shouldDraw = context.drawCustomFocusRing(path, element)`

If the given element is focused, and the user has configured his system to draw focus rings in a particular manner (for example, high contrast focus rings), draws a focus ring around the current default path or the given path and returns false.

Other

`draw` Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

`Sc` /web/20130103030411/http:  
the //whatwg.org/html

`context``context.clip(path)`

Further constrains the clipping region to the current default path or the given path.

`context.resetClip()`

Unconstraints the clipping region.

`context.isPointInPath(x, y)``context.isPointInPath(path, x, y)`

Returns true if the given point is in the current default path or the given path.

h small screens, where

The `beginPath()` method must empty the list of subpaths in the context's current default path so that it once again has zero subpaths.

Where the following method definitions use the term *intended path*, it means the `Path` argument, if one was provided, or the current default path otherwise.

When the intended path is a `Path` object, the coordinates and lines of its subpaths must be transformed according to the `CanvasRenderingContext2D` object's current transformation matrix when used by these methods (without affecting the `Path` object itself). When the intended path is the current default path, it is not affected by the transform. (This is because transformations already affect the current default path when it is constructed, so applying it when it is painted as well would result in a double transformation.)

The `fill()` method must fill all the subpaths of the intended path, using `fillStyle`, and using the non-zero winding number rule. Open subpaths must be implicitly closed when being filled (without affecting the actual subpaths).

**Note:** Thus, if two overlapping but otherwise independent subpaths have opposite windings, they cancel out and result in no fill. If they have the same winding, that area just gets painted once.

The `stroke()` method must trace the intended path, using the `CanvasRenderingContext2D` object for the line styles, and then fill the combined stroke area using the `strokeStyle` attribute.

**Note:** As a result of how the algorithm to trace a path is defined, overlapping parts of the paths in one stroke operation are treated as if their union was what was painted.

**Note:** The stroke style is affected by the transformation during painting, even if the intended path is the current default path.

Paths, when filled or stroked, must be painted without affecting the current default path or any `Path` objects, and must be subject to shadow effects, global alpha, the clipping region, and global composition operators. (The effect of transformations is described above and varies based on which path is being used.)

Zero-length line segments must be pruned before stroking a path. Empty subpaths must be ignored.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2. If the user has requested the use of particular focus rings (e.g. high-contrast focus rings), or if the `element` would have a focus ring drawn around it, then draw a focus ring of the appropriate style along the intended path, following platform conventions.

**Note:** Some platforms only draw focus rings around elements that have been focused from the keyboard, and not those focused from the mouse. Other platforms simply don't draw focus rings around some elements at all unless relevant accessibility features are enabled. This API is intended to follow these conventions. User agents that implement distinctions based on the manner in which the element was focused are encouraged to classify focus driven by the `focus()` method based on the kind of user interaction event from which the call was triggered (if any).

The sub-  
3. Opt  
If th  
The Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ition operators, but should be  
on which path is being used.)

ay wait until the next time

#### Otherwise

Add instructions to the scratch bitmap's list of pending interface actions that inform the user that the focus is at the location of the bitmap given by the intended path.

The `drawCustomFocusRing(element)` method, when invoked, must run the following steps:

1. If `element` is not focused or is not a descendant of the element with whose context the method is associated, then return false and abort these steps.
2. Let `result` be true.
3. If the user has requested the use of particular focus rings (e.g. high-contrast focus rings), then draw a focus ring of the appropriate style along the intended path, and set `result` to false.

The focus ring should not be subject to the shadow effects, the global alpha, or the global composition operators, but should be subject to the clipping region. (The effect of transformations is described above and varies based on which path is being used.)

4. Optionally, run the appropriate step from the following list:

#### If the `CanvasRenderingContext2D` object's context bitmap mode is fixed

Inform the user that the focus is at the location given by the intended path. The user agent may wait until the next time the event loop reaches its "update the rendering" step to optionally inform the user.

#### Otherwise

Add instructions to the scratch bitmap's list of pending interface actions that inform the user that the focus is at the location of the bitmap given by the intended path.

5. Return `result`.

The `scrollPathIntoView()` method, when invoked, if the `CanvasRenderingContext2D` object's context bitmap mode is fixed, must run the following steps; and otherwise, must add instructions to the scratch bitmap's list of pending interface actions that run the following steps:

1. Let `the specified rectangle` be the rectangle of the bounding box of the intended path.
2. Let `notional child` be a hypothetical element that is a rendered child of the `canvas` element whose dimensions are those of `the specified rectangle`.
3. Scroll `notional child` into view with the `align to top` flag set.
4. Optionally, inform the user that the caret or selection (or both) cover `the specified rectangle` of the canvas. If the `CanvasRenderingContext2D` object's context bitmap mode was fixed when the method was invoked, the user agent may wait until the next time the event loop reaches its "update the rendering" step to optionally inform the user.

**Note:** "Inform the user", as used in this section, could mean calling a system accessibility API, which would notify assistive technologies such as magnification tools. To properly drive magnification based on a focus change, a system accessibility API driving a screen magnifier needs the bounds for the newly focused object. The methods

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

The `clip()` method must create a new **clipping region** by calculating the intersection of the current clipping region and the area described by the intended path, using the non-zero winding number rule. Open subpaths must be implicitly closed when computing the clipping region, without affecting the actual subpaths. The new clipping region replaces the current clipping region.

When the context is initialized, the clipping region must be set to the rectangle with the top left corner at (0,0) and the width and height of the coordinate space.

The `resetClip()` method must create a new clipping region that is the rectangle with the top left corner at (0,0) and the width and height of the coordinate space. The new clipping region replaces the current clipping region.

The `isPointInPath()` method, when treated as coordinates non-zero winding number either of the

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>**

This is

```
<...>
<label><input type=checkbox id=showA> Show As</label>
<label><input type=checkbox id=showB> Show Bs</label>
<!-- ... -->
</canvas>
<script>
 function drawCheckbox(context, element, x, y, paint) {
 context.save();
 context.font = '10px sans-serif';
 context.textAlign = 'left';
 context.textBaseline = 'middle';
 var metrics = context.measureText(element.labels[0].textContent);
 if (paint) {
 context.beginPath();
 context.strokeStyle = 'black';
 context.rect(x-5, y-5, 10, 10);
 context.stroke();
 if (element.checked) {
 context.fillStyle = 'black';
 context.fill();
 }
 context.fillText(element.labels[0].textContent, x+5, y);
 }
 context.beginPath();
 context.rect(x-7, y-7, 12 + metrics.width+2, 14);
 if (paint && context.drawCustomFocusRing(element)) {
 context.strokeStyle = 'silver';
 context.stroke();
 }
 context.restore();
 }
 function drawBase() { /* ... */ }
 function drawAs() { /* ... */ }
 function drawBs() { /* ... */ }
 function redraw() {
 var canvas = document.getElementsByTagName('canvas')[0];
 var context = canvas.getContext('2d');
 context.clearRect(0, 0, canvas.width, canvas.height);
 drawCheckbox(context, document.getElementById('showA'), 20, 40, true);
 drawCheckbox(context, document.getElementById('showB'), 20, 60, true);
 drawBase();
 if (document.getElementById('showA').checked)
 drawAs();
 if (document.getElementById('showB').checked)
 drawBs();
 }
}
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
var y = event.clientY;
var node = event.target;
while (node) {
 x -= node.offsetLeft - node.scrollLeft;
 y -= node.offsetTop - node.scrollTop;
 node = node.offsetParent;
}
drawCheckbox(context, document.getElementById('showA'), 20, 40, false);
if (context.isPointInPath(x, y))
 .se);
 .redraw, true);
document.getElementsByTagName('canvas')[0].addEventListener('blur', redraw, true);
document.getElementsByTagName('canvas')[0].addEventListener('change', redraw, true);
document.getElementsByTagName('canvas')[0].addEventListener('click', processClick, false);
redraw();
</script>
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 4.8.11.2.14 Drawing images

To draw images, the `drawImage` method can be used.

This method can be invoked with three different sets of arguments:

- `drawImage(image, dx, dy)`
- `drawImage(image, dx, dy, dw, dh)`
- `drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)`

*This box is non-normative. Implementation requirements are given below this box.*

```
context.drawImage(image, dx, dy)
context.drawImage(image, dx, dy, dw, dh)
context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)
```

Draws the given image onto the canvas. The arguments are interpreted as follows:

Go

NOV JAN FEB

03

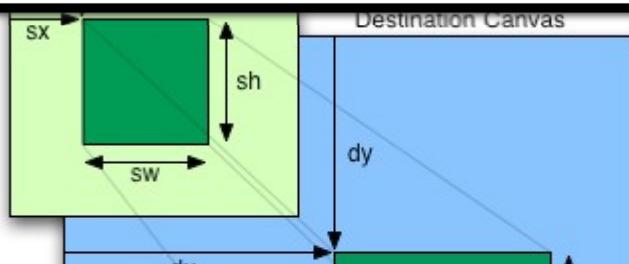
2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If the image has no image data, throws an `InvalidStateError` exception. If the image isn't yet fully decoded, then nothing is drawn.

When the `drawImage()` method is invoked, the user agent must run the following steps:

1. Check the usability of the `image` argument. If this returns `aborted`, then an exception has been thrown and the method doesn't return anything; abort these steps. If it returns `bad`, then abort these steps without drawing anything. Otherwise it returns `good`; continue with these steps.
2. Establish the source and destination rectangles as follows:

If not specified, the `dw` and `dh` arguments must default to the values of `sw` and `sh`, interpreted such that one CSS pixel in the image is treated as one unit in the scratch bitmap's coordinate space. If the `sx`, `sy`, `sw`, and `sh` arguments are omitted, they must default to `0, 0`, the image's intrinsic width in image pixels, and the image's intrinsic height in image pixels, respectively. If the image has no intrinsic dimensions, the `concrete object size` must be used instead, as determined using the CSS "Concrete Object Size Resolution" algorithm, with the `specified size` having neither a definite width nor height, nor any additional constraints, the object's intrinsic properties being those of the `image` argument, and the `default object size` being the size of the scratch bitmap. [CSSIMAGES]

The source rectangle is the rectangle whose corners are the four points  $(sx, sy), (sx+sw, sy), (sx+sw, sy+sh), (sx, sy+sh)$ .

The destination rectangle is the rectangle whose corners are the four points  $(dx, dy), (dx+dw, dy), (dx+dw, dy+dh), (dx, dy+dh)$ .

When the source rectangle is outside the source image, it must be clipped to the source image, and the destination rectangle must be clipped in the same proportion.

3. If one of the `sw` or `sh` arguments is zero, abort these steps. Nothing is painted.
4. Paint the region of the `image` argument specified by the source rectangle on the region of the rendering context's scratch bitmap specified by the destination rectangle, after applying the current transformation matrix to the points of the destination rectangle.

The image data must be processed in the original direction, even if the dimensions given are negative.

If the `imageSmoothingEnabled` attribute is set to true, then the user agent should attempt to apply a smoothing algorithm to the image data when it is scaled. Otherwise, the image must be rendered using nearest-neighbor interpolation.

**Note:** This specification does not define the precise algorithm to use when scaling an image when the `imageSmoothingEnabled` attribute is set to true.

**Note:** When a `canvas` or `CanvasRenderingContext2D` object is drawn onto itself, the drawing model requires the source to be copied before the image is drawn, so it is possible to copy parts of a `canvas` or

Go

NOV JAN FEB

03



2012 2013 2014

▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the filtering algorithm requires a pixel value from outside the original image data, it must instead use the value from the nearest edge pixel. (That is, the filter uses 'clamp-to-edge' behavior.) When the filtering algorithm requires a pixel value from outside the source rectangle but inside the original image data, then the value from the original image data must be used.

**Note:** Thus, scaling an image in parts or in whole will have the same effect. This does mean that when sprites coming from a single sprite sheet are to be scaled, adjacent images in the sprite sheet can interfere. This can be avoided by ensuring each sprite in the sheet is surrounded by a border of transparent black, or by copying sprites to be scaled into temporary canvas elements and drawing the scaled sprites from there.

Image  
glob Too slow? Try reading the multipage copy of  
5. If th the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

A hit region //whatwg.org/html

Each hit region.

- A set of pixels on the bitmap for which this region is responsible.
- A bounding circumference on the bitmap that surrounds the hit region's set of pixels as they stood when it was created.
- Optionally, a non-empty string representing an ID for distinguishing the region from others.
- Optionally, a reference to another region that acts as the parent for this one.
- A count of regions that have this one as their parent, known as the hit region's child count.
- A cursor specification, in the form of either a CSS cursor value, or the string "inherit" meaning that the cursor of the hit region's parent, if any, or of the canvas element, if not, is to be used instead.
- Optionally, either a control, or an unbacked region description.

A control is just a reference to an Element node, to which, in certain conditions, the user agent will route events, and from which the user agent will determine the state of the hit region for the purposes of accessibility tools. (The control is ignored when it is not a descendant of the canvas element.)

An unbacked region description consists of the following:

- Optionally, a label.
- An ARIA role, which, if the unbacked region description also has a label, could be the empty string.

This box is non-normative. Implementation requirements are given below this box.

#### context.addHitRegion(options)

Adds a hit region to the bitmap. The argument is an object with the following members:

##### path (default null)

A Path object that describes the pixels that form part of the region. If this member is not provided or is set to null, the current default path is used instead.

##### id (default empty string)

The ID to use for this region. This is used in MouseEvent events on the canvas (event.region) and as a way to reference this region in later calls to addHitRegion().

##### parentID (default null)

The ID of the parent region, for purposes of navigation by accessibility tools and for cursor fallback.

##### cursor (default "inherit")

The cursor to use when the mouse is over this region. The value "inherit" means to use the cursor for the parent region (as specified by the parentID member), if any, or to use the canvas element's cursor if the region has no parent.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

label (default null)

A text label for accessibility tools to use as a description of this region, if there is no control.

**role (default null)**

An ARIA role for accessibility tools to use to determine how to represent this region, if there is no control.

Hit regions can be used for a variety of purposes:

- With an ID, they can make hit detection easier by having the user agent check which region the mouse is over and

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

clicks on a canvas to

by touch on a mobile

it cursors, with the user

**context.removeHitRegion(options)**

Removes a hit region (and all its ancestors) from the canvas bitmap. The argument is the ID of a region added using `addHitRegion()`.

The pixels that were covered by this region and its descendants are effectively cleared by this operation, leaving the regions non-interactive. In particular, regions that occupied the same pixels before the removed regions were added, overlapping them, do not resume their previous role.

A hit region *A* is an **ancestor region** of a hit region *B* if *B* has a parent and its parent is either *A* or another hit region for which *A* is an ancestor region.

**The region identified by the ID *ID* in a bitmap *bitmap*** is the value returned by the following algorithm (which can return a hit region or nothing):

- If *ID* is null, return nothing and abort these steps.
- Let *list* be the hit region list associated with *bitmap*.
- If there is a hit region in *list* whose ID is a case-sensitive match for *ID*, then return that hit region and abort these steps.
- Otherwise, return nothing.

**The region representing the control *control* for a bitmap *bitmap*** is the value returned by the following algorithm (which can return a hit region or nothing):

- Let *list* be the hit region list associated with *bitmap*.
- If there is a hit region in *list* whose control is *control*, then return that hit region and abort these steps.
- Otherwise, return nothing.

**The control represented by a region *region* for a canvas element *ancestor*** is the value returned by the following algorithm (which can return an element or nothing):

- If *region* has no control, return nothing and abort these steps.
- Let *control* be *region*'s control.
- If *control* is not a descendant of *ancestor*, then return nothing and abort these steps.
- Otherwise, return *control*.

**The cursor for a hit region *region* of a canvas element *ancestor*** is the value returned by the following algorithm:

- Loop:** If *region* has a cursor specification other than "inherit", then return that hit region's cursor specification and abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**[3,445 captures](#)**

5 Jun 2004 - 22 Aug 2020

nothing):

1. Let *list* be the hit region list associated with *bitmap*.
2. If there is a hit region in *list* whose set of pixels contains *pixel*, then return that hit region and abort these steps.
3. Otherwise, return nothing.

To **clear regions that cover the pixels** *pixels* on a bitmap *bitmap*, the user agent must run the following steps:

1. Let *list* be the hit region list associated with *bitmap*.
2. Remove **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**
3. Garbage collect *bitmap*.
4. To **garbage collect**: /web/20130103030411/http://whatwg.org/html
5. Let *loop* be a step labeled *loop*.
6. Let *victim* be the hit region in *list* whose set of pixels contains *pixel*, if any. If there is no such hit region, abort these steps.
7. If *victim* has a parent, then decrement that hit region's child count by one.
8. Remove *victim* from *list*.
9. Jump back to the step labeled *loop*.

**Note:** Adding a new region and calling `clearRect()` are the two ways this clearing algorithm can be invoked. The hit region list itself is also reset when the rendering context is reset, e.g. when a `CanvasRenderingContext2D` object is bound to or unbound from a canvas, or when the dimensions of the bitmap are changed.

When the `addHitRegion()` method is invoked, the user agent must run the following steps:

1. Let *arguments* be the dictionary object provided as the method's argument.
2. If the *arguments* object's `path` member is not null, let *source path* be the `path` member's value. Otherwise, let it be the `CanvasRenderingContext2D` object's current default path.
3. Transform all the coordinates and lines in *source path* by the current transform matrix, if the *arguments* object's `path` member is not null.
4. If the *arguments* object's `id` member is the empty string, let it be null instead.
5. If the *arguments* object's `id` member is not null, then let *previous region for this ID* be the region identified by the ID given by the `id` member's value in this scratch bitmap, if any. If the `id` member is null or no such region currently exists, let *previous region for this ID* be null.
6. If the *arguments* object's `parent` member is the empty string, let it be null instead.
7. If the *arguments* object's `parent` member is not null, then let *parent region* be the region identified by the ID given by the `parent` member's value in the scratch bitmap, if any. If the `parent` member is null or no such region currently exists, let *parent region* be null.
8. If the *arguments* object's `label` member is the empty string, let it be null instead.
9. If any of the following conditions are met, throw a `NotSupportedError` exception and abort these steps.
  - o The *arguments* object's `control` and `label` members are both non-null.
  - o The *arguments* object's `control` and `role` members are both non-null.
  - o The *arguments* object's `role` member's value is the empty string, and the `label` member's value is either null or the empty string.
  - o The `path source path` describes a shape with no pixels.
  - o The *arguments* object's `control` member is not null but is neither an `a` element that represents a hyperlink, a `button` element, an `input` element whose `type` attribute is in one of the Checkbox or Radio Button states, nor an `input` element that is a button.
  - o The *parent region* is not null but has a control.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

11. If any of the following conditions are met, throw a `SyntaxError` exception and abort these steps.

- o The `arguments` object's `cursor` member is not null but is neither an ASCII case-insensitive match for the string "inherit", nor a valid CSS 'cursor' property value. [CSSUI]
- o The `arguments` object's `role` member is not null but its value is not an ordered set of unique space-separated tokens whose tokens are all case-sensitive matches for names of non-abstract WAI-ARIA roles. [ARIA]

12. Let `region` be a newly created hit region, with its information configured as follows:**Hit region's set of pixels**The pixels contained in `sourcePath`.

**Hit** **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>**

st case, this can just be the faces.)

**Hit** **/web/20130103030411/http://whatwg.org/html**

region has no id.

**Hit** **i**If `parent region` is not null: `parent region`. Otherwise, `region` has no parent.**Hit region's child count**

Initially zero.

**Hit region's cursor specification**If `parent region` is not null: `parent region`. Otherwise, `region` has no parent.**Hit region's control**If the `arguments` object's `control` member is not null: the value of the `control` member. Otherwise, `region` has no control.**Hit region's label**If the `arguments` object's `label` member is not null: the value of the `label` member. Otherwise, `region` has no label.**Hit region's ARIA role**If the `arguments` object's `role` member is not null: the value of the `role` member (which might be the empty string). Otherwise, if the `arguments` object's `label` member is not null: the empty string. Otherwise, `region` has no ARIA role.13. If the `arguments` object's `cursor` member is not null, then act as if a CSS rule for the `canvas` element setting its 'cursor' property had been seen, whose value was the hit region's cursor specification.

**Note:** For example, if the user agent prefetches cursor values, this would cause that to happen in response to an appropriately-formed `addHitRegion()` call.

14. If the `arguments` object's `control` member is not null, then let `previous region for the control` be the region representing the control given by the `control` member's value for this scratch bitmap, if any. If the `control` member is null or no such region currently exists, let `previous region for the control` be null.15. If there is a `previous region with this control`, remove it from the scratch bitmap's hit region list; then, if it had a parent region, decrement that hit region's child count by one.16. If there is a `previous region with this ID`, remove it, and all hit regions for which it is an ancestor region, from the scratch bitmap's hit region list; then, if it had a parent region, decrement that hit region's child count by one.17. If there is a `parent region`, increment its hit region's child count by one.18. Clear regions that cover the pixels in `region`'s set of pixels on this scratch bitmap.19. Add `region` to the scratch bitmap's element's hit region list.When the `removeHitRegion()` method is invoked, the user agent must run the following steps:

1. Let `region` be the region identified by the ID given by the method's argument in the rendering context's scratch bitmap. If no such region currently exists, abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `MouseEvent` interface is extended to support hit regions:

```
partial interface MouseEvent {
 readonly attribute DOMString? region;
};

partial dictionary MouseEventInit {
 DOMString? region;
};
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Links to the multipage copy are given below this box.

If the `region` attribute on `MouseEvent` objects must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the hit region's ID if the mouse was over a hit region when the event was fired.

When a `MouseEvent` is to be fired at a `canvas` element by the user agent in response to a pointing device action, if the `canvas` element has a hit region list, the user agent must instead follow these steps. If these steps say to *act as normal*, that means that the event must be fired as it would have had these requirements not been applied.

1. If the pointing device is not indicating a pixel on the `canvas`, act as normal and abort these steps.
2. Let `pixel` be the pixel indicated by the pointing device.
3. Let `region` be the hit region that is the region for the pixel `pixel` on this `canvas` element's bitmap, if any.
4. If there is no `region`, then act as normal and abort these steps.
5. Let `id` be the `region`'s ID, if any.
6. If there is an `id`, then initialize the event object's `region` attribute to `id`.
7. Let `control` be the control represented by `region` for this `canvas` element, if any.
8. If there is a `control`, then target the event object at `control` instead of the `canvas` element.
9. Continue dispatching the event, but with the updated event object and target as given in the above steps.

When a user's pointing device cursor is positioned over a `canvas` element, user agents should render the pointing device cursor according to the cursor specification described by the cursor for the hit region that is the region for the pixel that the pointing device designates on the `canvas` element's bitmap.

User agents are encouraged to make use of the information present in a `canvas` element's hit region list to improve the accessibility of `canvas` elements.

Each hit region should be handled in a fashion equivalent to a node in a virtual DOM tree rooted at the `canvas` element. The hierarchy of this virtual DOM tree must match the hierarchy of the hit regions, as described by the parent of each region. Regions without a parent must be treated as children of the `canvas` element for the purpose of this virtual DOM tree. For each node in such a DOM tree, the hit region's bounding circumference gives the region of the screen to use when representing the node (if appropriate).

The semantics of a hit region for the purposes of this virtual DOM tree are those of the the control represented by the region, if it has one, or else of a non-interactive element whose ARIA role, if any, is that given by the hit region's ARIA role, and whose textual representation, if any, is given by the hit region's label.

For the purposes of accessibility tools, when an element C is a descendant of a `canvas` element and there is a region representing the control C for that `canvas` element's bitmap, then the element's position relative to the document should be presented as if it was that region in the `canvas` element's virtual DOM tree.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Note: This, for instance, a user agent on a touch screen device could provide haptic feedback when the user crosses over a hit region's bounding circumference, and then read the hit region's label to the user. Similarly, a desktop user agent with a virtual accessibility focus separate from the keyboard input focus could allow the user to navigate through the hit regions, using the virtual DOM tree described above to enable hierarchical navigation. When an interactive control inside the canvas element is focused, if the control has a corresponding region, then that hit region's bounding circumference could be used to determine what area of the display to magnify.

#### 4.8.11.2.16 Pixel manipulation



ts are given below this box.

**imagedata** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

**imagedata** /web/20130103030411/http://whatwg.org/html/  
Ref: transparent black.  
trans. turned object are

**imagedata = context.createImageData(sw, sh)**

Returns an `ImageData` object whose dimensions equal the dimensions given in the arguments, multiplied by the number of pixels in the canvas bitmap that correspond to each coordinate space unit. All the pixels in the returned object are transparent black.

**imagedata = context.getImageData(sx, sy, sw, sh)**

Returns an `ImageData` object containing the image data for the given rectangle of the bitmap.

Throws an `IndexSizeError` exception if the either of the width or height arguments are zero.

The data will be returned with one pixel of image data for each coordinate space unit on the canvas (ignoring transforms).

**imagedata = context.getImageDataHD(sx, sy, sw, sh)**

Returns an `ImageData` object containing the image data for the given rectangle of the bitmap.

Throws an `IndexSizeError` exception if the either of the width or height arguments are zero.

The data will be returned at the same resolution as the canvas bitmap.

**imagedata.width**

**imagedata.height**

Returns the actual dimensions of the data in the `ImageData` object, in pixels. For objects returned by the non-HD variants of the methods in this API, this will correspond to the dimensions given to the methods. For the HD variants, the number of pixels might be different than the number of corresponding coordinate space units.

**imagedata.data**

Returns the one-dimensional array containing the data in RGBA order, as integers in the range 0 to 255.

**context.putImageData(imagedata, dx, dy [, dirtyX, dirtyY, dirtyWidth, dirtyHeight ])**

Paints the data from the given `ImageData` object onto the bitmap. If a dirty rectangle is provided, only the pixels from that rectangle are painted.

The `globalAlpha` and `globalCompositeOperation` attributes, as well as the shadow attributes, are ignored for the purposes of this method call; pixels in the canvas are replaced wholesale, with no composition, alpha blending, no shadows, etc.

Throws a `NotSupportedError` exception if any of the arguments are not finite.

Each pixel in the image data is mapped to one coordinate space unit on the bitmap.

**context.putImageDataHD(imagedata, dx, dy [, dirtyX, dirtyY, dirtyWidth, dirtyHeight ])**

Paints the data from the given `ImageData` object onto the bitmap, at the bitmap's native pixel density. If a dirty rectangle is provided, only the pixels from that rectangle are painted.

The `globalAlpha` and `globalCompositeOperation` attributes, as well as the shadow attributes, are ignored for the purposes of this method call; pixels in the canvas are replaced wholesale, with no composition, alpha blending, no shadows, etc.

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

▼ About this capture

When the `createImageData()` method is invoked with two arguments `sw` and `sh`, it must return a new `ImageData` object representing a rectangle with a width in equal to the absolute magnitude of `sw` and a height equal to the absolute magnitude of `sh`. When invoked with a single `imagedata` argument, it must return a new `ImageData` object representing a rectangle with the same dimensions as the `ImageData` object passed as the argument. The `ImageData` object returned must be filled with transparent black.

When the `createImageDataHD()` method is invoked (with its two arguments `sw` and `sh`) it must return a new `ImageData` object representing a rectangle with a width in equal to the absolute magnitude of `sw` multiplied by `scale` and a height equal to the absolute magnitude of `sh` multiplied by `scale`, where `scale` is the number of pixels in the scratch bitmap per coordinate space units. The `ImageData` object returned by `createImageDataHD()` has a `width` property with a value equal to the width of the rectangle represented by the `ImageData` object and a `height` property with a value equal to the height of the rectangle represented by the `ImageData` object.

The `getImage` exception; o must return the rectangle units. If the I returned abj the scratch t

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

In `IndexSizeError` or exception; otherwise, it  $i$  of that bitmap denoted by map's coordinate space  $e$  of each pixel in the coordinate. Pixels outside  $sha$  values

The `getImageDataHD(sx, sy, sw, sh)` method must, if either the `sw` or `sh` arguments are zero, throw an `IndexSizeError` exception; otherwise, if the scratch bitmap's origin-clean flag is set to false, it must throw a `SecurityError` exception; otherwise, it must return an `ImageData` object with width `sw` multiplied by `scale` and height `sh` multiplied by `scale` representing the scratch bitmap for the area of that bitmap denoted by the rectangle whose corners are the four points  $(sx, sy), (sx+sw, sy), (sx+sw, sy+sh), (sx, sy+sh)$ , in the bitmap's coordinate space units. Pixels outside the scratch bitmap must be returned as transparent black. Pixels must be returned as non-premultiplied alpha values.

New `ImageData` objects must be initialized so that their `width` attribute is set to the number of pixels per row in the image data, their `height` attribute is set to the number of rows in the image data, and their `data` attribute is initialized to a `Uint8ClampedArray` object. The `Uint8ClampedArray` object must use a Canvas Pixel `ArrayBuffer` for its storage, and must have a zero start offset and a length equal to the length of its storage, in bytes. The Canvas Pixel `ArrayBuffer` must contain the image data. At least one pixel's worth of image data must be returned. [TYPEDARRAY]

**A Canvas Pixel ArrayBuffer** is an `ArrayBuffer` that whose data is represented in left-to-right order, row by row top to bottom, starting with the top left, with each pixel's red, green, blue, and alpha components being given in that order for each pixel. Each component of each pixel represented in this array must be in the range 0..255, representing the 8 bit value for that component. The components must be assigned consecutive indices starting with 0 for the top left pixel's red component. [TYPEDARRAY]

The `putImageData()` and `putImageDataHD()` methods write data from `ImageData` structures back to the rendering context's scratch bitmap. Their arguments are: `imagedata`, `dx`, `dy`, `dirtyX`, `dirtyY`, `dirtyWidth`, and `dirtyHeight`.

When the last four arguments to these methods are omitted, they must be assumed to have the values 0, 0, the `width` member of the `imagedata` structure, and the `height` member of the `imagedata` structure, respectively.

When invoked, these methods must act as follows

1. If *dirtyWidth* is negative, let *dirtyX* be *dirtyX+dirtyWidth*, and let *dirtyWidth* be equal to the absolute magnitude of *dirtyWidth*.  
If *dirtyHeight* is negative, let *dirtyY* be *dirtyY+dirtyHeight*, and let *dirtyHeight* be equal to the absolute magnitude of *dirtyHeight*.
  2. If *dirtyX* is negative, let *dirtyWidth* be *dirtyWidth+dirtyX*, and let *dirtyX* be zero.  
If *dirtyY* is negative, let *dirtyHeight* be *dirtyHeight+dirtyY*, and let *dirtyY* be zero.
  3. If *dirtyX+dirtyWidth* is greater than the *width* attribute of the *imagedata* argument, let *dirtyWidth* be the value of that *width* attribute, minus the value of *dirtyX*.  
If *dirtyY+dirtyHeight* is greater than the *height* attribute of the *imagedata* argument, let *dirtyHeight* be the value of that *height* attribute, minus the value of *dirtyY*.
  4. If, after those changes, either *dirtyWidth* or *dirtyHeight* is negative or zero, stop these steps without affecting any bitmaps.
  5. Run the appropriate steps from the following list:

↳ If the method was `putImageData()`

Draw the region of the image data in the horizontal rectangle whose top left corner is at (*dirtyX*,*dirtyY*) and whose bottom right corner is at (*dirtyX*+*dirtyWidth*,*dirtyY*+*dirtyHeight*) onto the rendering context's scratch bitmap, aligned

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

### ↳ If the method was `putImageDataHD()`

1. Let  $dx_{device}$  be the x-coordinate of the device pixel in the scratch bitmap corresponding to the  $dx$  coordinate in the scratch bitmap's coordinate space.

Let  $dy_{device}$  be the y-coordinate of the device pixel in the scratch bitmap corresponding to the  $dy$  coordinate in the scratch bitmap's coordinate space.

2. For all integer values of  $x$  and  $v$  where  $dirtyX \leq x < dirtyX + dirtyWidth$  and  $dirtyY \leq y < dirtyY + dirtyHeight$ , set the  $alpha$  structure to the pixel with

## Too slow? Try reading the multipage copy of

The handling specification  
The spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

space is not defined by this

context

...for any value  $w$  and  $h$ ; and exception

```
context.createImageData(w, h);
context.getImageData(0, 0, w, h);
```

...must return `ImageData` objects with the same dimensions as each other, and the following two calls:

```
context.createImageDataHD(w, h);
context.getImageDataHD(0, 0, w, h);
```

...must also return `ImageData` objects with the same dimensions as each other, for any value of  $w$  and  $h$  in both cases. In other words, while user agents may round the arguments of these methods so that they map to device pixel boundaries, any rounding performed must be performed consistently for all of the methods described in this section.

**Note: Due to the lossy nature of converting to and from premultiplied alpha color values, pixels that have just been set using `putImageDataHD()` might be returned to an equivalent `getImageDataHD()` as different values.**

The current path, transformation matrix, shadow attributes, global alpha, the clipping region, and global composition operator must not affect the methods described in this section.

In the following example, the script generates an `ImageData` object so that it can draw onto it.

```
// canvas is a reference to a <canvas> element
var context = canvas.getContext('2d');

// create a blank slate
var data = context.createImageDataHD(canvas.width, canvas.height);

// create some plasma
FillPlasma(data, 'green'); // green plasma

// add a cloud to the plasma
AddCloud(data, data.width/2, data.height/2); // put a cloud in the middle

// paint the plasma+cloud on the canvas
context.putImageDataHD(data, 0, 0);

// support methods
function FillPlasma(data, color) { ... }
function AddCloud(data, x, y) { ... }
```

Here is an example of using `getImageDataHD()` and `putImageDataHD()` to implement an edge detection filter.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Edge detection demo</title>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

 }

function demo() {
 var canvas = document.getElementsByTagName('canvas')[0];
 var context = canvas.getContext('2d');

 // draw the image onto the canvas
 context.drawImage(image, 0, 0);

 // get the image data to manipulate
 var input = document.createElement('input');
 var output = document.createElement('output');

 input.type = 'file';
 output.type = 'arraybuffer';

 input.onchange = function() {
 var reader = new FileReader();
 reader.onload = function(e) {
 var arrayBuffer = e.target.result;
 var data = new Uint8Array(arrayBuffer);
 var w = input.width, h = input.height;
 var inputData = input.data;
 var outputData = output.data;

 // edge detection
 for (var y = 1; y < h-1; y += 1) {
 for (var x = 1; x < w-1; x += 1) {
 for (var c = 0; c < 3; c += 1) {
 var i = (y*w + x)*4 + c;
 outputData[i] = 127 + -inputData[i - w*4 - 4] - inputData[i - w*4] -
 inputData[i - w*4 + 4] +
 -inputData[i - 4] + 8*inputData[i] -
 inputData[i + 4] +
 -inputData[i + w*4 - 4] - inputData[i + w*4] -
 inputData[i + w*4 + 4];
 }
 outputData[(y*w + x)*4 + 3] = 255; // alpha
 }
 }
 }
 }

 // put the image data back after manipulation
 context.putImageDataHD(output, 0, 0);
}

</script>
</head>
<body onload="init()">
 <canvas></canvas>
</body>
</html>
```

#### 4.8.11.2.17 Compositing

This box is non-normative. Implementation requirements are given below this box.

**context.globalAlpha [= value]**

Returns the current alpha value applied to rendering operations.

Can be set, to change the alpha value. Values outside of the range 0.0 .. 1.0 are ignored.

**context.globalCompositeOperation [= value]**

Returns the current composition operation, from the list below.

Can be set, to change the composition operation. Unknown values are ignored.

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the context is created, the `globalAlpha` attribute must initially have the value 1.0.

The `globalCompositeOperation` attribute sets how shapes and images are drawn onto the scratch bitmap, once they have had `globalAlpha` and the current transformation matrix applied. It must be set to a value from the following list. In the descriptions below, the source image, *A*, is the shape or image being rendered, and the destination image, *B*, is the current state of the scratch bitmap.

`source-atop`

*A* atop *B*. *Display the source image wherever both images are opaque. Display the destination image wherever the destination image is opaque but the source image is transparent. Discard transparency elsewhere.*

`source-in`

*A* in *B*. *Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org*

`source-out`

*A* out *B*. *Same as source-in but using the destination image instead of the source image and vice versa.*

`source-over`

*A* over *B*. *Display the source image wherever the source image is opaque. Display the destination image elsewhere.*

`destination-atop`

*B* atop *A*. *Same as source-atop but using the destination image instead of the source image and vice versa.*

`destination-in`

*B* in *A*. *Same as source-in but using the destination image instead of the source image and vice versa.*

`destination-out`

*B* out *A*. *Same as source-out but using the destination image instead of the source image and vice versa.*

`destination-over`

*B* over *A*. *Same as source-over but using the destination image instead of the source image and vice versa.*

`lighter`

*A* plus *B*. *Display the sum of the source image and destination image, with color values approaching 255 (100%) as a limit.*

`copy`

*A* (*B* is ignored). *Display the source image instead of the destination image.*

`xor`

*A* xor *B*. *Exclusive OR of the source image and destination image.*

`vendorName-operationName`

Vendor-specific extensions to the list of composition operators should use this syntax.

The operators in the above list must be treated as described by the Porter-Duff operator given at the start of their description (e.g. *A* over *B*). They are to be applied as part of the drawing model, at which point the clipping region is also applied. (Without a clipping region, these operators act on the whole bitmap with every operation.) [PORTERDUFF]

These values are all case-sensitive — they must be used exactly as shown. User agents must not recognize values that are not a case-sensitive match for one of the values given above.

On setting, if the user agent does not recognize the specified value, it must be ignored, leaving the value of `globalCompositeOperation` unaffected.

When the context is created, the `globalCompositeOperation` attribute must initially have the value `source-over`.

#### 4.8.11.2.18 Image smoothing

This box is non-normative. Implementation requirements are given below this box.

`context.imageSmoothingEnabled [= value]`

Returns whether pattern fills and the `drawImage()` method will attempt to smooth images if they have to rescale them (as

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

The `imageSmoothingEnabled` attribute, on getting, must return the last value it was set to. On setting, it must be set to the new value. When the `CanvasRenderingContext2D` object is created, the attribute must be set to true.

#### 4.8.11.2.19 Shadows

All drawing operations are affected by the four global shadow attributes.

This box is non-normative. Implementation elements are given below this box.

`context``Ref`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>.

`context``context``Ref`

<https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`Ref`

Can be set, to change the shadow offset. Values that are not finite numbers are ignored.

`context.shadowBlur [= value]`

Returns the current level of blur applied to shadows.

Can be set, to change the blur level. Values that are not finite numbers greater than or equal to zero are ignored.

The `shadowColor` attribute sets the color of the shadow.

When the context is created, the `shadowColor` attribute initially must be fully-transparent black.

On getting, the serialization of the color must be returned.

On setting, the new value must be parsed as a CSS `<color>` value and the color assigned. If the value cannot be parsed as a CSS `<color>` value then it must be ignored, and the attribute must retain its previous value. [CSSCOLOR]

The `shadowOffsetX` and `shadowOffsetY` attributes specify the distance that the shadow will be offset in the positive horizontal and positive vertical distance respectively. Their values are in coordinate space units. They are not affected by the current transformation matrix.

When the context is created, the shadow offset attributes must initially have the value 0.

On getting, they must return their current value. On setting, the attribute being set must be set to the new value, except if the value is infinite or NaN, in which case the new value must be ignored.

The `shadowBlur` attribute specifies the level of the blurring effect. (The units do not map to coordinate space units, and are not affected by the current transformation matrix.)

When the context is created, the `shadowBlur` attribute must initially have the value 0.

On getting, the attribute must return its current value. On setting the attribute must be set to the new value, except if the value is negative, infinite or NaN, in which case the new value must be ignored.

**Shadows are only drawn if** the opacity component of the alpha component of the color of `shadowColor` is non-zero and either the `shadowBlur` is non-zero, or the `shadowOffsetX` is non-zero, or the `shadowOffsetY` is non-zero.

It is likely that this will change: browser vendors have indicated an interest in changing the processing model for shadows such that they only draw when the composition operator is "source-over" (the default). Read more...

When shadows are drawn, they must be rendered as follows:

1. Let A be an infinite transparent black bitmap on which the source image for which a shadow is being created has been rendered.
2. Let B be an infinite transparent black bitmap, with a coordinate space and an origin identical to A.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

1. Let  $\sigma$  be half the value of `shadowBlur`.
2. Perform a 2D Gaussian Blur on  $B$ , using  $\sigma$  as the standard deviation.

User agents may limit values of  $\sigma$  to an implementation-specific maximum value to avoid exceeding hardware limitations during the Gaussian blur operation.

5. Set the red, green, and blue components of every pixel in  $B$  to the red, green, and blue components (respectively) of the color of `shadowColor`.

6. Multi-

Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If the current

color.

7. The

te the shadow).

If the current

/web/20130103030411/http:

4.8.11.2.20 i

//whatwg.org/html

When a sha

y do):

1. Render the shape or image onto an infinite transparent black bitmap, creating image  $A$ , as described in the previous sections. For shapes, the current fill, stroke, and line styles must be honored, and the stroke must itself also be subjected to the current transformation matrix.

2. When shadows are drawn, render the shadow from image  $A$ , using the current shadow styles, creating image  $B$ .

3. When shadows are drawn, multiply the alpha component of every pixel in  $B$  by `globalAlpha`.

4. When shadows are drawn, composite  $B$  within the clipping region over the current scratch bitmap using the current composition operator.

5. Multiply the alpha component of every pixel in  $A$  by `globalAlpha`.

6. Composite  $A$  within the clipping region over the current scratch bitmap using the current composition operator.

#### 4.8.11.2.21 Best practices

This section is non-normative.

When a canvas is interactive, authors should include focusable elements in the element's fallback content corresponding to each focusable part of the canvas, as in the example above.

To indicate which focusable part of the canvas is currently focused, authors should use the `drawSystemFocusRing()` method, passing it the element for which a ring is being drawn. This method only draws the focus ring if the element is focused, so that it can simply be called whenever drawing the element, without checking whether the element is focused or not first.

Authors should avoid implementing text editing controls using the `canvas` element. Doing so has a large number of disadvantages:

- Mouse placement of the caret has to be reimplemented.
- Keyboard movement of the caret has to be reimplemented (possibly across lines, for multiline text input).
- Scrolling of the text field has to be implemented (horizontally for long lines, vertically for multiline input).
- Native features such as copy-and-paste have to be reimplemented.
- Native features such as spell-checking have to be reimplemented.
- Native features such as drag-and-drop have to be reimplemented.
- Native features such as page-wide text search have to be reimplemented.
- Native features specific to the user, for example custom text services, have to be reimplemented. This is close to impossible since each user might have different services installed, and there is an unbounded set of possible such services.
- Bidirectional text editing has to be reimplemented.
- For multiline text editing, line wrapping has to be implemented for all relevant languages.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- Platform-native input method editors (IMEs) have to be reimplemented.
- Undo and redo functionality has to be reimplemented.
- Accessibility features such as magnification following the caret or selection have to be reimplemented.

This is a huge amount of work, and authors are most strongly encouraged to avoid doing any of it by instead using the `input` element, the `textarea` element, or the `contenteditable` attribute.



#### 4.8.11.2.22 /

*This section*

Here

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```

var context = document.getElementById('myCanvas').getContext('2d');

var lastX = context.canvas.width * Math.random();
var lastY = context.canvas.height * Math.random();
var hue = 0;
function line() {
 context.save();
 context.translate(context.canvas.width/2, context.canvas.height/2);
 context.scale(0.9, 0.9);
 context.translate(-context.canvas.width/2, -context.canvas.height/2);
 context.beginPath();
 context.lineWidth = 5 + Math.random() * 10;
 context.moveTo(lastX, lastY);
 lastX = context.canvas.width * Math.random();
 lastY = context.canvas.height * Math.random();
 context.bezierCurveTo(context.canvas.width * Math.random(),
 context.canvas.height * Math.random(),
 context.canvas.width * Math.random(),
 context.canvas.height * Math.random(),
 lastX, lastY);

 hue = hue + 10 * Math.random();
 context.strokeStyle = 'hsl(' + hue + ', 50%, 50%)';
 context.shadowColor = 'white';
 context.shadowBlur = 10;
 context.stroke();
 context.restore();
}
setInterval(line, 50);

function blank() {
 context.fillStyle = 'rgba(0,0,0,0.1)';
 context.fillRect(0, 0, context.canvas.width, context.canvas.height);
}
setInterval(blank, 40);

</script>
```

#### 4.8.11.3 Pixel density

The user agent may use any square pixel density for the bitmaps of a `canvas` and its rendering contexts. Once a `canvas` has a bitmap, that canvas must keep its resolution for its lifetime.

**Note:** In general, user agents are encouraged to use a pixel density equal to the screen pixel density. Ideally, the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

pixel density.

```
partial interface Screen {
 readonly attribute double canvasResolution;
};
```



This box is non-normative. Implementation requirements are given below this box.

`window.screen.canvasResolution`

Ret

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 4.8.11.4 C

coordinate space units, that [CSSOMVIEW]

The `canvas` APIs must perform color correction at only two points: when rendering images with their own gamma correction and color space information onto a bitmap, to convert the image to the color space used by the bitmaps (e.g. using the 2D Context's `drawImage()` method with an `HTMLImageElement` object), and when rendering the actual canvas bitmap to the output device.

**Note:** Thus, in the 2D context, colors used to draw shapes onto the canvas will exactly match colors obtained through the `getImageDataHD()` method.

The `toDataURL()` and `toDataURLHD()` methods must not include color space information in the resources they return. Where the output format allows it, the color of pixels in resources created by `toDataURL()` must match those returned by the `getImageData()` method, and the color of pixels in resources created by `toDataURLHD()` must match those returned by the `getImageDataHD()` method.

In user agents that support CSS, the color space used by a `canvas` element must match the color space used for processing any colors for that element in CSS.

The gamma correction and color space information of images must be handled in such a way that an image rendered directly using an `img` element would use the same colors as one painted on a `canvas` element that is then itself rendered. Furthermore, the rendering of images that have no color correction information (such as those returned by the `toDataURL()` method) must be rendered with no color correction.

**Note:** Thus, in the 2D context, calling the `drawImage()` method to render the output of the `toDataURLHD()` method to the `canvas`, given the appropriate dimensions, has no visible effect.

#### 4.8.11.5 Serializing bitmaps to a file

When a user agent is to create a **serialization of the bitmap as a file**, optionally with some given `arguments`, and optionally with a `native` flag set, it must create an image file in the format given by the first value of `arguments`, or, if there are no `arguments`, in the PNG format. [PNG]

If the `native` flag is set, or if the bitmap has one pixel per coordinate space unit, then the image file must have the same pixel data (before compression, if applicable) as the bitmap, and if the file format used supports encoding resolution metadata, the resolution of that bitmap (device pixels per coordinate space units being interpreted as image pixels per CSS pixel) must be given as well.

Otherwise, the image file's pixel data must be the bitmap's pixel data scaled to one image pixel per coordinate space unit, and if the file format used supports encoding resolution metadata, the resolution must be given as 96dpi (one image pixel per CSS pixel).

If `arguments` is not empty, the first value must be interpreted as a MIME type giving the format to use. If the type has any parameters, it must be treated as not supported.

For example, the value "image/png" would mean to generate a PNG image, the value "image/jpeg" would mean to generate a JPEG image, and the value "image/svg+xml" would mean to generate an SVG image (which would require that the user agent track how the bitmap was generated, an unlikely, though potentially awesome, feature).

User agents must support PNG ("image/png"). User agents may support other types. If the user agent does not support the requested

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

background using the source-over operator

If the first argument in *arguments* gives a type corresponding to one of the types given in the first column of the following table, and the user agent supports that type, then the subsequent arguments, if any, must be treated as described in the second cell of that row.

Type	Arguments for serialization methods Other arguments	Reference
image/jpeg	The second argument, if it is a number in the range 0.0 to 1.0 inclusive, must be treated as the desired quality level. If it is not a number or is outside that range, the user agent must use its default value, as if the argument had been omitted.	[JPEG]

For the purpose of  
handling and

For the purpose of handling arguments, try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### **4.8.11.6 S**

*This section is non-normative.*

**Information leakage** can occur if scripts from one origin can access information (e.g. read pixels) from images from another origin (one that isn't the same).

To mitigate this, bitmaps used with canvas elements are defined to have a flag indicating whether they are origin-clean. All bitmaps start with their origin-clean set to true. The flag is set to false when cross-origin images or fonts are used.

The `toDataURL()`, `toDataURLHD()`, `toBlob()`, `getImageData()`, and `getImageDataHD()` methods check the flag and will throw a `SecurityError` exception rather than leak cross-origin data.

The flag can be reset in certain situations; for example, when a `CanvasRenderingContext2D` is bound to a new canvas, the bitmap is cleared and its flag reset.

## 4.8.12 The map element

## Categories:

- Flow content.  
Phrasing content.  
Palpable content.

#### **Contexts in which this element can be used:**

## Where phrasing content is expected

## **Content model:**

Transparent.

## **Content attributes:**

## Global attributes

## DOM interface:

```
IDL interface HTMLMapElement : HTMLElement {
 attribute DOMString name;
 readonly attribute HTMLCollection areas;
 readonly attribute HTMLCollection images
};
```

The `map` element, in conjunction with any `area` element descendants, defines an image map. The element represents its children.

The `name` attribute gives the map a name so that it can be referenced. The attribute must be present and must have a non-empty value with no space characters. The value of the `name` attribute must not be a compatibility-caseless match for the value of the `name` attribute of another `map` element in the same document. If the `id` attribute is also specified, both attributes must have the same value.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[map . Images](#)

Returns an `HTMLCollection` of the `img` and `object` elements that use the `map`.

The `areas` attribute must return an `HTMLCollection` rooted at the `map` element, whose filter matches only `area` elements.

The `images` attribute must return an `HTMLCollection` rooted at the `Document` node, whose filter matches only `img` and `object` elements that are associated with this `map` element according to the image map processing model.

The IDL attri-

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

This example is of a page with

```
<IMG SRC="/images/menu.gif"
 ALT="Babies™ navigation menu. Select a department to go to its page."
 USEMAP="#NAV">
</HEADER>
...
<FOOTER>
<MAP NAME="NAV">
<P>
Clothes
<AREA ALT="Clothes" COORDS="0,0,100,50" HREF="/clothes/"> |
Toys
<AREA ALT="Toys" COORDS="0,0,100,50" HREF="/toys/"> |
Food
<AREA ALT="Food" COORDS="0,0,100,50" HREF="/food/"> |
Books
<AREA ALT="Books" COORDS="0,0,100,50" HREF="/books/">
</MAP>
</FOOTER>
```

#### 4.8.13 The `area` element

##### Categories:

Flow content.  
Phrasing content.

##### Contexts in which this element can be used:

Where phrasing content is expected, but only if there is a `map` element ancestor.

##### Content model:

Empty.

##### Content attributes:

Global attributes  
`alt`  
`coords`  
`shape`  
`href`  
`target`  
`download`  
`ping`  
`rel`  
`hreflang`  
`type`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
attribute DOMString coords;
attribute DOMString shape;
stringifier attribute DOMString href;
attribute DOMString target;
attribute DOMString download;
attribute DOMString ping;
attribute DOMString rel;
readonly attribute DOMTokenList relList;
attribute DOMString hreflang;
attribute DOMString type;
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

};

The `area` element represents either a hyperlink with some text and a corresponding area on an image map, or a dead area on an image map.

If the `area` element has an `href` attribute, then the `area` element represents a hyperlink. In this case, the `alt` attribute must be present. It specifies the text of the hyperlink. Its value must be text that, when presented with the texts specified for the other hyperlinks of the image map, and with the alternative text of the image, but without the image itself, provides the user with the same kind of choice as the hyperlink would when used without its text but with its shape applied to the image. The `alt` attribute may be left blank if there is another `area` element in the same image map that points to the same resource and has a non-blank `alt` attribute.

If the `area` element has no `href` attribute, then the area represented by the element cannot be selected, and the `alt` attribute must be omitted.

In both cases, the `shape` and `coords` attributes specify the area.

The `shape` attribute is an enumerated attribute. The following table lists the keywords defined for this attribute. The states given in the first cell of the rows with keywords give the states to which those keywords map. Some of the keywords are non-conforming, as noted in the last column.

State	Keywords	Notes
Circle state	<code>circle</code>	
	<code>circ</code>	Non-conforming
Default state	<code>default</code>	
Polygon state	<code>poly</code>	
	<code>polygon</code>	Non-conforming
Rectangle state	<code>rect</code>	
	<code>rectangle</code>	Non-conforming

The `shape` attribute may be omitted. The *missing value* `default` is the rectangle state.

The `coords` attribute must, if specified, contain a valid list of integers. This attribute gives the coordinates for the shape described by the `shape` attribute. The processing for this attribute is described as part of the image map processing model.

In the **circle state**, `area` elements must have a `coords` attribute present, with three integers, the last of which must be non-negative. The first integer must be the distance in CSS pixels from the left edge of the image to the center of the circle, the second integer must be the distance in CSS pixels from the top edge of the image to the center of the circle, and the third integer must be the radius of the circle, again in CSS pixels.

In the **default state** state, `area` elements must not have a `coords` attribute. (The area is the whole image.)

In the **polygon state**, `area` elements must have a `coords` attribute with at least six integers, and the number of integers must be even. Each pair of integers must represent a coordinate given as the distances from the left and the top of the image in CSS pixels

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

side, and the distance from the top edge to the bottom side, all in CSS pixels.

When user agents allow users to follow hyperlinks or download hyperlinks created using the `area` element, as described in the next section, the `href`, `target`, `download`, and `ping` attributes decide how the link is followed. The `rel`, `hreflang`, and `type` attributes may be used to indicate to the user the likely nature of the target resource before the user follows the link.

The `target`, `download`, `ping`, `rel`, `hreflang`, and `type` attributes must be omitted if the `href` attribute is not present.

If the `itemprop` is specified on an `area` element, then the `href` attribute must also be specified

The activation

1. If th
2. If th
- the bro
- resl

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

3. Otherwise, the user agent must follow the hyperlink or download the hyperlink created by the `area` element, if any, and as determined by the `download` attribute and any expressed user preference.

/event being dispatched), and lying the rules for choosing a browsing context name, would tion and abort these steps.

The IDL attributes `alt`, `coords`, `href`, `target`, `download`, `ping`, `rel`, `hreflang`, and `type`, each must reflect the respective content attributes of the same name.

The IDL attribute `shape` must reflect the `shape` content attribute.

The IDL attribute `relList` must reflect the `rel` content attribute.

The `area` element also supports the complement of URL decomposition IDL attributes, `protocol`, `host`, `port`, `hostname`, `pathname`, `search`, and `hash`. These must follow the rules given for URL decomposition IDL attributes, with the input being the result of resolving the element's `href` attribute relative to the element, if there is such an attribute and resolving it is successful, or the empty string otherwise; and the common setter action being the same as setting the element's `href` attribute to the new output value.

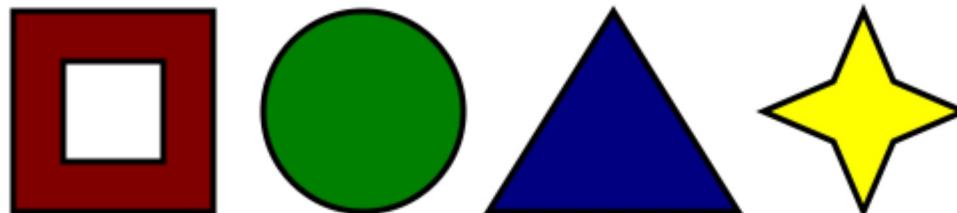
## 4.8.14 Image maps

### 4.8.14.1 Authoring

An **image map** allows geometric areas on an image to be associated with hyperlinks.

An image, in the form of an `img` element or an `object` element representing an image, may be associated with an image map (in the form of a `map` element) by specifying a `usemap` attribute on the `img` or `object` element. The `usemap` attribute, if specified, must be a valid hash-name reference to a `map` element.

Consider an image that looks as follows:



If we wanted just the colored areas to be clickable, we could do it as follows:

```
<p>
Please select a shape:

<area shape=poly coords="450,25,435,60,400,75,435,90,450,125,465,90,500,75,465,60"
 href="yellow.html" alt="Yellow star.">
</map>
</p>
```

#### 4.8.14.2 Processing model

If an `img` elem-

follows:

- Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**
1. Firs  
or n
  2. If th
  3. Oth



agents must process it as

either an element (the `map`)

all.

t those be the `areas`.

Having obtained the list of `area` elements that form the image map (`areas`), interactive user agents must process the list in one of two ways.

If the user agent intends to show the text that the `img` element represents, then it must use the following steps.

**Note: In user agents that do not support images, or that have images disabled, object elements cannot represent images, and thus this section never applies (the fallback content is shown instead). The following steps therefore only apply to `img` elements.**

1. Remove all the `area` elements in `areas` that have no `href` attribute.
2. Remove all the `area` elements in `areas` that have no `alt` attribute, or whose `alt` attribute's value is the empty string, if there is another `area` element in `areas` with the same value in the `href` attribute and with a non-empty `alt` attribute.
3. Each remaining `area` element in `areas` represents a hyperlink. Those hyperlinks should all be made available to the user in a manner associated with the text of the `img`.

In this context, user agents may represent `area` and `img` elements with no specified `alt` attributes, or whose `alt` attributes are the empty string or some other non-visible text, in a user-agent-defined fashion intended to indicate the lack of suitable author-provided text.

If the user agent intends to show the image and allow interaction with the image to select hyperlinks, then the image must be associated with a set of layered shapes, taken from the `area` elements in `areas`, in reverse tree order (so the last specified `area` element in the `map` is the bottom-most shape, and the first element in the `map`, in tree order, is the top-most shape).

Each `area` element in `areas` must be processed as follows to obtain a shape to layer onto the image:

1. Find the state that the element's `shape` attribute represents.
2. Use the rules for parsing a list of integers to parse the element's `coords` attribute, if it is present, and let the result be the `coords` list. If the attribute is absent, let the `coords` list be the empty list.
3. If the number of items in the `coords` list is less than the minimum number given for the `area` element's current state, as per the following table, then the shape is empty; abort these steps.

State	Minimum number of items
Circle state	3
Default state	0
Polygon state	6
Rectangle state	4

4. Check for excess items in the `coords` list as per the entry in the following list corresponding to the `shape` attribute's state:

##### ↪ Circle state

Drop any items in the list beyond the third.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

### ↪ Rectangle state

Drop any items in the list beyond the fourth.

5. If the `shape` attribute represents the rectangle state, and the first number in the list is numerically less than the third number in the list, then swap those two numbers around.

6. If the `shape` attribute represents the rectangle state, and the second number in the list is numerically less than the fourth number in the list, then swap those two numbers around.

7. If the `shape` attribute represents the rectangle state, and the third number in the list is numerically less than the fifth number in the list, then swap those two numbers around.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

8. Now that the shape is a rectangle, if the `shape` attribute is `circle`, then the shape is a circle whose center is  $x$  CSS pixels from the left edge of the image and  $y$  CSS pixels from the top edge of the image, and whose radius is  $r$  pixels.

9. If the `shape` attribute is not `circle`, then the shape is a rectangle that exactly covers the entire image.

### ↪ Default state

The shape is a rectangle that exactly covers the entire image.

### ↪ Polygon state

Let  $x_i$  be the  $(2i)$ th entry in `coords`, and  $y_i$  be the  $(2i+1)$ th entry in `coords` (the first entry in `coords` being the one with index 0).

Let the coordinates be  $(x_i, y_i)$ , interpreted in CSS pixels measured from the top left of the image, for all integer values of  $i$  from 0 to  $(N/2)-1$ , where  $N$  is the number of items in `coords`.

The shape is a polygon whose vertices are given by the coordinates, and whose interior is established using the even-odd rule. [GRAPHICS]

### ↪ Rectangle state

Let  $x_1$  be the first number in `coords`,  $y_1$  be the second number,  $x_2$  be the third number, and  $y_2$  be the fourth number.

The shape is a rectangle whose top-left corner is given by the coordinate  $(x_1, y_1)$  and whose bottom-right corner is given by the coordinate  $(x_2, y_2)$ , those coordinates being interpreted as CSS pixels from the top-left corner of the image.

For historical reasons, the coordinates must be interpreted relative to the *displayed* image after any stretching caused by the CSS 'width' and 'height' properties (or, for non-CSS browsers, the image element's `width` and `height` attributes — CSS browsers map those attributes to the aforementioned CSS properties).

**Note:** Browser zoom features and transforms applied using CSS or SVG do not affect the coordinates.

Pointing device interaction with an image associated with a set of layered shapes per the above algorithm must result in the relevant user interaction events being first fired to the top-most shape covering the point that the pointing device indicated, if any, or to the image element itself, if there is no shape covering that point. User agents may also allow individual `area` elements representing hyperlinks to be selected and activated (e.g. using a keyboard).

**Note:** Because a `map` element (and its `area` elements) can be associated with multiple `img` and `object` elements, it is possible for an `area` element to correspond to multiple focusable areas of the document.

Image maps are live; if the DOM is mutated, then the user agent must act as if it had rerun the algorithms for image maps.

## 4.8.15 MathML

The `math` element from the MathML namespace falls into the embedded content, phrasing content, and flow content categories for the purposes of the content models in this specification.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

purposes of MathML layout and rendering, by an `errorm` element in the MathML namespace containing some appropriate error message.

To enable authors to use MathML tools that only accept MathML in its XML form, interactive HTML user agents are encouraged to provide a way to export any MathML fragment as an XML namespace-well-formed XML fragment.

The semantics of MathML elements are defined by the MathML specification and other applicable specifications. [MATHML]

Here is an example of the use of MathML in an HTML document:

< Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```

<mi>x</mi>
<mo>=</mo>
<mfrac>
 <mrow>
 <mo form="prefix">-</mo> <mi>b</mi>
 <mo>±</mo>
 <msqrt>
 <msup> <mi>b</mi> <mn>2</mn> </msup>
 <mo>-</mo>
 <mn>4</mn> <mo></mo> <mi>a</mi> <mo></mo> <mi>c</mi>
 </msqrt>
 </mrow>
 <mrow>
 <mn>2</mn> <mo></mo> <mi>a</mi>
 </mrow>
</mfrac>
</math>
</p>
</body>
</html>
```

#### 4.8.16 SVG

The `svg` element from the SVG namespace falls into the embedded content, phrasing content, and flow content categories for the purposes of the content models in this specification.

To enable authors to use SVG tools that only accept SVG in its XML form, interactive HTML user agents are encouraged to provide a way to export any SVG fragment as an XML namespace-well-formed XML fragment.

When the SVG `foreignObject` element contains elements from the HTML namespace, such elements must all be flow content. [SVG]

The content model for `title` elements in the SVG namespace inside HTML documents is phrasing content. (This further constrains the requirements given in the SVG specification.)

The semantics of SVG elements are defined by the SVG specification and other applicable specifications. [SVG]

The SVG specification includes requirements regarding the handling of elements in the DOM that are not in the SVG namespace, that are in SVG fragments, and that are not included in a `foreignObject` element. This specification does not define any processing for elements in SVG fragments that are not in the HTML namespace; they are considered neither conforming nor non-conforming from the perspective of this specification.

#### 4.8.17 Dimension attributes

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The specified dimensions given may differ from the dimensions specified in the resource itself, since the resource may have a resolution that differs from the CSS pixel resolution. (On screens, CSS pixels have a resolution of 96ppi, but in general the CSS pixel resolution depends on the reading distance.) If both attributes are specified, then one of the following statements must be true:

- *specified width* - 0.5 ≤ *specified height* \* *target ratio* ≤ *specified width* + 0.5
- *specified height* - 0.5 ≤ *specified width* / *target ratio* ≤ *specified height* + 0.5
- *specified height* = *specified width* = 0

The *target ratio* values of the two attributes of the *width* and *height* attributes are the intrinsic height.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*Note:* [7](#)

**User agent requirements:** User agents are expected to use these attributes as hints for the rendering.

The **width** and **height** IDL attributes on the **iframe**, **embed**, **object**, and **video** elements must reflect the respective content attributes of the same name.

**Note:** For **iframe**, **embed**, and **object** the IDL attributes are **DOMString**; for **video** the IDL attributes are **unsigned long**.

**Note:** The corresponding IDL attributes for **img** and **input** elements are defined in those respective elements' sections, as they are slightly more specific to those elements' other behaviors.

## 4.9 Tabular data

### 4.9.1 The **table** element

#### Categories:

Flow content.  
Palpable content.

#### Contexts in which this element can be used:

Where flow content is expected.

#### Content model:

In this order: optionally a **caption** element, followed by zero or more **colgroup** elements, followed optionally by a **thead** element, followed optionally by a **tfoot** element, followed by either zero or more **tbody** elements or one or more **tr** elements, followed optionally by a **tfoot** element (but there can only be one **tfoot** element child in total).

#### Content attributes:

Global attributes  
**sortable**

#### DOM interface:

```
IDL interface HTMLTableElement : HTMLElement {
 attribute HTMLTableCaptionElement? caption;
 HTMLElement createCaption();
 void deleteCaption();
 attribute HTMLTableSectionElement? tHead;
 HTMLElement createTHead();
 void deleteTHead();
 attribute HTMLTableSectionElement? tFoot;
 HTMLElement createTFoot();
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
void deleteRow(long index);
 attribute boolean sortable;
void stopSorting();
};
```

The `table` element represents data with more than one dimension, in the form of a table.

The `table` element takes part in the table model. Tables have rows, columns, and cells given by their descendants. The rows and columns form

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

*in the description of*

*Note: F*  
*the tab* Authors are /web/20130103030411/http:  
information i //whatwg.org/html

If a `table` element is used as a layout table, the user agent may report the contents of that attribute to the user.

Tables must not be used as layout aids. Historically, some Web authors have misused tables in HTML as a way to control their page layout. This usage is non-conforming, because tools attempting to extract tabular data from such documents would obtain very confusing results. In particular, users of accessibility tools like screen readers are likely to find it very difficult to navigate pages with tables used for layout.

**Note:** There are a variety of alternatives to using HTML tables for layout, primarily using CSS positioning and the CSS table model. [CSS]

Tables can be complicated to understand and navigate. To help users with this, user agents should clearly delineate cells in a table from each other, unless the user agent has classified the table as a (non-conforming) layout table.

**Note:** Authors and implementors are encouraged to consider using some of the table design techniques described below to make tables easier to navigate for users.

User agents, especially those that do table analysis on arbitrary content, are encouraged to find heuristics to determine which tables actually contain data and which are merely being used for layout. This specification does not define a precise heuristic, but the following are suggested as possible indicators:

Feature	Indication
The use of the <code>role</code> attribute with the value <code>presentation</code>	Probably a layout table
The use of the non-conforming <code>border</code> attribute with the non-conforming value <code>0</code>	Probably a layout table
The use of the non-conforming <code>cellspacing</code> and <code>cellpadding</code> attributes with the value <code>0</code>	Probably a layout table
The use of <code>caption</code> , <code>thead</code> , or <code>th</code> elements	Probably a non-layout table
The use of the <code>headers</code> and <code>scope</code> attributes	Probably a non-layout table
The use of the non-conforming <code>border</code> attribute with a value other than <code>0</code>	Probably a non-layout table
Explicit visible borders set using CSS	Probably a non-layout table
The use of the <code>summary</code> attribute	Not a good indicator (both layout and non-layout tables have historically been given this attribute)

**Note:** It is quite possible that the above suggestions are wrong. Implementors are urged to provide feedback elaborating on their experiences with trying to create a layout table detection heuristic.

The `sortable` attribute is used in the table sorting model.

This box is non-normative. Implementation requirements are given below this box.

`table . caption [= value]`

Returns the table's `caption` element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Ensures the table has a `caption` element, and returns it.**`table.deleteCaption()`**Ensures the table does not have a `caption` element.**`table.tHead [= value]`**Returns the table's `thead` element.Can be set, to replace the `thead` element. If the new value is not a `thead` element, throws a `HierarchyRequestError` exception.**`thead =` Too slow? Try reading the multipage copy of****the spec instead: <https://web.archive.org>****`table.thead`**

En:

**`/web/20130103030411/http://whatwg.org/html`****`table.tFoot`**Returns the table's `tfoot` element.Can be set, to replace the `tfoot` element. If the new value is not a `tfoot` element, throws a `HierarchyRequestError` exception.**`tfoot = table.createTFoot()`**Ensures the table has a `tfoot` element, and returns it.**`table.deleteTFoot()`**Ensures the table does not have a `tfoot` element.**`table.tBodies`**Returns an `HTMLCollection` of the `tbody` elements of the table.**`tbody = table.createTBody()`**Creates a `tbody` element, inserts it into the table, and returns it.**`table.rows`**Returns an `HTMLCollection` of the `tr` elements of the table.**`tr = table.insertRow([index])`**Creates a `tr` element, along with a `tbody` if required, inserts them into the table at the position given by the argument, and returns the `tr`.The position is relative to the rows in the table. The index `-1`, which is the default if the argument is omitted, is equivalent to inserting at the end of the table.If the given position is less than `-1` or greater than the number of rows, throws an `IndexSizeError` exception.**`table.deleteRow(index)`**Removes the `tr` element with the given position in the table.The position is relative to the rows in the table. The index `-1` is equivalent to deleting the last row of the table.If the given position is less than `-1` or greater than the index of the last row, or if there are no rows, throws an `IndexSizeError` exception.

The `caption` IDL attribute must return, on getting, the first `caption` element child of the `table` element, if any, or null otherwise. On setting, if the new value is a `caption` element, the first `caption` element child of the `table` element, if any, must be removed, and the new value must be inserted as the first node of the `table` element. If the new value is not a `caption` element, then a `HierarchyRequestError` DOM exception must be thrown instead.

The `createCaption()` method must return the first `caption` element child of the `table` element, if any; otherwise a new `caption` element must be created, inserted as the first node of the `table` element, and then returned.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

element, if any, or at the end of the table if there are no such elements. If the new value is not a `thead` element, then a `HierarchyRequestError` DOM exception must be thrown instead.

The `createTHead()` method must return the first `thead` element child of the `table` element, if any; otherwise a new `thead` element must be created and inserted immediately before the first element in the `table` element that is neither a `caption` element nor a `colgroup` element, if any, or at the end of the table if there are no such elements, and then that new element must be returned.

The `deleteTHead()` method must remove the first `thead` element child of the `table` element, if any.

The `tFoot` `IF`

setting, if the value must be removed, and the new `tFoot` element, a `colgroup` element is not a `tFoot` element,

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

/web/20130103030411/http:

The `createTFoot()` method must be created, a `tbody` element, a `caption` element, a `colgroup` element, and then that new element

must be returned.

y, or null otherwise. On

wise a new `tFoot` element

The `deleteTFoot()` method must remove the first `tFoot` element child of the `table` element, if any.

The `tBodies` attribute must return an `HTMLCollection` rooted at the `table` node, whose filter matches only `tbody` elements that are children of the `table` element.

The `createTBody()` method must create a new `tbody` element, insert it immediately after the last `tbody` element in the `table` element, if any, or at the end of the `table` element if the `table` element has no `tbody` element children, and then must return the new `tbody` element.

The `rows` attribute must return an `HTMLCollection` rooted at the `table` node, whose filter matches only `tr` elements that are either children of the `table` element, or children of `thead`, `tbody`, or `tFoot` elements that are themselves children of the `table` element. The elements in the collection must be ordered such that those elements whose parent is a `thead` are included first, in tree order, followed by those elements whose parent is either a `tbody` element, again in tree order, followed finally by those elements whose parent is a `tFoot` element, still in tree order.

The behavior of the `insertRow(index)` method depends on the state of the table. When it is called, the method must act as required by the first item in the following list of conditions that describes the state of the table and the `index` argument:

↳ If `index` is less than `-1` or greater than the number of elements in `rows` collection:

The method must throw an `IndexSizeError` exception.

↳ If the `rows` collection has zero elements in it, and the `table` has no `tbody` elements in it:

The method must create a `tbody` element, then create a `tr` element, then append the `tr` element to the `tbody` element, then append the `tbody` element to the `table` element, and finally return the `tr` element.

↳ If the `rows` collection has zero elements in it:

The method must create a `tr` element, append it to the last `tbody` element in the `table`, and return the `tr` element.

↳ If `index` is `-1` or equal to the number of items in `rows` collection:

The method must create a `tr` element, and append it to the parent of the last `tr` element in the `rows` collection. Then, the newly created `tr` element must be returned.

↳ Otherwise:

The method must create a `tr` element, insert it immediately before the `index`th `tr` element in the `rows` collection, in the same parent, and finally must return the newly created `tr` element.

When the `deleteRow(index)` method is called, the user agent must run the following steps:

1. If `index` is equal to `-1`, then `index` must be set to the number of items in the `rows` collection, minus one.
2. Now, if `index` is less than zero, or greater than or equal to the number of elements in the `rows` collection, the method must instead throw an `IndexSizeError` exception, and these steps must be aborted.
3. Otherwise, the method must remove the `index`th element in the `rows` collection from its parent.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

such a table.

```
<section>
<style scoped>
 table { border-collapse: collapse; border: solid thick; }
 colgroup, tbody { border: solid medium; }
 td { border: solid thin; height: 1.4em; width: 1.4em; text-align: center; padding: 0; }
</style>
<h1>Today's Sudoku</h1>
<table border="1">
```

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
 <tr> <td> <td> <td> <td> <td> <td> <td> <td> <td>
 <tr> <td> 5 <td> <td> 9 <td> <td> 7 <td> <td> 1
<tbody>
 <tr> <td> 6 <td> <td> <td> 5 <td> <td> <td> <td> 2
 <tr> <td> <td> <td> <td> 7 <td> <td> <td> <td>
 <tr> <td> 9 <td> <td> 8 <td> <td> 2 <td> <td> 5
</table>
</section>
```

#### 4.9.1.1 Techniques for describing tables

For tables that consist of more than just a grid of cells with headers in the first row and headers in the first column, and for any table in general where the reader might have difficulty understanding the content, authors should include explanatory information introducing the table. This information is useful for all users, but is especially useful for users who cannot see the table, e.g. users of screen readers.

Such explanatory information should introduce the purpose of the table, outline its basic cell structure, highlight any trends or patterns, and generally teach the user how to use the table.

For instance, the following table:

Characteristics with positive and negative sides		
Negative	Characteristic	Positive
Sad	Mood	Happy
Failing	Grade	Passing

...might benefit from a description explaining the way the table is laid out, something like "Characteristics are given in the second column, with the negative side in the left column and the positive side in the right column".

There are a variety of ways to include this information, such as:

##### In prose, surrounding the table

```
<p>In the following table, characteristics are given in the second column, with the negative side in the left column and the positive side in the right column.</p>
<table>
<caption>Characteristics with positive and negative sides</caption>
<thead>
<tr>
 <th id="n"> Negative
 <th> Characteristic
 <th> Positive
<tbody>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

```
<td headers="n r2"> Failing
<th id="r2"> Grade
<td> Passing
</table>
```

**In the table's caption**

```
<table>
<caption>
```

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
<tbody>
<tr>
<td headers="n r1"> Sad
<th id="r1"> Mood
<td> Happy
<tr>
<td headers="n r2"> Failing
<th id="r2"> Grade
<td> Passing
</table>
```

**In the table's caption, in a details element**

```
<table>
<caption>
Characteristics with positive and negative sides.
<details>
<summary>Help</summary>
<p>Characteristics are given in the second column, with the
negative side in the left column and the positive side in the right
column.</p>
</details>
</caption>
<thead>
<tr>
<th id="n"> Negative
<th> Characteristic
<th> Positive
<tbody>
<tr>
<td headers="n r1"> Sad
<th id="r1"> Mood
<td> Happy
<tr>
<td headers="n r2"> Failing
<th id="r2"> Grade
<td> Passing
</table>
```

**Next to the table, in the same figure**

```
<figure>
<figcaption>Characteristics with positive and negative sides</figcaption>
<p>Characteristics are given in the second column, with the
negative side in the left column and the positive side in the right
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<th> Characteristic
<th> Positive
<tbody>
<tr>
<td headers="n r1"> Sad
<th id="r1"> Mood
<td> Happy
<tr>
<td headers="n r2"> Failing
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

Next to the /web/20130103030411/http://whatwg.org/html/

```
Characteristics with positive and negative sides
<p>Characteristics are given in the second column, with the negative side in the left column and the positive side in the right column.</p>
</figcaption>
<table>
<thead>
<tr>
<th id="n"> Negative
<th> Characteristic
<th> Positive
<tbody>
<tr>
<td headers="n r1"> Sad
<th id="r1"> Mood
<td> Happy
<tr>
<td headers="n r2"> Failing
<th id="r2"> Grade
<td> Passing
</table>
</figure>
```

Authors may also use other techniques, or combinations of the above techniques, as appropriate.

The best option, of course, rather than writing a description explaining the way the table is laid out, is to adjust the table such that no explanation is needed.

In the case of the table used in the examples above, a simple rearrangement of the table so that the headers are on the top and left sides removes the need for an explanation as well as removing the need for the use of `headers` attributes:

```
<table>
<caption>Characteristics with positive and negative sides</caption>
<thead>
<tr>
<th> Characteristic
<th> Negative
<th> Positive
<tbody>
<tr>
<th> Mood
<td> Sad
<td> Happy
<tr>
<th> Grade
<td> Failing
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Good table design is key to making tables more readable and usable.

In visual media, providing column and row borders and alternating row backgrounds can be very effective to make complicated tables more readable.

For tables with large volumes of numeric content, using monospaced fonts can help users see patterns, especially in situations where a user agent does not render the borders. (Unfortunately, for historical reasons, not rendering borders on tables is a common default.)

In speech media, table cells can be distinguished by reporting the corresponding headers before reading the cell's contents, and by allowing user agents to report the cells in source order.

Authors are [Too slow? Try reading the multipage copy of](#)

User agents [the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html](#)

S and the table is not

## 4.9.2 The `<caption>`

### Categories:

None.

### Contexts in which this element can be used:

As the first element child of a `table` element.

### Content model:

Flow content, but with no descendant `table` elements.

### Content attributes:

Global attributes

### DOM interface:

```
IDL interface HTMLTableCaptionElement : HTMLElement {};
```

The `caption` element represents the title of the `table` that is its parent, if it has a parent and that is a `table` element.

The `caption` element takes part in the table model.

When a `table` element is the only content in a `figure` element other than the `figcaption`, the `caption` element should be omitted in favor of the `figcaption`.

A caption can introduce context for a table, making it significantly easier to understand.

Consider, for instance, the following table:

1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11
7	8	9	10	11	12

In the abstract, this table is not clear. However, with a caption giving the table's number (for reference in the main prose) and explaining its use, it makes more sense:

```
<caption>
<p>Table 1.
<p>This table shows the total score obtained from rolling two
six-sided dice. The first row represents the value of the first die,
the first column the value of the second die. The total is given in
the cell that corresponds to the values of the two dice.
</caption>
```

This provides the user with more context:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11



#### 4.9.3 The `colgroup` element

Categories:

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

Contexts:

A [/whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html), `tfoot`, and `tr` elements.

Content:

If the `span` attribute is present, empty.  
If the `span` attribute is absent: Zero or more `col` elements.

Content attributes:

Global attributes  
`span`

DOM interface:

```
IDL interface HTMLTableColElement : HTMLElement {
 attribute unsigned long span;
};
```

The `colgroup` element represents a group of one or more columns in the `table` that is its parent, if it has a parent and that is a `table` element.

If the `colgroup` element contains no `col` elements, then the element may have a `span` content attribute specified, whose value must be a valid non-negative integer greater than zero.

The `colgroup` element and its `span` attribute take part in the table model.

The `span` IDL attribute must reflect the content attribute of the same name. The value must be limited to only non-negative numbers greater than zero.

#### 4.9.4 The `col` element

Categories:

None.

Contexts in which this element can be used:

As a child of a `colgroup` element that doesn't have a `span` attribute.

Content model:

Empty.

Content attributes:

Global attributes  
`span`

DOM interface:

`HTMLTableColElement`, same as for `colgroup` elements. This interface defines one member, `span`.

If a `col` element has a parent and that is a `colgroup` element that itself has a parent that is a `table` element, then the `col` element represents one or more columns in the column group represented by that `colgroup`.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

greater than zero.

## 4.9.5 The `tbody` element

### Categories:

None.

### Contexts in which this element can be used:

As

† Too slow? Try reading the multipage copy of

## Content:

z the spec instead: <https://web.archive.org>

## Content:

/web/20130103030411/http:

C

//whatwg.org/html

## DOM int.

.. if there are no `tr` elements

```
IDL interface HTMLTableSectionElement : HTMLElement {
 readonly attribute HTMLCollection rows;
 HTMLElement insertRow(optional long index = -1);
 void deleteRow(long index);
};
```

The `HTMLTableSectionElement` interface is also used for `thead` and `tfoot` elements.

The `tbody` element represents a block of rows that consist of a body of data for the parent `table` element, if the `tbody` element has a parent and it is a `table`.

The `tbody` element takes part in the table model.

This box is non-normative. Implementation requirements are given below this box.

#### `tbody.rows`

Returns an `HTMLCollection` of the `tr` elements of the table section.

#### `tr = tbody.insertRow( [ index ] )`

Creates a `tr` element, inserts it into the table section at the position given by the argument, and returns the `tr`.

The position is relative to the rows in the table section. The index `-1`, which is the default if the argument is omitted, is equivalent to inserting at the end of the table section.

If the given position is less than `-1` or greater than the number of rows, throws an `IndexSizeError` exception.

#### `tbody.deleteRow(index)`

Removes the `tr` element with the given position in the table section.

The position is relative to the rows in the table section. The index `-1` is equivalent to deleting the last row of the table section.

If the given position is less than `-1` or greater than the index of the last row, or if there are no rows, throws an `IndexSizeError` exception.

The `rows` attribute must return an `HTMLCollection` rooted at the element, whose filter matches only `tr` elements that are children of the element.

The `insertRow(index)` method must, when invoked on an element `table section`, act as follows:

If `index` is less than `-1` or greater than the number of elements in the `rows` collection, the method must throw an `IndexSizeError` exception.

If `index` is `-1` or equal to the number of items in the `rows` collection, the method must create a `tr` element, append it to the element

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `deleteRow(index)` method must remove the `index`th element in the `rows` collection from its parent. If `index` is less than zero or greater than or equal to the number of elements in the `rows` collection, the method must instead throw an `IndexSizeError` exception.

#### 4.9.6 The `thead` element

##### Categories:

None.



##### Contexts

<sup>A</sup> Too slow? Try reading the multipage copy of  
<sup>e</sup> the spec instead: <https://web.archive.org>

&gt;body, tfoot, and tr

##### Content

<sup>I</sup> [/web/20130103030411/http://whatwg.org/html](https://web/20130103030411/http://whatwg.org/html)

<sup>Z</sup>

##### Content

<sup>I</sup> [/web/20130103030411/http://whatwg.org/html](https://web/20130103030411/http://whatwg.org/html)

<sup>G</sup>

##### DOM interface:

`HTMLTableSectionElement`, as defined for `tbody` elements.

The `thead` element represents the block of rows that consist of the column labels (headers) for the parent `table` element, if the `thead` element has a parent and it is a `table`.

The `thead` element takes part in the table model.

This example shows a `thead` element being used. Notice the use of both `th` and `td` elements in the `thead` element: the first row is the headers, and the second row is an explanation of how to fill in the table.

```
<table>
 <caption> School auction sign-up sheet </caption>
 <thead>
 <tr>
 <th><label for=e1>Name</label>
 <th><label for=e2>Product</label>
 <th><label for=e3>Picture</label>
 <th><label for=e4>Price</label>
 <tr>
 <td>Your name here
 <td>What are you selling?
 <td>Link to a picture
 <td>Your reserve price
 <tbody>
 <tr>
 <td>Ms Danus
 <td>Doughnuts
 <td>
 <td>$45
 <tr>
 <td><input id=e1 type=text name=who required form=f>
 <td><input id=e2 type=text name=what required form=f>
 <td><input id=e3 type=url name=pic form=f>
 <td><input id=e4 type=number step=0.01 min=0 value=0 required form=f>
 </table>
 <form id=f action="/auction.cgi">
 <input type=button name=add value="Submit">
 </form>
```

#### 4.9.7 The `tfoot` element

##### Categories:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

As a child of a `table` element, after any `caption`, `colgroup`, `thead`, `tbody`, and `tr` elements, but only if there are no other `tfoot` elements that are children of the `table` element.

**Content model:**Zero or more `tr` elements**Content attributes:**

Global attributes

**DOM interface:**

H Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

rent `table` element, if theThe `tfoot` elementThe `tfoot` element//whatwg.org/html**4.9.8 The `tr` element****Categories:**

None.

**Contexts in which this element can be used:**As a child of a `thead` element.As a child of a `tbody` element.As a child of a `tfoot` element.As a child of a `table` element, after any `caption`, `colgroup`, and `thead` elements, but only if there are no `tbody` elements that are children of the `table` element.**Content model:**Zero or more `td` or `th` elements**Content attributes:**

Global attributes

**DOM interface:**

```
IDL interface HTMLTableRowElement : HTMLElement {
 readonly attribute long rowIndex;
 readonly attribute long sectionRowIndex;
 readonly attribute HTMLCollection cells;
 HTMLElement insertCell(optional long index);
 void deleteCell(long index);
};
```

The `tr` element represents a row of cells in a table.

The `tr` element takes part in the table model.

*This box is non-normative. Implementation requirements are given below this box.*

**`tr(rowIndex)`**

Returns the position of the row in the table's `rows` list.

Returns `-1` if the element isn't in a table.

**`tr(sectionRowIndex)`**

Returns the position of the row in the table section's `rows` list.

Returns `-1` if the element isn't in a table section.

**`tr(cells)`**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

The position is relative to the cells in the row. The index  $-1$ , which is the default if the argument is omitted, is equivalent to inserting at the end of the row.

If the given position is less than  $-1$  or greater than the number of cells, throws an `IndexSizeError` exception.

#### `tr.deleteCell(index)`

Removes the `td` or `th` element with the given position in the row.

The position is relative to the cells in the row. The index  $-1$  is equivalent to deleting the last cell of the row.

If the given position is less than  $-1$  or greater than the index of the last cell, or if there are no cells, throws an

Inc

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `row` parent element, the `grandparent` element, the

:`tfoot` element and a  
if there is no such `table`

The `section`

:`ent`, return the index of the  
collection; for table sections,  
that's the `HTMLTableElement.rows` collection). If there is no such parent element, then the attribute must return  $-1$ .

The `cells` attribute must return an `HTMLCollection` rooted at the `tr` element, whose filter matches only `td` and `th` elements that are children of the `tr` element.

The `insertCell(index)` method must act as follows:

If `index` is less than  $-1$  or greater than the number of elements in the `cells` collection, the method must throw an `IndexSizeError` exception.

If `index` is missing, equal to  $-1$ , or equal to the number of items in `cells` collection, the method must create a `td` element, append it to the `tr` element, and return the newly created `td` element.

Otherwise, the method must create a `td` element, insert it as a child of the `tr` element, immediately before the `index`th `td` or `th` element in the `cells` collection, and finally must return the newly created `td` element.

The `deleteCell(index)` method must remove the `index`th element in the `cells` collection from its parent. If `index` is less than zero or greater than or equal to the number of elements in the `cells` collection, the method must instead throw an `IndexSizeError` exception.

## 4.9.9 The `td` element

### Categories:

Sectioning root.

### Contexts in which this element can be used:

As a child of a `tr` element.

### Content model:

Flow content.

### Content attributes:

Global attributes  
`colspan`  
`rowspan`  
`headers`

### DOM interface:

```
IDL interface HTMLTableDataCellElement : HTMLTableCellElement {};
```

The `td` element represents a data cell in a table.

The `td` element and its `colspan`, `rowspan`, and `headers` attributes take part in the table model.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## 4.9.10 The `th` element

### Categories:

If the `th` element is a sorting interface `th` element: Interactive content.  
 Otherwise: None.

### Contexts in which this element can be used:

As a child of a `tr` element.



**Content:** `F` Too slow? Try reading the multipage copy of  
`a` the spec instead: <https://web.archive.org/web/20130103030411/http://www.whatwg.org/html>

s, and if the `th` element is

**Content:** `C` [/web/20130103030411/http://www.whatwg.org/html](https://web.archive.org/web/20130103030411/http://www.whatwg.org/html)  
`c` `r`

headers  
 scope  
 abbr  
 sorted

### DOM interface:

```
IDL interface HTMLTableHeaderCellElement : HTMLTableCellElement {
 attribute DOMString scope;
 attribute DOMString abbr;
 attribute DOMString sorted;
 void sort();
};
```

The `th` element represents a header cell in a table.

The `th` element may have a `scope` content attribute specified. The `scope` attribute is an enumerated attribute with five states, four of which have explicit keywords:

#### The `row` keyword, which maps to the `row` state

The `row` state means the header cell applies to some of the subsequent cells in the same row(s).

#### The `col` keyword, which maps to the `column` state

The `column` state means the header cell applies to some of the subsequent cells in the same column(s).

#### The `rowgroup` keyword, which maps to the `row group` state

The `row group` state means the header cell applies to all the remaining cells in the row group. A `th` element's `scope` attribute must not be in the `row group` state if the element is not anchored in a `row group`.

#### The `colgroup` keyword, which maps to the `column group` state

The `column group` state means the header cell applies to all the remaining cells in the `column group`. A `th` element's `scope` attribute must not be in the `column group` state if the element is not anchored in a `column group`.

#### The `auto` state

The `auto` state makes the header cell apply to a set of cells selected based on context.

The `scope` attribute's *missing value default* is the `auto` state.

The `th` element may have an `abbr` content attribute specified. Its value must be an alternative label for the header cell, to be used when referencing the cell in other contexts (e.g. when describing the header cells that apply to a data cell). It is typically an abbreviated form of the full header cell, but can also be an expansion, or merely a different phrasing.

The `sorted` attribute is used in the table sorting model.

The `th` element and its `colspan`, `rowspan`, `headers`, and `scope` attributes take part in the table model.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The following example shows how the `scope` attribute's `rowgroup` value affects which data cells a header cell applies to.

Here is a markup fragment showing a table:

```
<table>
 <thead>
 <tr> <th> ID <th> Measurement <th> Average <th> Maximum
 <tbody>
 <tr> <td> <th scope="rowgroup"> Cats <td> <td>
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

This \

ID	Measurement	Average	Maximum
Cats			
93	Legs	3.5	4
10	Tails	1	1
English speakers			
32	Legs	2.67	4
35	Tails	0.33	1

The headers in the first row all apply directly down to the rows in their column.

The headers with the explicit `scope` attributes apply to all the cells in their row group other than the cells in the first column.

The remaining headers apply just to the cells to the right of them.

ID	Measurement	Average	Maximum
	Cats		
93	Legs	3.5	4
10	Tails	1	1
English speakers			
32	Legs	2.67	4
35	Tails	0.33	1

#### 4.9.11 Attributes common to `td` and `th` elements

The `td` and `th` elements may have a `colspan` content attribute specified, whose value must be a valid non-negative integer greater than zero.

The `td` and `th` elements may also have a `rowspan` content attribute specified, whose value must be a valid non-negative integer. For this attribute, the value zero means that the cell is to span all the remaining rows in the row group.

These attributes give the number of columns and rows respectively that the cell is to span. These attributes must not be used to overlap cells, as described in the description of the table model.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

values include as one of their tokens the ID *id*. A `th` element *A* is said to be *targeted* by a `th` or `td` element *B* if either *A* is *directly targeted* by *B* or if there exists an element *C* that is itself *targeted* by the element *B* and *A* is *directly targeted* by *C*.

A `th` element must not be *targeted* by itself.

The `colspan`, `rowspan`, and `headers` attributes take part in the table model.

The `td` and `th` elements implement interfaces that inherit from the `HTMLTableCellElement` interface:

`intendedForTable`  
`[PagedModule]`  
`replaces`  
`};`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

...its are given below this box.

#### `cell.cellIndex`

Returns the position of the cell in the row's `cells` list. This does not necessarily correspond to the x-position of the cell in the table, since earlier cells might cover multiple rows or columns.

Returns `-1` if the element isn't in a row.

The `colspan` IDL attribute must reflect the `colspan` content attribute. Its default value is `1`.

The `rowSpan` IDL attribute must reflect the `rowspan` content attribute. Its default value is `1`.

The `headers` IDL attribute must reflect the `content` attribute of the same name.

The `cellIndex` IDL attribute must, if the element has a parent `tr` element, return the index of the cell's element in the parent element's `cells` collection. If there is no such parent element, then the attribute must return `-1`.

### 4.9.12 Processing model

The various table elements and their content attributes together define the **table model**.

A `table` consists of cells aligned on a two-dimensional grid of **slots** with coordinates  $(x, y)$ . The grid is finite, and is either empty or has one or more slots. If the grid has one or more slots, then the  $x$  coordinates are always in the range  $0 \leq x < x_{width}$ , and the  $y$  coordinates are always in the range  $0 \leq y < y_{height}$ . If one or both of  $x_{width}$  and  $y_{height}$  are zero, then the table is empty (has no slots). Tables correspond to `table` elements.

A **cell** is a set of slots anchored at a slot  $(cell_x, cell_y)$ , and with a particular  $width$  and  $height$  such that the cell covers all the slots with coordinates  $(x, y)$  where  $cell_x \leq x < cell_x + width$  and  $cell_y \leq y < cell_y + height$ . Cells can either be *data cells* or *header cells*. Data cells correspond to `td` elements, and header cells correspond to `th` elements. Cells of both types can have zero or more associated header cells.

It is possible, in certain error cases, for two cells to occupy the same slot.

A **row** is a complete set of slots from  $x=0$  to  $x=x_{width}-1$ , for a particular value of  $y$ . Rows usually correspond to `tr` elements, though a row group can have some implied rows at the end in some cases involving cells spanning multiple rows.

A **column** is a complete set of slots from  $y=0$  to  $y=y_{height}-1$ , for a particular value of  $x$ . Columns can correspond to `col` elements. In the absence of `col` elements, columns are implied.

A **row group** is a set of rows anchored at a slot  $(0, group_y)$  with a particular  $height$  such that the row group covers all the slots with coordinates  $(x, y)$  where  $0 \leq x < x_{width}$  and  $group_y \leq y < group_y + height$ . Row groups correspond to `tbody`, `thead`, and `tfoot` elements. Not every row is necessarily in a row group.

A **column group** is a set of columns anchored at a slot  $(group_x, 0)$  with a particular  $width$  such that the column group covers all the slots with coordinates  $(x, y)$  where  $group_x \leq x < group_x + width$  and  $0 \leq y < y_{height}$ . Column groups correspond to `colgroup` elements. Not every column is necessarily in a column group.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

In addition to cells, columns, rows, row groups, and column groups, tables can have a `caption` element associated with them. This gives the table a heading, or legend.

A **table model error** is an error with the data represented by `table` elements and their descendants. Documents must not have table model errors.

### 4.9.12.1 Forming a table

To determine which elements correspond to which slots in a table associated with a `table` element, to determine the dimensions of the table ( $xwidth$ ),

1. Let **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/table>**
2. Let ***the table's dimensions. The***
3. Let ***the table's dimensions. The***
4. Let ***the table's dimensions. The***

5. If the `table` element has no children elements, then return `the table` (which will be empty), and abort these steps.
6. Associate the first `caption` element child of the `table` element with `the table`. If there are no such children, then it has no associated `caption` element.
7. Let the `current element` be the first element child of the `table` element.

If a step in this algorithm ever requires the `current element` to be **advanced to the next child of the table** when there is no such next child, then the user agent must jump to the step labeled `end`, near the end of this algorithm.

8. While the `current element` is not one of the following elements, advance the `current element` to the next child of the `table`:

- o `colgroup`
- o `thead`
- o `tbody`
- o `tfoot`
- o `tr`

9. If the `current element` is a `colgroup`, follow these substeps:

1. **Column groups:** Process the `current element` according to the appropriate case below:

↳ **If the current element has any `col` element children**

Follow these steps:

1. Let  $x_{start}$  have the value of  $xwidth$ .
2. Let the `current column` be the first `col` element child of the `colgroup` element.
3. **Columns:** If the `current column` `col` element has a `span` attribute, then parse its value using the rules for parsing non-negative integers.

If the result of parsing the value is not an error or zero, then let `span` be that value.

Otherwise, if the `col` element has no `span` attribute, or if trying to parse the attribute's value resulted in an error or zero, then let `span` be 1.

4. Increase  $xwidth$  by `span`.
5. Let the last `span` columns in `the table` correspond to the `current column` `col` element.
6. If `current column` is not the last `col` element child of the `colgroup` element, then let the `current column` be the next `col` element child of the `colgroup` element, and return to the step labeled `columns`.
7. Let all the last columns in `the table` from  $x=x_{start}$  to  $x=xwidth-1$  form a new column group, anchored at the slot ( $x_{start}, 0$ ), with width  $xwidth-x_{start}$ , corresponding to the `colgroup` element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the result of parsing the value is not an error or zero, then let *span* be that value.

Otherwise, if the `colgroup` element has no `span` attribute, or if trying to parse the attribute's value resulted in an error or zero, then let *span* be 1.

2. Increase *xwidth* by *span*.

3. Let the last *span* columns in *the table* form a new column group, anchored at the slot (*xwidth*-*span*, 0), with width *span*, corresponding to the `colgroup` element.



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

!nt to the next child of the

4. If the *current element* is a `colgroup` element, jump to the step labeled *column groups* above.

10. Let *ycurrent* be zero.

11. Let the *list of downward-growing cells* be an empty list.

12. *Rows*: While the *current element* is not one of the following elements, advance the *current element* to the next child of the *table*:

- o `thead`
- o `tbody`
- o `tfoot`
- o `tr`

13. If the *current element* is a `tr`, then run the algorithm for processing rows, advance the *current element* to the next child of the *table*, and return to the step labeled *rows*.

14. Run the algorithm for ending a row group.

15. If the *current element* is a `tfoot`, then add that element to the list of *pending tfoot elements*, advance the *current element* to the next child of the *table*, and return to the step labeled *rows*.

16. The *current element* is either a `thead` or a `tbody`.

Run the algorithm for processing row groups.

17. Advance the *current element* to the next child of the *table*.

18. Return to the step labeled *rows*.

19. *End*: For each `tfoot` element in the list of *pending tfoot elements*, in tree order, run the algorithm for processing row groups.

20. If there exists a row or column in *the table* containing only slots that do not have a cell anchored to them, then this is a table model error.

21. Return the *table*.

The **algorithm for processing row groups**, which is invoked by the set of steps above for processing `thead`, `tbody`, and `tfoot` elements, is:

1. Let *ystart* have the value of *yheight*.
2. For each `tr` element that is a child of the element being processed, in tree order, run the algorithm for processing rows.
3. If *yheight* > *ystart*, then let all the last rows in *the table* from *y=ystart* to *y=yheight-1* form a new row group, anchored at the slot with coordinate (0, *ystart*), with height *yheight-ystart*, corresponding to the element being processed.
4. Run the algorithm for ending a row group.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2. Increase  $y_{current}$  by 1.
2. Empty the *list of downward-growing cells*.

The **algorithm for processing rows**, which is invoked by the set of steps above for processing `tr` elements, is:

1. If  $y_{height}$  is equal to  $y_{current}$ , then increase  $y_{height}$  by 1. ( $y_{current}$  is never greater than  $y_{height}$ .)

2. Let  $x_{current}$  be 0.

3. Run

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

abort this set of steps, and

4. If th

retu

5. Let

/web/20130103030411/http:

6. Cell

$x_{cur}$

7. If  $x_{current}$  is equal to  $x_{width}$ , increase  $x_{width}$  by 1. ( $x_{current}$  is never greater than  $x_{width}$ .)

8. If the *current cell* has a `colspan` attribute, then parse that attribute's value, and let `colspan` be the result.

If parsing that value failed, or returned zero, or if the attribute is absent, then let `colspan` be 1, instead.

9. If the *current cell* has a `rowspan` attribute, then parse that attribute's value, and let `rowspan` be the result.

If parsing that value failed or if the attribute is absent, then let `rowspan` be 1, instead.

10. If `rowspan` is zero and the `table` element's `Document` is not set to quirks mode, then let `cell grows downward` be true, and set `rowspan` to 1. Otherwise, let `cell grows downward` be false.

11. If  $x_{width} < x_{current} + \text{colspan}$ , then let  $x_{width}$  be  $x_{current} + \text{colspan}$ .

12. If  $y_{height} < y_{current} + \text{rowspan}$ , then let  $y_{height}$  be  $y_{current} + \text{rowspan}$ .

13. Let the slots with coordinates  $(x, y)$  such that  $x_{current} \leq x < x_{current} + \text{colspan}$  and  $y_{current} \leq y < y_{current} + \text{rowspan}$  be covered by a new cell  $c$ , anchored at  $(x_{current}, y_{current})$ , which has width `colspan` and height `rowspan`, corresponding to the *current cell* element.

If the *current cell* element is a `th` element, let this new cell  $c$  be a header cell; otherwise, let it be a data cell.

To establish which header cells apply to the *current cell* element, use the algorithm for assigning header cells described in the next section.

If any of the slots involved already had a cell covering them, then this is a table model error. Those slots now have two cells overlapping.

14. If `cell grows downward` is true, then add the tuple  $\{c, x_{current}, \text{colspan}\}$  to the *list of downward-growing cells*.

15. Increase  $x_{current}$  by `colspan`.

16. If *current cell* is the last `td` or `th` element in the `tr` element being processed, then increase  $y_{current}$  by 1, abort this set of steps, and return to the algorithm above.

17. Let *current cell* be the next `td` or `th` element in the `tr` element being processed.

18. Return to the step labelled *cells*.

When the algorithms above require the user agent to run the **algorithm for growing downward-growing cells**, the user agent must, for each  $\{cell, cell_x, width\}$  tuple in the *list of downward-growing cells*, if any, extend the cell *cell* so that it also covers the slots with coordinates  $(x, y_{current})$ , where  $cell_x \leq x < cell_x + width$ .

#### 4.9.12.2 Forming relationships between data cells and header cells

Each cell can be assigned zero or more header cells. The **algorithm for assigning header cells** to a cell *principal cell* is as follows.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. Take the value of the *principal cell's headers* attribute and split it on spaces, letting *id list* be the list of tokens obtained.
2. For each token in the *id list*, if the first element in the Document with an ID equal to the token is a cell in the same table, and that cell is not the *principal cell*, then add that cell to *header list*.

↳ If *principal cell* does not have a *headers* attribute specified

1. Let *principalWidth* be the width of the *principal cell*.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

5. If the *principal cell* is anchored in a row group, then add all header cells that are row group headers and are anchored in the same row group with an x-coordinate less than or equal to  $\text{principal}_x + \text{principalWidth} - 1$  and a y-coordinate less than or equal to  $\text{principal}_y + \text{principalHeight} - 1$  to *header list*.
6. If the *principal cell* is anchored in a column group, then add all header cells that are column group headers and are anchored in the same column group with an x-coordinate less than or equal to  $\text{principal}_x + \text{principalWidth} - 1$  and a y-coordinate less than or equal to  $\text{principal}_y + \text{principalHeight} - 1$  to *header list*.
4. Remove all the empty cells from the *header list*.
5. Remove any duplicates from the *header list*.
6. Remove *principal cell* from the *header list* if it is there.
7. Assign the headers in the *header list* to the *principal cell*.

The **internal algorithm for scanning and assigning header cells**, given a *principal cell*, a *header list*, an initial coordinate ( $\text{initial}_x$ ,  $\text{initial}_y$ ), and  $\Delta x$  and  $\Delta y$  increments, is as follows:

1. Let  $x$  equal  $\text{initial}_x$ .
2. Let  $y$  equal  $\text{initial}_y$ .
3. Let *opaque headers* be an empty list of cells.
4. ↳ If *principal cell* is a header cell

Let *in header block* be true, and let *headers from current header block* be a list of cells containing just the *principal cell*.

↳ Otherwise

Let *in header block* be false and let *headers from current header block* be an empty list of cells.

5. Loop: Increment  $x$  by  $\Delta x$ ; increment  $y$  by  $\Delta y$ .

**Note: For each invocation of this algorithm, one of  $\Delta x$  and  $\Delta y$  will be  $-1$ , and the other will be  $0$ .**

6. If either  $x$  or  $y$  is less than  $0$ , then abort this internal algorithm.
7. If there is no cell covering slot  $(x, y)$ , or if there is more than one cell covering slot  $(x, y)$ , return to the substep labeled *loop*.
8. Let *current cell* be the cell covering slot  $(x, y)$ .
9. ↳ If *current cell* is a header cell
  1. Set *in header block* to true.
  2. Add *current cell* to *headers from current header block*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the *current cell* is not a column header, then let *blocked* be true.

↳ If  $\Delta y$  is 0

If there are any cells in the *opaque headers* list anchored with the same y-coordinate as the *current cell*, and with the same height as *current cell*, then let *blocked* be true.

If the *current cell* is not a row header, then let *blocked* be true.

5. If *blocked* is false, then add the *current cell* to the *headers list*.

↳ Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the *opaque headers* list, and

10. Ret

A header ce  
following coi

olumn header if any of the

- The cell's `scope` attribute is in the column state, or
- The cell's `scope` attribute is in the auto state, and there are no data cells in any of the cells covering slots with y-coordinates  $y .. y+height-1$ .

A header cell anchored at the slot with coordinate  $(x, y)$  with width `width` and height `height` is said to be a **row header** if any of the following conditions are true:

- The cell's `scope` attribute is in the row state, or
- The cell's `scope` attribute is in the auto state, the cell is not a column header, and there are no data cells in any of the cells covering slots with x-coordinates  $x .. x+width-1$ .

A header cell is said to be a **column group header** if its `scope` attribute is in the column group state.

A header cell is said to be a **row group header** if its `scope` attribute is in the row group state.

A cell is said to be an **empty cell** if it contains no elements and its text content, if any, consists only of White\_Space characters.

#### 4.9.13 Table sorting model

The `sortable` attribute on `table` elements is a boolean attribute. When present, it indicates that the user agent is to allow the user to sort the table.

To make a column sortable in a `table` with a `thead`, the column needs to have `th` element that does not span multiple columns in a `thead` above any rows that it is to sort.

To make a column sortable in a `table` without a `thead`, the column needs to have `th` element that does not span multiple columns in the first `tr` element of the `table`, where that `tr` element is not in a `tfoot`.

When the user selects a column by which to sort, the user agent sets the `th` element's `sorted` attribute. This attribute can also be set manually, to indicate that the table should be automatically sorted, even when scripts modify the page on when the page is loaded.

The `sorted` attribute, if specified, must have a value that is a set of space-separated tokens consisting of optionally a token whose value is an ASCII case-insensitive match for the string "`reversed`", and optionally a token whose value is a valid non-negative integer greater than zero, in either order.

**Note:** In other words, ignoring spaces and case, the `sorted` attribute's value can be empty, "`reversed`", "`1`", "`reversed 1`", or "`1 reversed`", where "`1`" is any number equal to or greater than `1`.

While one or more `th` elements in the table have a `sorted` attribute, the user agent will keep the table's data rows sorted. The value of the attribute controls how the column is used in determining the sort order. The `reversed` keyword means that the **column sort direction** is `reversed`, rather than `normal`, which is the default if the keyword is omitted. The number, if present, indicates the **column key ordinality**; if the number is omitted, the column is the primary key, as if the value `1` had been specified.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

- It corresponds to a cell whose `width` is 1. (Specifically, a *header cell*, since this is a `th` element.)
  - There is no cell that corresponds to another sorting-capable `th` element that covers slots in the same column but on a higher (earlier) row.
  - If the cell's table has a row group corresponding to a `thead` element, the cell is in a row group that corresponds to the *first* `thead` element of the cell's table.
- Otherwise: the cell is not in a row group corresponding to a `tfoot` element, and the cell is the first row of the table.

*Note: If that spans 1* Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org> *ghost th in a thead row), assuming it*

The **sorting** `/web/20130103030411/http:`

A `table` element is a **table element** if it is a sorting-enabled `table` element whose cell's table is `table`.

A `th` element is a **sorting-enabled th element** if it is a sorting-capable `th` element and it has a `sorted` attribute.

The **sorting-enabled th elements of the table element** of the `table` element `table` are the sorting-enabled `th` elements whose cell's table is `table`.

A `table` element `table` is a **sorting-enabled table element** if there are one or more sorting-capable `th` elements of the `table` element `table`, and at least one of them is a sorting-enabled `th` element (i.e. at least one has a `sorted` attribute).

A `table` element is a **table element with a user-agent exposed sorting interface** if it is a sorting-capable `table` element and has a `sortable` attribute specified.

A **sorting interface th element** is a sorting-capable `th` element whose cell's table is a `table` element with a user-agent exposed sorting interface.

Each `table` element has a **currently-sorting flag**, which must initially be false.

The `sorted` attribute must not be specified on `th` elements that are not sorting-capable `th` elements. The `sortable` attribute must not be specified on `table` elements that are not sorting-capable `table` elements.

To determine a `th` element's `sorted` attribute's column sort direction and column key ordinality, user agents must use the following algorithm:

1. Let `direction` be *normal*.
2. Let `have explicit direction` be false.
3. Let `ordinality` be 1.
4. Let `have explicit ordinality` be false.
5. Let `tokens` be the result of splitting the attribute's value on spaces.
6. For each token `token` in `tokens`, run the appropriate steps from the following list:

↳ If `have explicit direction` is false and `token` is an ASCII case-insensitive match for the string "reversed"

Let `direction` be *reversed* and `have explicit direction` be true.

↳ If `have explicit ordinality` is false

Parse `token` as an integer. If this resulted in an error or the value zero, then ignore the token. Otherwise, set `ordinality` to the parsed value, and set `have explicit ordinality` to true.

↳ Otherwise

Ignore the token.

7. The column sort direction is the value of `direction`, and the column key ordinality is the value of `ordinality`.

A `table` must not have two `th` elements whose `sorted` attribute have the same column key ordinality.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. Set *table*'s currently-sorting flag to true.
4. Fire a simple event named `sort` that is cancelable at *table*.
5. If the event fired in the previous step was canceled, then jump to the step labeled *end* below.
6. If *table* is not now a sorting-enabled `table` element, then jump to the step labeled *end* below.

**Note:** Even if *table* was a sorting-enabled `table` element when the algorithm was invoked, the DOM might not yet be fully verified at this stage,

## Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org>

7. Let *table* be the `table` element pointed to by *row collection cursor*.
8. Sort *table*'s `tbody` children in tree order according to the `sort()` algorithm, with those having the same *key* value grouped together.
9. Let *row collection cursor* point to the first `tr` child of *table*, if any, and that is either a `tbody` or a `tr` element, assuming there is one. If there is no such child, then jump to the step labeled *end* below.
10. If *table* has no row group corresponding to a `thead` element, then set *ignore first group* to true. Otherwise, set it to false.
11. *Row loop*: Let *rows* be an empty list of `tr` elements.
12. Run the appropriate steps from the following list:

↳ If *row collection cursor* points to a `tr` element

1. *Collect*: Append the element pointed to by *row collection cursor* to *rows*.
2. If there are no `tr` or `tbody` children of *table* that are later siblings of the element pointed to by *row collection cursor*, or if the next such child is a `tbody` element, then jump to the step labeled *group* below.
3. Let *row collection cursor* point to the next `tr` child of *table* that is a later sibling of the element pointed to by *row collection cursor*.
4. Returned to the step labeled *collect* above.

↳ If *row collection cursor* points to a `tbody` element

1. Place all the `tr` element children of the element pointed to by *row collection cursor* into *rows*, in tree order.
2. If *rows* is empty, jump to the step labeled *increment loop* below.

13. *Group*: Let *groups* be an empty list of groups of `tr` elements.

14. Let *group* be an empty group of `tr` elements.

15. Let *group cursor* be a pointer to an element, initially pointing at the first `tr` element in *rows*.

16. *Start group*: Let *pending rows in group* be 1.

17. *Group loop*: Append the `tr` element pointed to by *group cursor* to *group*.

18. If there are any cells whose highest row's element is the one pointed to by *group cursor*, then let *tallest height* be the number of rows covered by the tallest such cell.

19. If *tallest height* is greater than *pending rows in group* then set *pending rows in group* to *tallest height*.

20. Decrement *pending rows in group* by one.

21. Let *group cursor* point to the next `tr` element in *rows*, if any; otherwise, let it be null.

22. If *group cursor* is not null and *pending rows in group* is not zero, return to the step labeled *group loop*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

26. If *ignore first group* is true, then drop the first group in *groups* and set *ignore first group* to false.
27. *Drop leading header groups*: If *groups* is now empty, jump to the step labeled *increment loop* below.
28. If the first group of *groups* consists of *tr* elements whose element children are all *th* elements, then drop the first group in *groups* and return to the previous step (labeled *drop leading header groups*).
29. Let *insertion point* be a placeholder in a DOM tree, which can be used to reinsert nodes at a specific point in the DOM. Insert *insertion point* into the parent of the first *tr* element of the first group in *groups*, immediately before that *tr* element.

30. Sort either

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

ups a and b (the algorithm  
ist.

When *direction* is toggled,  
value is *descending*, it must

3. *Column loop*. Let *th* be the key index in *th* in *key heading cells*.
4. If *th*'s *sorted* attribute's column sort direction is *reversed*, then toggle *direction*.
5. Let *tentative order* be the result of comparing two row groups using the *th* element *th*, with *a* and *b* as the rows.
6. If *tentative order* is not "equal", then jump to the step labeled *return* below.
7. Increment *key index*.
8. If *key index* still denotes a *th* element in *key heading cells*, then jump back to the step above labeled *column loop*.
9. If *a*'s *tr* elements precede *b*'s in tree order, then let *tentative order* be "a before b". Otherwise, let *tentative order* be "b before a".

10. *Return*: Return the relative order given by the matching option from the following list:

- ↪ If *direction* is *ascending* and *tentative order* is "a before b"  
Return that *a* comes before *b*.
- ↪ If *direction* is *ascending* and *tentative order* is "b before a"  
Return that *b* comes before *a*.
- ↪ If *direction* is *descending* and *tentative order* is "a before b"  
Return that *b* comes before *a*.
- ↪ If *direction* is *descending* and *tentative order* is "b before a"  
Return that *a* comes before *b*.

When the user agent is required to **compare two row groups using the *th* element *th***, with *a* and *b* being the two row groups respectively, the user agent must run the following steps:

1. Let *x* be the *x*-coordinate of the slots that *th* covers in its table.
2. Let *cell<sub>a</sub>* be the element corresponding to the cell in the first row of group *a* that covers the slot in that row whose *x*-coordinate is *x*.

Let *cell<sub>b</sub>* be the element corresponding to the cell in the first row of group *b* that covers the slot in that row whose *x*-coordinate is *x*.

In either case, if there's no cell that actually covers the slot, then use the value *null* instead.

3. Let *type<sub>a</sub>* and *value<sub>a</sub>* be the type and value of the cell *cell<sub>a</sub>*, as defined below.

Let *type<sub>b</sub>* and *value<sub>b</sub>* be the type and value of the cell *cell<sub>b</sub>*, as defined below.

The **type and value of the cell** *cell* are computed as follows.

1. If *cell* is *null*, then the type is "string" and the value is the empty string; abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

and that child is a `time` element, then the value is the machine-readable equivalent of the element's contents, if any, and the type is the kind of value that is thus obtained (a month, a date, a yearless date, a time, a local date and time, a time-zone offset, a global date and time, a week, a year, or a duration); abort these steps after completing this one.

If there is no machine-readable equivalent, then the type is "string" and the value is the empty string.

If the type is a month, a date, a week, or a year, then change the value to be the instant in time (with no time zone) that describes the earliest moment that the value represents, and change the type to be a local date

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4. The value is the element's `textContent`. The type is "string".
  4. If `typea` and `typeb` are not equal, then: return "a before b" if `typea` is earlier in the following list than `typeb`, otherwise, return "b before a"; then, abort these steps.
    1. time
    2. yearless date
    3. local date and time
    4. global date and time
    5. time-zone offset
    6. duration
    7. "string"
  5. If `valuea` and `valueb` are equal, then return "equal" and abort these steps.
  6. If `typea` and `typeb` are not "string", then: return "a before b" if `valuea` is earlier than `valueb`, and return "b before a" otherwise.
- Values sort in their natural order, with the following additional constraints:
- For time values, 00:00:00.000 is the earliest value and 23:59:59.999 is the latest value.
- For yearless date values, 01-01 is the earliest value and 12-31 is the latest value; 02-28 is earlier than 02-29 which is earlier than 03-01.
- Values that are local date and time compare as if they were in the same time zone.
- For time-zone offset values, -23:59 is the earliest value and +23:59 is the latest value.
7. Return "a before b" if `valuea` is earlier than `valueb`, and return "b before a" otherwise; for these purposes, the strings must be compared using collation order of the language of the `table` element to which `th`'s table corresponds.

31. Let `new order` be a list of `tr` elements consisting of the `tr` elements of all the groups in the newly ordered `groups`, with the `tr` elements being in the same order as the groups to which they belong are in `groups`, and the `tr` elements within each such group themselves being ordered in tree order.
32. Remove all the `tr` elements in `new order` from their parents, in tree order.
33. Insert all the `tr` elements in `new order` into the DOM at the location of `insertion point`, in the order these elements are found in `new order`.
34. Remove `insertion point` from the DOM.
35. *Increment loop*: If there are no `tr` or `tbody` children of `table` that are later siblings of the element pointed to by `row collection cursor`, then jump to the step labeled `end` below.
36. Let `row collection cursor` point to the next `tr` or `tbody` child of `table` that is a later sibling of the element pointed to by `row collection cursor`.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

table element becomes a sorting-enabled table element, the table element is said to become a table with a pending sort.

When the user agent is to **sort the tables with pending sorts**, which happens during the perform a microtask checkpoint algorithm, the user agent must run the following algorithm:

1. Let *tables* be a list of each table in the unit of related similar-origin browsing contexts that is a table with a pending sort, in the order that they became such, with those that became such at the same time being listed in tree order.
2. Let all the table elements in *tables* no longer be tables with a pending sort.
3. App<sup>l</sup>

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

When the us

1. Let *target* be the table element in *tables* whose target is the current document.
2. If *target* is not a table element, then abort these steps.

ction is *normal*, set that

3. Let *current headers* be the sorting-enabled th elements of the table element *target*, excluding *target*.
4. Sort *current headers* by their sorted attributes' column key ordinality, in ascending order, with elements that have the same column key ordinality being sorted in tree order.
5. Let *level* be 2.
6. For each th element *th* in *current headers*, in order, run the following substeps:
  1. If *th*'s sorted attribute's column sort direction is *normal*, then set *th*'s sorted attribute to a valid integer whose value is *level*. Otherwise, set it to the concatenation of the string "reversed", a U+0020 SPACE character, and a valid integer whose value is *level*.
  2. Increment *level* by 1.
7. Set *target*'s sorted attribute to the empty string.

The activation behavior of a sorting interface th element is to set the sort key to the th element.

**Note: The table will be sorted the next time the user agent performs a microtask checkpoint.**

This box is non-normative. Implementation requirements are given below this box.

#### th.sort()

Act as if the user had indicated that this was to be the new primary sort column.

The table won't actually be sorted until the script terminates.

#### table.stopSorting()

Removes all the sorted attributes that are causing the table to automatically sort its contents, if any.

The th element's sort() method, when invoked, must run the following steps:

1. If the th element is not a sorting-capable th element, then abort these steps.
2. Set the sort key to the th element.

**Note: The table will be sorted the next time the user agent performs a microtask checkpoint.**

The table element's stopSorting() method, when invoked, must remove the sorted attribute of all the sorting-enabled th elements of the table element on which the method was invoked.

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<table>
 <caption>Specification values: Steel, Castings,
 Ann. A.S.T.M. A27-16, Class B;* P max. 0.06; S max. 0.05.</caption>
 <thead>
 <tr>
 <th rowspan=2>Grade.</th>
 <th rowspan=2>Yield Point.</th>
 <th colspan=2>Ultimate tensile strength</th>
 </tr>
 <th>56.2</th>
 <th>80,000</th>
 </thead>
 <tbody>
 <tr>
 <td>Hard</td>
 <td>0.45 ultimate</td>
 <td>56.2</td>
 <td>80,000</td>
 </tr>
 <tr>
 <td>Medium</td>
 <td>0.45 ultimate</td>
 <td>49.2</td>
 <td>70,000</td>
 </tr>
 <tr>
 <td>Soft</td>
 <td>0.45 ultimate</td>
 <td>42.2</td>
 <td>60,000</td>
 </tr>
 </tbody>
</table>
```

This table could look like this:

Specification values: **Steel**, **Castings**, Ann. A.S.T.M. A27-16, Class B;\* P max. 0.06; S max. 0.05.

Grade.	Yield Point.	Ultimate tensile strength		Per cent elong. 50.8 mm or 2 in.	Per cent reduct. area.
		kg/mm <sup>2</sup>	lb/in <sup>2</sup>		
Hard	0.45 ultimate	56.2	80,000	15	20
Medium	0.45 ultimate	49.2	70,000	18	25
Soft	0.45 ultimate	42.2	60,000	22	30

The following shows how one might mark up the gross margin table on page 46 of Apple, Inc's 10-K filing for fiscal year 2008:

```
<table>
 <thead>
 <tr>
 <th></th>
 </tr>
 <tbody>
 <tr>
 <th>2008</th>
 <th>2007</th>
 </tbody>
</table>
```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

```

<td>$ 24,006
<td>$ 19,315
<tr>
<th>Cost of sales
<td> 21,334
<td> 15,852
<td> 13,717
<tbody>
<tr>
<th>
<tc
<tc
<tc
<tfo
<tr>
<th>
<tc
<tc
<td>29.0%
</table>

```

✖

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

This table could look like this:

	<u>2008</u>	<u>2007</u>	<u>2006</u>
Net sales	\$ 32,479	\$ 24,006	\$ 19,315
Cost of sales	21,334	15,852	13,717
Gross margin	<u>\$ 11,145</u>	<u>\$ 8,154</u>	<u>\$ 5,598</u>
Gross margin percentage	34.3%	34.0%	29.0%

The following shows how one might mark up the operating expenses table from lower on the same page of that document:

```

<table>
<colgroup> <col>
<colgroup> <col> <col> <col>
<thead>
<tr> <th>2008 <th>2007 <th>2006
<tbody>
<tr> <th scope=rowgroup> Research and development
 <td> $ 1,109 <td> $ 782 <td> $ 712
<tr> <th scope=row> Percentage of net sales
 <td> 3.4% <td> 3.3% <td> 3.7%
<tbody>
<tr> <th scope=rowgroup> Selling, general, and administrative
 <td> $ 3,761 <td> $ 2,963 <td> $ 2,433
<tr> <th scope=row> Percentage of net sales
 <td> 11.6% <td> 12.3% <td> 12.6%
</table>

```

This table could look like this:

	<u>2008</u>	<u>2007</u>	<u>2006</u>
Research and development	\$ 1,109	\$ 782	\$ 712
Percentage of net sales	3.4%	3.3%	3.7%
Selling, general, and administrative	\$ 3,761	\$ 2,963	\$ 2,433
Percentage of net sales	11.6%	12.3%	12.6%

## 4.10 Forms

### 4.10.1 Introduction

*This section is non-normative.*

A form is a component of a Web page that has form controls, such as text fields, buttons, checkboxes, range controls, or color pickers. A user can interact with such a form, providing data that can then be sent to the server for further processing (e.g. returning the results

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

#### 4.10.1.1 Writing a form's user interface

*This section is non-normative.*

For the purposes of this brief introduction, we will create a pizza ordering form.

Any form starts with a `form` element, inside which are placed the controls. Most controls are represented by the `input` element, which by default provides a one-line text field. To label a control, the `label` element is used; the `label` and the control itself go inside the `label` element.

Put this together: **Too slow? Try reading the multipage copy of**

`<form>` **the spec instead: https://web.archive.org**  
`<p><1>` **/web/20130103030411/http:**

To let the us **a type attrib**

`input` element, this time with  
on name using the `name`

attribute. To group a batch of controls together, such as, in this case, the radio buttons, one can use the `fieldset` element. The title of such a group of controls is given by the first element in the `fieldset`, which has to be a `legend` element.

```
<form>
<p><label>Customer name: <input></label></p>
<fieldset>
 <legend> Pizza Size </legend>
 <p><label> <input type=radio name=size> Small </label></p>
 <p><label> <input type=radio name=size> Medium </label></p>
 <p><label> <input type=radio name=size> Large </label></p>
</fieldset>
</form>
```

**Note: Changes from the previous step are highlighted.**

To pick toppings, we can use checkboxes. These use the `input` element with a `type` attribute with the value `checkbox`:

```
<form>
<p><label>Customer name: <input></label></p>
<fieldset>
 <legend> Pizza Size </legend>
 <p><label> <input type=radio name=size> Small </label></p>
 <p><label> <input type=radio name=size> Medium </label></p>
 <p><label> <input type=radio name=size> Large </label></p>
</fieldset>
<fieldset>
 <legend> Pizza Toppings </legend>
 <p><label> <input type=checkbox> Bacon </label></p>
 <p><label> <input type=checkbox> Extra Cheese </label></p>
 <p><label> <input type=checkbox> Onion </label></p>
 <p><label> <input type=checkbox> Mushroom </label></p>
</fieldset>
</form>
```

The pizzeria for which this form is being written is always making mistakes, so it needs a way to contact the customer. For this purpose, we can use form controls specifically for telephone numbers (`input` elements with their `type` attribute set to `tel`) and e-mail addresses (`input` elements with their `type` attribute set to `email`):

```
<form>
<p><label>Customer name: <input></label></p>
<p><label>Telephone: <input type=tel></label></p>
<p><label>E-mail address: <input type=email></label></p>
<fieldset>
 <legend> Pizza Size </legend>
 <p><label> <input type=radio name=size> Small </label></p>
 <p><label> <input type=radio name=size> Medium </label></p>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<p><label> <input type=checkbox> Extra Cheese </label></p>
<p><label> <input type=checkbox> Onion </label></p>
<p><label> <input type=checkbox> Mushroom </label></p>
</fieldset>
</form>
```

We can use an `input` element with its `type` attribute set to `time` to ask for a delivery time. Many of these form controls have attributes to control exactly what values can be specified; in this case, three attributes of particular interest are `min`, `max`, and `step`. These set the minimum time, the maximum time, and the interval between allowed values (in seconds). This pizzeria only delivers between 11a

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

/web/20130103030411/http://whatwg.org/html

```
<form>
<p><l
<p><l
<p><l
<fiel
<leq
<p><
<p><la
<p><la
<p><la
<p><la
<p><la
<fiel
<fiel
<legende
<p><la
<p><la
<p><la
<p><la
<p><la
<fiel
<fiel
<legend> Pizza Toppings </legend>
<p><label> <input type=checkbox> Bacon </label></p>
<p><label> <input type=checkbox> Extra Cheese </label></p>
<p><label> <input type=checkbox> Onion </label></p>
<p><label> <input type=checkbox> Mushroom </label></p>
<fiel
<p><label>Preferred delivery time: <input type=time min="11:00" max="21:00" step="900"></label>
<p>
</form>
```

The `textarea` element can be used to provide a free-form text field. In this instance, we are going to use it to provide a space for the customer to give delivery instructions:

```
<form>
<p><label>Customer name: <input></label></p>
<p><label>Telephone: <input type=tel></label></p>
<p><label>E-mail address: <input type=email></label></p>
<fieldset>
<legend> Pizza Size </legend>
<p><label> <input type=radio name=size> Small </label></p>
<p><label> <input type=radio name=size> Medium </label></p>
<p><label> <input type=radio name=size> Large </label></p>
<fieldset>
<fieldset>
<legend> Pizza Toppings </legend>
<p><label> <input type=checkbox> Bacon </label></p>
<p><label> <input type=checkbox> Extra Cheese </label></p>
<p><label> <input type=checkbox> Onion </label></p>
<p><label> <input type=checkbox> Mushroom </label></p>
<fieldset>
<p><label>Preferred delivery time: <input type=time min="11:00" max="21:00" step="900">
</label></p>
<p><label>Delivery instructions: <textarea></textarea></label></p>
</form>
```

Finally, to make the form submittable we use the `button` element:

```
<form>
<p><label>Customer name: <input></label></p>
<p><label>Telephone: <input type=tel></label></p>
<p><label>E-mail address: <input type=email></label></p>
<fieldset>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<fieldset>
 <legend> Pizza Toppings </legend>
 <p><label> <input type=checkbox> Bacon </label></p>
 <p><label> <input type=checkbox> Extra Cheese </label></p>
 <p><label> <input type=checkbox> Onion </label></p>
 <p><label> <input type=checkbox> Mushroom </label></p>
</fieldset>
<p><label>Preferred delivery time: <input type=time min="11:00" max="21:00" step="900">
</label></p>
<p><1
<p><t Too slow? Try reading the multipage copy of
</form> the spec instead: https://web.archive.org
```

#### 4.10.1.2 /web/20130103030411/http:

*This section* [//whatwg.org/html](http://whatwg.org/html)

The exact details of this introduction, we will assume that the script at <https://pizza.example.com/order.cgi> is configured to accept submissions using the application/x-www-form-urlencoded format, expecting the following parameters sent in an HTTP POST body:

**custname**

Customer's name

**custtel**

Customer's telephone number

**custemail**

Customer's e-mail address

**size**

The pizza size, either small, medium, or large

**toppings**

The topping, specified once for each selected topping, with the allowed values being bacon, cheese, onion, and mushroom

**delivery**

The requested delivery time

**comments**

The delivery instructions

#### 4.10.1.3 Configuring a form to communicate with a server

*This section is non-normative.*

Form submissions are exposed to servers in a variety of ways, most commonly as HTTP GET or POST requests. To specify the exact method used, the `method` attribute is specified on the `form` element. This doesn't specify how the form data is encoded, though; to specify that, you use the `enctype` attribute. You also have to specify the URL of the service that will handle the submitted data, using the `action` attribute.

For each form control you want submitted, you then have to give a name that will be used to refer to the data in the submission. We already specified the name for the group of radio buttons; the same attribute (`name`) also specifies the submission name. Radio buttons can be distinguished from each other in the submission by giving them different values, using the `value` attribute.

Multiple controls can have the same name; for example, here we give all the checkboxes the same name, and the server distinguishes which checkbox was checked by seeing which values are submitted with that name — like the radio buttons, they are also given unique values with the `value` attribute.

Given the settings in the previous section, this all becomes:

```
<form method="post"
 enctype="application/x-www-form-urlencoded"
 action="https://pizza.example.com/order.cgi">
```

Go

**NOV JAN FEB**

03

2012 2013 2014

A row of three social media icons: a black circle with a white person icon, a blue circle with a white question mark icon, a black circle with a white X icon, a blue square with a white 'f' icon for Facebook, and a blue square with a white bird icon for Twitter.

3,445 captures

5 Jun 2004 - 22 Aug 2020

```
<p><label> <input type=radio name=size value="small"value="medium"value="large"name="topping" value="bacon"name="topping" value="cheese"name="topping" value="onion"name="topping" value="pepperoni"
```

**The HTML syntax allows a variety of equally valid ways to specify attributes, as discussed in the syntax section.**

For example, if the customer entered "Denise Lawrence" as their name, "555-321-8642" as their telephone number, did not specify an e-mail address, asked for a medium-sized pizza, selected the Extra Cheese and Mushroom toppings, entered a delivery time of 7pm, and left the delivery instructions text field blank, the user agent would submit the following to the online Web service:

custname=Denise+Lawrence&custtel=555-321-8624&custemail=&size=medium&topping=cheese&topping=mushroom&delivery=19%3A00&comments=

#### **4.10.1.4 Client-side form validation**

*This section is non-normative.*

Forms can be annotated in such a way that the user agent will check the user's input before the form is submitted. The server still has to verify the input is valid (since hostile users can easily bypass the form validation), but it allows the user to avoid the wait incurred by having the server be the sole checker of the user's input.

The simplest annotation is the `required` attribute, which can be specified on `input` elements to indicate that the form is not to be submitted until a value is given. By adding this attribute to the customer name, pizza size, and delivery time fields, we allow the user agent to notify the user when the user submits the form without filling in those fields:

```
<form method="post"
 enctype="application/x-www-form-urlencoded"
 action="https://pizza.example.com/order.cgi">
<p><label>Customer name: <input name="custname" required></label></p>
<p><label>Telephone: <input type=tel name="custtel"></label></p>
<p><label>E-mail address: <input type=email name="custemail"></label></p>
<fieldset>
 <legend> Pizza Size </legend>
 <p><label> <input type=radio name=size required value="small"> Small </label></p>
 <p><label> <input type=radio name=size required value="medium"> Medium </label></p>
 <p><label> <input type=radio name=size required value="large"> Large </label></p>
</fieldset>
<fieldset>
 <legend> Pizza Toppings </legend>
 <p><label> <input type=checkbox name="topping" value="bacon"> Bacon </label></p>
 <p><label> <input type=checkbox name="topping" value="cheese"> Extra Cheese </label></p>
 <p><label> <input type=checkbox name="topping" value="onion"> Onion </label></p>
 <p><label> <input type=checkbox name="topping" value="mushroom"> Mushroom </label></p>
</fieldset>
<p><label>Preferred delivery time: <input type=time min="11:00" max="21:00" step="900"
name="delivery" required></label></p>
<p><label>Delivery instructions: <textarea name="comments"></textarea></label></p>
<p><button>Submit order</button></p>
</form>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

 enctype="application/x-www-form-urlencoded"
 action="https://pizza.example.com/order.cgi">
<p><label>Customer name: <input name="custname" required></label></p>
<p><label>Telephone: <input type="tel" name="custtel"></label></p>
<p><label>E-mail address: <input type="email" name="custemail"></label></p>
<fieldset>
 <legend> Pizza Size </legend>
 <p><label> <input type="radio" name="size" required value="small"> Small </label></p>
 <p><label> <input type="radio" name="size" required value="medium"> Medium </label></p>
 <p><label>
</fiel Too slow? Try reading the multipage copy of
<field the spec instead: https://web.archive.org
<legc />< />
<p>< />
<p>< />
<p>< />
<p>< />
</fie.
<p><label>Preferred delivery time: <input type="time" min="11:00" max="21:00" step="900" name="delivery" required></label></p>
<p><label>Delivery instructions: <textarea name="comments" maxlength=1000></textarea>
</label></p>
<p><button>Submit order</button></p>
</form>

```

#### 4.10.1.5 Enabling client-side automatic filling of form controls

*This section is non-normative.*

Some browsers attempt to aid the user by automatically filling form controls rather than having the user reenter their information each time. For example, a field asking for the user's telephone number can be automatically filled with the user's phone number.

To help the user agent with this, we can tell it what the field is using the `autocomplete` attribute. In the case of this form, we have three fields that can be usefully annotated in this way: the information about who the pizza is to be delivered to. Adding this information looks like this:

```

<form method="post"
 enctype="application/x-www-form-urlencoded"
 action="https://pizza.example.com/order.cgi">
<p><label>Customer name: <input name="custname" required autocomplete="shipping name">
</label></p>
<p><label>Telephone: <input type="tel" name="custtel" autocomplete="shipping tel"></label></p>
<p><label>E-mail address: <input type="email" name="custemail" autocomplete="shipping email">
</label></p>
<fieldset>
 <legend> Pizza Size </legend>
 <p><label> <input type="radio" name="size" required value="small"> Small </label></p>
 <p><label> <input type="radio" name="size" required value="medium"> Medium </label></p>
 <p><label> <input type="radio" name="size" required value="large"> Large </label></p>
</fieldset>
<fieldset>
 <legend> Pizza Toppings </legend>
 <p><label> <input type="checkbox" name="topping" value="bacon"> Bacon </label></p>
 <p><label> <input type="checkbox" name="topping" value="cheese"> Extra Cheese </label></p>
 <p><label> <input type="checkbox" name="topping" value="onion"> Onion </label></p>
 <p><label> <input type="checkbox" name="topping" value="mushroom"> Mushroom </label></p>
</fieldset>
<p><label>Preferred delivery time: <input type="time" min="11:00" max="21:00" step="900" name="delivery" required></label></p>
<p><label>Delivery instructions: <textarea name="comments" maxlength=1000></textarea>
</label></p>
<p><button>Submit order</button></p>
</form>

```

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

can provide the user with multiple input modalities. For example, when typing in a credit card number the user may wish to only see keys for digits 0-9, while when typing in their name they may wish to see a form field that by default capitalises each word.

Using the `inputmode` attribute we can select appropriate input modalities:

```
<form method="post"
 enctype="application/x-www-form-urlencoded"
 action="https://pizza.example.com/order.cgi">
 <p><label>Customer name: <input name="custname" required autocomplete="shipping name"
 inputmode="latin-name"></label></p>
 <p><1 Too slow? Try reading the multipage copy of
</labe the spec instead: https://web.archive.org
<fiel />
 <le<
 <p>< /web/20130103030411/http:
 <p>< //whatwg.org/html
</fie.
 <fieldset>
 <legend> Pizza Toppings </legend>
 <p><label> <input type="checkbox" name="topping" value="bacon"> Bacon </label></p>
 <p><label> <input type="checkbox" name="topping" value="cheese"> Extra Cheese </label></p>
 <p><label> <input type="checkbox" name="topping" value="onion"> Onion </label></p>
 <p><label> <input type="checkbox" name="topping" value="mushroom"> Mushroom </label></p>
 </fieldset>
 <p><label>Preferred delivery time: <input type="time" min="11:00" max="21:00" step="900"
 name="delivery" required></label></p>
 <p><label>Delivery instructions: <textarea name="comments" maxlength=1000 inputmode="latin-
 prose"></textarea></label></p>
 <p><button>Submit order</button></p>
</form>
```

### 4.10.1.7 The difference between the field type, the autofocus field name, and the input modality

*This section is non-normative.*

The `type`, `autocomplete`, and `inputmode` attributes can seem confusingly similar. For instance, in all three cases, the string "email" is a valid value. This section attempts to illustrate the difference between the three attributes and provides advice suggesting how to use them.

The `type` attribute on `input` elements decides what kind of control the user agent will use to expose the field. Chosing between different values of this attribute is the same choice as chosing whether to use an `input` element, a `textarea` element, a `select` element, a `keygen` element, etc.

The `autocomplete` attribute, in contrast, describes what the value that the user will enter actually represents. Chosing between different values of this attribute is the same choice as chosing what the label for the element will be.

First, consider telephone numbers. If a page is asking for a telephone number from the user, the right form control to use is `<input type="tel">`. However, which `autocomplete` value to use depends on which phone number the page is asking for, whether they expect a telephone number in the international format or just the local format, and so forth.

For example, a page that forms part of a checkout process on an e-commerce site for a customer buying a gift to be shipped to a friend might need both the buyer's telephone number (in case of payment issues) and the friend's telephone number (in case of delivery issues). If the site expects international phone numbers (with the country code prefix), this could thus look like this:

```
<p><label>Your phone number: <input type="tel" name="custtel" autocomplete="billing tel"></label>
<p><label>Recipient's phone number: <input type="tel" name="shiptel" autocomplete="shipping
 tel"></label>
<p>Please enter complete phone numbers including the country code prefix, as in "+1 555 123
 4567".
```

But if the site only supports British customers and recipients, it might instead look like this (notice the use of `tel-national` rather than `tel`):

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Now, consider a person's preferred languages. The right `autocomplete` value is `language`. However, there could be a number of different form controls used for the purpose: a free text field (`<input type="text">`), a drop-down list (`<select>`), radio buttons (`<input type="radio">`), etc. It only depends on what kind of interface is desired.

The `inputmode` decides what kind of input modality (e.g. keyboard) to use, when the control is a free-form text field.

Consider names. If a page just wants one name from the user, then the relevant control is `<input type="text">`. If the page is asking for the user's full name, then the relevant `autocomplete` value is `name`. But if the user is Japanese and the page is asking for the user's Japanese name, the `inputmode` is defaulted to a Japanese input mode (e.g. each character).

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```
<p><label>
<input type="text" name="jp_name" inputmode="text" />
<label>
<input type="text" name="name" inputmode="text" />
</p>
```

In this example, the two fields expect different names. Without them, the user agent could automatically fill the second field with the value given in the first field when the user gave a value to the first field.

**Note:** The "`-jp`" and "`-en`" parts of the keywords are opaque to the user agent; the user agent cannot guess, from those, that the two names are expected to be in Japanese and English respectively.

#### 4.10.1.8 Date, time, and number formats

This section is non-normative.

In this pizza delivery example, the times are specified in the format "HH:MM": two digits for the hour, in 24-hour format, and two digits for the time. (Seconds could also be specified, though they are not necessary in this example.)

In some locales, however, times are often expressed differently when presented to users. For example, in the United States, it is still common to use the 12-hour clock with an am/pm indicator, as in "2pm". In France, it is common to separate the hours from the minutes using an "h" character, as in "14h00".

Similar issues exist with dates, with the added complication that even the order of the components is not always consistent — for example, in Cyprus the first of February 2003 would typically be written "1/2/03", while that same date in Japan would typically be written as "2003年02月01日" — and even with numbers, where locales differ, for example, in what punctuation is used as the decimal separator and the thousands separator.

It is therefore important to distinguish the time, date, and number formats used in HTML and in form submissions, which are always the formats defined in this specification (and based on the well-established ISO 8601 standard for computer-readable date and time formats), from the time, date, and number formats presented to the user by the browser and accepted as input from the user by the browser.

The format used "on the wire", i.e. in HTML markup and in form submissions, is intended to be computer-readable and consistent irrespective of the user's locale. Dates, for instance, are always written in the format "YYYY-MM-DD", as in "2003-02-01". Users are not expected to ever see this format.

The time, date, or number given by the page in the wire format is then translated to the user's preferred presentation (based on user preferences or on the locale of the page itself), before being displayed to the user. Similarly, after the user inputs a time, date, or number using their preferred format, the user agent converts it back to the wire format before putting it in the DOM or submitting it.

This allows scripts in pages and on servers to process times, dates, and numbers in a consistent manner without needing to support dozens of different formats, while still supporting the users' needs.

**Note:** See also the implementation notes regarding localization of form controls.

#### 4.10.2 Categories

Mostly for historical reasons, elements in this section fall into several overlapping (but subtly different) categories in addition to the usual ones like flow content, phrasing content, and interactive content.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The form-associated elements fall into several subcategories:

### Listed elements

Denotes elements that are listed in the `form.elements` and `fieldset.elements` APIs.

⇒ `button`, `fieldset`, `input`, `keygen`, `object`, `output`, `select`, `textarea`

### Submittable elements

Denotes elements that can be used for constructing the form data set when a `form` element is submitted.



Son Too slow? Try reading the multipage copy of  
butt the spec instead: <https://web.archive.org>  
Resettable /web/20130103030411/http://whatwg.org/html

Some elements, not all of them form-associated, are categorized as **labelable elements**. These are elements that can be associated with a `label` element.

⇒ `button`, `input` (if the `type` attribute is *not* in the hidden state), `keygen`, `meter`, `output`, `progress`, `select`, `textarea`

## 4.10.3 The `form` element

### Categories:

Flow content.

Palpable content.

### Contexts in which this element can be used:

Where flow content is expected.

### Content model:

Flow content, but with no `form` element descendants.

### Content attributes:

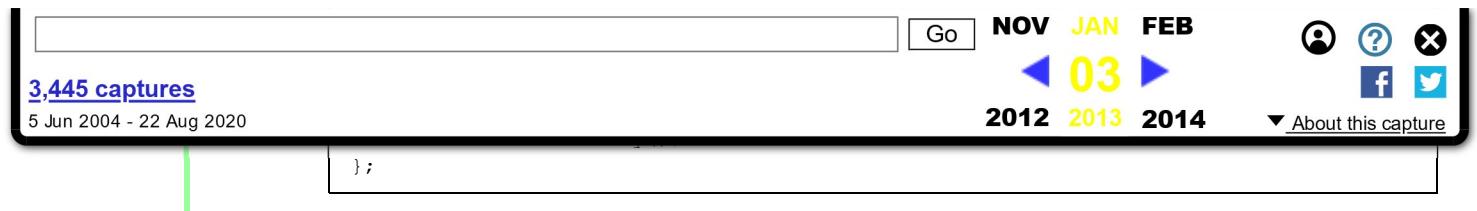
Global attributes

`accept-charset`  
`action`  
`autocomplete`  
`enctype`  
`method`  
`name`  
`novalidate`  
`target`

### DOM interface:

```
IDL [OverrideBuiltins]
interface HTMLFormElement : HTMLElement {
 attribute DOMString acceptCharset;
 attribute DOMString action;
 attribute DOMString autocomplete;
 attribute DOMString enctype;
 attribute DOMString encoding;
 attribute DOMString method;
 attribute DOMString name;
 attribute boolean noValidate;
 attribute DOMString target;

 readonly attribute HTMLFormControlsCollection elements;
 readonly attribute long length;
 getter Element (unsigned long index);
}
```



The `form` element represents a collection of form-associated elements, some of which can represent editable values that can be submitted to a server for processing.

The `accept-charset` attribute gives the character encodings that are to be used for the submission. If specified, the value must be an ordered set of unique space-separated tokens that are ASCII case-insensitive, and each token must be an ASCII case-insensitive match for the preferred MIME name of an ASCII-compatible character encoding. IANACHARS

The `name` attribute must be unique. The `autocomplete` keyword controls in the form.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

empty string, and the value is `on`. This applies to the `on` state, and the `off` state. The `off` state indicates that by default, form

The action, encrypt, merge, noauth, novalidate, and target attributes are attributes for form submission.

*This box is non-normative. Implementation requirements are given below this box.*

### *form . elements*

Returns an `HTMLCollection` of the form controls in the form (excluding image buttons for historical reasons).

***form.length***

Returns the number of form controls in the form (excluding image buttons for historical reasons).

*form*[*index*]

Returns the *index*th element in the form (excluding image buttons for historical reasons).

*form[name]*

Returns the form control (or, if there are several, a `RadioNodeList` of the form controls) in the form with the given ID or name (excluding image buttons for historical reasons); or, if there are none, returns the `img` element with the given ID.

Once an element has been referenced using a particular name, that name will continue being available as a way to reference that element in this method, even if the element's actual ID or name changes, for as long as the element remains in the Document.

If there are multiple matching items, then a `RadioNodeList` object containing all those elements is returned.

*form*.submit()

**Submits the form.**

*form*.reset()

**Resets the form**

*form*.checkValidity()

Returns true if the form's controls are all valid; otherwise, returns false.

The `autocomplete` IDL attribute must reflect the content attribute of the same name, limited to only known values.

The `name` IDL attribute must reflect the content attribute of the same name.

The `acceptCharset` IDL attribute must reflect the `accept-charset` content attribute.

The `elements` IDL attribute must return an `HTMLFormControlsCollection` rooted at the `Document` node while the `form` element is in a `Document` and rooted at the `form` element itself when it is not, whose filter matches listed elements whose form owner is the `form` element, with the exception of `input` elements whose `type` attribute is in the Image Button state, which must, for historical reasons, be excluded from this particular collection.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

When a `form` element is **indexed for indexed property retrieval**, the user agent must return the value returned by the `item` method on the `elements` collection, when invoked with the given index as its argument.

Each `form` element has a mapping of names to elements called the **past names map**. It is used to persist names of controls even when they change names.

The supported property names consist of the values of all the `id` and `name` attributes of all the listed elements and `img` elements that are descendants of the `form` element, and all the names currently in the past names map.

When a `for`

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

1. Let `candidates` be an empty `NodeList`.
2. If `candidates` contains more than one node, return `candidates` and abort these steps.
3. If `candidates` contains exactly one node, add a mapping from `name` to the node in `candidates` in the `form` element's past names map, replacing the previous entry with the same name, if any.

4. If `candidates` contains more than one node, return `candidates` and abort these steps.
5. Otherwise, `candidates` contains exactly one node. Add a mapping from `name` to the node in `candidates` in the `form` element's past names map, replacing the previous entry with the same name, if any.
6. Return the node in `candidates`.

If an element listed in the `form` element's past names map is removed from the Document, then its entries must be removed from the map.

The `submit()` method, when invoked, must submit the `form` element from the `form` element itself, with the `submitted from submit()` method flag set.

The `reset()` method, when invoked, must run the following steps:

1. If the `form` element is marked as `locked for reset`, then abort these steps.
2. Mark the `form` element as `locked for reset`.
3. Reset the `form` element.
4. Unmark the `form` element as `locked for reset`.

If the `checkValidity()` method is invoked, the user agent must statically validate the constraints of the `form` element, and return true if the constraint validation return a *positive* result, and false if it returned a *negative* result.

This example shows two search forms:

```
<form action="http://www.google.com/search" method="get">
 <label>Google: <input type="search" name="q"></label> <input type="submit"
 value="Search...">
</form>
<form action="http://www.bing.com/search" method="get">
 <label>Bing: <input type="search" name="q"></label> <input type="submit"
 value="Search...">
</form>
```

### 4.10.4 The `fieldset` element

#### Categories:

- Flow content.
- Sectioning root.
- Listed form-associated element.
- Palpable content.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**Content attributes:**

Global attributes  
**disabled**  
**form**  
**name**

**DOM interface:**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
readonly attribute boolean willValidate;
readonly attribute ValidityState validity;
readonly attribute DOMString validationMessage;
boolean checkValidity();
void setCustomValidity(DOMString error);
};
```

The `fieldset` element represents a set of form controls optionally grouped under a common name.

The name of the group is given by the first `legend` element that is a child of the `fieldset` element, if any. The remainder of the descendants form the group.

The `disabled` attribute, when specified, causes all the form control descendants of the `fieldset` element, excluding those that are descendants of the `fieldset` element's first `legend` element child, if any, to be disabled.

The `form` attribute is used to explicitly associate the `fieldset` element with its form owner. The `name` attribute represents the element's name.

*This box is non-normative. Implementation requirements are given below this box.*

**fieldset . type**

Returns the string "fieldset".

**fieldset . elements**

Returns an `HTMLFormControlsCollection` of the form controls in the element.

The `disabled` IDL attribute must reflect the content attribute of the same name.

The `type` IDL attribute must return the string "fieldset".

The `elements` IDL attribute must return an `HTMLFormControlsCollection` rooted at the `fieldset` element, whose filter matches listed elements.

The `willValidate`, `validity`, and `validationMessage` attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `form` and `name` IDL attributes are part of the element's forms API.

This example shows a `fieldset` element being used to group a set of related controls:

```
<fieldset>
<legend>Display</legend>
<p><label><input type=radio name=c value=0 checked> Black on White</label>
<p><label><input type=radio name=c value=1> White on Black</label>
<p><label><input type=checkbox name=g> Use grayscale</label>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
</datalist>
</fieldset>
```

The following snippet shows a fieldset with a checkbox in the legend that controls whether or not the fieldset is enabled. The contents of the fieldset consist of two required text fields and an optional year/month control.

```
<fieldset name="clubfields" disabled>
 <legend> <label>
 <input type=checkbox name=club onchange="form.clubfields.disabled = !checked">
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

You can also see so:

```
<.
 <legend> <label>
 <input type=checkbox name=club onchange="form.clubfields.disabled = !checked">
 Use Club Card
 </label> </legend>
 <p><label>Name on card: <input name=clubname required></label></p>
 <fieldset name="numfields">
 <legend> <label>
 <input type=radio checked name=clubtype onchange="form.numfields.disabled = !checked">
 My card has numbers on it
 </label> </legend>
 <p><label>Card number: <input name=clubnum required pattern="[-0-9]+"></label></p>
 </fieldset>
 <fieldset name="letfields" disabled>
 <legend> <label>
 <input type=radio name=clubtype onchange="form.letfields.disabled = !checked">
 My card has letters on it
 </label> </legend>
 <p><label>Card code: <input name=clublet required pattern="[A-Za-z]+>"></label></p>
 </fieldset>
</fieldset>
```

In this example, if the outer "Use Club Card" checkbox is not checked, everything inside the outer fieldset, including the two radio buttons in the legends of the two nested fieldsets, will be disabled. However, if the checkbox is checked, then the radio buttons will both be enabled and will let you select which of the two inner fieldsets is to be enabled.

#### 4.10.5 The legend element

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**Contexts in which this element can be used:**As the first child of a `fieldset` element.**Content model:**

Phrasing content.

**Content attributes:**

Global attributes

**DOM interface:**

( Too slow? Try reading the multipage copy of  
 the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `legend` element is a child of a `fieldset` element, if any.This box is non-normative. Implementation requirements are given below this box.**`legend`.`form`**Returns the element's `form` element, if any, or null otherwise.

The `form` IDL attribute's behavior depends on whether the `legend` element is in a `fieldset` element or not. If the `legend` has a `fieldset` element as its parent, then the `form` IDL attribute must return the same value as the `form` IDL attribute on that `fieldset` element. Otherwise, it must return null.

## 4.10.6 The `label` element

**Categories:**

- Flow content.
- Phrasing content.
- Interactive content.
- Form-associated element.
- Palpable content.

**Contexts in which this element can be used:**

Where phrasing content is expected.

**Content model:**Phrasing content, but with no descendant labelable elements unless it is the element's labeled control, and no descendant `label` elements.**Content attributes:**

- Global attributes
- `form`
- `for`

**DOM interface:**

```
IDL interface HTMLLabelElement : HTMLElement {
 readonly attribute HTMLFormElement? form;
 attribute DOMString htmlFor;
 readonly attribute HTMLElement? control;
};
```

The `label` represents a caption in a user interface. The caption can be associated with a specific form control, known as the `label` element's **labeled control**, either using `for` attribute, or by putting the form control inside the `label` element itself.

Except where otherwise specified by the following rules, a `label` element has no labeled control.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

If the `for` attribute is not specified, but the `label` element has a labelable element descendant, then the first such descendant in tree order is the `label` element's labeled control.

The `label` element's exact default presentation and behavior, in particular what its activation behavior might be, if anything, should match the platform's label behavior. The activation behavior of a `label` element for events targeted at interactive content descendants of a `label` element, and any descendants of those interactive content descendants, must be to do nothing.

For example, on platforms where clicking a checkbox label checks the checkbox, clicking the `label` in the following snippet could trigger the user agent to run synthetic click activation steps on the `input` element, even though the element itself had been triggered.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The `form` attribute is part of the element's forms API.

The `control` attribute is part of the element's forms API. It returns the right form control for users.

```
<p><label>Full name: <input name=fn> <small>Format: First Last</small></label></p>
<p><label>Age: <input name=age type=number min=0></label></p>
<p><label>Post code: <input name=pc> <small>Format: AB12 3CD</small></label></p>
```

*This box is non-normative. Implementation requirements are given below this box.*

#### `label.control`

Returns the form control that is associated with this element.

The `htmlFor` IDL attribute must reflect the `for` content attribute.

The `control` IDL attribute must return the `label` element's labeled control, if any, or null if there isn't one.

The `form` IDL attribute is part of the element's forms API.

*This box is non-normative. Implementation requirements are given below this box.*

#### `control.labels`

Returns a `NodeList` of all the `label` elements that the form control is associated with.

Labelable elements have a `NodeList` object associated with them that represents the list of `label` elements, in tree order, whose labeled control is the element in question. The `labels` IDL attribute of labelable elements, on getting, must return that `NodeList` object.

## 4.10.7 The `input` element

### Categories:

- Flow content.
- Phrasing content.
- If the `type` attribute is *not* in the Hidden state: Interactive content.
- If the `type` attribute is *not* in the Hidden state: Listed, labelable, submittable, and resettable form-associated element.
- If the `type` attribute is in the Hidden state: Listed, submittable, and resettable form-associated element.
- If the `type` attribute is *not* in the Hidden state: Palpable content.

### Contexts in which this element can be used:

- Where phrasing content is expected.

### Content model:

- Empty.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

autocomplete  
autofocus  
checked  
dirname  
disabled  
form  
formaction  
formenctype  
formmethod  
f

f Too slow? Try reading the multipage copy of  
h the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>  
l  
m

multiple  
name  
pattern  
placeholder  
readonly  
required  
size  
src  
step  
type  
value  
width

**DOM interface:**

```
interface HTMLInputElement : HTMLElement {
 attribute DOMString accept;
 attribute DOMString alt;
 attribute DOMString autocomplete;
 attribute boolean autofocus;
 attribute boolean defaultChecked;
 attribute boolean checked;
 attribute DOMString dirName;
 attribute boolean disabled;
 readonly attribute HTMLFormElement? form;
 readonly attribute FileList? files;
 attribute DOMString formAction;
 attribute DOMString formEnctype;
 attribute DOMString formMethod;
 attribute boolean formNoValidate;
 attribute DOMString formTarget;
 attribute unsigned long height;
 attribute boolean indeterminate;
 attribute DOMString inputMode;
 readonly attribute HTMLElement? list;
 attribute DOMString max;
 attribute long maxLength;
 attribute DOMString min;
 attribute boolean multiple;
 attribute DOMString name;
 attribute DOMString pattern;
 attribute DOMString placeholder;
 attribute boolean readOnly;
 attribute boolean required;
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
[TreatNullAs=EmptyString] attribute DOMString value;
attribute Date? valueAsDate;
attribute unrestricted double valueAsNumber;
attribute unsigned long width;

void stepUp(optional long n);
void stepDown(optional long n);

readonly attribute boolean willValidate;
```



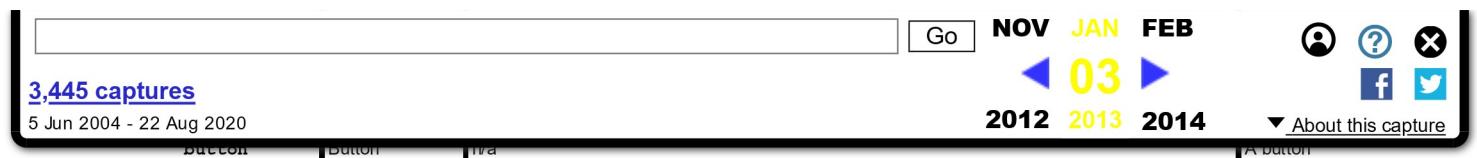
Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
attribute unsigned long selectionEnd;
attribute DOMString selectionDirection;
void setRangeText(DOMString replacement);
void setRangeText(DOMString replacement, unsigned long start, unsigned long end,
optional SelectionMode selectionMode);
void setSelectionRange(unsigned long start, unsigned long end, optional DOMString
direction);
};
```

The `input` element represents a typed data field, usually with a form control to allow the user to edit the data.

The `type` attribute controls the data type (and associated control) of the element. It is an enumerated attribute. The following table lists the keywords and states for the attribute — the keywords in the left column map to the states in the cell in the second column on the same row as the keyword.

Keyword	State	Data type	Control type
<code>hidden</code>	Hidden	An arbitrary string	n/a
<code>text</code>	Text	Text with no line breaks	A text field
<code>search</code>	Search	Text with no line breaks	Search field
<code>tel</code>	Telephone	Text with no line breaks	A text field
<code>url</code>	URL	An absolute URL	A text field
<code>email</code>	E-mail	An e-mail address or list of e-mail addresses	A text field
<code>password</code>	Password	Text with no line breaks (sensitive information)	A text field that obscures data entry
<code>datetime</code>	Date and Time	A date and time (year, month, day, hour, minute, second, fraction of a second) with the time zone set to UTC	A date and time control
<code>date</code>	Date	A date (year, month, day) with no time zone	A date control
<code>month</code>	Month	A date consisting of a year and a month with no time zone	A month control
<code>week</code>	Week	A date consisting of a week-year number and a week number with no time zone	A week control
<code>time</code>	Time	A time (hour, minute, seconds, fractional seconds) with no time zone	A time control
<code>datetime-local</code>	Local Date and Time	A date and time (year, month, day, hour, minute, second, fraction of a second) with no time zone	A date and time control
<code>number</code>	Number	A numerical value	A text field or spinner control
<code>range</code>	Range	A numerical value, with the extra semantic that the exact value is not important	A slider control or similar
<code>color</code>	Color	An sRGB color with 8-bit red, green, and blue components	A color well
<code>checkbox</code>	Checkbox	A set of zero or more values from a predefined list	A checkbox
<code>radio</code>	Radio Button	An enumerated value	A radio button
<code>file</code>	File Upload	Zero or more files each with a MIME type and optionally a file name	A label and a button
<code>submit</code>	Submit Button	An enumerated value, with the extra semantic that it must be the last value selected and initiates form submission	A button



The *missing value default* is the Text state.

Which of the accept, alt, autocomplete, checked, dirname, formaction, formenctype, formmethod, formnovalidate, formtarget, height, inputmode, list, max, maxlength, min, multiple, pattern, placeholder, readonly, required, size, src, step, and width **content attributes**, the checked, files, valueAsDate, valueAsNumber, and list IDL attributes, the select() method, the selectionStart, selectionEnd, and selectionDirection, IDL attributes, the setRangeText() and setSelectionRange() methods, the stepUp() and stepDown() methods, and the input and change events apply to an input element dependent on the type attribute? The correct answer is C. Which of those content attributes, if any, can be used to define the width of an input element?

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

kbox, dio ton	File Upload	Submit Button	Image Button	Re Bu Bu
---------------------	----------------	------------------	-----------------	----------------

---

### Content attribut

## **IDL attributes and methods**

Some states

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://>

Each input  
to a number  
to convert a  
stepUp().

**Algorithm to convert a string to object**, and an **algorithm stepDown()** and

Each input  
must be set to  
a non-zero value

the element is created, and  
set to true when the value is

programmatically changed, as described in the definition of the `value` IDL attribute.)

The `value` content attribute gives the default value of the `input` element. When the `value` content attribute is added, set, or removed, if the control's dirty value flag is false, the user agent must set the value of the element to the value of the `value` content attribute, if there is one, or the empty string otherwise, and then run the current value sanitization algorithm, if one is defined.

Each input element has a checkedness, which is exposed by the checked IDL attribute

Each `input` element has a boolean **`dirty checkedness`** flag. When it is true, the element is said to have a **`dirty checkedness`**. The `dirty checkedness` flag must be initially set to false when the element is created, and must be set to true whenever the user interacts with the control in a way that changes the checkedness.

The `checked` content attribute is a boolean attribute that gives the default checkedness of the `input` element. When the `checked` content attribute is added, if the control does not have *dirty checkedness*, the user agent must set the checkedness of the element to true; when the `checked` content attribute is removed, if the control does not have *dirty checkedness*, the user agent must set the checkedness of the element to false.

The reset algorithm for `input` elements is to set the dirty value flag and dirty checkedness flag back to false, set the value of the element to the value of the `value` content attribute, if there is one, or the empty string otherwise, set the checkedness of the element to true if the element has a `checked` content attribute and false if it does not, empty the list of selected files, and then invoke the value sanitization algorithm, if the `type` attribute's current state defines one.

Each `input` element can be *mutable*. Except where otherwise specified, an `input` element is always *mutable*. Similarly, except where otherwise specified, the user agent should not allow the user to modify the element's value or checkedness.

When an `input` element is disabled, it is not *mutable*.

**Note:** The `readonly` attribute can also in some cases (e.g. for the Date state, but not the Checkbox state) stop an input element from being mutable.

The cloning steps for `input` elements must propagate the value, dirty value flag, checkedness, and dirty checkedness flag from the node being cloned to the copy.

When an `input` element is first created, the element's rendering and behavior must be set to the rendering and behavior defined for the `type` attribute's state, and the value sanitization algorithm, if one is defined for the `type` attribute's state, must be invoked.

When an `input` element's `type` attribute changes state, the user agent must run the following steps:

1. If the previous state of the element's `type` attribute put the `value` IDL attribute in the `value` mode, and the element's value is not the empty string, and the new state of the element's `type` attribute puts the `value` IDL attribute in either the `default` mode or the `default/on` mode, then set the element's `value` content attribute to the element's value.
  2. Otherwise, if the previous state of the element's `type` attribute put the `value` IDL attribute in any mode other than the `value` mode, and the new state of the element's `type` attribute puts the `value` IDL attribute in the `value` mode, then set the value of the element to the value of the `value` content attribute, if there is one, or the empty string otherwise, and then set the control's

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `name` attribute represents the element's name. The `dirname` attribute controls how the element's directionality is submitted. The `disabled` attribute is used to make the control non-interactive and to prevent its value from being submitted. The `form` attribute is used to explicitly associate the `input` element with its form owner. The `autofocus` attribute controls focus. The `inputmode` attribute controls the user interface's input modality for the control. The `autocomplete` attribute controls how the user agent provides autofill behavior.

The `indeterminate` IDL attribute must initially be set to false. On getting, it must return the last value it was set to. On setting, it must be set to the new value. It has no effect except for changing the appearance of checkbox controls.

The `accept` IDL attribute reflects the respective `c` attribute. The `type` IDL attribute reflects the `type` attribute. The `maxLength` IDL attribute reflects the `maxLength` attribute. The `value` IDL attribute reflects the `value` attribute. The `checked` IDL attribute reflects the `checked` attribute. The `disabled` IDL attribute reflects the `disabled` attribute. The `form` IDL attribute reflects the `form` attribute. The `list` IDL attribute reflects the `list` attribute. The `multiple` IDL attribute reflects the `multiple` attribute. The `pattern` IDL attribute reflects the `pattern` attribute. The `step` IDL attribute reflects the `step` attribute. The `type` IDL attribute reflects the `type` attribute.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

`value` attributes must reflect the `value` attribute. The `checked` attribute reflects the `checked` content.

`value` values. The `inputMode` attribute reflects the `inputMode` attribute. The `maxLength` attribute reflects the `maxLength` attribute. The `type` attribute reflects the `type` attribute.

The IDL attributes `width` and `height` must return the rendered width and height of the image, in CSS pixels, if an image is being rendered, and is being rendered to a visual medium; or else the intrinsic width and height of the image, in CSS pixels, if an image is *available* but not being rendered to a visual medium; or else 0, if no image is *available*. When the `input` element's `type` attribute is not in the Image Button state, then no image is *available*. [CSS]

On setting, they must act as if they reflected the respective content attributes of the same name.

The `willValidate`, `validity`, and `validationMessage` IDL attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `labels` IDL attribute provides a list of the element's labels. The `select()`, `selectionStart`, `selectionEnd`, `selectionDirection`, `setRangeText()`, and `setSelectionRange()` methods and IDL attributes expose the element's text selection. The `autofocus`, `disabled`, `form`, and `name` IDL attributes are part of the element's forms API.

#### 4.10.7.1 States of the `type` attribute

##### 4.10.7.1.1 Hidden state (`type=hidden`)

When an `input` element's `type` attribute is in the Hidden state, the rules in this section apply.

The `input` element represents a value that is not intended to be examined or manipulated by the user.

**Constraint validation:** If an `input` element's `type` attribute is in the Hidden state, it is barred from constraint validation.

If the `name` attribute is present and has a value that is a case-sensitive match for the string "`_charset_`", then the element's `value` attribute must be omitted.

##### Bookkeeping details

- The `value` IDL attribute applies to this element and is in mode default.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `autocomplete`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `list`, `max`, `maxlength`, `min`, `multiple`, `pattern`, `placeholder`, `readonly`, `required`, `size`, `src`, `step`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `list`, `selectionStart`, `selectionEnd`, `selectionDirection`, `valueAsDate`, and `valueAsNumber` IDL attributes; `select()`, `setRangeText()`, `setSelectionRange()`, `stepDown()`, and `stepUp()` methods.
- The `input` and `change` events do not apply.

##### 4.10.7.1.2 Text (`type=text`) state and Search state (`type=search`)

When an `input` element's `type` attribute is in the Text state or the Search state, the rules in this section apply.

The `input` element represents a one line plain text edit control for the element's value.

**Note:** The difference between the Text state and the Search state is primarily stylistic: on platforms where search

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

U+000D CARRIAGE RETURN (CR) characters into the element's value.

If the element is *mutable*, the user agent should allow the user to change the writing direction of the element, setting it either to a left-to-right writing direction or a right-to-left writing direction. If the user does so, the user agent must then run the following steps:

1. Set the element's `dir` attribute to "`ltr`" if the user selected a left-to-right writing direction, and "`rtl`" if the user selected a right-to-left writing direction.
2. Queue a task to fire a simple event that bubbles named `input` at the `input` element.

The `value` attribute is ~~U+000A LINE FEED (LF)~~ ~~U+000D CARRIAGE RETURN (CR)~~ characters.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

**Bookkeeping**

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `maxlength`, `pattern`, `placeholder`, `readonly`, `required`, and `size` content attributes; `list`, `selectionStart`, `selectionEnd`, `selectionDirection`, and `value` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.
- The `value` IDL attribute is in mode `value`.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `max`, `min`, `multiple`, `src`, `step`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.

dirname, inputmode,  
lrt, selectionEnd,  
i.

### 4.10.7.1.3 Telephone state (`type=tel`)

When an `input` element's `type` attribute is in the Telephone state, the rules in this section apply.

The `input` element represents a control for editing a telephone number given in the element's value.

If the element is *mutable*, its value should be editable by the user. User agents may change the spacing and, with care, the punctuation of values that the user enters. User agents must not allow users to insert U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN (CR) characters into the element's value.

The `value` attribute, if specified, must have a value that contains no U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN (CR) characters.

**The value sanitization algorithm is as follows:** Strip line breaks from the value.

**Note: Unlike the URL and E-mail types, the Telephone type does not enforce a particular syntax. This is intentional; in practice, telephone number fields tend to be free-form fields, because there are a wide variety of valid phone numbers. Systems that need to enforce a particular format are encouraged to use the `pattern` attribute or the `setCustomValidity()` method to hook into the client-side validation mechanism.**

**Bookkeeping details**

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `maxlength`, `pattern`, `placeholder`, `readonly`, `required`, and `size` content attributes; `list`, `selectionStart`, `selectionEnd`, `selectionDirection`, and `value` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.
- The `value` IDL attribute is in mode `value`.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `max`, `min`, `multiple`, `src`, `step`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.

### 4.10.7.1.4 URL state (`type=url`)

When an `input` element's `type` attribute is in the URL state, the rules in this section apply.

The `input` element represents a control for editing a single absolute URL given in the element's value.

If the element is *mutable*, the user agent should allow the user to change the URL represented by its value. User agents may allow the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `value` attribute, if specified and not empty, must have a value that is a valid URL potentially surrounded by spaces that is also an absolute URL.

**The value sanitization algorithm is as follows:** Strip line breaks from the value, then strip leading and trailing whitespace from the value.

**Constraint validation:** While the value of the element is neither the empty string nor a valid absolute URL, the element is suffering from a type mismatch.



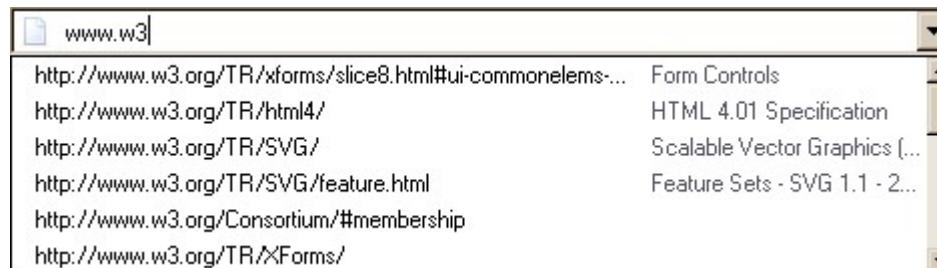
#### Bookkeeping details

- The following attributes are not part of the value sanitization algorithm: `list`, `maxlength`, `multiple`, `size`, `selectionDirection`, `stepDown()`, and `stepUp()`.
- The `value` attribute is part of the value sanitization algorithm.
- The `input` element is part of the value sanitization algorithm.
- The following attributes are part of the value sanitization algorithm: `form`, `formaction`, `formenctype`, `formmethod`, and `formtarget`.
- The following attributes are part of the value sanitization algorithm: `list`, `multiple`, `size`, and `selectionDirection`.
- The `stepDown()` and `stepUp()` methods are part of the value sanitization algorithm.

If a document contained the following markup:

```
<input type="url" name="location" list="urls">
<datalist id="urls">
 <option label="MIME: Format of Internet Message Bodies" value="http://tools.ietf.org/html/rfc2045">
 <option label="HTML 4.01 Specification" value="http://www.w3.org/TR/html4/>
 <option label="Form Controls" value="http://www.w3.org/TR/xforms/slice8.html#ui-common-elements-hint">
 <option label="Scalable Vector Graphics (SVG) 1.1 Specification" value="http://www.w3.org/TR/SVG/>
 <option label="Feature Sets - SVG 1.1 - 20030114" value="http://www.w3.org/TR/SVG/feature.html">
 <option label="The Single UNIX Specification, Version 3" value="http://www.unix-systems.org/version3/">
</datalist>
```

...and the user had typed "www.w3", and the user agent had also found that the user had visited `http://www.w3.org/Consortium/#membership` and `http://www.w3.org/TR/XForms/` in the recent past, then the rendering might look like this:



The first four URLs in this sample consist of the four URLs in the author-specified list that match the text the user has entered, sorted in some UA-defined manner (maybe by how frequently the user refers to those URLs). Note how the UA is using the knowledge that the values are URLs to allow the user to omit the scheme part and perform intelligent matching on the domain name.

The last two URLs (and probably many more, given the scrollbar's indications of more values being available) are the matches from the user agent's session history data. This data is not made available to the page DOM. In this particular case, the UA has no titles to provide for those values.

#### 4.10.7.1.5 E-mail state (`type=email`)

When an `input` element's `type` attribute is in the E-mail state, the rules in this section apply.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the element is *mutable*, the user agent should allow the user to change the e-mail address represented by its value. User agents may allow the user to set the value to a string that is not a valid e-mail address. The user agent should act in a manner consistent with expecting the user to provide a single e-mail address. User agents should allow the user to set the value to the empty string. User agents must not allow users to insert U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN (CR) characters into the value. User agents may transform the value for display and editing; in particular, user agents should convert punycode in the value to IDN in the display and vice versa.

**Constraint validation:** While the user interface is representing input that the user agent cannot convert to punycode, the control is suffering from bad input.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

l address.

eadings and trailing

rithm.

valid e-mail address, the

### When the `multiple` attribute is specified on the element

The element's values are the result of splitting on commas the element's value.

The `input` element represents a control for adding, removing, and editing the e-mail addresses given in the element's values.

If the element is *mutable*, the user agent should allow the user to add, remove, and edit the e-mail addresses represented by its values. User agents may allow the user to set any individual value in the list of values to a string that is not a valid e-mail address, but must not allow users to set any individual value to a string containing U+002C COMMA (,), U+000A LINE FEED (LF), or U+000D CARRIAGE RETURN (CR) characters. User agents should allow the user to remove all the addresses in the element's values. User agents may transform the values for display and editing; in particular, user agents should convert punycode in the value to IDN in the display and vice versa.

**Constraint validation:** While the user interface describes a situation where an individual value contains a U+002C COMMA (,) or is representing input that the user agent cannot convert to punycode, the control is suffering from bad input.

Whenever the user changes the element's values, the user agent must run the following steps:

1. Let *latest values* be a copy of the element's values.
2. Strip leading and trailing whitespace from each value in *latest values*.
3. Let the element's value be the result of concatenating all the values in *latest values*, separating each value from the next by a single U+002C COMMA character (,), maintaining the list's order.

The `value` attribute, if specified, must have a value that is a valid e-mail address list.

### The value sanitization algorithm is as follows:

1. Split on commas the element's value, strip leading and trailing whitespace from each resulting token, if any, and let the element's values be the (possibly empty) resulting list of (possibly empty) tokens, maintaining the original order.
2. Let the element's value be the result of concatenating the element's values, separating each value from the next by a single U+002C COMMA character (,), maintaining the list's order.

When the `multiple` attribute is set, the user agent must run the value sanitization algorithm.

**Constraint validation:** While the value of the element is not a valid e-mail address list, the element is suffering from a type mismatch.

A **valid e-mail address** is a string that matches the ABNF production `1*( atext / "." ) "@" ldh-str *( "." ldh-str )` where `atext` is defined in RFC 5322 section 3.2.3, and `ldh-str` is defined in RFC 1034 section 3.5. [ABNF] [RFC5322] [RFC1034]

**Note:** This requirement is a willful violation of RFC 5322, which defines a syntax for e-mail addresses that is simultaneously too strict (before the "@" character), too vague (after the "@" character), and too lax (allowing comments, whitespace characters, and quoted strings in manners unfamiliar to most users) to be of practical use

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

A **valid e-mail address list** is a set of comma-separated tokens, where each token is itself a valid e-mail address. To obtain the list of tokens from a valid e-mail address list, and implementation must split the string on commas.

#### **Bookkeeping details**

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `maxlength`, `multiple`, `pattern`, `placeholder`, `readonly`, `required`, and `size` content attributes; `list` and `value` IDL attributes.

- The `value` IDL attribute is in mode value.

- The `input` and `change` events apply.

- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `list`, `max`, `min`, `multiple`, `src`, `step`, and `width`.

- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `list`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

`formaction`, `formenctype`,

`i`, `selectionDirection`,

`!()` and `stepUp()` methods.

#### **4.10.7.1.6 P //whatwg.org/html**

When an `input` element's `type` attribute is in the Password state, the rules in this section apply.

The `input` element represents a one line plain text edit control for the element's value. The user agent should obscure the value so that people other than the user cannot see it.

If the element is *mutable*, its value should be editable by the user. User agents must not allow users to insert U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN (CR) characters into the value.

The `value` attribute, if specified, must have a value that contains no U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN (CR) characters.

**The value sanitization algorithm is as follows:** Strip line breaks from the value.

#### **Bookkeeping details**

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `maxlength`, `pattern`, `placeholder`, `readonly`, `required`, and `size` content attributes; `selectionStart`, `selectionEnd`, `selectionDirection`, and `value` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.

- The `value` IDL attribute is in mode value.

- The `input` and `change` events apply.

- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `list`, `max`, `min`, `multiple`, `src`, `step`, and `width`.

- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `list`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.

#### **4.10.7.1.7 Date and Time state (`type=date|datetime`)**

When an `input` element's `type` attribute is in the Date and Time state, the rules in this section apply.

The `input` element represents a control for setting the element's value to a string representing a specific global date and time. User agents may display the date and time in whatever time zone is appropriate for the user.

If the element is *mutable*, the user agent should allow the user to change the global date and time represented by its value, as obtained by parsing a global date and time from it. User agents must not allow the user to set the value to a non-empty string that is not a valid normalized forced-UTC global date and time string, though user agents may allow the user to set and view the time in another time zone and silently translate the time to and from the UTC time zone in the value. If the user agent provides a user interface for selecting a global date and time, then the value must be set to a valid normalized forced-UTC global date and time string representing the user's selection. User agents should allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid normalized forced-UTC global date and time string, the control is suffering from bad input.

**Note:** See the introduction section for a discussion of the difference between the input format and submission format for date, time, and number form controls, and the implementation notes regarding localization of form controls.

The `value` attribute, if specified and not empty, must have a value that is a valid global date and time string.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

value that is a valid global date and time string.

The `step` attribute is expressed in seconds. The step scale factor is 1000 (which converts the seconds to milliseconds, as used in the other algorithms). The default step is 60 seconds.

When the element is suffering from a step mismatch, the user agent may round the element's value to the nearest global date and time for which the element would not suffer from a step mismatch.

**The algorithm to convert a string to a number, given a string `input`, is as follows:** If parsing a global date and time from `input` results in an error, return an error; otherwise, return the number of milliseconds elapsed from midnight UTC on the morning of 1970-01-01/seconds.

**Too slow? Try reading the multipage copy of**

**the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**The algorithm to convert a string to a number, given a string `input`, is as follows:** If parsing a global date and time from `input` results in an error, return an error; otherwise, return the number of milliseconds elapsed from midnight UTC on the morning of 1970-01-01/UTC.

**The algorithm to convert a Date object to a string, given a Date object `input`, is as follows:** Return a valid normalized forced-UTC global date and time string that represents the global date and time that is represented by `input`.

***The Date and Time state (and other date- and time-related states described in subsequent sections) is not intended for the entry of values for which a precise date and time relative to the contemporary calendar cannot be established. For example, it would be inappropriate for the entry of times like "one millisecond after the big bang", "the early part of the Jurassic period", or "a winter around 250 BCE".***

***For the input of dates before the introduction of the Gregorian calendar, authors are encouraged to not use the Date and Time state (and the other date- and time-related states described in subsequent sections), as user agents are not required to support converting dates and times from earlier periods to the Gregorian calendar, and asking users to do so manually puts an undue burden on users. (This is complicated by the manner in which the Gregorian calendar was phased in, which occurred at different times in different countries, ranging from partway through the 16th century all the way to early in the 20th.) Instead, authors are encouraged to provide fine-grained input controls using the select element and input elements with the Number state.***

#### Bookkeeping details

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `max`, `min`, `readonly`, `required`, and `step` content attributes; `list`, `value`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.
- The `value` IDL attribute is in mode `value`.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `maxlength`, `multiple`, `pattern`, `placeholder`, `size`, `src`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `selectionStart`, `selectionEnd`, and `selectionDirection` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.

The following fragment shows part of a calendar application. A user can specify a date and time for a meeting (in his local time zone, probably, though the user agent can allow the user to change that), and since the submitted data includes the time-zone offset, the application can ensure that the meeting is shown at the correct time regardless of the time zones used by all the participants.

```
<fieldset>
 <legend>Add Meeting</legend>
 <p><label>Meeting name: <input type="text" name="meeting.label"></label>
 <p><label>Meeting time: <input type="datetime" name="meeting.start"></label>
</fieldset>
```

Had the application used the `datetime-local` type instead, the calendar application would have also had to explicitly determine which time zone the user intended.

#### 4.10.7.1.8 Date state (`type=date`)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

date from it. User agents must not allow the user to set the value to a non-empty string that is not a valid date string. If the user agent provides a user interface for selecting a date, then the value must be set to a valid date string representing the user's selection. User agents should allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid date string, the control is suffering from bad input.

**Note:** See the introduction section for a discussion of the difference between the input format and submission format for date, time, and number form controls, and the implementation notes regarding location of form controls.

The `value`: Too slow? Try reading the multipage copy of

The `value s` the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

set it to the empty string

The `min` attr valid date st /<https://web.archive.org/web/20130103030411/http://whatwg.org/html>

must have a value that is a

The `step` attr other algorithms). The default step is 1 day.

.milliseconds, as used in the

When the element is suffering from a step mismatch, the user agent may round the element's value to the nearest date for which the element would not suffer from a step mismatch.

**The algorithm to convert a string to a number, given a string `input`, is as follows:** If parsing a date from `input` results in an error, then return an error; otherwise, return the number of milliseconds elapsed from midnight UTC on the morning of 1970-01-01 (the time represented by the value "1970-01-01T00:00:00.0Z") to midnight UTC on the morning of the parsed date, ignoring leap seconds.

**The algorithm to convert a number to a string, given a number `input`, is as follows:** Return a valid date string that represents the date that, in UTC, is current `input` milliseconds after midnight UTC on the morning of 1970-01-01 (the time represented by the value "1970-01-01T00:00:00.0Z").

**The algorithm to convert a string to a Date object, given a string `input`, is as follows:** If parsing a date from `input` results in an error, then return an error; otherwise, return a new `Date` object representing midnight UTC on the morning of the parsed date.

**The algorithm to convert a Date object to a string, given a Date object `input`, is as follows:** Return a valid date string that represents the date current at the time represented by `input` in the UTC time zone.

**Note:** See the note on historical dates in the Date and Time state section.

#### Bookkeeping details

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `max`, `min`, `readonly`, `required`, and `step` content attributes; `list`, `value`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.
- The `value` IDL attribute is in mode value.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `maxlength`, `multiple`, `pattern`, `placeholder`, `size`, `src`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `selectionStart`, `selectionEnd`, and `selectionDirection` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.

#### 4.10.7.1.9 Month state (`type=month`)

When an `input` element's `type` attribute is in the Month state, the rules in this section apply.

The `input` element represents a control for setting the element's value to a string representing a specific month.

If the element is *mutable*, the user agent should allow the user to change the month represented by its value, as obtained by parsing a month from it. User agents must not allow the user to set the value to a non-empty string that is not a valid month string. If the user agent provides a user interface for selecting a month, then the value must be set to a valid month string representing the user's selection. User agents should allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid month string, the control is suffering from bad input.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**The value sanitization algorithm is as follows:** If the value of the element is not a valid month string, then set it to the empty string instead.

The `min` attribute, if specified, must have a value that is a valid month string. The `max` attribute, if specified, must have a value that is a valid month string.

The `step` attribute is expressed in months. The step scale factor is 1 (there is no conversion needed as the algorithms use months). The default step is 1 month.

When the element won't

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

The algorithm then returns

The algorithm then returns the month th

/web/20130103030411/http://whatwg.org/html

The algorithm

error, then return an error; otherwise, return a new `Date` object representing midnight UTC on the morning of the first day of the parsed month.

**The algorithm to convert a `Date` object to a string, given a `Date` object `input`, is as follows:** Return a valid month string that represents the month current at the time represented by `input` in the UTC time zone.

#### Bookkeeping details

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `max`, `min`, `readonly`, `required`, and `step` content attributes; `list`, `value`, `valueAsDate`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.
- The `value` IDL attribute is in mode value.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `maxlength`, `multiple`, `pattern`, `placeholder`, `size`, `src`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `selectionStart`, `selectionEnd`, and `selectionDirection` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.

#### 4.10.7.1.10 Week state (`type=week`)

When an `input` element's `type` attribute is in the Week state, the rules in this section apply.

The `input` element represents a control for setting the element's value to a string representing a specific week.

If the element is *mutable*, the user agent should allow the user to change the week represented by its value, as obtained by parsing a week from it. User agents must not allow the user to set the value to a non-empty string that is not a valid week string. If the user agent provides a user interface for selecting a week, then the value must be set to a valid week string representing the user's selection. User agents should allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid week string, the control is suffering from bad input.

**Note:** See the introduction section for a discussion of the difference between the `input` format and submission format for date, time, and number form controls, and the implementation notes regarding localization of form controls.

The `value` attribute, if specified and not empty, must have a value that is a valid week string.

**The value sanitization algorithm is as follows:** If the value of the element is not a valid week string, then set it to the empty string instead.

The `min` attribute, if specified, must have a value that is a valid week string. The `max` attribute, if specified, must have a value that is a valid week string.

The `step` attribute is expressed in weeks. The step scale factor is 604,800,000 (which converts the weeks to milliseconds, as used in the other algorithms). The default step is 1 week. The default step base is -259,200,000 (the start of week 1970-W01).

When the element is suffering from a step mismatch, the user agent may round the element's value to the nearest week for which the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

ignoring leap seconds.

**The algorithm to convert a number to a string, given a number *input*, is as follows:** Return a valid week string that represents the week that, in UTC, is current *input* milliseconds after midnight UTC on the morning of 1970-01-01 (the time represented by the value "1970-01-01T00:00:00.0Z").

**The algorithm to convert a string to a Date object, given a string *input*, is as follows:** If parsing a week from *input* results in an error, then return an error; otherwise, return a new Date object representing midnight UTC on the morning of the Monday of the parsed week.

**The algorithm to convert a number to a string, given a number *input*, is as follows:** Return a valid week string that represents the week that, in UTC, is current *input* milliseconds after midnight UTC on the morning of 1970-01-01 (the time represented by the value "1970-01-01T00:00:00.0Z").

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

#### Bookkeeping

- The following content attributes must not be specified and do not apply to the element: accept, alt, checked, dirname, formaction, formenctype, formmethod, formnovalidate, formtarget, height, inputmode, maxlength, multiple, pattern, placeholder, size, src, and width.
- The following IDL attributes and methods do not apply to the element: checked, files, selectionStart, selectionEnd, and selectionDirection IDL attributes; select(), setRangeText(), and setSelectionRange() methods.
- The value attribute represents the value of the element. It is a DOMString. It is mutable. It has the following additional methods: list, max, min, readonly, and stepUp() methods.
- The input element represents a control for setting the element's value to a string representing a specific time.

#### 4.10.7.1.11 Time state (*type=time*)

When an input element's type attribute is in the Time state, the rules in this section apply.

The input element represents a control for setting the element's value to a string representing a specific time.

If the element is mutable, the user agent should allow the user to change the time represented by its value, as obtained by parsing a time from it. User agents must not allow the user to set the value to a non-empty string that is not a valid time string. If the user agent provides a user interface for selecting a time, then the value must be set to a valid time string representing the user's selection. User agents should allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid time string, the control is suffering from bad input.

**Note:** See the introduction section for a discussion of the difference between the input format and submission format for date, time, and number form controls, and the implementation notes regarding localization of form controls.

The value attribute, if specified and not empty, must have a value that is a valid time string.

**The value sanitization algorithm is as follows:** If the value of the element is not a valid time string, then set it to the empty string instead.

The min attribute, if specified, must have a value that is a valid time string. The max attribute, if specified, must have a value that is a valid time string.

The step attribute is expressed in seconds. The step scale factor is 1000 (which converts the seconds to milliseconds, as used in the other algorithms). The default step is 60 seconds.

When the element is suffering from a step mismatch, the user agent may round the element's value to the nearest time for which the element would not suffer from a step mismatch.

**The algorithm to convert a string to a number, given a string *input*, is as follows:** If parsing a time from *input* results in an error, then return an error; otherwise, return the number of milliseconds elapsed from midnight to the parsed time on a day with no time changes.

**The algorithm to convert a number to a string, given a number *input*, is as follows:** Return a valid time string that represents the time that is *input* milliseconds after midnight on a day with no time changes.

**The algorithm to convert a string to a Date object, given a string *input*, is as follows:** If parsing a time from *input* results in an error, then return an error; otherwise, return a new Date object representing the parsed time in UTC on 1970-01-01.

**The algorithm to convert a Date object to a string, given a Date object *input*, is as follows:** Return a valid time string that

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- The value IDL attribute is in mode value.
- The input and change events apply.
- The following content attributes must not be specified and do not apply to the element: accept, alt, checked, dirname, formaction, formenctype, formmethod, formnovalidate, formtarget, height, inputmode, maxlength, multiple, pattern, placeholder, size, src, and width.
- The following IDL attributes and methods do not apply to the element: checked, files, selectionStart, selectionEnd, and selectionDirection IDL attributes; select(), setRangeText(), and setSelectionRange() methods.

#### **4.10.7.1.12 Local Date and Time state (`time-datetime-local`)**



When an `input` element has the `type="datetime-local"` attribute, its value is a string representing a local date and time, with no time-zone information.

The `input` element's `value` attribute must be a string representing a local date and time, with no time-zone information.

If the element's `value` attribute is an empty string, the user agent must parse a date and time from the user input, and then set the element's `value` attribute to the resulting local date and time string.

If the element's `value` attribute is not an empty string, the user agent must parse the date and time from the user input, and then set the element's `value` attribute to the resulting local date and time string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid normalized local date and time string, the control is suffering from bad input.

**Note:** See the introduction section for a discussion of the difference between the input format and submission format for date, time, and number form controls, and the implementation notes regarding localization of form controls.

The `value` attribute, if specified and not empty, must have a value that is a valid local date and time string.

**The value sanitization algorithm is as follows:** If the value of the element is a valid local date and time string, then set it to a valid normalized local date and time string representing the same date and time; otherwise, set it to the empty string instead.

The `min` attribute, if specified, must have a value that is a valid local date and time string. The `max` attribute, if specified, must have a value that is a valid local date and time string.

The `step` attribute is expressed in seconds. The step scale factor is 1000 (which converts the seconds to milliseconds, as used in the other algorithms). The default step is 60 seconds.

When the element is suffering from a step mismatch, the user agent may round the element's value to the nearest local date and time for which the element would not suffer from a step mismatch.

**The algorithm to convert a string to a number, given a string `input`, is as follows:** If parsing a date and time from `input` results in an error, then return an error; otherwise, return the number of milliseconds elapsed from midnight on the morning of 1970-01-01 (the time represented by the value "1970-01-01T00:00:00.0") to the parsed local date and time, ignoring leap seconds.

**The algorithm to convert a number to a string, given a number `input`, is as follows:** Return a valid normalized local date and time string that represents the date and time that is `input` milliseconds after midnight on the morning of 1970-01-01 (the time represented by the value "1970-01-01T00:00:00.0").

**Note:** See the note on historical dates in the Date and Time state section.

#### **Bookkeeping details**

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: autocomplete, list, max, min, readonly, required, and step content attributes; list, value, and valueAsNumber IDL attributes; stepDown() and stepUp() methods.
- The value IDL attribute is in mode value.
- The input and change events apply.
- The following content attributes must not be specified and do not apply to the element: accept, alt, checked, dirname, formaction, formenctype, formmethod, formnovalidate, formtarget, height, inputmode, maxlength, multiple, pattern, placeholder, size, src, and width.
- The following IDL attributes and methods do not apply to the element: checked, files, selectionStart, selectionEnd, selectionDirection, and valueAsDate IDL attributes; select(), setRangeText(), and setSelectionRange() methods.

The following example shows part of a flight booking application. The application uses an `input` element with its `type` attribute set to `datetime-local`, and it then interprets the given date and time in the time zone of the selected airport.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<datalist id=airports>
 <option value=ATL label="Atlanta">
 <option value=MEM label="Memphis">
 <option value=LHR label="London Heathrow">
 <option value=LAX label="Los Angeles">
 <option value=FRA label="Frankfurt">
</datalist>
```

If the application instead used the `datetime` type, then the user would have to work out time-zone conversions himself, which

## Too slow? Try reading the multipage copy of

**4.10.7.1.13** the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

When an `input` element is specified, the user agent must set the `value` attribute to the value obtained from applying the rules for parsing floating-point number values to it. User agents must not allow the user to set the value to a non-empty string that is not a valid floating-point number. If the user agent provides a user interface for selecting a number, then the value must be set to the best representation of the number representing the user's selection as a floating-point number. User agents should allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid floating-point number, the control is suffering from bad input.

**Note:** This specification does not define what user interface user agents are to use; user agent vendors are encouraged to consider what would best serve their users' needs. For example, a user agent in Persian or Arabic markets might support Persian and Arabic numeric input (converting it to the format required for submission as described above).

The `value` attribute, if specified and not empty, must have a value that is a valid floating-point number.

**The value sanitization algorithm is as follows:** If the value of the element is not a valid floating-point number, then set it to the empty string instead.

The `min` attribute, if specified, must have a value that is a valid floating-point number. The `max` attribute, if specified, must have a value that is a valid floating-point number.

The step scale factor is 1. The default step is 1 (allowing only integers, unless the `min` attribute has a non-integer value).

When the element is suffering from a step mismatch, the user agent may round the element's value to the nearest number for which the element would not suffer from a step mismatch. If there are two such numbers, user agents are encouraged to pick the one nearest positive infinity.

**The algorithm to convert a string to a number, given a string `input`, is as follows:** If applying the rules for parsing floating-point number values to `input` results in an error, then return an error; otherwise, return the resulting number.

**The algorithm to convert a number to a string, given a number `input`, is as follows:** Return a valid floating-point number that represents `input`.

### Bookkeeping details

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `max`, `min`, `placeholder`, `readonly`, `required`, and `step` content attributes; `list`, `value`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.
- The `value` IDL attribute is in mode value.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `maxlength`, `multiple`, `pattern`, `size`, `src`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `selectionStart`, `selectionEnd`, `selectionDirection`, and `valueAsDate` IDL attributes; `select()`, `setRangeText()`, and `setSelectionRange()` methods.

Here is an example of using a numeric input control:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*Speaking a number. For example, it would be inappropriate for credit card numbers or SS postal codes. A simple way of determining whether to use `type=number` is to consider whether it would make sense for the input control to have a spinbox interface (e.g. with "up" and "down" arrows). Getting a credit card number wrong by 1 in the last digit isn't a minor mistake, it's as wrong as getting every digit incorrect. So it would not make sense for the user to select a credit card number using "up" and "down" buttons. When a spinbox interface is not appropriate, `type=text` is probably the right choice (possibly with a `pattern` attribute).*

#### 4.10.7.1.14 Range state (`type=range`)

When an `input`

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

but with the caveat that the

exact value

*Note: If  
value tu*

re is no way to set the

If the element is suffering from bad input, the user agent must set the value to the empty string, as obtained from applying the rules for parsing floating-point number values to it. User agents must not allow the user to set the value to a string that is not a valid floating-point number. If the user agent provides a user interface for selecting a number, then the value must be set to a best representation of the number representing the user's selection as a floating-point number. User agents must not allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid floating-point number, the control is suffering from bad input.

The `value` attribute, if specified, must have a value that is a valid floating-point number.

**The value sanitization algorithm is as follows:** If the value of the element is not a valid floating-point number, then set it to a valid floating-point number that represents the default value.

The `min` attribute, if specified, must have a value that is a valid floating-point number. The default minimum is 0. The `max` attribute, if specified, must have a value that is a valid floating-point number. The default maximum is 100.

The **default value** is the minimum plus half the difference between the minimum and the maximum, unless the maximum is less than the minimum, in which case the default value is the minimum.

When the element is suffering from an underflow, the user agent must set the element's value to a valid floating-point number that represents the minimum.

When the element is suffering from an overflow, if the maximum is not less than the minimum, the user agent must set the element's value to a valid floating-point number that represents the maximum.

The step scale factor is 1. The default step is 1 (allowing only integers, unless the `min` attribute has a non-integer value).

When the element is suffering from a step mismatch, the user agent must round the element's value to the nearest number for which the element would not suffer from a step mismatch, and which is greater than or equal to the minimum, and, if the maximum is not less than the minimum, which is less than or equal to the maximum. If two numbers match these constraints, then user agents must use the one nearest to positive infinity.

For example, the markup `<input type="range" min=0 max=100 step=20 value=50>` results in a range control whose initial value is 60.

**The algorithm to convert a string to a number, given a string `input`, is as follows:** If applying the rules for parsing floating-point number values to `input` results in an error, then return an error; otherwise, return the resulting number.

**The algorithm to convert a number to a string, given a number `input`, is as follows:** Return a valid floating-point number that represents `input`.

##### Bookkeeping details

- The following common `input` element content attributes, IDL attributes, and methods apply to the element: `autocomplete`, `list`, `max`, `min`, and `step` content attributes; `list`, `value`, and `valueAsNumber` IDL attributes; `stepDown()` and `stepUp()` methods.
- The `value` IDL attribute is in mode value.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`,

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Here is an example of a range control using an autocomplete list with the `list` attribute. This could be useful if there are values along the full range of the control that are especially important, such as preconfigured light levels or typical speed limits in a range control used as a speed control. The following markup fragment:

```
<input type="range" min="-100" max="100" value="0" step="10" name="power" list="powers">
<datalist id="powers">
 <option value="0">
 <option value="-30">
 <option value="30">
```



Too slow? Try reading the multipage copy of  
 ...with the spec instead: [https://web.archive.org  
 /web/20130103030411/http:  
 //whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)



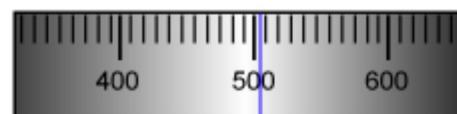
Note how the UA determined the orientation of the control from the ratio of the style-sheet-specified height and width properties. The colors were similarly derived from the style sheet. The tick marks, however, were derived from the markup. In particular, the `step` attribute has not affected the placement of tick marks, the UA deciding to only use the author-specified completion values and then adding longer tick marks at the extremes.

Note also how the invalid value `++50` was completely ignored.

For another example, consider the following markup fragment:

```
<input name=x type=range min=100 max=700 step=9.09090909 value=509.090909>
```

A user agent could display in a variety of ways, for instance:



Or, alternatively, for instance:



The user agent could pick which one to display based on the dimensions given in the style sheet. This would allow it to maintain the same resolution for the tick marks, despite the differences in width.

Finally, here is an example of a range control with two labeled values:

```
<input type="range" name="a" list="a-values">
<datalist id="a-values">
 <option value="10" label="Low">
 <option value="90" label="High">
</datalist>
```

With styles that make the control draw vertically, it might look as follows:

Go

NOV JAN FEB

03

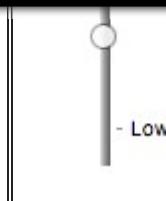
2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020



#### 4.10.7.1.15 Color

When an `input` element's `type` attribute is in the `color` state, the rules for `color` apply.

**Note:** If the element's `value` attribute is specified and not empty, must have a value that is a valid simple color.

`simple-color`

If the element's `value` attribute is specified and not empty, must have a value that is a simple color.

The user agent must convert the value to a lowercase simple color. If the user agent provides a user interface for selecting a color, then the value must be set to the result of using the rules for serializing simple color values to the user's selection. User agents must not allow the user to set the value to the empty string.

**Constraint validation:** While the user interface describes input that the user agent cannot convert to a valid lowercase simple color, the control is suffering from bad input.

The `value` attribute, if specified and not empty, must have a value that is a valid simple color.

**The value sanitization algorithm is as follows:** If the value of the element is a valid simple color, then set it to the value of the element converted to ASCII lowercase; otherwise, set it to the string "#000000".

##### Bookkeeping details

- The following common `input` element content attributes and IDL attributes apply to the element: `autocomplete` and `list` content attributes; `list` and `value` IDL attributes.
- The `value` IDL attribute is in mode `value`.
- The `input` and `change` events apply.
- The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `max`, `maxlength`, `min`, `multiple`, `pattern`, `placeholder`, `readonly`, `required`, `size`, `src`, `step`, and `width`.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `selectionStart`, `selectionEnd`, `selectionDirection`, `valueAsDate`, and `valueAsNumber` IDL attributes; `select()`, `setRangeText()`, `setSelectionRange()`, `stepDown()`, and `stepUp()` methods.

#### 4.10.7.1.16 Checkbox state (`type=checkbox`)

When an `input` element's `type` attribute is in the `checkbox` state, the rules in this section apply.

The `input` element represents a two-state control that represents the element's checkedness state. If the element's checkedness state is true, the control represents a positive selection, and if it is false, a negative selection. If the element's `indeterminate` IDL attribute is set to true, then the control's selection should be obscured as if the control was in a third, indeterminate, state.

**Note:** The control is never a true tri-state control, even if the element's `indeterminate` IDL attribute is set to true.

The `indeterminate` IDL attribute only gives the appearance of a third state.

If the element is `mutable`, then: The pre-click activation steps consist of setting the element's checkedness to its opposite value (i.e. true if it is false, false if it is true), and of setting the element's `indeterminate` IDL attribute to false. The canceled activation steps consist of setting the checkedness and the element's `indeterminate` IDL attribute back to the values they had before the pre-click activation steps were run. The activation behavior is to fire a simple event that bubbles named `change` at the element.

If the element is not `mutable`, it has no activation behavior.

**Constraint validation:** If the element is `required` and its checkedness is false, then the element is suffering from being missing.

This box is non-normative. Implementation requirements are given below this box.

Go

NOV JAN FEB



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

### **Bookkeeping details**

- The following common input element content attributes and IDL attributes apply to the element: checked, and required content attributes; checked and value IDL attributes.
  - The value IDL attribute is in mode default/on.
  - The change event applies.
  - The following content attributes must not be specified and do not apply to the element: accept, alt, autocomplete, dirname, formaction, formenctype, formmethod, formnovalidate, formtarget, height, inputmode, list, max, maxlength, min, multiple, pattern, placeholder, readonly, size, src, step, and width.
  - The following IDL attributes and methods do not apply to the element: files, list, selectionStart, selectionEnd, selectionDirection, valueAsDate, valueAsNumber, and stepUp() methods.

- The input Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

#### **4.10.7.1.17**

When an [in](#) /web/20130103030411/http://

The input: //whatwg.org/html

selected control in the group, and if it is false, it indicates a control in the group that is not selected.

The **radio button group** that contains an `input` element *a* also contains all the other `input` elements *b* that fulfill all of the following conditions:

- The `input` element *b*'s `type` attribute is in the Radio Button state.
  - Either *a* and *b* have the same form owner, or they both have no form owner.
  - Both *a* and *b* are in the same home subtree.
  - They both have a `name` attribute, their `name` attributes are not empty, and the value of *a*'s `name` attribute is a compatibility caseless match for the value of *b*'s `name` attribute.

A document must not contain an `input` element whose *radio button group* contains only that element.

When any of the following phenomena occur, if the element's checkedness state is true after the occurrence, the checkedness state of all the other elements in the same *radio button group* must be set to false:

- The element's checkedness state is set to true (for whatever reason)
  - The element's `name` attribute is set, changed, or removed.
  - The element's form owner changes.

If the element is *mutable*, then: The pre-click activation steps consist of setting the element's checkedness to true. The canceled activation steps consist of setting the element's checkedness to false. The activation behavior is to fire a simple event that bubbles named `change` at the element. .

If the element is not *mutable*, it has no activation behavior.

**Constraint validation:** If an element in the *radio button group* is *required*, and all of the `input` elements in the *radio button group* have a checkedness that is false, then the element is suffering from being missing.

**Note:** If none of the radio buttons in a radio button group are checked when they are inserted into the document, then they will all be initially unchecked in the interface, until such time as one of them is checked (either by the user or by script).

#### ***Bookkeeping details***

- The following common `input` element content attributes and IDL attributes apply to the element: `checked` and `required` content attributes; `checked` and `value` IDL attributes.
  - The `value` IDL attribute is in mode `default/on`.
  - The `change` event applies.
  - The following content attributes must not be specified and do not apply to the element: `accept`, `alt`, `autocomplete`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `list`, `max`, `maxlength`, `min`, `multiple`, `pattern`, `placeholder`, `readonly`, `size`, `src`, `step`, and `width`.
  - The following IDL attributes and methods do not apply to the element: `files`, `list`, `selectionStart`, `selectionEnd`, `selectionDirection`,

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When an `input` element's `type` attribute is in the File Upload state, the rules in this section apply.

The `input` element represents a list of **selected files**, each file consisting of a file name, a file type, and a file body (the contents of the file).

File names must not contain path components, even in the case that a user has selected an entire directory hierarchy or multiple files with the same name from different directories. Path components are those separated by U+005C REVERSE SOLIDUS character (\) characters.

If the element is from the file: **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

**Constraint:** Unless the attribute `accept` is present, the element is suffering from being missing.

Unless the attribute `accept` is present, the element is suffering from being missing.

The `accept` attribute specifies a set of comma-separated tokens, each of which must be an ASCII case-insensitive match for one of the following:

#### The string `audio/*`

Indicates that sound files are accepted.

#### The string `video/*`

Indicates that video files are accepted.

#### The string `image/*`

Indicates that image files are accepted.

#### A valid MIME type with no parameters

Indicates that files of the specified type are accepted.

#### A string whose first character is a U+002E FULL STOP character (.)

Indicates that files with the specified file extension are accepted.

The tokens must not be ASCII case-insensitive matches for any of the other tokens (i.e. duplicates are not allowed). To obtain the list of tokens from the attribute, the user agent must split the attribute value on commas.

User agents may use the value of this attribute to display a more appropriate user interface than a generic file picker. For instance, given the value `image/*`, a user agent could offer the user the option of using a local camera or selecting a photograph from their photo collection; given the value `audio/*`, a user agent could offer the user the option of recording a clip using a headset microphone.

User agents should prevent the user from selecting files that are not accepted by one (or more) of these tokens.

**Note:** Authors are encouraged to specify both any MIME types and any corresponding extensions when looking for data in a specific format.

For example, consider an application that converts Microsoft Word documents to Open Document Format files. Since Microsoft Word documents are described with a wide variety of MIME types and extensions, the site can list several, as follows:

```
<input type="file" accept=".doc,.docx,.xml,application/msword,application/vnd.openxmlformats-officedocument.wordprocessingml.document">
```

On platforms that only use file extensions to describe file types, the extensions listed here can be used to filter the allowed documents, while the MIME types can be used with the system's type registration table (mapping MIME types to extensions used by the system), if any, to determine any other extensions to allow. Similarly, on a system that does not have file names or extensions but labels documents with MIME types internally, the MIME types can be used to pick the allowed files, while the extensions can be used if the system has an extension registration table that maps known extensions to MIME types used by the system.

**⚠Warning!** Extensions tend to be ambiguous (e.g. there are an untold number of formats that use the ".dat" extension, and users can typically quite easily rename their files to have a ".doc" extension even if they are not Microsoft Word documents), and MIME types tend to be unreliable (e.g. many formats have no formally registered types, and many formats are in practice labeled using a number of different MIME types). Authors are reminded that, as usual, data received from a client should be

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

attribute in a backwards-compatible way is non-trivial. The following function extracts the filename in a suitably compatible manner:

```
function extractFilename(path) {
 if (path.substr(0, 12) == "C:\\fakepath\\")
 return path.substr(12); // modern browser
 var x;
 x = path.lastIndexOf('/');
 if (x >= 0) // Unix-based path
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

This [//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
<_> </p>
<p>The name of the file you picked is: (none)</p>
<script>
 function updateFilename(path) {
 var name = extractFilename(path);
 document.getElementById('filename').textContent = name;
 }
</script>
```

#### **Bookkeeping details**

- The following common `input` element content attributes and IDL attributes apply to the element: `accept`, `multiple`, and `required` content attributes; `files` and `value` IDL attributes.
- The `value` IDL attribute is in mode `filename`.
- The `change` event applies.
- The following content attributes must not be specified and do not apply to the element: `alt`, `autocomplete`, `checked`, `dirname`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `inputmode`, `list`, `max`, `maxlength`, `min`, `pattern`, `placeholder`, `readonly`, `size`, `src`, `step`, and `width`.
- The element's `value` attribute must be omitted.
- The following IDL attributes and methods do not apply to the element: `checked`, `list`, `selectionStart`, `selectionEnd`, `selectionDirection`, `valueAsDate`, and `valueAsNumber` IDL attributes; `select()`, `setRangeText()`, `setSelectionRange()`, `stepDown()`, and `stepUp()` methods.
- The `input` event does not apply.

#### **4.10.7.1.19 Submit Button state (`type=submit`)**

When an `input` element's `type` attribute is in the Submit Button state, the rules in this section apply.

The `input` element represents a button that, when activated, submits the form. If the element has a `value` attribute, the button's label must be the value of that attribute; otherwise, it must be an implementation-defined string that means "Submit" or some such. The element is a button, specifically a submit button.



If the element is `mutable`, then the element's activation behavior is as follows: if the element has a form owner, submit the form owner from the `input` element; otherwise, do nothing.

If the element is not `mutable`, it has no activation behavior.

The `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget` attributes are attributes for form submission.

**Note:** The `formnovalidate` attribute can be used to make submit buttons that do not trigger the constraint validation.

#### **Bookkeeping details**

- The following common `input` element content attributes and IDL attributes apply to the element: `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget` content attributes; `value` IDL attribute.
- The `value` IDL attribute is in mode `default`.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- The input and change events do not apply.

#### 4.10.7.1.20 Image Button state (`type=image`)

When an `input` element's `type` attribute is in the Image Button state, the rules in this section apply.

The `input` element represents either an image from which a user can select a coordinate and submit the form, or alternatively a button from which the user can submit the form. The element is a button, specifically a submit button.



*Note: T  
from th  
coordi*

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

*he element, derived  
ponents of the*

The image is surrounded by `/web/20130103030411/http://whatwg.org/html`

empty URL potentially  
ed nor scripted.

When any of the user agent only fetches elements on demand, or the `src` attribute's value is the empty string, the user agent must resolve the value of the `src` attribute, relative to the element, and if that is successful, must fetch the resulting absolute URL:

- The `input` element's `type` attribute is first set to the Image Button state (possibly when the element is first created), and the `src` attribute is present.
- The `input` element's `type` attribute is changed back to the Image Button state, and the `src` attribute is present, and its value has changed since the last time the `type` attribute was in the Image Button state.
- The `input` element's `type` attribute is in the Image Button state, and the `src` attribute is set or changed.

Fetching the image must delay the load event of the element's document until the task that is queued by the networking task source once the resource has been fetched (defined below) has been run.

If the image was successfully obtained, with no network errors, and the image's type is a supported image type, and the image is a valid image of that type, then the image is said to be **available**. If this is true before the image is completely downloaded, each task that is queued by the networking task source while the image is being fetched must update the presentation of the image appropriately.

The user agents should apply the image sniffing rules to determine the type of the image, with the image's associated Content-Type headers giving the **official type**. If these rules are not applied, then the type of the image must be the type given by the image's associated Content-Type headers.

User agents must not support non-image resources with the `input` element. User agents must not run executable code embedded in the image resource. User agents must only display the first page of a multipage resource. User agents must not allow the resource to act in an interactive fashion, but should honor any animation in the resource.

The task that is queued by the networking task source once the resource has been fetched, must, if the download was successful and the image is **available**, queue a task to fire a simple event named `load` at the `input` element; and otherwise, if the fetching process fails without a response from the remote server, or completes but the image is not a valid or supported image, queue a task to fire a simple event named `error` on the `input` element.

The `alt` attribute provides the textual label for the button for users and user agents who cannot use the image. The `alt` attribute must be present, and must contain a non-empty string.

The `input` element supports dimension attributes.

If the `src` attribute is set, and the image is **available** and the user agent is configured to display that image, then: The element represents a control for selecting a coordinate from the image specified by the `src` attribute; if the element is **mutable**, the user agent should allow the user to select this coordinate, and the element's activation behavior is as follows: if the element has a form owner, take the user's selected coordinate, and submit the `input` element's form owner from the `input` element. If the user activates the control without explicitly selecting a coordinate, then the coordinate (0,0) must be assumed.

Otherwise, the element represents a submit button whose label is given by the value of the `alt` attribute; if the element is **mutable**, then the element's activation behavior is as follows: if the element has a form owner, set the selected coordinate to (0,0), and submit the `input` element's form owner from the `input` element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `x`-component must be a valid integer representing a number  $x$  in the range  $-(\text{borderleft}+\text{paddingleft}) \leq x \leq \text{width}+\text{borderright}+\text{paddingright}$ , where `width` is the rendered width of the image, `borderleft` is the width of the border on the left of the image, `paddingleft` is the width of the padding on the left of the image, `borderright` is the width of the border on the right of the image, and `paddingright` is the width of the padding on the right of the image, with all dimensions given in CSS pixels.

The `y`-component must be a valid integer representing a number  $y$  in the range  $-(\text{bordertop}+\text{paddingtop}) \leq y \leq \text{height}+\text{borderbottom}+\text{paddingbottom}$ , where `height` is the rendered height of the image, `bordertop` is the width of the border above the image, `paddingtop` is the width of the padding above the image, `borderbottom` is the width of the border below the image, and `paddingbottom` is the width of the padding below the image, with all dimensions given in CSS pixels.

Where a `border` is present, the rendered width of the `img` element is the sum of the `borderwidth`, the `paddingwidth`, and the `width` of the image. The `formaction` attribute specifies the URL to which the form is submitted. The `formmethod` attribute specifies the HTTP method used to submit the form. The `formenctype` attribute specifies the encoding type used to encode the form data. The `formnovalidate` attribute indicates whether the form should be validated before submission. The `formtarget` attribute specifies the target frame or window where the form should be submitted. The `height` attribute specifies the height of the image. The `src` attribute specifies the URL of the image. The `width` attribute specifies the width of the image.

ibutes for form submission.

ts are given below this box.

These attributes return the actual rendered dimensions of the image, or zero if the dimensions are not known.

They can be set, to change the corresponding content attributes.

#### **Bookkeeping details**

- The following common `input` element content attributes and IDL attributes apply to the element: `alt`, `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `height`, `src`, and `width` content attributes; `value` IDL attribute.
- The `value` IDL attribute is in mode default.
- The following content attributes must not be specified and do not apply to the element: `accept`, `autocomplete`, `checked`, `dirname`, `inputmode`, `list`, `max`, `maxlength`, `min`, `multiple`, `pattern`, `placeholder`, `readonly`, `required`, `size`, and `step`.
- The element's `value` attribute must be omitted.
- The following IDL attributes and methods do not apply to the element: `checked`, `files`, `list`, `selectionStart`, `selectionEnd`, `selectionDirection`, `valueAsDate`, and `valueAsNumber` IDL attributes; `select()`, `setRangeText()`, `setSelectionRange()`, `stepDown()`, and `stepUp()` methods.
- The `input` and `change` events do not apply.

**Note:** Many aspects of this state's behavior are similar to the behavior of the `img` element. Readers are encouraged to read that section, where many of the same requirements are described in more detail.

Take the following form:

```
<form action="process.cgi">
 <input type=image src=map.png name=where>
</form>
```

If the user clicked on the image at coordinate (127,40) then the URL used to submit the form would be "process.cgi?where.x=127&where.y=40".

#### **4.10.7.1.21 Reset Button state (`type=reset`)**

When an `input` element's `type` attribute is in the Reset Button state, the rules in this section apply.

The `input` element represents a button that, when activated, resets the form. If the element has a `value` attribute, the button's label must be the value of that attribute; otherwise, it must be an implementation-defined string that means "Reset" or some such. The element is a button.

If the element is `mutable`, then the element's activation behavior, if the element has a form owner, is to reset the form owner; otherwise, it is to do nothing.



If the element is not `mutable`, it has no activation behavior.

**Constraint validation:** The element is barred from constraint validation.

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

- The following IDL attributes and methods do not apply to the element: checked, files, list, selectionStart, selectionEnd, selectionDirection, valueAsDate, and valueAsNumber **IDL attributes**; select(), setRangeText(), setSelectionRange(), stepDown(), and stepUp() **methods**.
- The input and change events do not apply.

#### **4.10.7.1.22 Button state (*type=button*)**

When an `input` element's `type` attribute is in the Button state, the rules in this section apply.

#### *Bookkeeping details*

- The value IDL attribute applies to this element and is in mode default.
  - The following content attributes must not be specified and do not apply to the element: accept, alt, autocomplete, checked, dirname, formaction, formenctype, formmethod, formnovalidate, formtarget, height, inputmode, list, max, maxlength, min, multiple, pattern, placeholder, readonly, required, size, src, step, and width.
  - The following IDL attributes and methods do not apply to the element: checked, files, list, selectionStart, selectionEnd, selectionDirection, valueAsDate, and valueAsNumber IDL attributes; select(), setRangeText(), setSelectionRange(), stepDown(), and stepUp() methods.
  - The input and change events do not apply.

#### **4.10.7.2 Implementation notes regarding localization of form controls**

*This section is non-normative.*

The formats shown to the user in date, time, and number controls is independent of the format used for form submission.

Browsers are encouraged to use user interfaces that present dates, times, and numbers according to the conventions of either the locale implied by the `input` element's language or the user's preferred locale. Using the page's locale will ensure consistency with page-provided data.

For example, it would be confusing to users if an American English page claimed that a Cirque De Soleil show was going to be showing on 02/03, but their browser, configured to use the British English locale, only showed the date 03/02 in the ticket purchase date picker. Using the page's locale would at least ensure that the date was presented in the same format everywhere. (There's still a risk that the user would end up arriving a month late, of course, but there's only so much that can be done about such cultural differences...)

#### 4.10.7.3 Common input element attributes

These attributes only apply to an `input` element if its `type` attribute is in a state whose definition declares that the attribute applies. When an attribute **doesn't apply** to an `input` element, user agents must ignore the attribute, regardless of the requirements and definitions below.

#### 4.10.7.3.1 The `maxLength` attribute

The `maxlength` attribute, when it applies, is a form control `maxlength` attribute controlled by the `input` element's dirty value flag.

If the `input` element has a maximum allowed value length, then the code-unit length of the value of the element's `value` attribute must be equal to or less than the element's maximum allowed value length.

The following extract shows how a messaging client's text entry could be arbitrarily restricted to a fixed number of characters, thus forcing any conversation through this medium to be terse and discouraging intelligent discourse.

<label>What are you doing? <input name=status maxlength=140></label>

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

### 3,445 captures

5 Jun 2004 - 22 Aug 2020

The `size` attribute, if specified, must have a value that is a valid non-negative integer greater than zero.

If the attribute is present, then its value must be parsed using the rules for parsing non-negative integers, and if the result is a number greater than zero, then the user agent should ensure that at least that many characters are visible.

The `size` IDL attribute is limited to only non-negative numbers greater than zero and has a default value of 20.

#### 4.10.7.3.3 The `readonly` attribute

The `readonly`

element is n

**Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org**

*Note: This text contains some obfuscation.*

`/web/20130103030411/http://whatwg.org/html`

rol. When specified, the

from constraint validation.

able, so the user can  
For this reason, only  
tances.)

In the following example, the existing product identifiers cannot be modified, but they are still displayed as part of the form, for consistency with the row representing a new product (where the identifier is not yet filled in).

```
<form action="products.cgi" method="post" enctype="multipart/form-data">
 <table>
 <tr> <th> Product ID <th> Product name <th> Price <th> Action
 <tr>
 <td> <input readonly="readonly" name="1.pid" value="H412">
 <td> <input required="required" name="1.pname" value="Floor lamp Ulke">
 <td> $<input required="required" type="number" min="0" step="0.01" name="1.pprice" value="49.99">
 <td> <button formnovalidate="formnovalidate" name="action" value="delete:1">Delete</button>
 <tr>
 <td> <input readonly="readonly" name="2.pid" value="FG28">
 <td> <input required="required" name="2.pname" value="Table lamp Ulke">
 <td> $<input required="required" type="number" min="0" step="0.01" name="2.pprice" value="24.99">
 <td> <button formnovalidate="formnovalidate" name="action" value="delete:2">Delete</button>
 <tr>
 <td> <input required="required" name="3.pid" value="" pattern="[A-Z0-9]+>
 <td> <input required="required" name="3.pname" value="">
 <td> $<input required="required" type="number" min="0" step="0.01" name="3.pprice" value="">
 <td> <button formnovalidate="formnovalidate" name="action" value="delete:3">Delete</button>
 </table>
 <p> <button formnovalidate="formnovalidate" name="action" value="add">Add</button> </p>
 <p> <button name="action" value="update">Save</button> </p>
 </form>
```

#### 4.10.7.3.4 The `required` attribute

The `required` attribute is a boolean attribute. When specified, the element is **required**.

**Constraint validation:** If the element is *required*, and its value IDL attribute applies and is in the mode value, and the element is *mutable*, and the element's value is the empty string, then the element is suffering from being missing.

The following form has two required fields, one for an e-mail address and one for a password. It also has a third field that is only considered valid if the user types the same password in the password field and this third field.

```
<h1>Create new account</h1>
<form action="/newaccount" method=post
 oninput="up2.setCustomValidity(up2.value != up.value ? 'Passwords do not match.' :
```

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB  
 ◀ 03 ▶  
 2012 2013 2014

About this capture

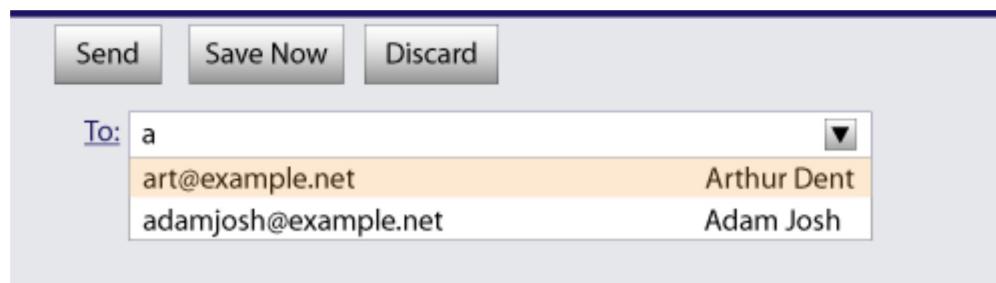
```
<label for="password1">Password:</label>
<input id="password1" type=password required name=up>
<p>
 <label for="password2">Confirm password:</label>
 <input id="password2" type=password name=up2>
<p>
 <input type=submit value="Create account">
</form>
```

#### 4.10.7.3.5 T Too slow? Try reading the multipage copy of

The `multiple` attribute on the `input` element allows for more than one value.

The `multiple` attribute on the `input` element allows for more than one value.  
 If the user agent suggests two e-mail addresses to the user, it should suggest both of them.

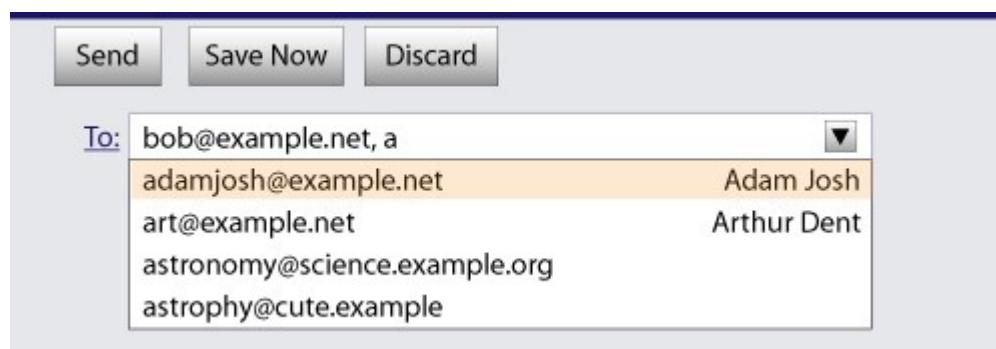
If the user agent suggests both "Arthur Dent" (with address "art@example.net") and "Adam Josh" (with address "adamjosh@example.net"), then, after the user has typed "a", the user agent might suggest these two e-mail addresses to the user.



The page could also link in the user's contacts database from the site:

```
<label>Cc: <input type=email multiple name=cc list=contacts></label>
...
<datalist id="contacts">
 <option value="hedral@damowmow.com">
 <option value="pillar@example.com">
 <option value="astrophysics@cute.example">
 <option value="astronomy@science.example.org">
</datalist>
```

Suppose the user had entered "bob@example.net" into this text field, and then started typing a second e-mail address starting with "a". The user agent might show both the two friends mentioned earlier, as well as the "astrophysics" and "astronomy" values given in the `datalist` element.



The following extract shows how an e-mail client's "Attachments" field could accept multiple files for upload.

```
<label>Attachments: <input type=file multiple name=att></label>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

If specified, the attribute's value must match the JavaScript *Pattern* production. [ECMA262]

If an `input` element has a `pattern` attribute specified, and the attribute's value, when compiled as a JavaScript regular expression with the `global`, `ignoreCase`, and `multiline` flags *disabled* (see ECMA262 Edition 5, sections 15.10.7.2 through 15.10.7.4), compiles successfully, then the resulting regular expression is the element's **compiled pattern regular expression**. If the element has no such attribute, or if the value doesn't compile successfully, then the element has no compiled pattern regular expression. [ECMA262]

**Constraint validation:** If the element's value is not the empty string, and either the element's `multiple` attribute is not specified or it does not apply to the `input` element given its `type` attribute's current state, and the element has a compiled pattern regular expression but the value does not match the entire string, then the element is suffering from a pattern mismatch.

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

**Note:** This implies that the regular expression language used for this attribute is the same as that used in JavaScript, except that the `pattern` attribute is matched against the entire value, not just any subset (somewhat as if it implied a `^?: at the start of the pattern and a ) $ at the end`).

When an `input` element has a `pattern` attribute specified, authors should include a `title` attribute to give a description of the pattern. User agents may use the contents of this attribute, if it is present, when informing the user that the pattern is not matched, or at any other suitable time, such as in a tooltip or read out by assistive technology when the control gains focus.

For example, the following snippet:

```
<label> Part number:
<input pattern="[0-9][A-Z]{3}" name="part"
 title="A part number is a digit followed by three uppercase letters." />
</label>
```

...could cause the UA to display an alert such as:

A part number is a digit followed by three uppercase letters.  
You cannot submit this form when the field is incorrect.

When a control has a `pattern` attribute, the `title` attribute, if used, must describe the pattern. Additional information could also be included, so long as it assists the user in filling in the control. Otherwise, assistive technology would be impaired.

For instance, if the `title` attribute contained the caption of the control, assistive technology could end up saying something like The text you have entered does not match the required pattern. Birthday, which is not useful.

UAs may still show the `title` in non-error situations (for example, as a tooltip when hovering over the control), so authors should be careful not to word `titles` as if an error has necessarily occurred.

### 4.10.7.3.7 The `min` and `max` attributes

The `min` and `max` attributes indicate the allowed range of values for the element.

Their syntax is defined by the section that defines the `type` attribute's current state.

If the element has a `min` attribute, and the result of applying the algorithm to convert a string to a number to the value of the `min` attribute is a number, then that number is the element's **minimum**; otherwise, if the `type` attribute's current state defines a **default minimum**, then that is the minimum; otherwise, the element has no minimum.

**Constraint validation:** When the element has a minimum, and the result of applying the algorithm to convert a string to a number to the string given by the element's value is a number, and the number obtained from that algorithm is less than the minimum, the element is suffering from an underflow.

The `min` attribute also defines the step base.

If the element has a `max` attribute, and the result of applying the algorithm to convert a string to a number to the value of the `max`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

element is suffering from an overflow.

The `max` attribute's value (the maximum) must not be less than the `min` attribute's value (its minimum).

**Note:** If an element has a maximum that is less than its minimum, then so long as the element has a value, it will either be suffering from an underflow or suffering from an overflow.

An element has **range limitations** if it has a defined minimum or a defined maximum.



The `fr`'  
**Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**  
**4.10.7.3.8 T**

The `step` attribute indicates the granularity that is expected (and required) of the value, by limiting the allowed values. The section that defines the `type` attribute's current state also defines the **default step**, the **step scale factor**, and in some cases the **default step base**, which are used in processing the attribute as described below.

The `step` attribute, if specified, must either have a value that is a valid floating-point number that parses to a number that is greater than zero, or must have a value that is an ASCII case-insensitive match for the string "any".

The attribute provides the **allowed value step** for the element, as follows:

1. If the attribute is absent, then the allowed value step is the default step multiplied by the step scale factor.
2. Otherwise, if the attribute's value is an ASCII case-insensitive match for the string "any", then there is no allowed value step.
3. Otherwise, if the rules for parsing floating-point number values, when they are applied to the attribute's value, return an error, zero, or a number less than zero, then the allowed value step is the default step multiplied by the step scale factor.
4. Otherwise, the allowed value step is the number returned by the rules for parsing floating-point number values when they are applied to the attribute's value, multiplied by the step scale factor.

The **step base** is the value returned by the following algorithm:

1. If the element has a `min` content attribute, and the result of applying the algorithm to convert a string to a number to the value of the `min` content attribute is not an error, then return that result and abort these steps.
2. If the element has a `value` content attribute, and the result of applying the algorithm to convert a string to a number to the value of the `value` content attribute is not an error, then return that result and abort these steps.
3. If a default step base is defined for this element given its `type` attribute's state, then return it and abort these steps.
4. Return zero.

**Constraint validation:** When the element has an allowed value step, and the result of applying the algorithm to convert a string to a number to the string given by the element's value is a number, and that number subtracted from the step base is not an integral multiple of the allowed value step, the element is suffering from a step mismatch.

The following range control only accepts values in the range 0..1, and allows 256 steps in that range:

```
<input name=opacity type=range min=0 max=1 step=0.00392156863>
```

The following control allows any time in the day to be selected, with any accuracy (e.g. thousandth-of-a-second accuracy or more):

```
<input name=favtime type=time step=any>
```

Normally, time controls are limited to an accuracy of one minute.

#### 4.10.7.3.9 The `list` attribute

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

attribute, if that element is a `datalist` element. If there is no `list` attribute, or if there is no element with that ID, or if the first element with that ID is not a `datalist` element, then there is no suggestions source element.

If there is a suggestions source element, then, when the user agent is allowing the user to edit the `input` element's value, the user agent should offer the suggestions represented by the suggestions source element to the user in a manner suitable for the type of control used. The user agent may use the suggestion's label to identify the suggestion if appropriate.

How user selections of suggestions are handled depends on whether the element is a control accepting a single value only, or whether it accepts multiple values:

↳ If the element is a control accepting a single value only:  
**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

not apply  
suggestion's value, as if the

element's values, whose values to have the value value, or edited an existing

entity to be that value. Which behavior is to be applied depends on the user interface in a user-agent-defined manner.

If the `list` attribute does not apply, there is no suggestions source element.

This URL field offers some suggestions.

```
<label>Homepage: <input name=hp type=url list=hpuurls></label>
<datalist id=hpuurls>
 <option value="http://www.google.com/" label="Google">
 <option value="http://www.reddit.com/" label="Reddit">
</datalist>
```

Other URLs from the user's history might show also; this is up to the user agent.

This example demonstrates how to design a form that uses the autocomplete list feature while still degrading usefully in legacy user agents.

If the autocomplete list is merely an aid, and is not important to the content, then simply using a `datalist` element with children `option` elements is enough. To prevent the values from being rendered in legacy user agents, they need to be placed inside the `value` attribute instead of inline.

```
<p>
 <label>
 Enter a breed:
 <input type="text" name="breed" list="breeds">
 <datalist id="breeds">
 <option value="Abyssinian">
 <option value="Alpaca">
 <!-- ... -->
 </datalist>
</label>
</p>
```

However, if the values need to be shown in legacy UAs, then fallback content can be placed inside the `datalist` element, as follows:

```
<p>
 <label>
 Enter a breed:
 <input type="text" name="breed" list="breeds">
 </label>
 <datalist id="breeds">
 <label>
 or select one from the list:
 <select name="breed">
 <option value=""> (none selected)
 </label>
 </datalist>
</p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
</datalist>
</p>
```

The fallback content will only be shown in UAs that don't support `datalist`. The options, on the other hand, will be detected by all UAs, even though they are not children of the `datalist` element.

Note that if an `option` element used in a `datalist` is selected, it will be selected by default by legacy UAs (because it affects the `select`), but it will not have any effect on the `input` element in UAs that support `datalist`.



#### 4.10.7.3.10 Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

The `placeholder` has no value that contains  
The `placeholder` attribute is n

data entry when the control specified, must have a value

visory text, the `title`

**Note:** These mechanisms are very similar but subtly different: the hint given by the control's `label` is shown at all times; the short hint given in the `placeholder` attribute is shown before the user enters a value; and the hint in the `title` attribute is shown when the user requests further help.

User agents should present this hint to the user, after having stripped line breaks from it, when the element's value is the empty string or the control is not focused (or both), e.g. by displaying it inside a blank unfocused control and hiding it otherwise.

Here is an example of a mail configuration user interface that uses the `placeholder` attribute:

```
<fieldset>
 <legend>Mail Account</legend>
 <p><label>Name: <input type="text" name="fullname" placeholder="John Ratzenberger">
 </label></p>
 <p><label>Address: <input type="email" name="address" placeholder="john@example.net">
 </label></p>
 <p><label>Password: <input type="password" name="password"></label></p>
 <p><label>Description: <input type="text" name="desc" placeholder="My Email Account">
 </label></p>
</fieldset>
```

In situations where the control's content has one directionality but the placeholder needs to have a different directionality, Unicode's bidirectional-algorithm formatting characters can be used in the attribute value:

```
<input name=t1 type=tel placeholder="‫ رقم الهاتف 1">
<input name=t2 type=tel placeholder="‫ رقم الهاتف 2">
```

For slightly more clarity, here's the same example using numeric character references instead of inline Arabic:

```
<input name=t1 type=tel
placeholder="‫رقم الهاتف 1‫
<input name=t2 type=tel
placeholder="‫رقم الهاتف 2‫"
```

#### 4.10.7.4 Common `input` element APIs

This box is non-normative. Implementation requirements are given below this box.

`input.value [= value]`

Returns the current value of the form control.

Can be set, to change the value.

Throws an `InvalidStateError` exception if it is set to any value other than the empty string when the control is a file upload control.

`input.checked [= value]`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Returns a `FileList` object listing the selected files of the form control.Returns `null` if the control isn't a file control.**`input.valueAsDate [= value]`**Returns a `Date` object representing the form control's value, if applicable; otherwise, returns `null`.

Can be set, to change the value.

Throws an `InvalidStateError` exception if the control isn't date- or time-based.**`input.v`****Re: Too slow? Try reading the multipage copy of****the spec instead: https://web.archive.org****Thru https://web.archive.org/web/20130103030411/http://whatwg.org/html****`input.v`****`input.v`**Changes the form control's `value` by the value given in the `step` attribute, multiplied by `n`. The default value for `n` is 1.Throws `InvalidStateError` exception if the control is neither date- or time-based nor numeric, or if the `step` attribute's value is "any". Does nothing if it would take the control's value out of range or if the control currently has no value.**`input.list`**Returns the `datalist` element indicated by the `list` attribute.

The `value` IDL attribute allows scripts to manipulate the value of an `input` element. The attribute is in one of the following modes, which define its behavior:

**`value`**

On getting, it must return the current value of the element. On setting, it must set the element's value to the new value, set the element's dirty value flag to true, invoke the value sanitization algorithm, if the element's `type` attribute's current state defines one, and then, if the element has a text entry cursor position, should move the text entry cursor position to the end of the text field, unselecting any selected text and resetting the selection direction to `none`.

**`default`**

On getting, if the element has a `value` attribute, it must return that attribute's value; otherwise, it must return the empty string. On setting, it must set the element's `value` attribute to the new value.

**`default/on`**

On getting, if the element has a `value` attribute, it must return that attribute's value; otherwise, it must return the string "on". On setting, it must set the element's `value` attribute to the new value.

**`filename`**

On getting, it must return the string "C:\fakepath\" followed by the filename of the first file in the list of selected files, if any, or the empty string if the list is empty. On setting, if the new value is the empty string, it must empty the list of selected files; otherwise, it must throw an `InvalidStateError` exception.

**Note: This "fakepath" requirement is a sad accident of history. See the example in the File Upload state section for more information.**

The `checked` IDL attribute allows scripts to manipulate the checkedness of an `input` element. On getting, it must return the current checkedness of the element; and on setting, it must set the element's checkedness to the new value and set the element's dirty checkedness flag to true.

The `files` IDL attribute allows scripts to access the element's selected files. On getting, if the IDL attribute applies, it must return a `FileList` object that represents the current selected files. The same object must be returned until the list of selected files changes. If the IDL attribute does not apply, then it must instead return `null`. [FILEAPI]

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

On setting, if the `valueAsDate` attribute does not apply, as defined for the `input` element's `type` attribute's current state, then throw an `InvalidStateError` exception; otherwise, if the new value is null or a `Date` object representing the NaN time value, then set the value of the element to the empty string; otherwise, run the algorithm to convert a `Date` object to a string, as defined for that state, on the new value, and set the value of the element to resulting string.

The `valueAsNumber` IDL attribute represents the value of the element, interpreted as a number.

On getting, if the `valueAsNumber` IDL attribute does not apply, as defined for the `input` element's `type` attribute's current state, then return a `Not-a-Number` object defined from midnight value. Otherwise, if the new value is null or a `Date` object representing the NaN time value, then convert a string to a `Date` object to a string, as defined for that state, on the new value, and set the value of the element to resulting string.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

On setting, if the `valueAsNumber` IDL attribute does not apply, as defined for the `input` element's `type` attribute's current state, then throw an `InvalidStateError` exception; otherwise, if the new value is null or a `Date` object representing the NaN time value, then convert a string to a `Date` object to a string, as defined for that state, on the new value, and set the value of the element to resulting string. Otherwise, run the algorithm to convert a number to a string, as defined for that state, on the new value, and set the value of the element to resulting string.

The `stepDown(n)` and `stepUp(n)` methods, when invoked, must run the following algorithm:

1. If the `stepDown()` and `stepUp()` methods do not apply, as defined for the `input` element's `type` attribute's current state, then throw an `InvalidStateError` exception, and abort these steps.
2. If the element has no allowed value step, then throw an `InvalidStateError` exception, and abort these steps.
3. If the element has a minimum and a maximum and the minimum is greater than the maximum, then abort these steps.
4. If the element has a minimum and a maximum and there is no value greater than or equal to the element's minimum and less than or equal to the element's maximum that, when subtracted from the step base, is an integral multiple of the allowed value step, then abort these steps.
5. If applying the algorithm to convert a string to a number to the string given by the element's value results in an error, then abort these steps; otherwise, let `value` be the result of that algorithm.
6. Let `n` be the argument, or 1 if the argument was omitted.
7. Let `delta` be the allowed value step multiplied by `n`.
8. If the method invoked was the `stepDown()` method, negate `delta`.
9. Let `value` be the result of adding `delta` to `value`.
10. If the element has a minimum, and the `value` is less than that minimum, then set `value` to minimum.
11. If the element has a maximum, and the `value` is greater than that maximum, then set `value` to maximum.
12. If `value` subtracted from the step base is not an integral multiple of the allowed value step, then set `value` to the nearest value that is greater than or equal to the element's minimum, if any, and less than or equal to the element's maximum, if any, that, when subtracted from the step base, is an integral multiple of the allowed value step, rounding down if the method invoked was the `stepDown()` and rounding up otherwise.
13. Let `value as string` be the result of running the algorithm to convert a number to a string, as defined for the `input` element's `type` attribute's current state, on `value`.
14. Set the value of the element to `value as string`.

The `list` IDL attribute must return the current suggestions source element, if any, or null otherwise.

### 4.10.7.5 Common event behaviors

When the `input` event applies, any time the user causes the element's value to change, the user agent must queue a task to fire a

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

field, or undoing an edit in that field. Some user interactions do not cause changes to the value, e.g. hitting the "delete" key in an empty text field, or replacing some text in the field with text from the clipboard that happens to be exactly the same text.

When the `change` event applies, if the element does not have an activation behavior defined but uses a user interface that involves an explicit commit action, then any time the user commits a change to the element's value or list of selected files, the user agent must queue a task to fire a simple event that bubbles named `change` at the `input` element.

An example of a user interface with a commit action would be a File Upload control that consists of a single button that brings up a file selection dialog: when the dialog is closed, if that the file selection changed as a result, then the user has committed a new file.

A note from the spec:  
Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ext-based user input and selecting a date from the drop

When the user follows these steps:  
 /web/20130103030411/http://whatwg.org/html

, the user agent must

1. If the `change` event applies, queue a task to fire a simple event that bubbles named `change` at the `input` element.

2. If the `change` event applies, queue a task to fire a simple event that bubbles named `change` at the `input` element.

**Note:** In addition, when the `change` event applies, `change` events can also be fired as part of the element's activation behavior and as part of the unfocusing steps.

The task source for these tasks is the user interaction task source.

#### 4.10.8 The `button` element

##### Categories:

- Flow content.
- Phrasing content.
- Interactive content.
- Listed, labelable, and submittable form-associated element.
- Palpable content.

##### Contexts in which this element can be used:

Where phrasing content is expected.

##### Content model:

Phrasing content, but there must be no interactive content descendant.

##### Content attributes:

- Global attributes
- `autofocus`
- `disabled`
- `form`
- `formaction`
- `formenctype`
- `formmethod`
- `formnovalidate`
- `formtarget`
- `name`
- `type`
- `value`
- `menu`

##### DOM interface:

```
IDL interface HTMLButtonElement : HTMLElement {
 attribute boolean autofocus;
 attribute boolean disabled;
 readonly attribute HTMLFormElement? form;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
attribute DOMString name;
attribute DOMString type;
attribute DOMString value;
attribute HTMLMenuElement? menu;

readonly attribute boolean willValidate;
readonly attribute ValidityState validity;
readonly attribute DOMString validationMessage;
boolean checkValidity();
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The `button`

The element

The `type` attribute controls the behavior of the button when it is activated. It is an enumerated attribute. The following table lists the keywords and states for the attribute — the keywords in the left column map to the states in the cell in the second column on the same row as the keyword.

Keyword	State	Brief description
<code>submit</code>	Submit Button	Submits the form.
<code>reset</code>	Reset Button	Resets the form.
<code>button</code>	Button	Does nothing.
<code>menu</code>	Menu	Shows a menu.

The *missing value default* is the Submit Button state.

If the `type` attribute is in the Submit Button state, the element is specifically a submit button.

**Constraint validation:** If the `type` attribute is in the Reset Button state, the Button state, or the Menu state, the element is barred from constraint validation.

When a `button` element is not disabled, its activation behavior element is to run the steps defined in the following list for the current state of the element's `type` attribute:

#### Submit Button

If the element has a form owner, the element must submit the form owner from the `button` element.

#### Reset Button

If the element has a form owner, the element must reset the form owner.

#### Button

Do nothing.

#### Menu

The element must follow these steps:

1. If the `button` is not being rendered, abort these steps.
2. Let `menu` be the element's designated pop-up menu, if any. If there isn't one, then abort these steps.
3. Fire a simple event named `show` that is cancelable at `menu`.
4. If the event is not canceled, then construct and show the menu for `menu`.

The `form` attribute is used to explicitly associate the `button` element with its form owner. The `name` attribute represents the element's name. The `disabled` attribute is used to make the control non-interactive and to prevent its value from being submitted. The `autofocus` attribute controls focus. The `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget` attributes are attributes for form submission.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

attribute is not in the Submit Button state.

The `value` attribute gives the element's value for the purposes of form submission. The element's value is the value of the element's `value` attribute, if there is one, or the empty string otherwise.

**Note:** A button (and its value) is only included in the form submission if the button itself was used to initiate the form submission.

If the element'

the ID of a m  
specified if t  
A button e  
button ele  
has no desi  
**Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html**

The `value` a.ent's menu. The value must be  
attribute must not beD is that given by the  
state; otherwise, the element

The `type` IDL attribute must reflect the content attribute of the same name, limited to only known values.

The `willValidate`, `validity`, and `validationMessage` IDL attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `labels` IDL attribute provides a list of the element's labels. The `autofocus`, `disabled`, `form`, and `name` IDL attributes are part of the element's forms API.

The following button is labeled "Show hint" and pops up a dialog box when activated:

```
<button type=button
 onclick="alert('This 15-20 minute piece was composed by George Gershwin.')">
 Show hint
</button>
```

#### 4.10.9 The select element

##### Categories:

- Flow content.
- Phrasing content.
- Interactive content.
- Listed, labelable, submittable, and resettable form-associated element.
- Palpable content.

##### Contexts in which this element can be used:

Where phrasing content is expected.

##### Content model:

Zero or more `option` or `optgroup` elements.

##### Content attributes:

- Global attributes
- `autofocus`
- `disabled`
- `form`
- `multiple`
- `name`
- `required`
- `size`

##### DOM interface:

```
IDL interface HTMLSelectElement : HTMLElement {
 attribute boolean autofocus;
 attribute boolean disabled;
```

Go

**NOV JAN FEB**

03

2012 2013 2014

▼ About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

```
readonly attribute DOMString type;

readonly attribute HTMLOptionsCollection options;
 attribute unsigned long length;
getter Element item(unsigned long index);
object namedItem(DOMString name);
void add((HTMLOptionElement or HTMLOptGroupElement) element, optional
'HTMLElement or long)? before = null);
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
readonly attribute DOMString validationMessage;
boolean checkValidity();
void setCustomValidity(DOMString error);

readonly attribute NodeList labels;
;
```

The `select` element represents a control for selecting amongst a set of options.

The `multiple` attribute is a boolean attribute. If the attribute is present, then the `select` element represents a control for selecting zero or more options from the list of options. If the attribute is absent, then the `select` element represents a control for selecting a single option from the list of options.

The `size` attribute gives the number of options to show to the user. The `size` attribute, if specified, must have a value that is a valid non-negative integer greater than zero.

The **display size** of a `select` element is the result of applying the rules for parsing non-negative integers to the value of element's `size` attribute, if it has one and parsing it is successful. If applying those rules to the attribute's value is not successful, or if the `size` attribute is absent, then the element's display size is 4 if the element's `multiple` content attribute is present, and 1 otherwise.

The **list of options** for a `select` element consists of all the `option` element children of the `select` element, and all the `option` element children of all the `optgroup` element children of the `select` element, in tree order.

The `required` attribute is a boolean attribute. When specified, the user will be required to select a value before submitting the form.

If a `select` element has a `required` attribute specified, does not have a `multiple` attribute specified, and has a display size of 1; and if the value of the first `option` element in the `select` element's list of options (if any) is the empty string, and that `option` element's parent node is the `select` element (and not an `optgroup` element), then that `option` is the `select` element's `placeholder` label option.

If a `select` element has a `required` attribute specified, does not have a `multiple` attribute specified, and has a display size of 1, then the `select` element must have a `placeholder` label option.

**Constraint validation:** If the element has its `required` attribute specified, and either none of the `option` elements in the `select` element's list of options have their `selectedness` set to true, or the only `option` element in the `select` element's list of options with its `selectedness` set to true is the placeholder label option, then the element is suffering from being missing.

If the `multiple` attribute is absent, and the element is not disabled, then the user agent should allow the user to pick an `option` element in its list of options that is itself not disabled. Upon this `option` element being **picked** (either through a click, or through unfocusing the element after changing its value, or through a menu command, or through any other mechanism), and before the relevant user interaction event is queued (e.g. before the `click` event), the user agent must set the selectedness of the picked `option` element to true and then queue a task to fire a simple event that bubbles named `change` at the `select` element, using the user interaction task source as the task source.

If the `multiple` attribute is absent, whenever an `option` element in the `select` element's list of options has its `selected`ness set to true,

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

before the relevant user interaction event is queued (e.g. before the `click` event), the user agent must set the selectedness of that `option` element to false and then queue a task to fire a simple event that bubbles named `change` at the `select` element, using the user interaction task source as the task source.

If nodes are inserted or nodes are removed causing the list of options to gain or lose one or more `option` elements, or if an `option` element in the list of options **asks for a reset**, then, if the `select` element's `multiple` attribute is absent, the `select` element's display size is 1, and no `option` elements in the `select` element's list of options have their selectedness set to true, the user agent must set the selectedness of the first `option` element in the list of options in tree order that is not disabled, if any, to true.

If the `multiple` attribute is set to true, then, if the `select` element's display size is 1, and no `option` elements in the list of options have their selectedness set to true, the user agent must set the selectedness of the first `option` element in the list of options in tree order that is not disabled, if any, to true. The user agent must then queue a task to fire a simple event that bubbles named `change` at the `select` element, using the user interaction task source as the task source.

The `reset` algorithm for the `select` element is as follows. If the `multiple` attribute is set to true, then, if the `select` element's display size is 1, and no `option` elements in the list of options have their selectedness set to true, the user agent must set the selectedness of the first `option` element in the list of options in tree order that is not disabled, if any, to true.

user to **toggle** the selectedness of the first `option` element in the list of options in tree order that is not disabled, if any, to true. The user agent must then queue a task to fire a simple event that bubbles named `change` at the `select` element, using the user interaction task source as the task source.

options, set their selectedness to true. The user agent must then queue a task to fire a simple event that bubbles named `change` at the `select` element, using the user interaction task source as the task source.

The `form` attribute is used to explicitly associate the `select` element with its form owner. The `name` attribute represents the element's name. The `disabled` attribute is used to make the control non-interactive and to prevent its value from being submitted. The `autofocus` attribute controls focus.

A `select` element that is not disabled is *mutable*.

*This box is non-normative. Implementation requirements are given below this box.*

#### `select.type`

Returns "select-multiple" if the element has a `multiple` attribute, and "select-one" otherwise.

#### `select.options`

Returns an `HTMLCollection` of the list of options.

#### `select.length [= value]`

Returns the number of elements in the list of options.

When set to a smaller number, truncates the number of `option` elements in the `select`.

When set to a greater number, adds new blank `option` elements to the `select`.

#### `element = select.item(index)`

#### `select[index]`

Returns the item with index `index` from the list of options. The items are sorted in tree order.

#### `element = select.namedItem(name)`

Returns the item with ID or name `name` from the list of options.

If there are multiple matching items, then a `NodeList` object containing all those elements is returned.

Returns null if no element with that ID could be found.

#### `select.add(element [, before])`

Inserts `element` before the node given by `before`.

The `before` argument can be a number, in which case `element` is inserted before the item with that number, or an element from the list of options, in which case `element` is inserted before that element.

If `before` is omitted, null, or a number out of range, then `element` will be added at the end of the list.

This method will throw a `HierarchyRequestError` exception if `element` is an ancestor of the element into which it is to be inserted.

#### `select.selectedOptions`

Returns an `HTMLCollection` of the list of options that are selected.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

`select.value [= value]`

Returns the value of the first selected item, if any, or the empty string if there is no selected item.

Can be set, to change the selection.

The `type` IDL attribute, on getting, must return the string "select-one" if the `multiple` attribute is absent, and the string "select-multiple" if the `multiple` attribute is present.



e filter matches the elements

The `option` in the list of

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ces at any instant are the

The `option` indices supp

The `length` attribute of t

tting, it must act like the

The `item(index)` method must return the same `options` collection, when invoked with the same argument.

The `namedItem(name)` method must return the value returned by the method of the same name on the `options` collection, when invoked with the same argument.

When the user agent is to **set the value of a new indexed property** for a given property index `index` to a new value `value`, it must instead set the value of a new indexed property with the given property index `index` to the new value `value` on the `options` collection.

Similarly, the `add()` and `remove()` methods must act like their namesake methods on that same `options` collection.

The `selectedOptions` IDL attribute must return an `HTMLCollection` rooted at the `select` node, whose filter matches the elements in the list of options that have their selectedness set to true.

The `selectedIndex` IDL attribute, on getting, must return the index of the first `option` element in the list of options in tree order that has its selectedness set to true, if any. If there isn't one, then it must return -1.

On setting, the `selectedIndex` attribute must set the selectedness of all the `option` elements in the list of options to false, and then the `option` element in the list of options whose index is the given new value, if any, must have its selectedness set to true.

**Note:** This can result in no element having a selectedness set to true even in the case of the `select` element having no `multiple` attribute and a display size of 1.

The `value` IDL attribute, on getting, must return the value of the first `option` element in the list of options in tree order that has its selectedness set to true, if any. If there isn't one, then it must return the empty string.

On setting, the `value` attribute must set the selectedness of all the `option` elements in the list of options to false, and then the first `option` element in the list of options, in tree order, whose value is equal to the given new value, if any, must have its selectedness set to true.

**Note:** This can result in no element having a selectedness set to true even in the case of the `select` element having no `multiple` attribute and a display size of 1.

The `multiple`, `required`, and `size` IDL attributes must reflect the respective content attributes of the same name. The `size` IDL attribute has a default value of zero.

**Note:** For historical reasons, the default value of the `size` IDL attribute does not return the actual size used, which, in the absence of the `size` content attribute, is either 1 or 4 depending on the presence of the `multiple` attribute.

The `willValidate`, `validity`, and `validationMessage` IDL attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `labels` IDL attribute provides a list of the element's labels. The `autofocus`, `disabled`, `form`, and `name` IDL attributes are part of the element's forms API.

The following example shows how a `select` element can be used to offer the user with a set of options from which the user can select a single option. The default option is preselected.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

```
<option value="3" selected> Snipey </option>
<option value="4"> Max </option>
<option value="5"> Firebot </option>
</select>
</p>
```

When there is no default option, a placeholder can be used instead:

```
<select name="unittype" required>
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Here [//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html) options are selected.

```
<p>
<label for="allowedunits">Select unit types to enable on this map:</label>
<select id="allowedunits" name="allowedunits" multiple>
<option value="1" selected> Miner </option>
<option value="2" selected> Puffer </option>
<option value="3" selected> Snipey </option>
<option value="4" selected> Max </option>
<option value="5" selected> Firebot </option>
</select>
</p>
```

Sometimes, a user has to select one or more items. This example shows such an interface.

```
<p>Select the songs from that you would like on your Act II Mix Tape:</p>
<select multiple required name="act2">
<option value="s1">It Sucks to Be Me (Reprise)
<option value="s2">There is Life Outside Your Apartment
<option value="s3">The More You Ruv Someone
<option value="s4">Schadenfreude
<option value="s5">I Wish I Could Go Back to College
<option value="s6">The Money Song
<option value="s7">School for Monsters
<option value="s8">The Money Song (Reprise)
<option value="s9">There's a Fine, Fine Line (Reprise)
<option value="s10">What Do You Do With a B.A. in English? (Reprise)
<option value="s11">For Now
</select>
```

#### 4.10.10 The `datalist` element

##### Categories:

Flow content.  
Phrasing content.

##### Contexts in which this element can be used:

Where phrasing content is expected.

##### Content model:

Either: phrasing content.  
Or: Zero or more `option` elements.

##### Content attributes:

Global attributes

##### DOM interface:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `datalist` element represents a set of `option` elements that represent predefined options for other controls. The contents of the element represents fallback content for legacy user agents, intermixed with `option` elements that represent the predefined options. In the rendering, the `datalist` element represents nothing and it, along with its children, should be hidden.

The `datalist` element is hooked up to an `input` element using the `list` attribute on the `input` element.

Each `option` element that is a descendant of the `datalist` element, that is not disabled, and whose value is a string that isn't the empty string, represents a suggestion. Each suggestion has a value and a label.



`Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html`

*ts are given below this box.*

Re:

The `option` element natches `option` elements.

**Constraint validation:** If an element has a `datalist` element ancestor, it is barred from constraint validation.

#### 4.10.11 The `optgroup` element

##### Categories:

None.

##### Contexts in which this element can be used:

As a child of a `select` element.

##### Content model:

Zero or more `option` elements.

##### Content attributes:

Global attributes  
`disabled`  
`label`

##### DOM interface:

```
IDL interface HTMLOptGroupElement : HTMLElement {
 attribute boolean disabled;
 attribute DOMString label;
};
```

The `optgroup` element represents a group of `option` elements with a common label.

The element's group of `option` elements consists of the `option` elements that are children of the `optgroup` element.

When showing `option` elements in `select` elements, user agents should show the `option` elements of such groups as being related to each other and separate from other `option` elements.

The `disabled` attribute is a boolean attribute and can be used to disable a group of `option` elements together.

The `label` attribute must be specified. Its value gives the name of the group, for the purposes of the user interface. User agents should use this attribute's value when labelling the group of `option` elements in a `select` element.

The `disabled` and `label` attributes must reflect the respective content attributes of the same name.

The following snippet shows how a set of lessons from three courses could be offered in a `select` drop-down widget:

```
<form action="coursesselector.dll" method="get">
<p>Which course would you like to watch today?
<p><label>Course:
<select name="c">
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

<option value="8.02.1">Lecture 01: What holds our world together?
<option value="8.02.2">Lecture 02: Electric Field
<option value="8.02.3">Lecture 03: Electric Flux
<optgroup label="8.03 Physics III: Vibrations and Waves">
 <option value="8.03.1">Lecture 01: Periodic Phenomenon
 <option value="8.03.2">Lecture 02: Beats
 <option value="8.03.3">Lecture 03: Forced Oscillations with Damping
</optgroup>
</label>
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 4.10.12 T

##### Categories

- ↳ <http://whatwg.org/html>

##### Contexts in which this element can be used.

- As a child of a `select` element.
- As a child of a `datalist` element.
- As a child of an `optgroup` element.

##### Content model:

Text.

##### Content attributes:

Global attributes  
`disabled`  
`label`  
`selected`  
`value`

##### DOM interface:

```

IDL [NamedConstructor=Option(),
 NamedConstructor=Option(DOMString text),
 NamedConstructor=Option(DOMString text, DOMString value),
 NamedConstructor=Option(DOMString text, DOMString value, boolean defaultSelected),
 NamedConstructor=Option(DOMString text, DOMString value, boolean defaultSelected,
 boolean selected)]
interface HTMLOptionElement : HTMLElement {
 attribute boolean disabled;
 readonly attribute HTMLFormElement? form;
 attribute DOMString label;
 attribute boolean defaultSelected;
 attribute boolean selected;
 attribute DOMString value;

 attribute DOMString text;
 readonly attribute long index;
};
```

The `option` element represents an option in a `select` element or as part of a list of suggestions in a `datalist` element.

In certain circumstances described in the definition of the `select` element, an `option` element can be a `select` element's placeholder label option. A placeholder label option does not represent an actual option, but instead represents a label for the `select` control.

The `disabled` attribute is a boolean attribute. An `option` element is **disabled** if its `disabled` attribute is present or if it is a child of an `optgroup` element whose `disabled` attribute is present.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The `label` content attribute, if specified, must not be empty. If the attribute is not specified, then the element itself must not be empty.

The `value` attribute provides a value for element. The `value` of an `option` element is the value of the `value` content attribute, if there is one, or, if there is not, the value of the element's `text` IDL attribute.

The `selected` attribute is a boolean attribute. It represents the default selectedness of the element.

The `selectedness` of an `option` element is a boolean state, initially false. Except where otherwise specified, when the element is created, its `selected` attribute is false. When the element is destroyed, its `selected` attribute is set to the element's selectedness.

## Too slow? Try reading the multipage copy of

*Note: This section is non-normative. It contains implementation details and is not part of the standard. The fourth argument is not part of the standard.*

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

A `select` element has a `selected` attribute, which is a boolean attribute. It represents the selectedness of the option element with its selected attribute.

An `option` element's `index` is the number of `option` elements that are in the same list of options but that come before it in tree order. If the `option` element is not in a list of options, then the `option` element's index is zero.

*This box is non-normative. Implementation requirements are given below this box.*

### `option . selected`

Returns true if the element is selected, and false otherwise.

Can be set, to override the current state of the element.

### `option . index`

Returns the index of the element in its `select` element's `options` list.

### `option . form`

Returns the element's `form` element, if any, or null otherwise.

### `option . text`

Same as `textContent`, except that spaces are collapsed.

### `option = new Option([ text [, value [, defaultSelected [, selected ]]]])`

Returns a new `option` element.

The `text` argument sets the contents of the element.

The `value` argument sets the `value` attribute.

The `defaultSelected` argument sets the `selected` attribute.

The `selected` argument sets whether or not the element is selected. If it is omitted, even if the `defaultSelected` argument is true, the element is not selected.

The `disabled` IDL attribute must reflect the `content` attribute of the same name. The `defaultSelected` IDL attribute must reflect the `selected` content attribute.

The `label` IDL attribute, on getting, must return the element's `label`. On setting, the element's `label` content attribute must be set to the new value.

The `value` IDL attribute, on getting, must return the element's `value`. On setting, the element's `value` content attribute must be set to the new value.

The `selected` IDL attribute, on getting, must return true if the element's selectedness is true, and false otherwise. On setting, it must set the element's selectedness to the new value, and then cause the element to ask for a reset.

The `index` IDL attribute must return the element's `index`.



Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
attribute DOMString defaultValue;
[TreatNullAs=EmptyString] attribute DOMString value;
readonly attribute unsigned long textLength;

readonly attribute boolean willValidate;
readonly attribute ValidityState validity;
readonly attribute DOMString validationMessage;
boolean checkValidity();
void setCustomValidity(DOMString error);
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```
unsigned long end,
optional SelectionMode selectionMode);
void setSelectionRange(unsigned long start, unsigned long end, optional DOMString
direction);
};
```

The `textarea` element represents a multiline plain text edit control for the element's **raw value**. The contents of the control represent the control's default value.

The raw value of a `textarea` control must be initially the empty string.

A newline in a `textarea` element, and in its raw value, should separate paragraphs for the purposes of the Unicode bidirectional algorithm. This requirement may be implemented indirectly through the style layer. For example, an HTML+CSS user agent could implement these requirements by implementing the CSS 'unicode-bidi' property. [BIDI] [CSS]

The `readonly` attribute is a boolean attribute used to control whether the text can be edited by the user or not.

In this example, a text field is marked read-only because it represents a read-only file:

```
Filename: <code>/etc/bash.bashrc</code>
<textarea name="buffer" readonly>
System-wide .bashrc file for interactive bash(1) shells.

To enable the settings / commands in this file for login shells as well,
this file has to be sourced in /etc/profile.

If not running interactively, don't do anything
[-z "$PS1"] && return

...</textarea>
```

**Constraint validation:** If the `readonly` attribute is specified on a `textarea` element, the element is barred from constraint validation.

A `textarea` element is mutable if it is neither disabled nor has a `readonly` attribute specified.

When a `textarea` is mutable, its raw value should be editable by the user: the user agent should allow the user to edit, insert, and remove text, and to insert and remove line breaks in the form of U+000A LINE FEED (LF) characters. Any time the user causes the element's raw value to change, the user agent must queue a task to fire a simple event that bubbles named `input` at the `textarea` element. User agents may wait for a suitable break in the user's interaction before queuing the task; for example, a user agent could wait for the user to have not hit a key for 100ms, so as to only fire the event when the user pauses, instead of continuously for each keystroke.

A `textarea` element has a **dirty value flag**, which must be initially set to false, and must be set to true whenever the user interacts with the control in a way that changes the raw value.

When the `textarea` element's `textContent` IDL attribute changes value, if the element's dirty value flag is false, then the element's

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

right writing direction or a right-to-left writing direction. If the user does so, the user agent must then run the following steps:

1. Set the element's `dir` attribute to "ltr" if the user selected a left-to-right writing direction, and "rtl" if the user selected a right-to-left writing direction.
2. Queue a task to fire a simple event that bubbles named `input` at the `textarea` element.

The `cols` attribute specifies the expected maximum number of characters per line. If the `cols` attribute is specified, its value must be a valid non-negative integer greater than zero. If applying the rules for parsing non-negative integers to the attribute's value results in a number greater than zero, then the element's `character width` is that value; otherwise, it is 20.

The user agent prefers the server's preferred character renderings, the user agent should wrap the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `rows` attribute specifies the expected maximum number of lines of text. If the `rows` attribute is specified, its value must be a valid non-negative integer greater than zero, then the user agent should wrap the text at the given height.

Visual user agents should set the height of the control to the number of lines given by the `rows` attribute.

The `wrap` attribute is an enumerated attribute with two keywords and states: the `soft` keyword which maps to the Soft state, and the `hard` keyword which maps to the Hard state. The *missing value default* is the Soft state.

The **Soft** state indicates that the text in the `textarea` is not to be wrapped when it is submitted (though it can still be wrapped in the rendering).

The **Hard** state indicates that the text in the `textarea` is to have newlines added by the user agent so that the text is wrapped when it is submitted.

If the element's `wrap` attribute is in the Hard state, the `cols` attribute must be specified.

For historical reasons, the element's value is normalised in three different ways for three different purposes. The raw value is the value as it was originally set. It is not normalized. The API value is the value used in the `value` IDL attribute. It is normalized so that line breaks use U+000A LINE FEED (LF) characters. Finally, there is the form submission value. It is normalized so that line breaks use U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pairs, and in addition, if necessary given the element's `wrap` attribute, additional line breaks are inserted to wrap the text at the given width.

The element's **API value** is defined to be the element's raw value with the following transformation applied:

1. Replace every U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pair from the raw value with a single U+000A LINE FEED (LF) character.
2. Replace every remaining U+000D CARRIAGE RETURN character from the raw value with a single U+000A LINE FEED (LF) character.

The element's value is defined to be the element's raw value with the following transformation applied:

1. Replace every occurrence of a U+000D CARRIAGE RETURN (CR) character not followed by a U+000A LINE FEED (LF) character, and every occurrence of a U+000A LINE FEED (LF) character not preceded by a U+000D CARRIAGE RETURN (CR) character, by a two-character string consisting of a U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pair.
2. If the element's `wrap` attribute is in the Hard state, insert U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pairs into the string using a UA-defined algorithm so that each line has no more than character width characters. For the purposes of this requirement, lines are delimited by the start of the string, the end of the string, and U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pairs.

The `maxlength` attribute is a form control `maxlength` attribute controlled by the `textarea` element's dirty value flag.

If the `textarea` element has a maximum allowed value length, then the element's children must be such that the code-unit length of the value of the element's `textContent` IDL attribute is equal to or less than the element's maximum allowed value length.

The `required` attribute is a boolean attribute. When specified, the user will be required to enter a value before submitting the form.

**Constraint validation:** If the element has its `required` attribute specified, and the element is mutable, and the element's value is the empty string, then the element is suffering from being missing.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

attribute is more appropriate.

**Note:** These mechanisms are very similar but subtly different: the hint given by the control's `label` is shown at all times; the short hint given in the `placeholder` attribute is shown before the user enters a value; and the hint in the `title` attribute is shown when the user requests further help.

User agents should present this hint to the user, after having stripped line breaks from it, when the element's value is the empty string and the control is not focused (e.g. by displaying it inside a blank unfocused control).

The `name` attribute  
disabled attribute  
used to explain  
attribute controls  
autofill behavior

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/textarea>

ctionality is submitted. The `value` attribute is used. The `form` attribute is set. The `inputmode` attribute provides information about how the user agent provides

ts are given below this box.

Returns the string "textarea".

`textarea.value`

Returns the current value of the element.

Can be set, to change the value.

The `cols`, `placeholder`, `required`, `rows`, and `wrap` attributes must reflect the respective content attributes of the same name. The `cols` and `rows` attributes are limited to only non-negative numbers greater than zero. The `cols` attribute's default value is 20. The `rows` attribute's default value is 2. The `dirName` IDL attribute must reflect the `dirname` content attribute. The `inputMode` IDL attribute must reflect the `inputmode` content attribute, limited to only known values. The `maxLength` IDL attribute must reflect the `maxlength` content attribute, limited to only non-negative numbers. The `readonly` IDL attribute must reflect the `readonly` content attribute.

The `type` IDL attribute must return the value "textarea".

The `defaultValue` IDL attribute must act like the element's `textContent` IDL attribute.

The `value` attribute must, on getting, return the element's API value; on setting, it must set the element's raw value to the new value, set the element's dirty value flag to true, and should then move the text entry cursor position to the end of the text field, unselecting any selected text and resetting the selection direction to `none`.

The `textLength` IDL attribute must return the code-unit length of the element's API value.

The `willValidate`, `validity`, and `validationMessage` IDL attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `labels` IDL attribute provides a list of the element's labels. The `select()`, `selectionStart`, `selectionEnd`, `selectionDirection`, `setRangeText()`, and `setSelectionRange()` methods and IDL attributes expose the element's text selection. The `autofocus`, `disabled`, `form`, and `name` IDL attributes are part of the element's forms API.

Here is an example of a `textarea` being used for unrestricted free-form text input in a form:

```
<p>If you have any comments, please let us know: <textarea cols=80 name=comments>
</textarea></p>
```

To specify a maximum length for the comments, one can use the `maxlength` attribute:

```
<p>If you have any short comments, please let us know: <textarea cols=80 name=comments
maxlength=200></textarea></p>
```

To give a default value, text can be included inside the element:

```
<p>If you have any comments, please let us know: <textarea cols=80 name=comments>You
rock!</textarea></p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

#### 4.10.14 The `keygen` element

##### Categories:

Flow content.  
Phrasing content.  
Interactive content.  
Listed, labelable, submittable, and resettable form-associated element.



**Contexts** Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**Content** /web/20130103030411/http:

**Content** //whatwg.org/html

Global attributes:  
autofocus  
challenge  
disabled  
form  
keytype  
name

##### DOM interface:

```
IDL interface HTMLKeygenElement : HTMLElement {
 attribute boolean autofocus;
 attribute DOMString challenge;
 attribute boolean disabled;
 readonly attribute HTMLFormElement? form;
 attribute DOMString keytype;
 attribute DOMString name;

 readonly attribute DOMString type;

 readonly attribute boolean willValidate;
 readonly attribute ValidityState validity;
 readonly attribute DOMString validationMessage;
 boolean checkValidity();
 void setCustomValidity(DOMString error);

 readonly attribute NodeList labels;
};
```

The `keygen` element represents a key pair generator control. When the control's form is submitted, the private key is stored in the local keystore, and the public key is packaged and sent to the server.

The `challenge` attribute may be specified. Its value will be packaged with the submitted key.

The `keytype` attribute is an enumerated attribute. The following table lists the keywords and states for the attribute — the keywords in the left column map to the states listed in the cell in the second column on the same row as the keyword. User agents are not required to support these values, and must only recognize values whose corresponding algorithms they support.

Keyword	State
rsa	RSA

The `invalid value default` state is the `unknown` state. The `missing value default` state is the `RSA` state, if it is supported, or the `unknown` state otherwise.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

generator, e.g. the key length.

The reset algorithm for `keygen` elements is to set these various configuration settings back to their defaults.

The element's value is the string returned from the following algorithm:

1. Use the appropriate step from the following list:

↳ If the `keytype` attribute is in the RSA state

Generate an RSA key pair using the settings given by the user, if appropriate, using the

D5 and the RSA encryption defined in RFC 2313.

**Too slow? Try reading the multipage copy of**

↳ **the spec instead: https://web.archive.org**

**/web/20130103030411/http:**

Let `//whatwg.org/html`

Let `push` be a function that takes a string and pushes it onto a stack.

Let `signature algorithm` be the selected signature algorithm.

2. If the element has a `challenge` attribute, then let `challenge` be that attribute's value. Otherwise, let `challenge` be the empty string.
3. Let `algorithm` be an ASN.1 `AlgorithmIdentifier` structure as defined by RFC 5280, with the `algorithm` field giving the ASN.1 OID used to identify `signature algorithm`, using the OIDs defined in section 2.2 ("Signature Algorithms") of RFC 3279, and the `parameters` field set up as required by RFC 3279 for `AlgorithmIdentifier` structures for that algorithm. [X690] [RFC5280] [RFC3279]
4. Let `spki` be an ASN.1 `SubjectPublicKeyInfo` structure as defined by RFC 5280, with the `algorithm` field set to the `algorithm` structure from the previous step, and the `subjectPublicKey` field set to the BIT STRING value resulting from ASN.1 DER encoding the `public key`. [X690] [RFC5280]
5. Let `publicKeyAndChallenge` be an ASN.1 `PublicKeyAndChallenge` structure as defined below, with the `spki` field set to the `spki` structure from the previous step, and the `challenge` field set to the string `challenge` obtained earlier. [X690]
6. Let `signature` be the BIT STRING value resulting from ASN.1 DER encoding the signature generated by applying the `signature algorithm` to the byte string obtained by ASN.1 DER encoding the `publicKeyAndChallenge` structure, using `private key` as the signing key. [X690]
7. Let `signedPublicKeyAndChallenge` be an ASN.1 `SignedPublicKeyAndChallenge` structure as defined below, with the `publicKeyAndChallenge` field set to the `publicKeyAndChallenge` structure, the `signatureAlgorithm` field set to the `algorithm` structure, and the `signature` field set to the BIT STRING `signature` from the previous step. [X690]
8. Return the result of base64 encoding the result of ASN.1 DER encoding the `signedPublicKeyAndChallenge` structure. [RFC4648] [X690]

The data objects used by the above algorithm are defined as follows. These definitions use the same "ASN.1-like" syntax defined by RFC 5280. [RFC5280]

```

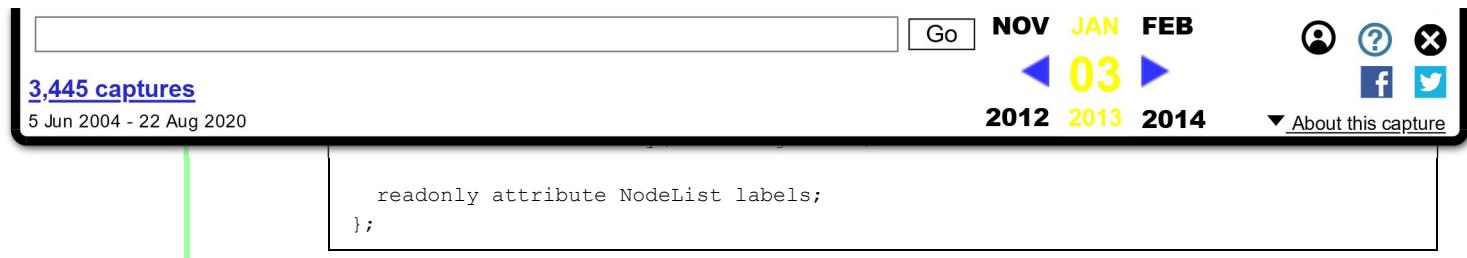
PublicKeyAndChallenge ::= SEQUENCE {
 spki SubjectPublicKeyInfo,
 challenge IA5STRING
}

SignedPublicKeyAndChallenge ::= SEQUENCE {
 publicKeyAndChallenge PublicKeyAndChallenge,
 signatureAlgorithm AlgorithmIdentifier,
 signature BIT STRING
}

```

**Constraint validation:** The `keygen` element is barred from constraint validation.





The `output` element represents the result of a calculation or user action.

The `for` content attribute allows an explicit relationship to be made between the result of a calculation and the elements that represent the values that went into the calculation or that otherwise influenced the calculation. The `for` attribute, if specified, must contain a string consisting of one or more space-separated tokens, each of which must have the value of an ID of an element.

Too slow? Try reading the multipage copy of

The formal name instead: <https://web.archive.org>

Name: \_\_\_\_\_

The element //whatwg.org/html

The element `//www.whatwg.org/html`

When the value mode flag is in mode `default`, the contents of the element represent both the value of the element and its default value. When the value mode flag is in mode `value`, the contents of the element represent the value of the element only, and the default value is only accessible using the `defaultValue` IDL attribute.

Whenever the element's descendants are changed in any way, if the value mode flag is in mode *default*, the element's default value must be set to the value of the element's `textContent` IDL attribute.

The reset algorithm for output elements is to set the element's value mode flag to *default* and then to set the element's `textContent` IDL attribute to the value of the element's default value (thus replacing the element's child nodes).

*This box is non-normative. Implementation requirements are given below this box.*

*output*.value [= value]

Returns the element's current value

Can be set, to change the value.

**output.defaultValue [= value]**

Returns the element's current default value

Can be set to change the default value

### *output*.type

Returns the string "output"

The `value` IDL attribute must act like the element's `textContent` IDL attribute, except that on setting, in addition, before the child nodes are changed, the element's `valueMode` flag must be set to `value`.

The `defaultValue` IDL attribute, on getting, must return the element's default value. On setting, the attribute must set the element's default value, and if the element's value mode flag is in the mode `default`, set the element's `textContent` IDL attribute as well.

The `type` attribute must return the string "output".

The `htmlFor` IDL attribute must reflect the `for` content attribute.

The `willValidate`, `validity`, and `validationMessage` IDL attributes, and the `checkValidity()` and `setCustomValidity()` methods, are part of the constraint validation API. The `labels` IDL attribute provides a list of the element's labels. The `form` and `name` IDL attributes are part of the element's forms API.

A simple calculator could use output for its display of calculated results.

```
<form onsubmit="return false" oninput="o.value = a.valueAsNumber + b.valueAsNumber">
 <input name=a type=number step=any> +
 <input name=b type=number step=any> =
 <output name=o></output>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**Categories:**

Flow content.  
Phrasing content.  
Labelable element.  
Palpable content.

**Contexts in which this element can be used:**

Where phrasing content is expected.

**Content model:**

F Too slow? Try reading the multipage copy of

**Content:** the spec instead: <https://web.archive.org>

C /web/20130103030411/http:

DOM interface [//whatwg.org/html](http://whatwg.org/html)

**IDL**

```
interface HTMLProgressElement : HTMLElement {
 attribute double value;
 attribute double max;
 readonly attribute double position;
 readonly attribute NodeList labels;
};
```

The `progress` element represents the completion progress of a task. The progress is either indeterminate, indicating that progress is being made but that it is not clear how much more work remains to be done before the task is complete (e.g. because the task is waiting for a remote host to respond), or the progress is a number in the range zero to a maximum, giving the fraction of work that has so far been completed.

There are two attributes that determine the current task completion represented by the element. The `value` attribute specifies how much of the task has been completed, and the `max` attribute specifies how much work the task requires in total. The units are arbitrary and not specified.

**Note:** To make a determinate progress bar, add a `value` attribute with the current progress (either a number from 0.0 to 1.0, or, if the `max` attribute is specified, a number from 0 to the value of the `max` attribute). To make an indeterminate progress bar, remove the `value` attribute.

Authors are encouraged to also include the current value and the maximum value inline as text inside the element, so that the progress is made available to users of legacy user agents.

Here is a snippet of a Web application that shows the progress of some automated task:

```
<section>
 <h2>Task Progress</h2>
 <p>Progress: <progress id="p" max=100>0%</progress></p>
 <script>
 var progressBar = document.getElementById('p');
 function updateProgress(newValue) {
 progressBar.value = newValue;
 progressBar.getElementsByTagName('span')[0].textContent = newValue;
 }
 </script>
</section>
```

(The `updateProgress()` method in this example would be called by some other code on the page to update the actual progress bar as the task progressed.)

The `value` and `max` attributes, when present, must have values that are valid floating-point numbers. The `value` attribute, if present, must have a value equal to or greater than zero, and less than or equal to the value of the `max` attribute, if present, or 1.0, otherwise. The `max` attribute, if present, must have a value greater than zero.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**User agent requirements:** If the `value` attribute is omitted, then the progress bar is an indeterminate progress bar. Otherwise, it is a determinate progress bar.

If the progress bar is a determinate progress bar and the element has a `max` attribute, the user agent must parse the `max` attribute's value according to the rules for parsing floating-point number values. If this does not result in an error, and if the parsed value is greater than zero, then the **maximum value** of the progress bar is that value. Otherwise, if the element has no `max` attribute, or if it has one but parsing it resulted in an error, or if the parsed value was less than or equal to zero, then the maximum value of the progress bar is 1.0.

If the progress bar is a determinate progress bar, user agents must parse the `value` attribute's value according to the rules for parsing floating-point numbers. If this does not result in an error, and if the parsed value is maximum value and greater than zero, then the maximum value of the progress bar is greater than or equal to the value. Otherwise, if parsing the `value` attribute resulted in an error, or if the parsed value was less than or equal to zero, then the maximum value of the progress bar is zero.

**UA requirement:** Whether it is the UA should indicate whether the current position of the current value relative to the maximum value is determinate or indeterminate.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

This box is non-normative. Implementation requirements are given below this box.

#### progress . position

For a determinate progress bar (one with known current and maximum values), returns the result of dividing the current value by the maximum value.

For an indeterminate progress bar, returns -1.

If the progress bar is an indeterminate progress bar, then the `position` IDL attribute must return -1. Otherwise, it must return the result of dividing the current value by the maximum value.

If the progress bar is an indeterminate progress bar, then the `value` IDL attribute, on getting, must return 0. Otherwise, it must return the current value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then the `value` content attribute must be set to that string.

**Note:** Setting the `value` IDL attribute to itself when the corresponding content attribute is absent would change the progress bar from an indeterminate progress bar to a determinate progress bar with no progress.

The `max` IDL attribute must reflect the content attribute of the same name, limited to numbers greater than zero. The default value for `max` is 1.0.

The `labels` IDL attribute provides a list of the element's labels.

### 4.10.17 The meter element

#### Categories:

- Flow content.
- Phrasing content.
- Labelable element.
- Palpable content.

#### Contexts in which this element can be used:

Where phrasing content is expected.

#### Content model:

Phrasing content, but there must be no `meter` element descendants.

#### Content attributes:

- Global attributes
- `value`
- `min`
- `max`
- `low`
- `high`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
attribute double value;
attribute double min;
attribute double max;
attribute double low;
attribute double high;
attribute double optimum;
readonly attribute NodeList labels;
};
```

The `meter` element is used to represent a scalar value of arbitrary range — for example, it would be wrong to use this to report a weight, or height, unless there is a known maximum value.

This is also true for the `progress` element.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**Note:** This is also true for the `progress` element.

At role, HTML provides

**Note:** The `meter` element also does not represent a scalar value of arbitrary range — for example, it would be wrong to use this to report a weight, or height, unless there is a known maximum value.

There are six attributes that determine the semantics of the gauge represented by the element.

The `min` attribute specifies the lower bound of the range, and the `max` attribute specifies the upper bound. The `value` attribute specifies the value to have the gauge indicate as the "measured" value.

The other three attributes can be used to segment the gauge's range into "low", "medium", and "high" parts, and to indicate which part of the gauge is the "optimum" part. The `low` attribute specifies the range that is considered to be the "low" part, and the `high` attribute specifies the range that is considered to be the "high" part. The `optimum` attribute gives the position that is "optimum"; if that is higher than the "high" value then this indicates that the higher the value, the better; if it's lower than the "low" mark then it indicates that lower values are better, and naturally if it is in between then it indicates that neither high nor low values are good.

**Authoring requirements:** The `value` attribute must be specified. The `value`, `min`, `low`, `high`, `max`, and `optimum` attributes, when present, must have values that are valid floating-point numbers.

In addition, the attributes' values are further constrained:

Let `value` be the `value` attribute's number.

If the `min` attribute is specified, then let `minimum` be that attribute's value; otherwise, let it be zero.

If the `max` attribute is specified, then let `maximum` be that attribute's value; otherwise, let it be 1.0.

The following inequalities must hold, as applicable:

- $\text{minimum} \leq \text{value} \leq \text{maximum}$
- $\text{minimum} \leq \text{low} \leq \text{maximum}$  (if `low` is specified)
- $\text{minimum} \leq \text{high} \leq \text{maximum}$  (if `high` is specified)
- $\text{minimum} \leq \text{optimum} \leq \text{maximum}$  (if `optimum` is specified)
- $\text{low} \leq \text{high}$  (if both `low` and `high` are specified)

**Note:** If no minimum or maximum is specified, then the range is assumed to be 0..1, and the value thus has to be within that range.

Authors are encouraged to include a textual representation of the gauge's state in the element's contents, for users of user agents that do not support the `meter` element.

The following examples show three gauges that would all be three-quarters full:

```
Storage space usage: <meter value=6 max=8>6 blocks used (out of 8 total)</meter>
Voter turnout: <meter value=0.75></meter>
Tickets sold: <meter min="0" max="100" value="75"></meter>
```

The following example is incorrect use of the element, because it doesn't give a range (and since the default maximum is 1, both of the gauges would end up looking maxed out):

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<p>The grapefruit pie had a radius of 12cm and a height of
2cm.</p>
<dl>
 <dt>Radius: <dd> <meter min=0 max=20 value=12>12cm</meter>
 <dt>Height: <dd> <meter min=0 max=10 value=2>2cm</meter>
</dl>
```

There is no explicit way to specify units in the `meter` element, but the units may be specified in the `title` attribute in free-form text.

The `c`

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**User agent** <http://whatwg.org/html>

`:m</meter>  
:/meter>`

utes using the rules for

User agents must then use all these numbers to obtain values for six points on the gauge, as follows. (The order in which these are evaluated is important, as some of the values refer to earlier ones.)

### The **minimum value**

If the `min` attribute is specified and a value could be parsed out of it, then the minimum value is that value. Otherwise, the minimum value is zero.

### The **maximum value**

If the `max` attribute is specified and a value could be parsed out of it, then the candidate maximum value is that value. Otherwise, the candidate maximum value is 1.0.

If the candidate maximum value is greater than or equal to the minimum value, then the maximum value is the candidate maximum value. Otherwise, the maximum value is the same as the minimum value.

### The **actual value**

If the `value` attribute is specified and a value could be parsed out of it, then that value is the candidate actual value. Otherwise, the candidate actual value is zero.

If the candidate actual value is less than the minimum value, then the actual value is the minimum value.

Otherwise, if the candidate actual value is greater than the maximum value, then the actual value is the maximum value.

Otherwise, the actual value is the candidate actual value.

### The **low boundary**

If the `low` attribute is specified and a value could be parsed out of it, then the candidate low boundary is that value. Otherwise, the candidate low boundary is the same as the minimum value.

If the candidate low boundary is less than the minimum value, then the low boundary is the minimum value.

Otherwise, if the candidate low boundary is greater than the maximum value, then the low boundary is the maximum value.

Otherwise, the low boundary is the candidate low boundary.

### The **high boundary**

If the `high` attribute is specified and a value could be parsed out of it, then the candidate high boundary is that value. Otherwise, the candidate high boundary is the same as the maximum value.

If the candidate high boundary is less than the low boundary, then the high boundary is the low boundary.

Otherwise, if the candidate high boundary is greater than the maximum value, then the high boundary is the maximum value.

Otherwise, the high boundary is the candidate high boundary.

### The **optimum point**

If the `optimum` attribute is specified and a value could be parsed out of it, then the candidate optimum point is that value. Otherwise, the candidate optimum point is the midpoint between the minimum value and the maximum value.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

All of which will result in the following inequalities all being true:

- minimum value ≤ actual value ≤ maximum value
- minimum value ≤ low boundary ≤ high boundary ≤ maximum value
- minimum value ≤ optimum point ≤ maximum value

**UA requirements for regions of the gauge:** If the optimum point is equal to the low boundary or the high boundary, or anywhere in between them, then the region between the low and high boundaries of the gauge must be treated as the optimum region, and the low and high parts, if any, must be treated as suboptimal. Otherwise, if the optimum point is less than the low boundary, then the region between the minimum value and the low boundary must be treated as the optimum region, the region from the low boundary up to the high boundary is a suboptimal region, and the region from the high boundary down to the maximum value is a suboptimal region. Finally, if the optimum point is greater than the high boundary and the maximum value, then the region between the high boundary and the maximum value must be treated as the optimum region.

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

**UA requirements for regions of the gauge**: If the optimum point is equal to the low boundary or the high boundary, or anywhere in between them, then the region between the low and high boundaries of the gauge must be treated as the optimum region, and the low and high parts, if any, must be treated as suboptimal. Otherwise, if the optimum point is less than the low boundary, then the region between the minimum value and the low boundary must be treated as the optimum region, the region from the low boundary up to the high boundary is a suboptimal region, and the region from the high boundary down to the maximum value is a suboptimal region. Finally, if the optimum point is greater than the high boundary and the maximum value, then the region between the high boundary and the maximum value must be treated as the optimum region.

**UA requirements for regions of the gauge**: If the optimum point is equal to the low boundary or the high boundary, or anywhere in between them, then the region between the low and high boundaries of the gauge must be treated as the optimum region, and the low and high parts, if any, must be treated as suboptimal. Otherwise, if the optimum point is less than the low boundary, then the region between the minimum value and the low boundary must be treated as the optimum region, the region from the low boundary up to the high boundary is a suboptimal region, and the region from the high boundary down to the maximum value is a suboptimal region. Finally, if the optimum point is greater than the high boundary and the maximum value, then the region between the high boundary and the maximum value must be treated as the optimum region.

**UA requirements for regions of the gauge**: If the optimum point is equal to the low boundary or the high boundary, or anywhere in between them, then the region between the low and high boundaries of the gauge must be treated as the optimum region, and the low and high parts, if any, must be treated as suboptimal. Otherwise, if the optimum point is less than the low boundary, then the region between the minimum value and the low boundary must be treated as the optimum region, the region from the low boundary up to the high boundary is a suboptimal region, and the region from the high boundary down to the maximum value is a suboptimal region. Finally, if the optimum point is greater than the high boundary and the maximum value, then the region between the high boundary and the maximum value must be treated as the optimum region.

**UA requirements for regions of the gauge**: If the optimum point is equal to the low boundary or the high boundary, or anywhere in between them, then the region between the low and high boundaries of the gauge must be treated as the optimum region, and the low and high parts, if any, must be treated as suboptimal. Otherwise, if the optimum point is less than the low boundary, then the region between the minimum value and the low boundary must be treated as the optimum region, the region from the low boundary up to the high boundary is a suboptimal region, and the region from the high boundary down to the maximum value is a suboptimal region. Finally, if the optimum point is greater than the high boundary and the maximum value, then the region between the high boundary and the maximum value must be treated as the optimum region.

The following markup:

```
<h3>Suggested groups</h3>
<menu type="toolbar">
 Hide suggested groups
</menu>

 <p>comp.infosystems.www.authoring.stylesheets -
 join</p>
 <p>Group description: Layout/presentation on the WWW.</p>
 <p><meter value="0.5">Moderate activity,</meter> Usenet, 618 subscribers</p>

 <p>netscape.public.mozilla.xpininstall -
 join</p>
 <p>Group description: Mozilla XPIInstall discussion.</p>
 <p><meter value="0.25">Low activity,</meter> Usenet, 22 subscribers</p>

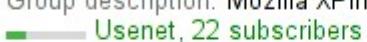
 <p>mozilla.dev.general -
 join</p>
 <p><meter value="0.25">Low activity,</meter> Usenet, 66 subscribers</p>


```

Might be rendered as follows:

**Suggested groups** - [Hide suggested groups](#)

[comp.infosystems.www.authoring.stylesheets](#) - [join](#)  
 Group description: Layout/presentation on the WWW.  
 Usenet, 618 subscribers

[netscape.public.mozilla.xpininstall](#) - [join](#)  
 Group description: Mozilla XPIInstall discussion.  
 Usenet, 22 subscribers

[mozilla.dev.general](#) - [join](#)  
 Usenet, 66 subscribers

User agents may combine the value of the title attribute and the other attributes to provide context-sensitive help or inline text detailing the actual values.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

|| second line.

The `value` IDL attribute, on getting, must return the actual value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then the `value` content attribute must be set to that string.

The `min` IDL attribute, on getting, must return the minimum value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then the `min` content attribute must be set to that string.

The `max` IDL attribute, on getting, must return the maximum value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then the `max` content attribute must be set to that string.

The `low` IDL attribute, on getting, must return the minimum value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then the `low` content attribute must be set to that string.

The `high` IDL attribute, on getting, must return the maximum value. On setting, the given value must be converted to the best representation of the number as a floating-point number and then the `high` content attribute must be set to that string.

The `representative` IDL attribute, on getting, must return the best representation of the number as a floating-point number and then the `representative` content attribute must be set to that string.

The `optimal` IDL attribute, on getting, must return the best representation of the number as a floating-point number and then the `optimal` content attribute must be set to that string.

The `labels` IDL attribute provides a list of the element's labels.

|| The following example shows how a gauge could fall back to localized or pretty-printed text.

```
<p>Disk usage: <meter min=0 value=170261928 max=233257824>170 261 928 bytes used
out of 233 257 824 bytes available</meter></p>
```

#### 4.10.18 Form control infrastructure

##### 4.10.18.1 A form control's value

Form controls have a `value` and a `checkedness`. (The latter is only used by `input` elements.) These are used to describe how the user interacts with the control.

To define the behaviour of constraint validation in the face of the `input` element's `multiple` attribute, `input` elements can also have separately defined `values`.

##### 4.10.18.2 Mutability

A form control can be designated as `mutable`.

**Note:** This determines (by means of definitions and requirements in this specification that rely on whether an element is so designated) whether or not the user can modify the value or checkedness of a form control, or whether or not a control can be automatically prefilled.

##### 4.10.18.3 Association of controls and forms

A form-associated element can have a relationship with a `form` element, which is called the element's **form owner**. If a form-associated element is not associated with a `form` element, its form owner is said to be null.

A form-associated element is, by default, associated with its nearest ancestor `form` element (as described below), but may have a `form` attribute specified to override this.

**Note:** This feature allows authors to work around the lack of support for nested `form` elements.

If a form-associated element has a `form` attribute specified, then that attribute's value must be the ID of a `form` element in the element's owner Document.

**Note:** The rules in this section are complicated by the fact that although conforming documents will never contain nested `form` elements, it is quite possible (e.g. using a script that performs DOM manipulation) to generate documents that have such nested elements. They are also complicated by rules in the HTML parser that, for historical reasons, can result in a form-associated element being associated with a `form` element that is not its

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When a form-associated element or one of its ancestors is inserted into a Document, then the user agent must reset the form owner of that form-associated element. **The HTML parser overrides this requirement when inserting form controls.**

When an element is removed from a Document resulting in a form-associated element and its form owner (if any) no longer being in the same home subtree, then the user agent must reset the form owner of that form-associated element.

When a form-associated element's `form` attribute is set, changed, or removed, then the user agent must reset the form owner of that element.



When a form agent must

## Too slow? Try reading the multipage copy of

changes, then the user

When a form the user age

## the spec instead: [https://web.archive.org](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

ed from the Document, then

When the us

## /web/20130103030411/http:

eps:

1. If th

## //whatwg.org/html

↓ the element's form owner is  
its nearest `form` element ancestor after the change to the ancestor chain, then do nothing, and abort these steps.

2. Let the element's form owner be null.

3. If the element has a `form` content attribute and is itself in a Document, then run these substeps:

1. If the first element in the Document to have an ID that is case-sensitively equal to the element's `form` content attribute's value is a `form` element, then associate the form-associated element with that `form` element.

2. Abort the "reset the form owner" steps.

4. Otherwise, if the form-associated element in question has an ancestor `form` element, then associate the form-associated element with the nearest such ancestor `form` element.

5. Otherwise, the element is left unassociated.

In the following non-conforming snippet:

```
...
<form id="a">
 <div id="b"></div>
</form>
<script>
 document.getElementById('b').innerHTML =
 '<table><tr><td><form id="c"><input id="d"></table>' +
 '<input id="e">';
</script>
...
```

The form owner of "d" would be the inner nested form "c", while the form owner of "e" would be the outer form "a".

This happens as follows: First, the "e" node gets associated with "c" in the HTML parser. Then, the `innerHTML` algorithm moves the nodes from the temporary document to the "b" element. At this point, the nodes see their ancestor chain change, and thus all the "magic" associations done by the parser are reset to normal ancestor associations.

This example is a non-conforming document, though, as it is a violation of the content models to nest `form` elements.

*This box is non-normative. Implementation requirements are given below this box.*

### `element.form`

Returns the element's form owner.

Returns null if there isn't one.

Form-associated elements have a `form` IDL attribute, which, on getting, must return the element's form owner, or null if there isn't one.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

object. If the attribute is specified, its value must not be the empty string.

Any non-empty value for `name` is allowed, but the names "`_charset_`" and "`isindex`" are special:

### `isindex`

This value, if used as the name of a Text control that is the first control in a form that is submitted using the `application/x-www-form-urlencoded` mechanism, causes the submission to only include the value of this control, with no name.

### `_charset_`

This value, if used as the name of a Text control that is the first control in a form that is submitted using the `application/x-www-form-urlencoded` mechanism, causes the submission to only include the value of this control, with no name.

The `name` `IC` **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

### 4.10.19.2 /web/20130103030411/http:

The `dirname` `//whatwg.org/html`

field that con nt, and gives the name of the the empty string.

In this example, a form contains a text field and a submission button:

```
<form action="addcomment.cgi" method=post>
<p><label>Comment: <input type=text name="comment" dirname="comment.dir" required>
</label></p>
<p><button name="mode" type=submit value="add">Post Comment</button></p>
</form>
```

When the user submits the form, the user agent includes three fields, one called "comment", one called "comment.dir", and one called "mode"; so if the user types "Hello", the submission body might be something like:

comment=Hello&**comment.dir=ltr**&mode=add

If the user manually switches to a right-to-left writing direction and enters "مرحبا", the submission body might be something like:

comment=%D9%85%D8%B1%D8%AD%D8%A8%D9%8B%D8%A7&**comment.dir=rtl**&mode=add

### 4.10.19.3 Limiting user input length: the `maxlength` attribute

A form control `maxlength` attribute, controlled by a *dirty value flag*, declares a limit on the number of characters a user can input.

If an element has its form control `maxlength` attribute specified, the attribute's value must be a valid non-negative integer. If the attribute is specified and applying the rules for parsing non-negative integers to its value results in a number, then that number is the element's **maximum allowed value length**. If the attribute is omitted or parsing its value results in an error, then there is no maximum allowed value length.

**Constraint validation:** If an element has a maximum allowed value length, its *dirty value flag* is true, its value was last changed by a user edit (as opposed to a change made by a script), and the code-unit length of the element's value is greater than the element's maximum allowed value length, then the element is suffering from being too long.

User agents may prevent the user from causing the element's value to be set to a value whose code-unit length is greater than the element's maximum allowed value length.

### 4.10.19.4 Enabling and disabling form controls: the `disabled` attribute

The `disabled` content attribute is a boolean attribute.

A form control is **disabled** if its `disabled` attribute is set, or if it is a descendant of a `fieldset` element whose `disabled` attribute is set and is *not* a descendant of that `fieldset` element's first `legend` element child, if any.

A form control that is disabled must prevent any `click` events that are queued on the user interaction task source from being dispatched on the element.

**Constraint validation:** If an element is disabled, it is barred from constraint validation.

The `disabled` IDL attribute must reflect the `disabled` content attribute.

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

The attributes for form submission that may be specified on form elements are action, enctype, method, novalidate, and target.

The corresponding attributes for form submission that may be specified on submit buttons are `formaction`, `formenctype`, `formmethod`, `formnovalidate`, and `formtarget`. When omitted, they default to the values given on the corresponding attributes on the `form` element.

The `action` and `formaction` content attributes, if specified, must have a value that is a valid non-empty URL potentially surrounded by spaces.

The `action` attribute, or Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

The method /web/20130103030411/http://... and states:

- The [/whatwg.org/html](http://whatwg.org/html)
  - The `dialog` element
  - The keyword `dialog`, mapping to the state `dialog`, indicating that submitting the `form` is intended to close the `dialog` box in which the form finds itself, if any, and otherwise not submit.

The *invalid value default* for these attributes is the GET state. (There is no *missing value default*.)

The **method** of an element is one of those states. If the element is a submit button and has a `formmethod` attribute, then the element's method is that attribute's state; otherwise, it is the form owner's `method` attribute's state.

The `enctype` and `formenctype` content attributes are enumerated attributes with the following keywords and states:

- The "application/x-www-form-urlencoded" keyword and corresponding state.
  - The "multipart/form-data" keyword and corresponding state.
  - The "text/plain" keyword and corresponding state.

The *invalid value default* for these attributes is the application/x-www-form-urlencoded state. (There is no *missing value default*.)

The **enctype** of an element is one of those three states. If the element is a submit button and has a `formenctype` attribute, then the element's enctype is that attribute's state; otherwise, it is the form owner's `enctype` attribute's state.

The `target` and `formtarget` content attributes, if specified, must have values that are valid browsing context names or keywords.

The **target** of an element is the value of the element's `formtarget` attribute, if the element is a submit button and has such an attribute; or the value of its form owner's `target` attribute, if *it* has such an attribute; or, if the Document contains a `base` element with a `target` attribute, then the value of the `target` attribute of the first such `base` element; or, if there is no such element, the empty string.

The `novalidate` and `formnovalidate` content attributes are boolean attributes. If present, they indicate that the form is not to be validated during submission.

The **no-validate state** of an element is true if the element is a submit button and the element's `formnovalidate` attribute is present, or if the element's form owner's `novalidate` attribute is present, and false otherwise.

This attribute is useful to include "save" buttons on forms that have validation constraints, to allow users to save their progress even though they haven't fully entered the data in the form. The following example shows a simple form that has two required fields. There are three buttons: one to submit the form, which requires both fields to be filled in; one to save the form so that the user can come back and fill it in later; and one to cancel the form altogether.

```
<form action="editor.cgi" method="post">
<p><label>Name: <input required name=fn></label></p>
<p><label>Essay: <textarea required name=essay></textarea></label></p>
<p><input type=submit name=submit value="Submit essay"></p>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

The `action` IDL attribute must reflect the content attribute of the same name, except that on getting, when the content attribute is missing or its value is the empty string, the document's address must be returned instead. The `target` IDL attribute must reflect the content attribute of the same name. The `method` and `enctype` IDL attributes must reflect the respective content attributes of the same name, limited to only known values. The `encoding` IDL attribute must reflect the `enctype` content attribute, limited to only known values. The `novalidate` IDL attribute must reflect the `novalidate` content attribute. The `formAction` IDL attribute must reflect the `formaction` content attribute, except that on getting, when the content attribute is missing or its value is the empty string, the document's address must be returned instead. The `formEnctype` IDL attribute must reflect the `formenctype` content attribute, limited to only known values. The `formMethod` IDL attribute must reflect the `formmethod` content attribute, limited to only known values. The `formNoValidate` IDL attribute must reflect the `formnovalidate` content attribute. The `formTarget` IDL attribute must reflect the `formtarget` content attribute.

## Too slow? Try reading the multipage copy of

**4.10.19.6** the spec instead: <https://web.archive.org>

The `autofocus` attribute is a boolean attribute. It indicates whether the page is loaded or as soon as the `dial` control is focused.

The page is loaded or as soon as the `dial` control is focused.

The `autofocus` attribute is a boolean attribute.

An element's **nearest ancestor autofocus scoping root element** is the element itself if the element is a `dialog` element, or else is the element's nearest ancestor `dialog` element, if any, or else is the element's root element.

There must not be two elements with the same nearest ancestor autofocus scoping root element that both have the `autofocus` attribute specified.

When an element with the `autofocus` attribute specified is inserted into a document, user agents should run the following steps:

1. Let `target` be the element's `Document`.
2. If `target` has no browsing context, abort these steps.
3. If `target`'s browsing context has no top-level browsing context (e.g. it is a nested browsing context with no parent browsing context), abort these steps.
4. If `target`'s active sandboxing flag set has the `sandboxed automatic features` browsing context flag, abort these steps.
5. If `target`'s origin is not the same as the origin of the `Document` of the currently focused element in `target`'s top-level browsing context, abort these steps.
6. If `target`'s origin is not the same as the origin of the active document of `target`'s top-level browsing context, abort these steps.
7. If the user agent has already reached the last step of this list of steps in response to an element being inserted into a `Document` whose top-level browsing context's active document is the same as `target`'s top-level browsing context's active document, abort these steps.
8. If the user has indicated (for example, by starting to type in a form control) that he does not wish focus to be changed, then optionally abort these steps.
9. Queue a task that checks to see if the element is focusable, and if so, runs the focusing steps for that element. User agents may also change the scrolling position of the document, or perform some other action that brings the element to the user's attention. The task source for this task is the DOM manipulation task source.

**Note:** This handles the automatic focusing during document load. The `showModal()` method of `dialog` elements also processes the `autofocus` attribute.

**Note:** Focusing the control does not imply that the user agent must focus the browser window if it has lost focus.

The `autofocus` IDL attribute must reflect the content attribute of the same name.

In the following snippet, the text control would be focused when the document was loaded.

```
<input maxlength="256" name="q" value="" autofocus>
<input type="submit" value="Search">
```

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

User agents must recognise all the keywords and corresponding states given below, but need not support all of the corresponding states. If a keyword's state is not supported, the user agent must act as if the keyword instead mapped to the given state's fallback state, as defined below. This fallback behaviour is transitive.

**Note:** For example, if a user agent with a QWERTY keyboard layout does not support text prediction and automatic capitalization, then it could treat the latin-prose keyword in the same way as the verbatim keyword, following the chain Latin Prose → Latin Text → Latin Verbatim.

The possible keywords and states for the attributes are listed in the following table. The keyword is listed in the first column. Each maps to the state given in the cell in the third column. Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/latin-name>

verbatim			passwords, product codes.
latin			ids enabled (e.g. text prediction). arch fields.
latin-name			tended for entering human names
			enabled (e.g. text prediction from the user's contact list and automatic capitalisation at every word). Intended for situations such as customer name fields.
latin-prose	Latin Prose	Latin Text	Latin-script input in the user's preferred language(s), with aggressive typing aids intended for human-to-human communications enabled (e.g. text prediction and automatic capitalisation at the start of sentences). Intended for situations such as e-mails and instant messaging.
full-width-latin	Full-width Latin	Latin Prose	Latin-script input in the user's secondary language(s), using full-width characters, with aggressive typing aids intended for human-to-human communications enabled (e.g. text prediction and automatic capitalisation at the start of sentences). Intended for latin text embedded inside CJK text.
kana	Kana	Default	Kana or romaji input, typically hiragana input, using full-width characters, with support for converting to kanji. Intended for Japanese text input.
katakana	Katakana	Kana	Katakana input, using full-width characters, with support for converting to kanji. Intended for Japanese text input.
numeric	Numeric	Default	Numeric input, including keys for the digits 0 to 9, the user's preferred thousands separator character, and the character for indicating negative numbers. Intended for numeric codes, e.g. credit card numbers. (For numbers, prefer "<input type=number>".)
tel	Telephone	Numeric	Telephone number input, including keys for the digits 0 to 9, the "#" character, and the "*" character. In some locales, this can also include alphabetic mnemonic labels (e.g. in the US, the key labeled "2" is historically also labeled with the letters A, B, and C). Rarely necessary; use "<input type=tel>" instead.
email	E-mail	Default	Text input in the user's locale, with keys for aiding in the input of e-mail addresses, such as that for the "@" character and the "." character. Rarely necessary; use "<input type=email>" instead.
url	URL	Default	Text input in the user's locale, with keys for aiding in the input of Web addresses, such as that for the "/" and "." characters and for quick input of strings commonly found in domain names such as "www." or ".co.uk". Rarely necessary; use "<input type=url>" instead.

The last three keywords listed above are only provided for completeness, and are rarely necessary, as dedicated input controls exist for their usual use cases (as described in the table above).

User agents must all support the **Default** input mode state, which corresponds to the user agent's default input modality. This specification does not define how the user agent's default modality is to operate. The *missing value default* is the default input mode state.

User agents should use the input modality corresponding to the state of the `inputmode` attribute when exposing a user interface for editing the value of a form control to which the attribute applies. An input modality corresponding to a state is one designed to fit the description of the state in the table above. This value can change dynamically; user agents should update their interface as the attribute changes state, unless that would go against the user's wishes.

### 4.10.19.8 Autofilling form controls: the `autocomplete` attribute

User agents sometimes have features for helping users fill forms in, for example prefilling the user's address based on earlier user input. The `autocomplete` content attribute can be used to hint to the user agent how to, or indeed whether to, provide such a feature.

The attribute, if present, must have a value that is a set of space-separated tokens consisting of either a single token that is an ASCII case-insensitive match for the string "off", or a single token that is an ASCII case-insensitive match for the string "on", or the following, in the order given below:

1. Optionally, a token whose first eight characters are an ASCII case-insensitive match for the string "section-", meaning that

Go NOV JAN FEB  
 ◀ 03 ▶  
 2012 2013 2014 ▾ About this capture

[3,445 captures](#)  
 5 Jun 2004 - 22 Aug 2020

```
<legend>Ship the blue gift to...</legend>
<p> <label> Address: <input name=ba autocomplete="section-blue shipping street-address"> </label>
<p> <label> City: <input name=bc autocomplete="section-blue shipping region"> </label>
<p> <label> Postal Code: <input name=bp autocomplete="section-blue shipping postal-code"> </label>
</fieldset>
<fieldset>
```



Too slow? Try reading the multipage copy of  
 the spec instead: [https://web.archive.org  
 /web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

*i shipping street-*

*i shipping*

*i shipping*

2. Optionally, a token that is an ASCII case-insensitive match for one of the following strings:

- "shipping", meaning the field is part of the shipping address or contact information
- "billing", meaning the field is part of the billing address or contact information

3. Either of the following two options:

- A token that is an ASCII case-insensitive match for one of the following autofill field strings:

- "name"
- "honorific-prefix"
- "given-name"
- "additional-name"
- "family-name"
- "honorific-suffix"
- "nickname"
- "organization-title"
- "organization"
- "street-address"
- "address-line1"
- "address-line2"
- "address-line3"
- "locality"
- "region"
- "country"
- "postal-code"
- "cc-name"
- "cc-given-name"
- "cc-additional-name"
- "cc-family-name"
- "cc-number"
- "cc-exp"
- "cc-exp-month"
- "cc-exp-year"
- "cc-csc"
- "cc-type"
- "language"
- "bday"
- "bday-day"
- "bday-month"
- "bday-year"
- "sex"
- "url"
- "photo"

(See the table below for descriptions of these values.)

- The following, in the given order:

1. Optionally, a token that is an ASCII case-insensitive match for one of the following strings:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

2. A token that is an ASCII case-insensitive match for one of the following autofocus field strings:

- "tel"
- "tel-country-code"
- "tel-national"
- "tel-area-code"
- "tel-local"
- "tel-local-prefix"
- "tel-local-suffix"
- "tel-extension"



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The "off" keyword indicates that the user agent is allowed to provide the user with completion values, or explicitly enter the document provides its own autocor-

ivation code for a nuclear weapon); or the user will therefore have to t the document provides its

The "on" keyword indicates that the user agent is allowed to provide the user with completion values, but does not provide any further information about what kind of data the user might be expected to enter. User agents would have to use heuristics to decide what completion values to suggest.

The **autofill fields** names listed above indicate that the user agent is allowed to provide the user with completion values, and specifies what kind of value is expected. The keywords relate to each other as described in the table below. Each field name listed on a row of this table corresponds to the meaning given in the cell for that row in the column labeled "Meaning". Some fields correspond to subparts of other fields; for example, a credit card expiry date can be expressed as one field giving both the month and year of expiry ("cc-exp"), or as two fields, one giving the month ("cc-exp-month") and one the year ("cc-exp-year"). In such cases, the names of the broader fields cover multiple rows, in which the narrower fields are defined.

**Note: Generally, authors are encouraged to use the broader fields rather than the narrower fields, as the narrower fields tend to expose Western biases. For example, while it is common in some Western cultures to have a given name and a family name, in that order (and thus often referred to as a first name and a surname), many cultures put the family name first and the given name second, and many others simply have one name (a mononym). Having a single field is therefore more flexible.**

Field name	Meaning	Example
"name"	Full name	Sir Timothy John Berners-Lee, OM, KBE, FRS, FREng, FRSA
"honorific-prefix"	Prefix or title (e.g. "Mr.", "Ms.", "Dr.", "M <sup>lle</sup> ")	Sir
"given-name"	Given name (in some Western cultures, also known as the <i>first name</i> )	Timothy
"additional-name"	Additional names (in some Western cultures, also known as <i>middle names</i> , forenames other than the first name)	John
"family-name"	Family name (in some Western cultures, also known as the <i>last name</i> or <i>surname</i> )	Berners-Lee
"honorific-suffix"	Suffix (e.g. "Jr.", "B.Sc.", "MBASW", "II")	OM, KBE, FRS, FREng, FRSA
"nickname"	Nickname, screen name, handle: a typically short name used instead of the full name	Tim
"organization-title"	Job title (e.g. "Software Engineer", "Senior Vice President", "Deputy Managing Director")	Professor
"organization"	Company name corresponding to the person, address, or contact information in the other fields associated with this field	World Wide Web Consortium
"street-address"	Street address (as one line)	32 Vassar Street; MIT Room 32-G524
"address-line1"	Street address (as multiple lines)	32 Vassar Street
"address-line2"		MIT Room 32-G524
"address-line3"		
"locality"	City, town, village, or other locality within which the relevant street address is found	Cambridge
"region"	Province such as a state, county, or canton within which the locality is found	MA
"country"	Country	USA
"postal-code"	Postal code, post code, ZIP code	02139
"cc-name"	Full name as given on the payment instrument	Tim Berners-Lee
"cc-given-name"	Given name as given on the payment instrument (in some Western cultures, also known as the <i>first name</i> )	Tim

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

"cc-number"	Code identifying the payment instrument (e.g. the credit card number, bank account number)	4114360123456785
"cc-exp"	Expiration date of the payment instrument	2014-12
"cc-exp-month"	Month component of the expiration date of the payment instrument	12
"cc-exp-year"	Year component of the expiration date of the payment instrument	2014
"cc-csc"	Security code for the payment instrument (also known as the card security code (CSC), card validation code (CVC), card verification value (CVV), signature panel code (SPC), credit card ID (CCID), etc)	419
"cc-type"	Type of payment instrument	Visa
"language"		English
"bday"		955-06-08
"bday-4"		une
"bday-3"		955
"bday-2"		fale
"sex"		http://www.w3.org/People/Berners-ee/
"url"		
"photo"	Photograph, icon, or other image corresponding to the company, person, address, or contact information in the other fields associated with this field	http://www.w3.org/Press/Stock/Berners-Lee/2001-europaeum-eighth.jpg
"tel"	Full telephone number, including country code	+1 617 253 5702
"tel-country-code"	Country code component of the telephone number	+1
"tel-national"	Telephone number without the country code component	617 253 5702
"tel-area-code"	Area code component of the telephone number	617
"tel-local"	Telephone number without the country code and area code components	2535702
"tel-local-prefix"	First part of the component of the telephone number that follows the area code, when that component is split into two components	253
"tel-local-suffix"	Second part of the component of the telephone number that follows the area code, when that component is split into two components	5702
"tel-extension"	Telephone number internal extension code	1000
"email"	E-mail address	timbl@w3.org
"impp"	URL representing an instant messaging protocol endpoint (for example, "aim:goim?screenname=example" or xmpp:fred@example.net")	irc://example.org/timbl,isuser

If the `autocomplete` attribute is omitted, the default value corresponding to the state of the element's form owner's `autocomplete` attribute is used instead (either "on" or "off"). If there is no form owner, then the value "on" is used.

Each `input`, `select`, and `textarea` element has an **autofill hint set**, an **autofill scope**, an **autofill field name**, and an **IDL-exposed autofill value** whose values are defined as the result of running the following algorithm:

1. If the element has no `autocomplete` attribute, then jump to the step labeled *default*.
2. Let *tokens* be the result of splitting the attribute's value on spaces.
3. If *tokens* is empty, then jump to the step labeled *default*.
4. Let *index* be the index of the last token in *tokens*.
5. If the *index* token in *tokens* is not an ASCII case-insensitive match for one of the tokens given in the first column of the following table, or if the number of tokens in *tokens* is greater than the maximum number given in the cell in the second column of that token's row, then jump to the step labeled *default*. Otherwise, let *field* be the string given in the cell of the first column of the matching row, and let *category* be the value of the cell in the third column of that same row.

Token	Maximum number of tokens	Category
"off"	1	Off
"on"	1	Automatic
"name"	3	Normal
"honorific-prefix"	3	Normal
"given-name"	3	Normal

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

"nickname"	3	Normal
"organization-title"	3	Normal
"organization"	3	Normal
"street-address"	3	Normal
"address-line1"	3	Normal
"address-line2"	3	Normal
"address-line3"	3	Normal
"locality"	3	Normal



"rec" Too slow? Try reading the multipage copy of  
"co1" the spec instead: [https://web.archive.org  
"po:](https://web.archive.org/web/20130103030411/http://whatwg.org/html) /web/20130103030411/http:  
"cc: //whatwg.org/html

"cc-n"		
"cc-exp"	3	Normal
"cc-exp-month"	3	Normal
"cc-exp-year"	3	Normal
"cc-csc"	3	Normal
"cc-type"	3	Normal
"language"	3	Normal
"bday"	3	Normal
"bday-day"	3	Normal
"bday-month"	3	Normal
"bday-year"	3	Normal
"sex"	3	Normal
"url"	3	Normal
"photo"	3	Normal
"tel"	4	Contact
"tel-country-code"	4	Contact
"tel-national"	4	Contact
"tel-area-code"	4	Contact
"tel-local"	4	Contact
"tel-local-prefix"	4	Contact
"tel-local-suffix"	4	Contact
"tel-extension"	4	Contact
"email"	4	Contact
"impp"	4	Contact

6. If **category** is Off, let the element's **autocomplete** field name be the string "off", let its **autocompletetype** hint set be empty, and let its IDL-exposed **autocomplete** value be the string "off". Then, abort these steps.
7. If **category** is Automatic, let the element's **autocomplete** field name be the string "on", let its **autocompletetype** hint set be empty, and let its IDL-exposed **autocomplete** value be the string "on". Then, abort these steps.
8. Let **scope tokens** be an empty list.
9. Let **hint tokens** be an empty set.
10. Let **IDL value** have the same value as **field**.
11. If the **index** token in **tokens** is the first entry, then skip to the step labeled **done**.
12. Decrement **index** by one.
13. If **category** is Contact and the **index** token in **tokens** is an ASCII case-insensitive match for one of the strings in the following list, then run the substeps that follow:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The substeps are:

1. Let *contact* be the matching string from the list above.
2. Insert *contact* at the start of *scope tokens*.
3. Add *contact* to *hint tokens*.
4. Let *IDL value* be the concatenation of *contact*, a U+0020 SPACE character, and the previous value of *IDL value* (which at this point will always be *field*).

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

14. If the *sub* step is not part of a *for* loop, then let *tokens* be the concatenation of *tokens* and the current value of *IDL value*, then run the following steps:

The substeps are:

1. Let *mode* be the matching string from the list above.
2. Insert *mode* at the start of *scope tokens*.
3. Add *mode* to *hint tokens*.
4. Let *IDL value* be the concatenation of *mode*, a U+0020 SPACE character, and the previous value of *IDL value* (which at this point will either be *field* or the concatenation of *contact*, a space, and *field*).
5. If the *index* entry in *tokens* is the first entry, then skip to the step labeled *done*.
6. Decrement *index* by one.
15. If the *index* entry in *tokens* is not the first entry, then jump to the step labeled *default*.
16. If the first eight characters of the *index* entry in *tokens* are not an ASCII case-insensitive match for the string "section-", then jump to the step labeled *default*.
17. Let *section* be the *index* entry in *tokens*, converted to ASCII lowercase.
18. Insert *section* at the start of *scope tokens*.
19. Let *IDL value* be the concatenation of *section*, a U+0020 SPACE character, and the previous value of *IDL value*.
20. *Done*: Let the element's autofill hint set be *hint tokens*.
21. Let the element's autofill scope be *scope tokens*.
22. Let the element's autofill field name be *field*.
23. Let the element's IDL-exposed autofill value be *IDL value*.
24. Abort these steps.
25. *Default*: Let the element's IDL-exposed autofill value be the empty string, and its autofill hint set and autofill scope be empty.
26. Let *form* be the element's form owner, if any, or null otherwise.
27. If *form* is not null and *form*'s *autocomplete* attribute is in the off state, then let the element's autofill field name be "off". Otherwise, let the element's autofill field name be "on".

When an element's autofill field name is "off", the user agent should not remember the control's value, and should not offer past values to the user.

**Note: In addition, when an element's autofill field name is "off", values are reset when traversing the history.**

Go

NOV JAN FEB

03

2012 2013 2014



## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When an element's `autofill` field name is *not* "off", the user agent may store the control's value, and may offer previously stored values to the user.

When the `autofill` field name is "on", the user agent should attempt to use heuristics to determine the most appropriate values to offer the user, e.g. based on the element's `name` value, the position of the element in the document's DOM, what other fields exist in the form, and so forth.

When the `autofill` field name is one of the names of the `autofill` fields described above, the user agent should provide suggestions that match the meaning of the field name as given in the table earlier in this section. The `autofill hint set` should be used to select amongst multiple possibl<sup>1</sup> suggestions.

For example, if the user agent knows that the user has entered "123" into a field with `autofill="given-name"`, it would suggest another address into fields for form controls whose `name` is "given-name".

When the user agent finds a `select` element with `autofill="given-name"`, it must use data relating to the same person as the user agent has previously seen.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Suppose a user agent knows of two phone numbers, +1 555 123 1234 and +1 555 666 7777. It would not be conforming for the user agent to fill a field with `autocomplete="shipping tel-local-prefix"` with the value "123" and another field in the same form with `autocomplete="shipping tel-local-suffix"` with the value "7777". The only valid prefilled values given the aforementioned information would be "123" and "1234", or "666" and "7777", respectively.

Similarly, if a form for some reason contained both a "cc-exp" field and a "cc-exp-month" field, and the user agent prefilled the form, then the month component of the former would have to match the latter.

The autocompletion mechanism must be implemented by the user agent acting as if the user had modified the element's value, and must be done at a time where the element is *mutable* (e.g. just after the element has been inserted into the document, or when the user agent stops parsing). User agents must only prefill controls using values that the user could have entered.

For example, if a `select` element only has `option` elements with values "Steve" and "Rebecca", "Jay", and "Bob", and has an `autofill` field name "given-name", but the user agent's only idea for what to prefill the field with is "Evan", then the user agent cannot prefill the field. It would not be conforming to somehow set the `select` element to the value "Evan", since the user could not have done so themselves.

A user agent prefiling a form control's value must not cause that control to suffer from a type mismatch, suffer from a pattern mismatch, suffer from being too long, suffer from an underflow, suffer from an overflow, or suffer from a step mismatch. Where possible, user agents should use heuristics to attempt to convert values so that they can be used.

For example, if the user agent knows that the user's middle name is "Ines", and attempts to prefill a form control that looks like this:

```
<input name=middle-initial maxlength=1 autocomplete="additional-name">
```

...then the user agent could convert "Ines" to "I" and prefill it that way.

A more elaborate example would be with month values. If the user agent knows that the user's birthday is the 27th of July 2012, then it might try to prefill all of the following controls with slightly different values, all driven from this information:

<code>&lt;input name=b type=month autocomplete="bday"&gt;</code>	2012-07	The day is dropped since the Month state only accepts a month/year combination.
------------------------------------------------------------------	---------	---------------------------------------------------------------------------------

<code>&lt;select name=c autocomplete="bday"&gt;</code> <code>&lt;option&gt;Jan</code> <code>&lt;option&gt;Feb</code> <code>...</code> <code>&lt;option&gt;Jul</code> <code>&lt;option&gt;Aug</code> <code>...</code> <code>&lt;/select&gt;</code>	July	The user agent picks the month from the listed options, either by noticing there are twelve options and picking the 7th, or by recognising that one of the strings (three characters "Jul" followed by a newline and a space) is a close match for the name of the month (July) in one of the user agent's supported languages, or through some other similar mechanism.
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>&lt;input name=a type=number min=1 max=12 autocomplete="bday-month"&gt;</code>	7	User agent converts "July" to a month number in the range 1..12, like the field.
--------------------------------------------------------------------------------------	---	----------------------------------------------------------------------------------

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<input name=a type=number min=1
max=11 autocomplete="bday-
month">
```

A user agent may allow the user to override an element's autofocus field name, e.g. to change it from "off" to "on" to allow values to be remembered and prefilled despite the page author's objections, or to always "off", never remembering values. However, user agents should not allow users to trivially override the autofocus field name from "off" to "on" or other values, as there are significant security implications for the user if all values are always remembered, regardless of the site's preferences.

The `autoco`

content attril

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

**4.10.20 A**The `input`:

vo\_

```
attribute unsigned long selectionStart;
attribute unsigned long selectionEnd;
attribute DOMString selectionDirection;
void setRangeText(DOMString replacement);
void setRangeText(DOMString replacement, unsigned long start, unsigned long end, optional
SelectionMode selectionMode);
void setSelectionRange(unsigned long start, unsigned long end, optional DOMString
direction = "preserve");
```

The `setRangeText` method uses the following enumeration:

```
enum SelectionMode {
 "select",
 "start",
 "end",
 "preserve",
};
```

These methods and attributes expose and control the selection of `input` and `textarea` text fields.

*This box is non-normative. Implementation requirements are given below this box.*

**`element.select()`**

Selects everything in the text field.

**`element.selectionStart [= value]`**

Returns the offset to the start of the selection.

Can be set, to change the start of the selection.

**`element.selectionEnd [= value]`**

Returns the offset to the end of the selection.

Can be set, to change the end of the selection.

**`element.selectionDirection [= value]`**

Returns the current direction of the selection.

Can be set, to change the direction of the selection.

The possible values are "forward", "backward", and "none".

**`element.setSelectionRange(start, end [, direction] )`**

Changes the selection to cover the given substring in the given direction. If the direction is omitted, it will be reset to be the platform default (none or forward).

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**"select"**

Selects the newly inserted text.

**"start"**

Moves the selection to just before the inserted text.

**"end"**

Moves the selection to just after the selected text



## <sup>P1</sup> Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

For input elements, throw an `InputEvent`

while they don't apply, must

For `input` elements, these methods and attributes must operate on the element's value. For `textarea` elements, these methods and attributes must operate on the element's raw value.

Where possible, user interface features for changing the text selection in `input` and `textarea` elements must be implemented in terms of the DOM API described in this section, so that, e.g., all the same events fire.

The selections of `input` and `textarea` elements have a `direction`, which is either `forward`, `backward`, or `none`. This direction is set when the user manipulates the selection. The exact meaning of the selection direction depends on the platform.

**Note:** On Windows, the `direction` indicates the position of the caret relative to the selection: a forward selection has the caret at the end of the selection and a backward selection has the caret at the start of the selection. Windows has no `none` direction. On Mac, the `direction` indicates which end of the selection is affected when the user adjusts the size of the selection using the arrow keys with the Shift modifier: the forward direction means the end of the selection is modified, and the backwards direction means the start of the selection is modified. The `none` direction is the default on Mac, it indicates that no particular direction has yet been selected. The user sets the direction implicitly when first adjusting the selection, based on which directional arrow key was used.

The `select()` method must cause the contents of the text field to be fully selected, with the selection direction being `none`, if the platform supports selections with the direction `none`, or otherwise `forward`. The user agent must then queue a task to fire a simple event that bubbles named `select` at the element, using the user interaction task source as the task source.

The `selectionStart` attribute must, on getting, return the offset (in logical order) to the character that immediately follows the start of the selection. If there is no selection, then it must return the offset (in logical order) to the character that immediately follows the text entry cursor.

On setting, it must act as if the `setSelectionRange()` method had been called, with the new value as the first argument; the current value of the `selectionEnd` attribute as the second argument, unless the current value of the `selectionEnd` is less than the new value, in which case the second argument must also be the new value; and the current value of the `selectionDirection` as the third argument.

The `selectionEnd` attribute must, on getting, return the offset (in logical order) to the character that immediately follows the end of the selection. If there is no selection, then it must return the offset (in logical order) to the character that immediately follows the text entry cursor.

On setting, it must act as if the `setSelectionRange()` method had been called, with the current value of the `selectionStart` attribute as the first argument, the new value as the second argument, and the current value of the `selectionDirection` as the third argument.

The `selectionDirection` attribute must, on getting, return the string corresponding to the current selection direction: if the direction is `forward`, "forward"; if the direction is `backward`, "backward"; and otherwise, "none".

On setting, it must act as if the `setSelectionRange()` method had been called, with the current value of the `selectionStart` attribute as the first argument, the current value of the `selectionEnd` attribute as the second argument, and the new value as the third argument.

The `setSelectionRange(start, end, direction)` method must set the selection of the text field to the sequence of characters

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

a case-sensitive match for the string "forward" or if the platform does not support selections with the direction `none`, and `none` otherwise (including if the argument is omitted). The user agent must then queue a task to fire a simple event that bubbles named `select` at the element, using the user interaction task source as the task source.

The `setRangeText(replacement, start, end, selectMode)` method must run the following steps:

1. If the method has only one argument, then let `start` and `end` have the values of the `selectionStart` attribute and the `selectionEnd` attribute respectively.

Otherwise

2. If `start` is less than `end`, then let `replacement` be the value of the text field, and let `new length` be the length of the value of the text field.
3. If `end` is greater than `start`, then let `new start` be `start`, let `new end` be `end`, and let `new length` be the length of the value of the text field.
4. Let `old length` be the length of the value of the text field.
5. Let `delta` be `new length` minus `old length`.

6. If `start` is less than `end`, delete the sequence of characters starting with the character at the `start`th position (in logical order) and ending with the character at the `(end-1)`th position.

7. Insert the value of the first argument into the text of the value of the text field, immediately before the `start`th character.

8. Let `new length` be the length of the value of the first argument.

9. Let `new end` be the sum of `start` and `new length`.

10. Run the appropriate set of substeps from the following list:

↳ If the fourth argument's value is "select"

Let `selection start` be `start`.

Let `selection end` be `new end`.

↳ If the fourth argument's value is "start"

Let `selection start` and `selection end` be `start`.

↳ If the fourth argument's value is "end"

Let `selection start` and `selection end` be `new end`.

↳ If the fourth argument's value is "preserve", or if the argument was omitted

1. Let `old length` be `end` minus `start`.

2. Let `delta` be `new length` minus `old length`.

3. If `selection start` is greater than `end`, then increment it by `delta`. (If `delta` is negative, i.e. the new text is shorter than the old text, then this will decrease the value of `selection start`.)

Otherwise: if `selection start` is greater than `start`, then set it to `start`. (This snaps the start of the selection to the start of the new text if it was in the middle of the text that it replaced.)

4. If `selection end` is greater than `end`, then increment it by `delta` in the same way.

Otherwise: if `selection end` is greater than `start`, then set it to `new end`. (This snaps the end of the selection to the end of the new text if it was in the middle of the text that it replaced.)

11. Set the selection of the text field to the sequence of characters starting with the character at the `selection start`th position (in logical order) and ending with the character at the `(selection end-1)`th position. In UAs where there is no concept of an empty selection, this must set the cursor to be just before the character with offset `end`. The direction of the selection must be set to `forward` if the platform does not support selections with the direction `none`, and `none` otherwise.

12. Queue a task to fire a simple event that bubbles named `select` at the element, using the user interaction task source as the task source.

All elements to which this API applies have either a selection or a text entry cursor position at all times (even for elements that are not

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

To obtain the currently selected text, the following JavaScript suffices:

```
var selectionText = control.value.substring(control.selectionStart, control.selectionEnd);
...where control is the input or textarea element.
```

To add some text at the start of a text control, while maintaining the text selection, the three attributes must be preserved:

```
var oldStart = control.selectionStart;
var oldEnd = control.selectionEnd;
var
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

## 4.10.21 c

### 4.10.21.1 Definitions

A submittable element is a **candidate for constraint validation** except when a condition has **barred the element from constraint validation**. (For example, an element is barred from constraint validation if it is an `object` element.)

An element can have a **custom validity error message** defined. Initially, an element must have its custom validity error message set to the empty string. When its value is not the empty string, the element is suffering from a custom error. It can be set using the `setCustomValidity()` method. The user agent should use the custom validity error message when alerting the user to the problem with the control.

An element can be constrained in various ways. The following is the list of **validity states** that a form control can be in, making the control invalid for the purposes of constraint validation. (The definitions below are non-normative; other parts of this specification define more precisely when each state applies or does not.)

#### Suffering from being missing

When a control has no value but has a `required` attribute (`input required`, `select required`, `textarea required`), or, in the case of an element in a `radio button group`, any of the other elements in the group has a `required` attribute.

#### Suffering from a type mismatch

When a control that allows arbitrary user input has a value that is not in the correct syntax (E-mail, URL).

#### Suffering from a pattern mismatch

When a control has a value that doesn't satisfy the `pattern` attribute.

#### Suffering from being too long

When a control has a value that is too long for the form control `maxlength` attribute (`input maxlength`, `textarea maxlength`).

#### Suffering from an underflow

When a control has a value that is too low for the `min` attribute.

#### Suffering from an overflow

When a control has a value that is too high for the `max` attribute.

#### Suffering from a step mismatch

When a control has a value that doesn't fit the rules given by the `step` attribute.

#### Suffering from bad input

When a control has incomplete input and the user agent does not think the user ought to be able to submit the form in its current state.

#### Suffering from a custom error

When a control's custom validity error message (as set by the element's `setCustomValidity()` method) is not the empty string.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

#### 4.10.21.2 Constraint validation

When the user agent is required to **statically validate the constraints** of `form` element `form`, it must run the following steps, which return either a *positive* result (all the controls in the form are valid) or a *negative* result (there are invalid controls) along with a (possibly empty) list of elements that are invalid and for which no script has claimed responsibility:

1. Let `controls` be a list of all the submittable elements whose form owner is `form`, in tree order.
2. Let `ir`
3. For **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**
4. If `invalidControls` is `null`, let `invalidControls` be an initially empty list of elements.
5. Let `UnhandledInvalidControls` be an initially empty list of elements.
6. For each element `field` in `invalidControls`, if any, in tree order, run the following substeps:
  1. Fire a simple event named `invalid` that is cancelable at `field`.
  2. If the event was not canceled, then add `field` to `UnhandledInvalidControls`.
7. Return a *negative* result with the list of elements in the `UnhandledInvalidControls` list.

If a user agent is to **interactively validate the constraints** of `form` element `form`, then the user agent must run the following steps:

1. Statically validate the constraints of `form`, and let `UnhandledInvalidControls` be the list of elements returned if the result was *negative*.
2. If the result was *positive*, then return that result and abort these steps.
3. Report the problems with the constraints of at least one of the elements given in `UnhandledInvalidControls` to the user. User agents may focus one of those elements in the process, by running the focusing steps for that element, and may change the scrolling position of the document, or perform some other action that brings the element to the user's attention. User agents may report more than one constraint violation. User agents may coalesce related constraint violation reports if appropriate (e.g. if multiple radio buttons in a group are marked as required, only one error need be reported). If one of the controls is not being rendered (e.g. it has the `hidden` attribute set) then user agents may report a script error.
4. Return a *negative* result.

#### 4.10.21.3 The constraint validation API

This box is non-normative. Implementation requirements are given below this box.

##### `element.willValidate`

Returns true if the element will be validated when the form is submitted; false otherwise.

##### `element.setCustomValidity(message)`

Sets a custom error, so that the element would fail to validate. The given message is the message to be shown to the user when reporting the problem to the user.

If the argument is the empty string, clears the custom error.

##### `element.validity.valueMissing`

Returns true if the element has no value but is a required field; false otherwise.

##### `element.validity.typeMismatch`

Returns true if the element's value is not in the correct syntax; false otherwise.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Returns true if the element's value is longer than the provided maximum length; false otherwise.

**`element.validity.rangeUnderflow`**

Returns true if the element's value is lower than the provided minimum; false otherwise.

**`element.validity.rangeOverflow`**

Returns true if the element's value is higher than the provided maximum; false otherwise.

**`element.validity.stepMismatch`**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

3.

Convert to a value; false

**`element.validity.valid`**

Returns true if the element's value has no validity problems; false otherwise.

**`valid = element.checkValidity()`**Returns true if the element's value has no validity problems; false otherwise. Fires an `invalid` event at the element in the latter case.**`element.validationMessage`**

Returns the error message that would be shown to the user if the element was to be checked for validity.

The `willValidate` attribute must return true if an element is a candidate for constraint validation, and false otherwise (i.e. false if any conditions are barring it from constraint validation).

The `setCustomValidity(message)`, when invoked, must set the custom validity error message to the value of the given `message` argument.

In the following example, a script checks the value of a form control each time it is edited, and whenever it is not a valid value, uses the `setCustomValidity()` method to set an appropriate message.

```
<label>Feeling: <input name=f type="text" oninput="check(this)"></label>
<script>
function check(input) {
 if (input.value == "good" ||
 input.value == "fine" ||
 input.value == "tired") {
 input.setCustomValidity("'" + input.value + "' is not a feeling.");
 } else {
 // input is fine -- reset the error message
 input.setCustomValidity('');
 }
}</script>
```

The `validity` attribute must return a `ValidityState` object that represents the validity states of the element. This object is live, and the same object must be returned each time the element's `validity` attribute is retrieved.

```
interface ValidityState {
 readonly attribute boolean valueMissing;
 readonly attribute boolean typeMismatch;
 readonly attribute boolean patternMismatch;
 readonly attribute boolean tooLong;
 readonly attribute boolean rangeUnderflow;
```

3,445 captures

Go NOV JAN FEB  
◀ 03 ▶  
2012 2013 2014

5 Jun 2004 - 22 Aug 2020

};

A `ValidityState` object has the following attributes. On getting, they must return true if the corresponding condition given in the following list is true, and false otherwise.

#### `valueMissing`

The control is suffering from being missing.



#### `typeMismatch`

The `patternMismatch` condition is true if the user input does not match the regular expression pattern defined by the `pattern` attribute. The `tooLong` condition is true if the user input is longer than the maximum allowed length defined by the `maxLength` attribute.

The `tooLong` condition is true if the user input is longer than the maximum allowed length defined by the `maxLength` attribute.

#### `rangeUnderflow`

The control is suffering from an underflow.

#### `rangeOverflow`

The control is suffering from an overflow.

#### `stepMismatch`

The control is suffering from a step mismatch.

#### `badInput`

The control is suffering from bad input.

#### `customError`

The control is suffering from a custom error.

#### `valid`

None of the other conditions are true.

When the `checkValidity()` method is invoked, if the element is a candidate for constraint validation and does not satisfy its constraints, the user agent must fire a simple event named `invalid` that is cancelable (but in this case has no default action) at the element and return false. Otherwise, it must only return true without doing anything else.

The `validationMessage` attribute must return the empty string if the element is not a candidate for constraint validation or if it is one but it satisfies its constraints; otherwise, it must return a suitably localized message that the user agent would show the user if this were the only form control with a validity constraint problem. If the user agent would not actually show a textual message in such a situation (e.g. it would show a graphical cue instead), then the attribute must return a suitably localized message that expresses (one or more of) the validity constraint(s) that the control does not satisfy. If the element is a candidate for constraint validation and is suffering from a custom error, then the custom validity error message should be present in the return value.

### 4.10.21.4 Security

Servers should not rely on client-side validation. Client-side validation can be intentionally bypassed by hostile users, and unintentionally bypassed by users of older user agents or automated tools that do not implement these features. The constraint validation features are only intended to improve the user experience, not to provide any kind of security mechanism.

### 4.10.22 Form submission

#### 4.10.22.1 Introduction

*This section is non-normative.*

When a form is submitted, the data in the form is converted into the structure specified by the `enctype`, and then sent to the destination specified by the `action` using the given method.

3,445 captures

5 Jun 2004 - 22 Aug 2020

---

Go

NOV JAN FEB

1

03

2012 2013 2014



▼ About this capture

```
<input type=submit>
</form>
```

If the user types in "cats" in the first field and "fur" in the second, and then hits the submit button, then the user agent will load </find.cgi?t=cats&q=fur>.

On the other hand, consider this form:

```
<form action="/find.cgi" method=post enctype="multipart/form-data">
 <input type=text name=*>
```

Given the size of the file, it's probably best to download it and read it locally. If you're reading this on a mobile device, consider switching to a desktop browser or using a desktop computer to read the spec.

⇒ POST to the given URL,

-----  
Content //whatwg.org/html

```
cats
-----kYFrd4jNJEgCervE
Content-Disposition: form-data; name="q"
```

fur  
-----kYFrd4jNJEqCervE--

#### **4.10.22.2 Implicit submission**

A `form` element's **default button** is the first submit button in tree order whose `form` owner is that `form` element.

If the user agent supports letting the user submit a form implicitly (for example, on some platforms hitting the "enter" key while a text field is focused implicitly submits the form), then doing so for a form whose default button has a defined activation behavior must cause the user agent to run synthetic click activation steps on that default button.

**Note:** Consequently, if the default button is disabled, the form is not submitted when such an implicit submission mechanism is used. (A button has no activation behavior when disabled.)

**Note:** There are pages on the Web that are only usable if there is a way to implicitly submit forms, so user agents are strongly encouraged to support this.

If the form has no submit button, then the implicit submission mechanism must do nothing if the form has more than one *field that blocks implicit submission*, and must submit the `form` element from the `form` element itself otherwise.

For the purpose of the previous paragraph, an element is a *field that blocks implicit submission* of a `form` element if it is an `input` element whose form owner is that `form` element and whose `type` attribute is in one of the following states: Text, Search, URL, Telephone, E-mail, Password, Date and Time, Date, Month, Week, Time, Local Date and Time, Number.

#### 4.10.22.3 Form submission algorithm

When a `form` element `form` is **submitted** from an element `submitter` (typically a button), optionally with a `submitted from submit()` `method` flag set, the user agent must run the following steps:

1. Let *form document* be the *form's Document*.
  2. If *form document* has no associated browsing context or its active sandboxing flag set has its sandboxed forms browsing context flag set, then abort these steps without doing anything.
  3. Let *form browsing context* be the browsing context of *form document*.
  4. If the *submitted from submit () method* flag is not set, and the *submitter element's no-validate state* is false, then interactively validate the constraints of *form* and examine the result: if the result is negative (the constraint validation concluded that there were invalid fields and probably informed the user of this) then abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

7. Let *action* be the *submitter* element's action.
8. If *action* is the empty string, let *action* be the document's address of the *form document*.

**Note:** This step is a willful violation of RFC 3986, which would require base URL processing here. This violation is motivated by a desire for compatibility with legacy content. [RFC3986]

9. Resolve the URL *action*, relative to the *submitter* element. If this fails, abort these steps. Otherwise, let *action* be the resulting absolute URL.

10. Let **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

11. Let **/web/20130103030411/http://whatwg.org/html**

12. Let **//whatwg.org/html**

13. Let **//whatwg.org/html**

14. If the *browsing context* be that browsing context. Otherwise, apply the rules for choosing a browsing context given a browsing context name using *target* as the name and *form browsing context* as the context in which the algorithm is executed, and let *target browsing context* be the resulting browsing context.

15. If *target browsing context* was created in the previous step, or, alternatively, if the *form document* has not yet completely loaded and the *submitted from submit() method* is set, then let *replace* be true. Otherwise, let it be false.

16. If the value of *method* is dialog then jump to the submit dialog steps.

Otherwise, select the appropriate row in the table below based on the value of *scheme* as given by the first cell of each row. Then, select the appropriate cell on that row based on the value of *method* as given in the first cell of each column. Then, jump to the steps named in that cell and defined below the table.

	GET	POST
<b>http</b>	Mutate action URL	Submit as entity body
<b>https</b>	Mutate action URL	Submit as entity body
<b>ftp</b>	Get action URL	Get action URL
<b>javascript</b>	Get action URL	Get action URL
<b>data</b>	Get action URL	Post to data:
<b>mailto</b>	Mail with headers	Mail as body

If *scheme* is not one of those listed in this table, then the behavior is not defined by this specification. User agents should, in the absence of another specification defining this, act in a manner analogous to that defined in this specification for similar schemes.

The behaviors are as follows:

#### **Mutate action URL**

Let *query* be the result of encoding the *form data set* using the `application/x-www-form-urlencoded` encoding algorithm, interpreted as a US-ASCII string.

Let *destination* be a new URL that is equal to the *action* except that its `<query>` component is replaced by *query* (adding a U+003F QUESTION MARK character (?)) if appropriate.

Navigate *target browsing context* to *destination*. If *replace* is true, then *target browsing context* must be navigated with replacement enabled.

#### **Submit as entity body**

Let *entity body* be the result of encoding the *form data set* using the appropriate form encoding algorithm.

Let *MIME type* be determined as follows:

##### **If enctype is application/x-www-form-urlencoded**

Let *MIME type* be "application/x-www-form-urlencoded".

##### **If enctype is multipart/form-data**

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Let *MIME type* be "text/plain".

Otherwise, navigate *target browsing context* to *action* using the HTTP method given by *method* and with *entity body* as the entity body, of type *MIME type*. If *replace* is true, then *target browsing context* must be navigated with replacement enabled.

#### **Get action URL**

Navigate *target browsing context* to *action*. If *replace* is true, then *target browsing context* must be navigated with replacement enabled.

**Pos**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

algorithm.

cape all bytes in *data* that, if syntax, and then, treating the in the resulting string and [RFC3986]

a row, but not four), then

%-escape all characters in *data* that, if interpreted as US-ASCII, do not match the unreserved production in the URI Generic Syntax, and then, treating the result as a US-ASCII string, replace the first occurrence of "%%" in *action* with the resulting escaped string. [RFC3986]

Navigate *target browsing context* to the potentially modified *action* (which will be a `data: URL`). If *replace* is true, then *target browsing context* must be navigated with replacement enabled.

#### **Mail with headers**

Let *headers* be the resulting encoding the *form data set* using the `application/x-www-form-urlencoded` encoding algorithm, interpreted as a US-ASCII string.

Replace occurrences of U+002B PLUS SIGN characters (+) in *headers* with the string "%20".

Let *destination* consist of all the characters from the first character in *action* to the character immediately before the first U+003F QUESTION MARK character (?), if any, or the end of the string if there are none.

Append a single U+003F QUESTION MARK character (?) to *destination*.

Append *headers* to *destination*.

Navigate *target browsing context* to *destination*. If *replace* is true, then *target browsing context* must be navigated with replacement enabled.

#### **Mail as body**

Let *body* be the resulting encoding the *form data set* using the appropriate form encoding algorithm and then %-escaping all the bytes in the resulting byte string that, when interpreted as US-ASCII, do not match the unreserved production in the URI Generic Syntax. [RFC3986]

Let *destination* have the same value as *action*.

If *destination* does not contain a U+003F QUESTION MARK character (?), append a single U+003F QUESTION MARK character (?) to *destination*. Otherwise, append a single U+0026 AMPERSAND character (&).

Append the string "body=" to *destination*.

Append *body*, interpreted as a US-ASCII string, to *destination*.

Navigate *target browsing context* to *destination*. If *replace* is true, then *target browsing context* must be navigated with replacement enabled.

#### **Submit dialog**

Let *dialog* be the nearest ancestor `dialog` element of *form*, if any.

If there isn't one, do nothing. Otherwise, proceed as follows:

If *submitter* is an `input` element whose `type` attribute is in the Image Button state, then let *result* be the string formed by concatenating the selected coordinate's x-component, expressed as a base-ten number using ASCII digits, a U+002C COMMA character (,), and the selected coordinate's y-component, expressed in the same way as the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Then, close the dialog *dialog*. If there is a *result*, let that be the return value.

The **appropriate form encoding algorithm** is determined as follows:

**If *enctype* is application/x-www-form-urlencoded**

    Use the application/x-www-form-urlencoded encoding algorithm.

**If *enctype* is multipart/form-data**

    Use the multipart/form-data encoding algorithm.



**If *enctype* is other** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 4.10.22.4 /web/20130103030411/http://whatwg.org/html

The algorithm specified on the page is as follows. If not

1. Let *controls* be a list of all the submittable elements whose form owner is *form*, in tree order.

2. Let the *form data set* be a list of name-value-type tuples, initially empty.

3. *Loop*: For each element *field* in *controls*, in tree order, run the following substeps:

1. If any of the following conditions are met, then skip these substeps for this element:
  - The *field* element has a `datalist` element ancestor.
  - The *field* element is disabled.
  - The *field* element is a button but it is not *submitter*.
  - The *field* element is an `input` element whose `type` attribute is in the Checkbox state and whose checkedness is false.
  - The *field* element is an `input` element whose `type` attribute is in the Radio Button state and whose checkedness is false.
  - The *field* element is not an `input` element whose `type` attribute is in the Image Button state, and either the *field* element does not have a `name` attribute specified, or its `name` attribute's value is the empty string.
  - The *field* element is an `object` element that is not using a plugin.

Otherwise, process *field* as follows:

2. Let *type* be the value of the `type` IDL attribute of *field*.

3. If the *field* element is an `input` element whose `type` attribute is in the Image Button state, then run these further nested substeps:
  1. If the *field* element has a `name` attribute specified and its value is not the empty string, let *name* be that value followed by a single U+002E FULL STOP character (.). Otherwise, let *name* be the empty string.
  2. Let *name<sub>x</sub>* be the string consisting of the concatenation of *name* and a single U+0078 LATIN SMALL LETTER X character (x).
  3. Let *name<sub>y</sub>* be the string consisting of the concatenation of *name* and a single U+0079 LATIN SMALL LETTER Y character (y).

4. The *field* element is *submitter*, and before this algorithm was invoked the user indicated a coordinate. Let *x* be the *x*-component of the coordinate selected by the user, and let *y* be the *y*-component of the coordinate selected by the user.

5. Append an entry to the *form data set* with the name *name<sub>x</sub>*, the value *x*, and the type *type*.

6. Append an entry to the *form data set* with the name *name<sub>y</sub>* and the value *y*, and the type *type*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

5. If the *field* element is a `select` element, then for each `option` element in the `select` element whose selectedness is true, append an entry to the *form data set* with the *name* as the name, the value of the `option` element as the value, and *type* as the type.

6. Otherwise, if the *field* element is an `input` element whose *type* attribute is in the Checkbox state or the Radio Button state, then run these further nested substeps:

1. If the *field* element has a *value* attribute specified, then let *value* be the value of that attribute; otherwise, let *value* be the string "on".

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

and *type* as the type.

load state, then for each file  
› name, the file (consisting of  
ted files, then append an  
application/octet-

ue from the plugin, and if that  
is successful, append an entry to the *form data set* with *name* as the name, the returned form submission value as the value, and the string "object" as the type.

9. Otherwise, append an entry to the *form data set* with *name* as the name, the value of the *field* element as the value, and *type* as the type.

10. If the element has a *dirname* attribute, and that attribute's value is not the empty string, then run these substeps:

1. Let *dirname* be the value of the element's *dirname* attribute.
2. Let *dir* be the string "ltr" if the directionality of the element is 'ltr', and "rtl" otherwise (i.e. when the directionality of the element is 'rtl').
3. Append an entry to the *form data set* with *dirname* as the name, *dir* as the value, and the string "direction" as the type.

**Note: An element can only have a *dirname* attribute if it is a `textarea` element or an `input` element whose *type* attribute is in either the Text state or the Search state.**

4. *End*: For the name of each entry in the *form data set*, and for the value of each entry in the *form data set* whose type is not "file" or "textarea", replace every occurrence of a U+000D CARRIAGE RETURN (CR) character not followed by a U+000A LINE FEED (LF) character, and every occurrence of a U+000A LINE FEED (LF) character not preceded by a U+000D CARRIAGE RETURN (CR) character, by a two-character string consisting of a U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pair.

**Note: In the case of the value of `textarea` elements, this newline normalization is already performed during the conversion of the control's raw value into the control's value (which also performs any necessary line wrapping). In the case of `input` elements *type* attributes in the File Upload state, the value is not normalized.**

5. Return the *form data set*.

#### 4.10.22.5 URL-encoded form data

**Note: This form data set encoding is in many ways an aberrant monstrosity, the result of many years of implementation accidents and compromises leading to a set of requirements necessary for interoperability, but in no way representing good design practices. In particular, readers are cautioned to pay close attention to the twisted details involving repeated (and in some cases nested) conversions between character encodings and byte sequences.**

The `application/x-www-form-urlencoded` encoding algorithm is as follows:

1. Let *result* be the empty string.
2. If the `form` element has an `accept-charset` attribute, then, taking into account the characters found in the *form data set*'s

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

character encoding.

Otherwise, let the selected character encoding be UTF-8.

3. Let *charset* be the preferred MIME name of the selected character encoding.

4. For each entry in the *form data set*, perform these substeps:

1. If the entry's name is "`_charset_`" and its type is "hidden", replace its value with *charset*.



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

selected character encoding,  
+0023 NUMBER SIGN  
ter in base ten, and finally a

ime and value are now byte

5. For each byte in the entry's name and value, apply the appropriate subsubsteps from the following list:

↳ **The byte is 0x20 (U+0020 SPACE if interpreted as ASCII)**

Replace the byte with a single 0x2B byte (U+002B PLUS SIGN character (+) if interpreted as ASCII).

↳ **If the byte is in the range 0x2A, 0x2D, 0x2E, 0x30 to 0x39, 0x41 to 0x5A, 0x5F, 0x61 to 0x7A**

Leave the byte as is.

↳ **Otherwise**

1. Let *s* be a string consisting of a U+0025 PERCENT SIGN character (%) followed by uppercase ASCII hex digits representing the hexadecimal value of the byte in question (zero-padded if necessary).

2. Encode the string *s* as US-ASCII, so that it is now a byte string.

3. Replace the byte in question in the name or value being processed by the bytes in *s*, preserving their relative order.

6. Interpret the entry's name and value as Unicode strings encoded in US-ASCII. (All of the bytes in the string will be in the range 0x00 to 0x7F; the high bit will be zero throughout.) The entry's name and value are now Unicode strings again.

7. If the entry's name is "isindex", its type is "text", and this is the first entry in the *form data set*, then append the value to *result* and skip the rest of the substeps for this entry, moving on to the next entry, if any, or the next step in the overall algorithm otherwise.

8. If this is not the first entry, append a single U+0026 AMPERSAND character (&) to *result*.

9. Append the entry's name to *result*.

10. Append a single U+003D EQUALS SIGN character (=) to *result*.

11. Append the entry's value to *result*.

5. Encode *result* as US-ASCII and return the resulting byte stream.

To **decode application/x-www-form-urlencoded payloads**, the following algorithm should be used. This algorithm uses as inputs the payload itself, *payload*, consisting of a Unicode string using only characters in the range U+0000 to U+007F; a default character encoding *encoding*; and optionally an *isindex* flag indicating that the payload is to be processed as if it had been generated for a form containing an *isindex* control. The output of this algorithm is a sorted list of name-value pairs. If the *isindex* flag is set and the first control really was an *isindex* control, then the first name-value pair will have as its name the empty string.

1. Let *strings* be the result of strictly splitting the string *payload* on U+0026 AMPERSAND characters (&).
2. If the *isindex* flag is set and the first string in *strings* does not contain a U+003D EQUALS SIGN character (=), insert a U+003D EQUALS SIGN character (=) at the start of the first string in *strings*.
3. Let *pairs* be an empty list of name-value pairs.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

EQUALS SIGN character (=) is the first character, then *name* will be the empty string. If it is the last character, then *value* will be the empty string.

Otherwise, *string* contains no U+003D EQUALS SIGN characters (=). Let *name* have the value of *string* and let *value* be the empty string.

2. Replace any U+002B PLUS SIGN characters (+) in *name* and *value* with U+0020 SPACE characters.
3. Replace any escape in *name* and *value* with the character represented by the escape. This replacement must not be recursive.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

ual to the value of the two  
il number (in the range

↳ 25AA%21" becomes the

4. Convert the *name* and *value* strings to their byte representation in ISO-8859-1 (i.e. convert the Unicode string to a byte string, mapping code points to byte values directly).
5. Add a pair consisting of *name* and *value* to *pairs*.
5. If any of the name-value pairs in *pairs* have a name component consisting of the string "`_charset_`" encoded in US-ASCII, and the value component of the first such pair, when decoded as US-ASCII, is the name of a supported character encoding, then let *encoding* be that character encoding (replacing the default passed to the algorithm).
6. Convert the name and value components of each name-value pair in *pairs* to Unicode by interpreting the bytes according to the encoding *encoding*.
7. Return *pairs*.

**Note:** Parameters on the `application/x-www-form-urlencoded` MIME type are ignored. In particular, this MIME type does not support the `charset` parameter.

#### 4.10.22.6 Multipart form data

The `multipart/form-data` encoding algorithm is as follows:

1. Let *result* be the empty string.
2. If the algorithm was invoked with an explicit character encoding, let the selected character encoding be that encoding. (This algorithm is used by other specifications, which provide an explicit character encoding to avoid the dependency on the `form` element described in the next paragraph.)

Otherwise, if the `form` element has an `accept-charset` attribute, then, taking into account the characters found in the `form` data set's names and values, and the character encodings supported by the user agent, select a character encoding from the list given in the `form`'s `accept-charset` attribute that is an ASCII-compatible character encoding. If none of the encodings are supported, or if none are listed, then let the selected character encoding be UTF-8.

Otherwise, if the document's character encoding is an ASCII-compatible character encoding, then that is the selected character encoding.

Otherwise, let the selected character encoding be UTF-8.

3. Let *charset* be the preferred MIME name of the selected character encoding.
4. For each entry in the `form` data set, perform these substeps:

1. If the entry's name is "`_charset_`" and its type is "hidden", replace its value with *charset*.
2. For each character in the entry's name and value that cannot be expressed using the selected character encoding, replace the character by a string consisting of a U+0026 AMPERSAND character (&), a U+0023 NUMBER SIGN character (#), one or more ASCII digits representing the Unicode code point of the character in base ten, and finally a

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Each entry in the *form data set* is a *field*, the name of the entry is the *field name* and the value of the entry is the *field value*.The order of parts must be the same as the order of fields in the *form data set*. Multiple entries with the same name must be treated as distinct fields.

**Note:** In particular, this means that multiple files submitted as part of a single `<input type="file" multiple>` element will result in each file having its own field; the "sets of files" feature ("multipart/mixed") of RFC 2388 is not used.

The

Typ  
part

File

enc  
file |

repl

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

must not have a Content-type selected above (field names in

st use the character  
es could be removed from  
acter encoding could be  
FC 2388.

The boundary is the `boundary` attribute of the `multipart/form-data` boundary string. (This value is used to generate the MIME type of the form submission payload generated by this algorithm.)

For details on how to interpret `multipart/form-data` payloads, see RFC 2388. [RFC2388]

#### 4.10.22.7 Plain text form data

The `text/plain` encoding algorithm is as follows:

1. Let *result* be the empty string.
2. If the `form` element has an `accept-charset` attribute, then, taking into account the characters found in the *form data set*'s names and values, and the character encodings supported by the user agent, select a character encoding from the list given in the `form`'s `accept-charset` attribute. If none of the encodings are supported, or if none are listed, then let the selected character encoding be UTF-8.

Otherwise, the selected character encoding is the document's character encoding.

3. Let *charset* be the preferred MIME name of the selected character encoding.
4. If the entry's name is `_charset_` and its type is `"hidden"`, replace its value with *charset*.
5. If the entry's type is `"file"`, replace its value with the file's filename only.
6. For each entry in the *form data set*, perform these substeps:
  1. Append the entry's name to *result*.
  2. Append a single U+003D EQUALS SIGN character (=) to *result*.
  3. Append the entry's value to *result*.
  4. Append a U+000D CARRIAGE RETURN (CR) U+000A LINE FEED (LF) character pair to *result*.
7. Encode *result* using the selected character encoding and return the resulting byte stream.

Payloads using the `text/plain` format are intended to be human readable. They are not reliably interpretable by computer, as the format is ambiguous (for example, there is no way to distinguish a literal newline in a value from the newline at the end of the value).

#### 4.10.23 Resetting a form

When a `form` element `form` is `reset`, the user agent must fire a simple event named `reset`, that bubbles and is cancelable, at `form`, and then, if that event is not canceled, must invoke the `reset` algorithm of each resettable element whose `form` owner is `form`.Each resettable element defines its own `reset` algorithm. Changes made to form controls as part of these algorithms do not count as changes caused by the user (and thus, e.g., do not cause `input` events to fire).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Flow content.  
Sectioning root.  
Interactive content.  
Palpable content.

**Contexts in which this element can be used:**

Where flow content is expected.

**Content model:**

Or

**Content:** Too slow? Try reading the multipage copy of  
 ↗ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**DOM intc:** /web/20130103030411/http:  
 ( //whatwg.org/html

{ };



The `details` element represents a disclosure widget from which the user can obtain additional information or controls.

**Note:** The `details` element is not appropriate for footnotes. Please see the section on footnotes for details on how to mark up footnotes.

The first `summary` element child of the element, if any, represents the summary or legend of the details. If there is no child `summary` element, the user agent should provide its own legend (e.g. "Details").

The rest of the element's contents represents the additional information or controls.

The `open` content attribute is a boolean attribute. If present, it indicates that both the summary and the additional information is to be shown to the user. If the attribute is absent, only the summary is to be shown.

When the element is created, if the attribute is absent, the additional information should be hidden; if the attribute is present, that information should be shown. Subsequently, if the attribute is removed, then the information should be hidden; if the attribute is added, the information should be shown.

The user agent should allow the user to request that the additional information be shown or hidden. To honor a request for the details to be shown, the user agent must set the `open` attribute on the element to the value `open`. To honor a request for the information to be hidden, the user agent must remove the `open` attribute from the element.

The `open` IDL attribute must reflect the `open` content attribute.

The following example shows the `details` element being used to hide technical details in a progress report.

```
<section class="progress window">
 <h1>Copying "Really Achieving Your Childhood Dreams"</h1>
 <details>
 <summary>Copying... <progress max="375505392" value="97543282"></progress> 25%</summary>
 <dl>
 <dt>Transfer rate:</dt> <dd>452KB/s</dd>
 <dt>Local filename:</dt> <dd>/home/rpausch/raycd.m4v</dd>
 <dt>Remote filename:</dt> <dd>/var/www/lectures/raycd.m4v</dd>
 <dt>Duration:</dt> <dd>01:16:27</dd>
 <dt>Color profile:</dt> <dd>SD (6-1-6)</dd>
 <dt>Dimensions:</dt> <dd>320×240</dd>
 </dl>
 </details>
</section>
```

The following shows how a `details` element can be used to hide some controls by default:

```
<details>
```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

One could use this in conjunction with other details in a list to allow the user to collapse a set of fields down to a small set of headings, with the ability to open each one.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The screenshot shows a Mac OS X-style PDF viewer window titled "Pillar Magazine.pdf Info". The file is a PDF named "Pillar Magazine.pdf" with a size of 804 KB, modified on Sep 6, 2009 at 23:02. Below the viewer is a preview of the magazine cover. The cover features a close-up of a cat's face with the title "Pillar magazine" in a stylized font. Below the title, it says "Summer 2009". The preview lists several articles: "HOW TO ASK FOR FOOD – AND GET IT!", "THE SECRET LIFE OF THE MICE", "WHERE TO FIND INSECTS, STICKS, AND LEAVES", "28 WAYS TO GET IN A BOX", "THE SECRET TO SLEEPING IN: 24 HOURS A DAY", and a barcode. At the bottom of the preview, there is a link "► Sharing & Permissions:".

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Modified: Sep 6, 2009 23:02

PDF **NOV JAN FEB**  
 ◀ **03** ▶  
**2012 2013 2014**

About this capture

**Spotlight Comments:**

- ▶ General:
- ▶ More Info:
- ▼ Name & Extension:  
**Dollar Magazine.pdf**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



▶ Sharing & Permissions:

In these examples, the summary really just summarises what the controls can change, and not the actual values, which is less than ideal.

Because the `open` attribute is added and removed automatically as the user interacts with the control, it can be used in CSS to style the element differently based on its state. Here, a stylesheet is used to animate the color of the summary when the element is opened or closed:

```
<style>
 details > summary { transition: color 1s; color: black; }
 details[open] > summary { color: red; }
</style>
<details>
 <summary>Automated Status: Operational</summary>
 <p>Velocity: 12m/s</p>
 <p>Direction: North</p>
</details>
```

## 4.11.2 The `summary` element

Categories:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

**Content model:**

Phrasing content.

**Content attributes:**

Global attributes

**DOM interface:**Uses `HTMLElement`.

The `summary` element represents a summary, caption, or legend for the rest of the contents of its element's parent `details` element.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

**Category:** /web/20130103030411/http:

F //whatwg.org/html

**Contexts in which this element can be used:**

Where flow content is expected.

If the element's `type` attribute is in the popup menu state: as the child of a `menu` element whose `type` attribute is in the popup menu state.**Content model:**If the element's `type` attribute is in the toolbar state: either zero or more `li` elements, or flow content.If the element's `type` attribute is in the popup menu state: in any order, zero or more `menuitem` elements, zero or more `hr` elements, and zero or more `menu` elements whose `type` attributes are in the popup menu state.**Content attributes:**

Global attributes

`type``label`**DOM interface:**

```
IDL interface HTMLMenuElement : HTMLElement {
 attribute DOMString type;
 attribute DOMString label;
};
```

The `menu` element represents a list of commands.

The `type` attribute is an enumerated attribute indicating the kind of menu being declared. The attribute has two states. The `popup` keyword maps to the `popup menu` state, in which the element is declaring a context menu or the menu for a menu button. The `toolbar` keyword maps to the `toolbar` state, in which the element is declaring a toolbar. The attribute may also be omitted. The *missing value default* is the popup menu state if the parent element is a `menu` element whose `type` attribute is in the popup menu state; otherwise, it is the toolbar state.

If a `menu` element's `type` attribute is in the popup menu state, then the element represents the commands of a popup menu, and the user can only examine and interact with the commands if that popup menu is activated through some other element, either via the `contextmenu` attribute or the `button` element's `menu` attribute.

If a `menu` element's `type` attribute is in the toolbar state, then the element represents a toolbar consisting of its contents, in the form of either an unordered list of items (represented by `li` elements), each of which represents a command that the user can perform or activate, or, if the element has no `li` element children, flow content describing available commands.

The `label` attribute gives the label of the menu. It is used by user agents to display nested menus in the UI: a context menu containing another menu would use the nested menu's `label` attribute for the submenu's menu label. The `label` attribute must only be specified on `menu` elements whose parent element is a `menu` element whose `type` attribute is in the popup menu state.

A `menu` is a **currently relevant menu element** if it is the child of a currently relevant `menu` element, or if it is the designated pop-up

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- Commands, which can be marked as default commands (`menuitem`)
- Separators (`hr`)
- Other menus, which allows the list to be nested (`menu`)

To construct and show a menu for a particular `menu` element, the user agent must run the following steps:

1. Let the menu be an empty list of the type described above.
2. Run the menu builder steps for the `menu` element using the menu prepared in the previous list as the output.

The `in tr` Too slow? Try reading the multipage copy of each child node of the `menu`

- ↳ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

as a default attribute, mark

↳ If the child is a `menu` element with no `label` attribute

Append a separator to the menu, then run the menu builder steps using this child `menu` element for the same menu, then append another separator to the menu.

↳ If the child is a `menu` element with a `label` attribute

Create a new submenu as an empty list of the type described above, and construct it by running the menu builder steps for the child `menu` element using the new submenu as the output. Then, append the submenu to the menu, using the value of the child `menu` element's `label` attribute as the label of the submenu.

↳ Any other node

Ignore the node.

3. Remove any submenu with no label, or whose label is the empty string, in the menu or any submenus.
4. Remove any menu item with no label, or whose label is the empty string, in the menu or any submenus.
5. Collapse all sequences of two or more adjacent separators in the menu or any submenus to a single separator.
6. Remove all separators at the start or end of the menu and any submenus.
7. Display the menu to the user, and let the algorithm that invoked this one continue.

If the user selects a menu item that corresponds to an element that still represents a command when the user selects it, then the UA must invoke that command's Action.

Pop-up menus must not, while being shown, reflect changes in the DOM. The menu is constructed from the DOM before being shown, and is then immutable.

The `type` and `label` IDL attributes must reflect the respective content attributes of the same name.

In this example, the `menu` element is used to describe a toolbar with three menu buttons on it, each of which has a dropdown menu with a series of options:

```
<menu>

 <button type=menu value="File" menu="filemenu">
 <menu id="filemenu" type="popup">
 <menuitem onclick="fnew()" label="New...">
 <menuitem onclick="fopen()" label="Open...">
 <menuitem onclick="fsave()" label="Save">
 <menuitem onclick="fsaveas()" label="Save as...">
 </menu>

 <button type=menu value="Edit" menu="editmenu">
 <menu id="editmenu" type="popup">
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

 <button type="menu" value="Help" menu="helpmenu">
 <menu id="helpmenu" type="popup">
 <MenuItem onclick="location='help.html'" label="Help">
 <MenuItem onclick="location='about.html'" label="About">
 </menu>

</menu>
```

In a s:

 Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

d button):

#### 4.11.4 The `MenuItem` element

##### Categories:

None.

##### Contexts in which this element can be used:

As a child of a `menu` element whose `type` attribute is in the `popup` menu state.

##### Content model:

Empty.

##### Content attributes:

Global attributes

`type`  
`label`  
`icon`  
`disabled`  
`checked`  
`radiogroup`  
`default`  
`command`

Also, the `title` attribute has special semantics on this element.

##### DOM interface:

```
IDL interface HTMLMenuItemElement : HTMLElement {
 attribute DOMString type;
 attribute DOMString label;
 attribute DOMString icon;
 attribute boolean disabled;
 attribute boolean checked;
 attribute DOMString radiogroup;
 attribute boolean default;
 readonly attribute HTMLElement? command;
};
```

The `MenuItem` element represents a command that the user can invoke from a popup menu (either a context menu or the menu of a menu button).

A `MenuItem` element that uses one or more of the `type`, `label`, `icon`, `disabled`, `checked`, and `radiogroup` attributes defines a new command.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

mode. When the element is in the indirect command mode, the element must not have any of the following attributes specified: `type`, `label`, `icon`, `disabled`, `checked`, `radiogroup`.

The `type` attribute indicates the kind of command: either a normal command with an associated action, or a state or option that can be toggled, or a selection of one item from a list of items.

The attribute is an enumerated attribute with three keywords and states. The "command" keyword maps to the Command state, the "checkbox" keyword maps to the Checkbox state, and the "radio" keyword maps to the Radio state. The *missing value default* is the Command state.

**The Command**: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**The Checkbox**: /web/20130103030411/http:

**The Radio**: //whatwg.org/html

The `label` attribute gives the name of the command, as shown to the user. The `label` attribute must be specified if the element is in the explicit command mode. If the attribute is specified, it must have a value that is not the empty string.

The `icon` attribute gives a picture that represents the command. If the attribute is specified, the attribute's value must contain a valid non-empty URL potentially surrounded by spaces. To obtain the absolute URL of the icon when the attribute's value is not the empty string, the attribute's value must be resolved relative to the element. When the attribute is absent, or its value is the empty string, or resolving its value fails, there is no icon.

The `disabled` attribute is a boolean attribute that, if present, indicates that the command is not available in the current state.

**Note:** The distinction between `disabled` and `hidden` is subtle. A command would be disabled if, in the same context, it could be enabled if only certain aspects of the situation were changed. A command would be marked as hidden if, in that situation, the command will never be enabled. For example, in the context menu for a water faucet, the command "open" might be disabled if the faucet is already open, but the command "eat" would be marked hidden since the faucet could never be eaten.

The `checked` attribute is a boolean attribute that, if present, indicates that the command is selected. The attribute must be omitted unless the `type` attribute is in either the Checkbox state or the Radio state.

The `radiogroup` attribute gives the name of the group of commands that will be toggled when the command itself is toggled, for commands whose `type` attribute has the value "radio". The scope of the name is the child list of the parent element. The attribute must be omitted unless the `type` attribute is in the Radio state.

If a `menuitem` element *slave* has a `command` attribute, and *slave* is in a Document, and there is an element in that Document whose ID has a value equal to the value of *slave*'s `command` attribute, and the first such element in tree order, hereafter *master*, itself defines a command and either is not a `menuitem` element or does not itself have a `command` attribute, then the **master command** of *slave* is *master*.

An element with a `command` attribute must have a master command.

The `title` attribute gives a hint describing the command, which might be shown to the user to help him.

The `default` attribute indicates, if present, that the command is the one that would have been invoked if the user had directly activated the menu's subject instead of using the menu. The `default` attribute is a boolean attribute.

In this trivial example, a submit button is given a context menu that has two options, one to reset the form, and one to submit the form. The submit command is marked as being the default.

```
<form action="dosearch.pl">
<p><label>Enter search terms: <input type="text" name="terms"></label></p>
<p><input type=submit contextmenu=formmenu id="submitbutton"></p>
<p hidden><input type=reset id="resetbutton"></p>
<menu type=popup id=formmenu>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The `type` IDL attribute must reflect the content attribute of the same name, limited to only known values.

The `label`, `icon`, `disabled`, `checked`, and `radiogroup`, and `default` IDL attributes must reflect the respective content attributes of the same name.

The `command` IDL attribute must return the master command, if any, or null otherwise.

If the element's Disabled State is false (enabled) then the element's activation behavior depends on the element's `type` and `command` attributes, as:

↳ If the element's `type` is `radio` or `checkbox`:  
Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ If the element's `type` is `radio` or `checkbox`:  
<https://web.archive.org/web/20130103030411/http://whatwg.org/html>  
must add a `checked` attribute.

↳ If the element's `type` attribute is in the `Radio State`:

If the element has a parent, then the UA must walk the list of child nodes of that parent element, and for each node that is a `menuitem` element, if that element has a `radiogroup` attribute whose value exactly matches the current element's (treating missing `radiogroup` attributes as if they were the empty string), and has a `checked` attribute, must remove that attribute.

Then, the element's `checked` attribute attribute must be set to the literal value `checked`.

↳ Otherwise:

The element's activation behavior is to do nothing.

**Note:** Firing a synthetic click event at the element does not cause any of the actions described above to happen.

If the element's Disabled State is true (disabled) then the element has no activation behavior.

**Note:** The `menuitem` element is not rendered except as part of a popup menu.

Here is an example of a pop-up menu button with three options that let the user toggle between left, center, and right alignment. One could imagine such a toolbar as part of a text editor. The menu also has a separator followed by another menu item labeled "Publish", though that menu item is disabled.

```
<button type=menu menu=editmenu>Commands...</button>
<menu type="popup" id="editmenu">
 <menuitem type="radio" radiogroup="alignment" checked="checked"
 label="Left" icon="icons/allL.png" onclick="setAlign('left')">
 <menuitem type="radio" radiogroup="alignment"
 label="Center" icon="icons/alc.png" onclick="setAlign('center')">
 <menuitem type="radio" radiogroup="alignment"
 label="Right" icon="icons/alR.png" onclick="setAlign('right')">
 <hr>
 <menuitem type="command" disabled
 label="Publish" icon="icons/pub.png" onclick="publish()">
</menu>
```

## 4.11.5 Context menus

### 4.11.5.1 Declaring a context menu

The `contextmenu` attribute gives the element's context menu. The value must be the ID of a `menu` element in the same home subtree whose `type` attribute is in the `popup` menu state.

**Note:** When a user right-clicks on an element with a `contextmenu` attribute, the user agent will first fire a `contextmenu` event at the element, and then, if that event is not canceled, a `show` event at the `menu` element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<menuitem label="Pick random name" onclick="document.forms.npc.elements.char.value = getRandomName()">
 <menuitem label="Prefill other fields based on name"
 onclick="prefillFields(document.forms.npc.elements.char.value)">
</menu>
</form>
```

This adds two items to the control's context menu, one called "Pick random name", and one called "Prefill other fields based on name". They invoke scripts that are not shown in the example above.



## 4.11.5.2 P Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If the there is:

such elemen

Otherwise, t

/web/20130103030411/http:

s home subtree, or if the first

is no assigned context menu.

When an ele  
agent must a,

ntext menu key), the user

### ↳ If the user requested a context menu using a pointing device

The user agent must fire a trusted event with the name `contextmenu`, that bubbles and is cancelable, and that uses the `MouseEvent` interface, at the element for which the menu was requested. The context information of the event must be initialized to the same values as the last `MouseEvent` user interaction event that was fired as part of the gesture that that was interpreted as a request for the context menu.

### ↳ Otherwise

The user agent must fire a synthetic mouse event named `contextmenu` that bubbles and is cancelable at the element for which the menu was requested.

**Note:** Typically, therefore, the firing of the `contextmenu` event will be the default action of a `mouseup` or `keyup` event. The exact sequence of events is UA-dependent, as it will vary based on platform conventions.

The default action of the `contextmenu` event depends on whether or not the element or one of its ancestors has a context menu assigned (using the `contextmenu` attribute) when the event dispatch has completed. If there is no context menu assigned, the default action must be for the user agent to show its default context menu, if it has one.

If the element or one of its ancestors does have a context menu assigned, then the user agent must fire a simple event named `show` that is cancelable at the `menu` element of the context menu of the nearest ancestor (including the element itself) with one assigned.

The default action of *this* event is that the user agent must construct and show the menu for the `menu` element.

The user agent may also provide access to its default context menu, if any, with the context menu shown. For example, it could merge the menu items from the two menus together, or provide the page's context menu as a submenu of the default menu.

User agents may provide means for bypassing the context menu processing model, ensuring that the user can always access the UA's default context menus. For example, the user agent could handle right-clicks that have the Shift key depressed in such a way that it does not fire the `contextmenu` event and instead always shows the default context menu.

The `contextMenu` IDL attribute must reflect the `contextmenu` content attribute.

## 4.11.6 Commands

### 4.11.6.1 Facets

A **command** is the abstraction behind menu items, buttons, and links. Once a command is defined, other parts of the interface can refer to the same command, allowing many access points to a single feature to share facets such as the Disabled State.

Commands are defined to have the following **facets**:

#### Type

The kind of command: "command", meaning it is a normal command; "radio", meaning that triggering the command will, amongst other things, set the Checked State to true (and probably uncheck some other commands); or "checkbox", meaning that triggering the command will, amongst other things, toggle the value of the Checked State.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The name of the command as seen by the user.

**Hint**

A helpful or descriptive string that can be shown to the user.

**Icon**

An absolute URL identifying a graphical image that represents the action. A command might not have an Icon.

**Access Key**A key combination selected by the user agent that triggers the command. A command ~~may not~~ have an Access Key.

**Hidden State** Too slow? Try reading the multipage copy of  
 Who the spec instead: <https://web.archive.org>

**Disabled State** Who /web/20130103030411/http:

**Checked State** Who //whatwg.org/html

**Action**

The actual effect that triggering the command will have. This could be a scripted event handler, a URL to which to navigate, or a form submission.

These facets are exposed on elements using the **command API**:

This box is non-normative. Implementation requirements are given below this box.

**element.commandType**

Exposes the Type facet of the command.

**element.id**

Exposes the ID facet of the command.

**element.commandLabel**

Exposes the Label facet of the command.

**element.title**

Exposes the Hint facet of the command.

**element.commandIcon**

Exposes the Icon facet of the command.

**element.accessKeyLabel**

Exposes the Access Key facet of the command.

**element.commandHidden**

Exposes the Hidden State facet of the command.

**element.commandDisabled**

Exposes the Disabled State facet of the command.

**element.commandChecked**

Exposes the Checked State facet of the command.

**element.click()**

Triggers the Action of the command.

The **commandType** attribute must return a string whose value is either "command", "radio", or "checkbox", depending on whether the Type of the command defined by the element is "command", "radio", or "checkbox" respectively. If the element does not define a command, it must return null.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

The `commandHidden` attribute must return true if the command's Hidden State is that the command is hidden, and false if the command is not hidden. If the element does not define a command, the attribute must return null.

The `commandDisabled` attribute must return true if the command's Disabled State is that the command is disabled, and false if the command is not disabled. This attribute is not affected by the command's Hidden State. If the element does not define a command, the attribute must return null.

The `commandChecked` attribute must return true if the command's Checked State is that the command is checked, and false if it is that the command is not checked. If the element does not define a command, the attribute must return null.

**Note: 7 Access** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/docum>

*L attribute, and the**ts are given below this box.*

Return

Save IDs.

The `commands` attribute of the document's `Document` interface must return an `HTMLCollection` rooted at the `Document` node, whose filter matches only elements that define commands and have IDs.

User agents may expose the commands that match the following criteria:

- The Hidden State facet is false (visible)
- The element is in a `Document` that has an associated browsing context.
- Neither the element nor any of its ancestors has a `hidden` attribute specified.
- The element is not a `menuitem` element, or it is a child of a currently relevant `menu` element, or it has an Access Key.

User agents are encouraged to do this especially for commands that have Access Keys, as a way to advertise those keys to the user.

For example, such commands could be listed in the user agent's menu bar.

#### 4.11.6.2 Using the `a` element to define a command

An `a` element with an `href` attribute defines a command.

The Type of the command is "command".

The ID of the command is the value of the `id` attribute of the element, if the attribute is present and not empty. Otherwise the command is an anonymous command.

The Label of the command is the string given by the element's `textContent` IDL attribute.

The Hint of the command is the value of the `title` attribute of the element. If the attribute is not present, the Hint is the empty string.

The Icon of the command is the absolute URL obtained from resolving the value of the `src` attribute of the first `img` element descendant of the element in tree order, relative to that element, if there is such an element and resolving its attribute is successful. Otherwise, there is no icon for the command.

The AccessKey of the command is the element's assigned access key, if any.

The Hidden State of the command is true (hidden) if the element has a `hidden` attribute, and false otherwise.

The Disabled State facet of the command is true if the element or one of its ancestors is inert, and false otherwise.

The Checked State of the command is always false. (The command is never checked.)

The Action of the command, if the element has a defined activation behavior, is to run synthetic click activation steps on the element. Otherwise, it is just to fire a `click` event at the element.

#### 4.11.6.3 Using the `button` element to define a command

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The Disabled State of the command is true if the element or one of its ancestors is inert, or if the element's disabled state is set, and false otherwise.

#### 4.11.6.4 Using the `input` element to define a command

An `input` element whose `type` attribute is in one of the Submit Button, Reset Button, Image Button, Button, Radio Button, or Checkbox states defines a command.

The Type of the command is "radio" if the `type` attribute is in the Radio Button state. "checkbox" if the `type` attribute is in the Checkbox state.

The ID of the command is the value of the `id` attribute of the `input` element. The Label of the command is the value of the `label` attribute of the first `label` element in tree order whose labeled control is the element in question. The Hint of the command is the value of the `title` attribute of the `input` element. If the `title` attribute is present, it is the Label. Otherwise, the command is an anonymous command.

If the Type is "radio" or "checkbox", the Label is the value of the `label` attribute of the first `label` element in tree order whose labeled control is the element in question.

If the Type is "checkbox", the Label is the value of the `label` attribute of the first `label` element in tree order whose labeled control is the element in question.

Otherwise, the Type is "radio" or "checkbox". If the element is a labeled control, the `textContent` of the first `label` element in tree order whose labeled control is the element in question is the Label (in DOM terms, this is the string given by `element.labels[0].textContent`). Otherwise, the value of the `value` attribute, if present, is the Label. Otherwise, the Label is the empty string.

The Hint of the command is the value of the `title` attribute of the `input` element. If the attribute is not present, the Hint is the empty string.

If the element's `type` attribute is in the Image Button state, and the element has a `src` attribute, and that attribute's value can be successfully resolved relative to the element, then the Icon of the command is the absolute URL obtained from resolving that attribute that way. Otherwise, there is no Icon for the command.

The AccessKey of the command is the element's assigned access key, if any.

The Hidden State of the command is true (hidden) if the element has a `hidden` attribute, and false otherwise.

The Disabled State of the command is true if the element or one of its ancestors is inert, or if the element's disabled state is set, and false otherwise.

The Checked State of the command is true if the command is of Type "radio" or "checkbox" and the element is checked attribute, and false otherwise.

The Action of the command, if the element has a defined activation behavior, is to run synthetic click activation steps on the element. Otherwise, it is just to fire a `click` event at the element.

#### 4.11.6.5 Using the `option` element to define a command

An `option` element with an ancestor `select` element and either no `value` attribute or a `value` attribute that is not the empty string defines a command.

The Type of the command is "radio" if the `option`'s nearest ancestor `select` element has no `multiple` attribute, and "checkbox" if it does.

The ID of the command is the value of the `id` attribute of the element, if the attribute is present and not empty. Otherwise the command is an anonymous command.

The Label of the command is the value of the `option` element's `label` attribute, if there is one, or else the value of `option` element's `textContent` IDL attribute, with leading and trailing whitespace stripped, and with any sequences of two or more space characters replaced by a single U+0020 SPACE character.

The Hint of the command is the string given by the element's `title` attribute, if any, and the empty string if the attribute is absent.

There is no Icon for the command.

The AccessKey of the command is the element's assigned access key, if any.

The Hidden State of the command is true (hidden) if the element has a `hidden` attribute, and false otherwise.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The Action of the command depends on its Type. If the command is of Type "radio" then it must pick the `option` element. Otherwise, it must toggle the `option` element.

#### 4.11.6.6 Using the `menuitem` element to define a command

A `menuitem` element that does not have a `command` attribute defines a command.

The Type of the command is "radio" if the `menuitem`'s `type` attribute is "radio", "checkbox" if the attribute's value is "checkbox", and "command" otherwise.

The ID of the command is an anonymous command. Otherwise the command

The Label of the command is the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The Hint of the command is the Hint of the element, if it has one.

The Icon for the command is the icon of the element, if it has one.

The AccessKey of the command is the element's assigned access key, if any.

The Hidden State of the command is true (hidden) if the element has a `hidden` attribute, and false otherwise.

The Disabled State of the command is true if the element or one of its ancestors is inert, or if the element has a `disabled` attribute, and false otherwise.

The Checked State of the command is true (checked) if the element has a `checked` attribute, and false otherwise.

The Action of the command, if the element has a defined activation behavior, is to run synthetic click activation steps on the element. Otherwise, it is just to fire a `click` event at the element.

#### 4.11.6.7 Using the `command` attribute on `menuitem` elements to define a command indirectly

A `menuitem` element with a master command defines a command.

The Type of the command is the Type of the master command.

The ID of the command is the value of the `id` attribute of the element, if the attribute is present and not empty. Otherwise the command is an anonymous command.

The Label of the command is the Label of the master command.

If the element has a `title` attribute, then the Hint of the command is the value of that `title` attribute. Otherwise, the Hint of the command is the Hint of the master command.

The Icon of the command is the Icon of the master command.

The AccessKey of the command is the element's assigned access key, if any.

The Hidden State of the command is the Hidden State of the master command.

The Disabled State of the command is the Disabled State of the master command.

The Checked State of the command is the Checked State of the master command.

The Action of the command is to invoke the Action of the master command.

#### 4.11.6.8 Using the `accesskey` attribute on a `label` element to define a command

A `label` element that has an assigned access key and a labeled control and whose labeled control defines a command, itself defines a command.

The Type of the command is "command".

The ID of the command is the value of the `id` attribute of the element, if the attribute is present and not empty. Otherwise the command

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

There is no icon for the command.

The AccessKey of the command is the element's assigned access key.

The Hidden State, Disabled State, and Action facets of the command are the same as the respective facets of the element's labeled control.

The Checked State of the command is always false. (The command is never checked.)



#### **4.11.6.9 U Too slow? Try reading the multipage copy of**

A legend element is a descendant of the parent of the command that defines a command but is not a descendant of the legend element and is neither a label nor a legend element.

The Type of the command is "command".

The ID of the command is the value of the id attribute of the element.

A legend element is a descendant of the parent of the command that defines a command but is not a descendant of the legend element and is neither a label nor a legend element.

If the command itself defines a command, it is a command.

If the command is empty, it is an anonymous command.

The Label of the command is the string given by the element's textContent IDL attribute.

The Hint of the command is the value of the title attribute of the element.

There is no icon for the command.

The AccessKey of the command is the element's assigned access key.

The Hidden State, Disabled State, and Action facets of the command are the same as the respective facets of the first element in tree order that is a descendant of the parent of the legend element that defines a command but is not a descendant of the legend element and is neither a label nor a legend element.

The Checked State of the command is always false. (The command is never checked.)

#### **4.11.6.10 Using the accesskey attribute to define a command on other elements**

An element that has an assigned access key defines a command.

If one of the earlier sections that define elements that define commands define that this element defines a command, then that section applies to this element, and this section does not. Otherwise, this section applies to that element.

The Type of the command is "command".

The ID of the command is the value of the id attribute of the element, if the attribute is present and not empty. Otherwise the command is an anonymous command.

The Label of the command depends on the element. If the element is a labeled control, the textContent of the first label element in tree order whose labeled control is the element in question is the Label (in DOM terms, this is the string given by element.labels[0].textContent). Otherwise, the Label is the textContent of the element itself.

The Hint of the command is the value of the title attribute of the element. If the attribute is not present, the Hint is the empty string.

There is no icon for the command.

The AccessKey of the command is the element's assigned access key.

The Hidden State of the command is true (hidden) if the element has a hidden attribute, and false otherwise.

The Disabled State of the command is true if the element or one of its ancestors is inert, and false otherwise.

The Checked State of the command is always false. (The command is never checked.)

The Action of the command is to run the following steps:

1. If the element is focusable, run the focusing steps for the element.
2. If the element has a defined activation behavior, run synthetic click activation steps on the element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Categories:**

Flow content.  
Sectioning root.

**Contexts in which this element can be used:**

Where flow content is expected.  
As a child of a `dt` element.  
As a child of a `th` element.

**Content model**

F Too slow? Try reading the multipage copy of

**Content:** C the spec instead: <https://web.archive.org>

o /web/20130103030411/http://whatwg.org/html

**DOM interface**

```
interface boolean open;
attribute DOMString returnValue;
void show(optional (MouseEvent or Element) anchor);
void showModal(optional (MouseEvent or Element) anchor);
void close(optional DOMString returnValue);
};
```

The `dialog` element represents a part of an application that a user interacts with to perform a task, for example a dialog box, inspector, or window.

The `open` attribute is a boolean attribute. When specified, it indicates that the `dialog` element is active and that the user can interact with it.

A `dialog` element without an `open` attribute specified should not be shown to the user. This requirement may be implemented indirectly through the style layer. For example, user agents that support the suggested default rendering implement this requirement using the CSS rules described in the rendering section.

*This box is non-normative. Implementation requirements are given below this box.*

**`dialog . show( [ anchor ] )`**

Displays the `dialog` element.

The argument, if provided, provides an anchor point to which the element will be fixed.

**`dialog . showModal( [ anchor ] )`**

Displays the `dialog` element and makes it the top-most modal dialog.

The argument, if provided, provides an anchor point to which the element will be fixed.

This method honors the `autofocus` attribute.

**`dialog . close( [ result ] )`**

Closes the `dialog` element.

The argument, if provided, provides a return value.

**`dialog . returnValue [ = result ]`**

Returns the `dialog`'s return value.

Can be set, to update the return value.

When the `show()` method is invoked, the user agent must run the following steps:

1. If the element already has an `open` attribute, then abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Each Document has a stack of dialog elements known as the **pending dialog stack**. When a Document is created, this stack must be initialized to be empty.

When an element is added to the pending dialog stack, it must also be added to the top layer layer. When an element is removed from the pending dialog stack, it must be removed from the top layer. [FULLSCREEN]

When the `showModal()` method is invoked, the user agent must run the following steps:

1. Let `dialog` be the `dialog` element on which the method was invoked.
2. If `di` **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html** steps.
3. If `di`
4. Add `/web/20130103030411/http://whatwg.org/html`
5. If th at argument as the anchor.  
Oth
6. Let `dialog`'s Document be blocked by the modal dialog `dialog`.
7. Push `dialog` onto `dialog`'s Document's pending dialog stack.
8. Let `control` be the first element in tree order whose nearest ancestor `dialog` element is `dialog` and that has an `autofocus` attribute specified, if any.
9. If there is no `control`, then abort these steps.
10. Run the focusing steps for `control`.

If at any time a `dialog` element is removed from a Document, then if that `dialog` is in that Document's pending dialog stack, the following steps must be run:

1. Let `dialog` be that `dialog` element and `document` be the Document from which it is being removed.
2. Remove `dialog` from `document`'s pending dialog stack.
3. If `document`'s pending dialog stack is not empty, then let `document` be blocked by the modal dialog that is at the top of `document`'s pending dialog stack. Otherwise, let `document` be no longer blocked by a modal dialog at all.

When the `close()` method is invoked, the user agent must close the dialog that the method was invoked on. If the method was invoked with an argument, that argument must be used as the return value; otherwise, there is no return value.

When a `dialog` element `dialog` is to be **closed**, optionally with a return value `result`, the user agent must run the following steps:

1. If `dialog` does not have an `open` attribute, then throw an `InvalidStateError` exception and abort these steps.
2. Remove `dialog`'s `open` attribute.
3. If the argument was passed a `result`, then set the `returnValue` attribute to the value of `result`.
4. If `dialog` is in its Document's pending dialog stack, then run these substeps:
  1. Remove `dialog` from that pending dialog stack.
  2. If that pending dialog stack is not empty, then let `dialog`'s Document be blocked by the modal dialog that is at the top of the pending dialog stack. Otherwise, let `document` be no longer blocked by a modal dialog at all.
5. Queue a task to fire a simple event named `close` at `dialog`.

The `returnValue` IDL attribute, on getting, must return the last value to which it was set. On setting, it must be set to the new value. When the element is created, it must be set to the empty string.

**Cancelling dialogs:** When a Document's pending dialog stack is not empty, user agents may provide a user interface that, upon activation, queues a task to fire a simple event named `cancel` that is cancelable at the top `dialog` element on the Document's pending dialog stack. The default action of this event must be to close the dialog with no return value.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

that its top static position, for the purposes of calculating the used value of the 'top' property, is the value that would place the element's top margin edge as far from the top of the viewport as the element's bottom margin edge from the bottom of the viewport, if the element's height is less than the height of the viewport, and otherwise is the value that would place the element's top margin edge at the top of the viewport.

This top static position must remain the element's top static position until it is next changed by the above algorithm or the next one. (The element's static position is only used in calculating the used value of the 'top' property in certain situations; it's not used, for instance, to position the element if its 'position' property is set to 'static'.)

If there is an element whose top static position was last set by the set up the default static position algorithm (without an anchor) when its browsing context's document object's `defaultView` property's value's `getComputedStyle` method's return value's `top` property's value is the same as the element's `top` property's value, then let `dialog` be the element.

all such elements. **Too slow? Try reading the multipage copy of**

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. If `ai`

/web/20130103030411/http://whatwg.org/html

2. Let `anchor element` be an anonymous element rendered as a box with zero height and width (so its margin and border boxes both just form a point), positioned so that its top and left are at the coordinate identified by the event, and whose properties all compute to their initial values.

Otherwise, let `anchor element` be `anchor`.

2. Let `dialog` be magically aligned to `anchor element`.

While an element *A* is **magically aligned** to an element *B*, *A* and *B* both have rendered boxes, and *B* is not a descendant of *A*, the following requirements apply:

- *A*'s 'position' property must compute to the keyword '**absolute-anchored**'.
- Let *A* and *B*'s anchor points be defined as per the appropriate entry in the following list:

↳ **If the computed value of 'anchor-point' is 'none' on both *A* and *B***

The anchor points of *A* and *B* are the center points of their respective first boxes' border boxes.

↳ **If the computed value of 'anchor-point' is 'none' on *A* and a specific point on *B***

The anchor point of *B* is the point given by its 'anchor-point' property.

If the anchor point of *B* is the center point of *B*'s first box's border box, then *A*'s anchor point is the center point of its first box's margin box.

Otherwise, *A*'s anchor point is on one of its margin edges. Consider four hypothetical half-infinite lines L1, L2, L3, and L4 that each start in the center of *B*'s first box's border box, and that extend respectively through the top left corner, top right corner, bottom right corner, and bottom left corner of *B*'s first box's border box. *A*'s anchor point is determined by the location of *B*'s anchor point relative to these four hypothetical lines, as follows:

If the anchor point of *B* lies on L1 or L2, or inside the area bounded by L1 and L2 that also contains the points above *B*'s first box's border box, then let *A*'s anchor point be the horizontal center of *A*'s bottom margin edge.

Otherwise, if the anchor point of *B* lies on L3 or L4, or inside the area bounded by L4 and L4 that also contains the points below *B*'s first box's border box, then let *A*'s anchor point be the horizontal center of *A*'s top margin edge.

Otherwise, if the anchor point of *B* lies inside the area bounded by L4 and L1 that also contains the points to the left of *B*'s first box's border box, then let *A*'s anchor point be the vertical center of *A*'s right margin edge.

Otherwise, the anchor point of *B* lies inside the area bounded by L2 and L3 that also contains the points to the right of *B*'s first box's border box; let *A*'s anchor point be the vertical center of *A*'s left margin edge.

↳ **If the computed value of 'anchor-point' is a specific point on *A* and 'none' on *B***

The anchor point of *A* is the point given by its 'anchor-point' property.

If the anchor point of *A* is the center point of *A*'s first box's margin box, then *B*'s anchor point is the center point of its first box's border box.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the anchor point of  $A$  lies on  $L_1$  or  $L_2$ , or inside the area bounded by  $L_1$  and  $L_2$  that also contains the points above  $A$ 's first box's margin box, then let  $B$ 's anchor point be the horizontal center of  $B$ 's bottom border edge.

Otherwise, if the anchor point of  $A$  lies on  $L_3$  or  $L_4$ , or inside the area bounded by  $L_4$  and  $L_4$  that also contains the points below  $A$ 's first box's margin box, then let  $B$ 's anchor point be the horizontal center of  $B$ 's top border edge.

Otherwise, if the anchor point of  $A$  lies inside the area bounded by  $L_4$  and  $L_1$  that also contains the points to the left of  $A$ 's first box's margin box, then let  $B$ 's anchor point be the vertical center of  $B$ 's right border edge.

Too slow? Try reading the multipage copy of

↳ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*(Note: The 'position' property may be not entirely clear (e.g. in the face of margin collapsing and 'clear' handling of in-flow blocks).)*

contains the points to the border edge.

roperties.

*the while A always has  
in the box it is  
margin box whose*

In cases where  $B$  does not have a border box but its border box is used by the algorithm above, user agents must use its first box's content area instead. (This is in particular an issue with boxes in tables that have 'border-collapse' set to 'collapse').

**Note:** *The trivial example of an element that does not have a rendered box is one whose 'display' property computes to 'none'. However, there are many other cases; e.g. table columns do not have boxes (their properties merely affect other boxes).*

When an element's 'position' property must compute to 'absolute-anchored', the 'float' property does not apply and must compute to 'none', the 'display' property must compute to a value as described by the table in the section of CSS 2.1 describing the *relationships between 'display', 'position', and 'float'*, and the element's box must be positioned using the rules for absolute positioning but with its static position set such that if the box is positioned in its static position, its anchor point is exactly aligned over the anchor point of the element to which it is magically aligned.

**Note:** *The 'absolute-anchored' keyword is not a keyword that can be specified in CSS; the 'position' property can only compute to this value if the dialog element is positioned via the APIs described above.*

The `open` IDL attribute must reflect the `open` content attribute.

#### 4.11.7.1 Anchor points

This section will eventually be moved to a CSS specification; it is specified here only on an interim basis until an editor can be found to own this.

##### 'anchor-point'

Value:	none   <position>
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	refer to width or height of box; see prose
Media:	visual
Computed value:	The specified value, but with any lengths replaced by their corresponding absolute length
Animatable:	no
Canonical order:	per grammar

The 'anchor-point' property specifies a point to which dialog boxes are to be aligned.

If the value is a <position>, the alignment point is the point given by the value, which must be interpreted relative to the element's first rendered box's margin box. Percentages must be calculated relative to the element's first rendered box's margin box (specifically, its width for the horizontal position and its height for the vertical position). [CSSVALUES] [CSS]

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

## 4.12 LINKS

### 4.12.1 Introduction

Links are a conceptual construct, created by `a`, `area`, and `link` elements, that represent a connection between two resources, one of which is the current Document. There are two kinds of links in HTML:

#### *Links to external resources*

These are links to resources that are to be used to augment the current document, generally automatically processed by the user agent.

**Hyperlinks** Too slow? Try reading the multipage copy of  
The age the spec instead: <https://web.archive.org>

For link elec /web/20130103030411/http:  
for those key //whatwg.org/html

Similarly, for attribute as defined for those keywords in the link types section. Unlike `link` elements, however, a and `area` element with an `href` attribute that either do not have a `rel` attribute, or whose `rel` attribute has no keywords that are defined as specifying hyperlinks, must also create a hyperlink. This implied hyperlink has no special meaning (it has no link type) beyond linking the element's document to the resource given by the element's `href` attribute.

A hyperlink can have one or more **hyperlink annotations** that modify the processing semantics of that hyperlink.

### 4.12.2 Links created by a and area elements

The `href` attribute on `a` and `area` elements must have a value that is a valid URL potentially surrounded by spaces.

**Note:** The `href` attribute on `a` and `area` elements is not required; when those elements do not have `href` attributes they do not create hyperlinks.

The `target` attribute, if present, must be a valid browsing context name or keyword. It gives the name of the browsing context that will be used. User agents use this name when following hyperlinks.

When an `a` or `area` element's activation behavior is invoked, the user agent may allow the user to indicate a preference regarding whether the hyperlink is to be used for navigation or whether the resource it specifies is to be downloaded.

In the absence of a user preference, the default should be navigation if the element has no `download` attribute, and should be to download the specified resource if it does.

Whether determined by the user's preferences or via the presence or absence of the attribute, if the decision is to use the hyperlink for navigation then the user agent must follow the hyperlink, and if the decision is to use the hyperlink to download a resource, the user agent must download the hyperlink. These terms are defined in subsequent sections below.

The `download` attribute, if present, indicates that the author intends the hyperlink to be used for downloading a resource. The attribute may have a value; the value, if any, specifies the default filename that the author recommends for use in labeling the resource in a local file system. There are no restrictions on allowed values, but authors are cautioned that most file systems have limitations with regard to what punctuation is supported in file names, and user agents are likely to adjust file names accordingly.

The `ping` attribute, if present, gives the URLs of the resources that are interested in being notified if the user follows the hyperlink. The value must be a set of space-separated tokens, each of which must be a valid non-empty URL. The value is used by the user agent for hyperlink auditing.

The `rel` attribute on `a` and `area` elements controls what kinds of links the elements create. The attribute's value must be a set of space-separated tokens. The allowed keywords and their meanings are defined below.

The `rel` attribute has no default value. If the attribute is omitted or if none of the values in the attribute are recognized by the user agent, then the document has no particular relationship with the destination resource other than there being a hyperlink between the two.

The `hreflang` attribute on `a` and `area` elements that create hyperlinks, if present, gives the language of the linked resource. It is purely advisory. The value must be a valid BCP 47 language tag. [BCP47] User agents must not consider this attribute authoritative —

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

included in the link to the resource to determine its type.

#### 4.12.3 Following hyperlinks

When a user **follows a hyperlink** created by an element, the user agent must resolve the URL given by the `href` attribute of that element, relative to that element, and if that is successful, must navigate a browsing context to the resulting absolute URL. In the case of server-side image maps, that absolute URL must have its *hyperlink suffix* appended to it before the navigation is started.

If resolving the URL fails, the user agent may navigate to an error page to report the error.

If the user is navigating a multipage copy of the spec instead:

Otherwise, if the user agent is navigating a multipage copy of the spec instead:

Otherwise, if the hyperlink is a **sidebar hyperlink** and the user agent implements a feature that can be considered a secondary browsing context, such a secondary browsing context may be selected as the browsing context to be navigated.

Otherwise, if the element is an `a` or `area` element with no `target` attribute, but the `Document` contains a `base` element with a `target` attribute, then the browsing context that is navigated must be chosen by applying the rules for choosing a browsing context given a browsing context name, using the value of the `target` attribute of the first such `base` element as the browsing context name. If these rules result in the creation of a new browsing context, it must be navigated with replacement enabled.

Otherwise, the browsing context that must be navigated is the same browsing context as the one which the element itself is in.

The navigation must be done with the browsing context that contains the `Document` object with which the element in question is associated as the source browsing context.

#### 4.12.4 Downloading resources

In some cases, resources are intended for later use rather than immediate viewing. To indicate that a resource is intended to be downloaded for use later, rather than immediately used, the `download` attribute can be specified on the `a` or `area` element that creates the hyperlink to that resource.

The `download` attribute can furthermore be given a value, to specify the filename that user agents are to use when storing the resource in a file system. This value can be overridden by the `Content-Disposition` HTTP header's `filename` parameters. [RFC6266]

In cross-origin situations, the `download` attribute has to be combined with the `Content-Disposition` HTTP header, specifically with the `attachment` disposition type, to avoid the user being warned of possibly nefarious activity. (This is to protect users from being made to download sensitive personal or confidential information without their full understanding.)

When a user **downloads a hyperlink** created by an element, the user agent must run the following steps:

1. Resolve the URL given by the `href` attribute of that element, relative to that element.
2. If resolving the URL fails, the user agent may report the error to the user in a user-agent-specific manner, may navigate to an error page to report the error, or may ignore the error and do nothing. In either case, the user agent must abort these steps.
3. Otherwise, let `URL` be the resulting absolute URL.
4. In the case of server-side image maps, append the *hyperlink suffix* to `URL`.
5. Return to whatever algorithm invoked these steps and continue these steps asynchronously.
6. Fetch `URL` and handle the resulting resource as a download.

When a user agent is to handle a resource obtained from a fetch algorithm **as a download**, it should provide the user with a way to save the resource for later use, if a resource is successfully obtained; or otherwise should report any problems downloading the file to the user.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. Let *filename* be the void value.
2. If the resource has a Content-Disposition header, that header specifies the attachment disposition type, and the header includes filename information, then let *filename* have the value specified by the header, and jump to the step labeled "sanitize" below. [RFC6266]
3. Let *resource origin* be the origin of the resource being downloaded.
4. Let *interface origin* be the origin of the Document in which the download or navigate action resulting in the download was initiated, if any.

5. If the *sam*  
    6. If the *tn*  
       info  
       / web/20130103030411/http:  
    7. If the *whi*  
       while  
       its val

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

true if *resource origin* is the  
    er includes filename  
    "sanitize" below. [RFC6266]

ent of the hyperlink from  
    re was such an attribute but  
    proposed *filename*.

8. Let *proposed filename* have the value of the download attribute of the element of the hyperlink that initiated the download at the time the download was initiated.
9. If *trusted operation* is true, let *filename* have the value of *proposed filename*, and jump to the step labeled "sanitize" below.
10. If the resource has a Content-Disposition header and that header specifies the attachment disposition type, let *filename* have the value of *proposed filename*, and jump to the step labeled "sanitize" below. [RFC6266]
11. *No proposed filename*: If *trusted operation* is true, or if the user indicated a preference for having the resource in question downloaded, let *filename* have a value derived from the URL of the resource in a user-agent-defined manner, and jump to the step labeled "sanitize" below.
12. Act in a user-agent-defined manner to safeguard the user from a potentially hostile cross-origin download. If the download is not to be aborted, then let *filename* be set to the user's preferred file name or to a file name selected by the user agent, and jump to the step labeled "sanitize" below.

*If the algorithm reaches this step, then a download was begun from a different origin than the resource being downloaded, and the origin did not mark the file as suitable for downloading, and the download was not initiated by the user. This could be because a download attribute was used to trigger the download, or because the resource in question is not of a type that the user agent supports.*

*This could be dangerous, because, for instance, a hostile server could be trying to get a user to unknowingly download private information and then re-upload it to the hostile server, by tricking the user into thinking the data is from the hostile server.*

*Thus, it is in the user's interests that the user be somehow notified that the resource in question comes from quite a different source, and to prevent confusion, any suggested filename from the potentially hostile interface origin should be ignored.*

13. *Sanitize*: Optionally, allow the user to influence *filename*. For example, a user agent could prompt the user for a file name, potentially providing the value of *filename* as determined above as a default value.
14. Adjust *filename* to be suitable for the local file system.
  - For example, this could involve removing characters that are not legal in file names, or trimming leading and trailing whitespace.
15. If the platform conventions do not in any way use extensions to determine the types of file on the file system, then return *filename* as the file name and abort these steps.
16. Let *claimed type* be the type given by the resource's Content-Type metadata, if any is known. Let *named type* be the type given by *filename*'s extension, if any is known. For the purposes of this step, a *type* is a mapping of a MIME type to an extension.
17. If *named type* is consistent with the user's preferences (e.g. because the value of *filename* was determined by prompting the user), then return *filename* as the file name and abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Otherwise, if `named type` is known to be potentially dangerous (e.g. it will be treated by the platform conventions as a native executable, shell script, HTML application, or executable-macro-capable document) then optionally alter `filename` to add a known-safe extension (e.g. `".txt"`).

**Note:** This last step would make it impossible to download executables, which might not be desirable. As always, implementors are forced to balance security and usability in this matter.

20. Return `filename` as the file name.

For the purpose of identifying the file name to

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

User agents deciding whether

conventions dictate will be used for the last dot (".") in the executed.

by download attribute, in

#### 4.12.4.1 H [//whatwg.org/html](#)

If a hyperlink created by an `a` or `area` element has a `ping` attribute, and the user follows the hyperlink, and the value of the element's `href` attribute can be resolved, relative to the element, without failure, then the user agent must take the `ping` attribute's value, split that string on spaces, resolve each resulting token relative to the element, and then each of the resulting absolute URLs should be fetched from the origin of the `Document` containing the hyperlink (as described below). (Tokens that fail to resolve are ignored.) This may be done in parallel with the primary request, and is independent of the result of that request.

User agents should allow the user to adjust this behavior, for example in conjunction with a setting that disables the sending of HTTP `Referer` (sic) headers. Based on the user's preferences, UAs may either ignore the `ping` attribute altogether, or selectively ignore URLs in the list (e.g. ignoring any third-party URLs).

For URLs that are HTTP URLs, the requests must be performed using the POST method, with an entity body with the MIME type `text/ping` consisting of the four-character string "PING". All relevant cookie and HTTP authentication headers must be included in the request. Which other headers are required depends on the URLs involved.

↳ **If both the address of the Document object containing the hyperlink being audited and the ping URL have the same origin**

The request must include a `Ping-From` HTTP header with, as its value, the address of the document containing the hyperlink, and a `Ping-To` HTTP header with, as its value, the address of the absolute URL of the target of the hyperlink. The request must not include a `Referer` (sic) HTTP header.

↳ **Otherwise, if the origins are different, but the document containing the hyperlink being audited was not retrieved over an encrypted connection**

The request must include a `Referer` (sic) HTTP header with, as its value, the address of the document containing the hyperlink, a `Ping-From` HTTP header with the same value, and a `Ping-To` HTTP header with, as its value, the address of the target of the hyperlink.

↳ **Otherwise, the origins are different and the document containing the hyperlink being audited was retrieved over an encrypted connection**

The request must include a `Ping-To` HTTP header with, as its value, the address of the target of the hyperlink. The request must neither include a `Referer` (sic) HTTP header nor include a `Ping-From` HTTP header.

**Note:** To save bandwidth, implementors might also wish to consider omitting optional headers such as `Accept` from these requests.

User agents must, unless otherwise specified by the user, honor the HTTP headers (including, in particular, redirects and HTTP cookie headers), but must ignore any entity bodies returned in the responses. User agents may close the connection prematurely once they start receiving an entity body. [COOKIES]

When the `ping` attribute is present, user agents should clearly indicate to the user that following the hyperlink will also cause secondary requests to be sent in the background, possibly including listing the actual target URLs.

For example, a visual user agent could include the hostnames of the target ping URLs along with the hyperlink's actual URL in a status bar or tooltip.

**The `ping` attribute is redundant with pre-existing technologies like HTTP redirects and JavaScript in allowing Web pages to track which off-site links are most popular or allowing advertisers to track click-through rates.**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- It allows the user to disable the notifications without losing the underlying link functionality.
- It allows the UA to optimize the use of available network bandwidth so that the target page loads faster.

Thus, while it is possible to track users without this feature, authors are encouraged to use the `ping` attribute so that the user agent can make the user experience more transparent.

#### 4.12.5 Link

The following is a list of link types for the `link` attribute.

In this section, the term `document` refers to the document being processed.

To determine the type of a link, the tokens are checked in the following order:

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ative; the actual definitions

the link, and the term *current*

lit on spaces. The resulting

Except where otherwise specified, a keyword must not be specified more than once per `rel` attribute.

Link types are always ASCII case-insensitive, and must be compared as such.

|| Thus, `rel="next"` is the same as `rel="NEXT"`.

Link type	Effect on...		Brief description
	link	a and area	
alternate	Hyperlink	Hyperlink	Gives alternate representations of the current document.
author	Hyperlink	Hyperlink	Gives a link to the author of the current document or article.
bookmark	<i>not allowed</i>	Hyperlink	Gives the permalink for the nearest ancestor section.
help	Hyperlink	Hyperlink	Provides a link to context-sensitive help.
icon	External Resource	<i>not allowed</i>	Imports an icon to represent the current document.
license	Hyperlink	Hyperlink	Indicates that the main content of the current document is covered by the copyright license described by the referenced document.
next	Hyperlink	Hyperlink	Indicates that the current document is a part of a series, and that the next document in the series is the referenced document.
nofollow	<i>not allowed</i>	Annotation	Indicates that the current document's original author or publisher does not endorse the referenced document.
noreferrer	<i>not allowed</i>	Annotation	Requires that the user agent not send an HTTP Referer (sic) header if the user follows the hyperlink.
prefetch	External Resource	External Resource	Specifies that the target resource should be preemptively cached.
prev	Hyperlink	Hyperlink	Indicates that the current document is a part of a series, and that the previous document in the series is the referenced document.
search	Hyperlink	Hyperlink	Gives a link to a resource that can be used to search through the current document and its related pages.
stylesheet	External Resource	<i>not allowed</i>	Imports a stylesheet.
tag	<i>not allowed</i>	Hyperlink	Gives a tag (identified by the given address) that applies to the current document.

Some of the types described below list synonyms for these values. These are to be handled as specified by user agents, but must not be used in documents.

##### 4.12.5.1 Link type "alternate"

The `alternate` keyword may be used with `link`, `a`, and `area` elements.

The meaning of this keyword depends on the values of the other attributes.

↳ If the element is a `link` element and the `rel` attribute also contains the keyword `stylesheet`

The `alternate` keyword modifies the meaning of the `stylesheet` keyword in the way described for that keyword. The `alternate` keyword does not create a link of its own.

↳ The `alternate` keyword is used with the `type` attribute set to the value `application/rss+xml` or the value

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

attribute set to the value `application/rss+xml` or the value `application/atom+xml` must be treated as the default syndication feed for the purposes of feed autodiscovery.

The following `link` element gives the syndication feed for the current page:

```
<link rel="alternate" type="application/atom+xml" href="data.xml">
```

The following extract offers various different syndication feeds:

```
<link rel="alternate" hreflang="en" type="text/html" />
<link rel="alternate" hreflang="fr" type="text/html" />
<link rel="alternate" hreflang="de" type="text/html" />
<link rel="alternate" hreflang="es" type="text/html" />
```

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### ↳ Otherwise

The keyword creates a hyperlink referencing an alternate representation of the current document.

The nature of the referenced document is given by the `hreflang`, and `type` attributes.

If the `alternate` keyword is used with the `hreflang` attribute, and that attribute's value differs from the root element's language, it indicates that the referenced document is a translation.

If the `alternate` keyword is used with the `type` attribute, it indicates that the referenced document is a reformulation of the current document in the specified format.

The `hreflang` and `type` attributes can be combined when specified with the `alternate` keyword.

For example, the following link is a French translation that uses the PDF format:

```
<link rel=alternate type=application/pdf hreflang=fr href=manual-fr>
```

This relationship is transitive — that is, if a document links to two other documents with the link type "alternate", then, in addition to implying that those documents are alternative representations of the first document, it is also implying that those two documents are alternative representations of each other.

### 4.12.5.2 Link type "author"

The `author` keyword may be used with `link`, `a`, and `area` elements. This keyword creates a hyperlink.

For `a` and `area` elements, the `author` keyword indicates that the referenced document provides further information about the author of the nearest `article` element ancestor of the element defining the hyperlink, if there is one, or of the page as a whole, otherwise.

For `link` elements, the `author` keyword indicates that the referenced document provides further information about the author for the page as a whole.

**Note:** The "referenced document" can be, and often is, a `mailto:` URL giving the e-mail address of the author.  
[MAILTO]

**Synonyms:** For historical reasons, user agents must also treat `link`, `a`, and `area` elements that have a `rev` attribute with the value "made" as having the `author` keyword specified as a link relationship.

### 4.12.5.3 Link type "bookmark"

The `bookmark` keyword may be used with `a` and `area` elements. This keyword creates a hyperlink.

The `bookmark` keyword gives a permalink for the nearest ancestor `article` element of the linking element in question, or of the section the linking element is most closely associated with, if there are no ancestor `article` elements.

The following snippet has three permalinks. A user agent could determine which permalink applies to which part of the spec by

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<div id="a">
 <h2>First example</h2>
 <p>This permalink applies to
only the content from the first H2 to the second H2. The DIV isn't
exactly that section, but it roughly corresponds to it.</p>
</div>
<h2>Second example</h2>
<article id="b">
 <p>This permalink applies to
the content from the second H2 to the end of the article.</p>
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 4.12.5.4 Link type "help"

The `help` keyword may be used with `link`, `a`, and `area` elements. This keyword creates a hyperlink.

For `a` and `area` elements, the `help` keyword indicates that the referenced document provides further help information for the parent of the element defining the hyperlink, and its children.

In the following example, the form control has associated context-sensitive help. The user agent could use this information, for example, displaying the referenced document if the user presses the "Help" or "F1" key.

```
<p><label> Topic: <input name=topic> (Help)
</label></p>
```

For `link` elements, the `help` keyword indicates that the referenced document provides help for the page as a whole.

For `a` and `area` elements, on some browsers, the `help` keyword causes the link to use a different cursor.

#### 4.12.5.5 Link type "icon"

The `icon` keyword may be used with `link` elements. This keyword creates an external resource link.

The specified resource is an icon representing the page or site, and should be used by the user agent when representing the page in the user interface.

Icons could be auditory icons, visual icons, or other kinds of icons. If multiple icons are provided, the user agent must select the most appropriate icon according to the `type`, `media`, and `sizes` attributes. If there are multiple equally appropriate icons, user agents must use the last one declared in tree order at the time that the user agent collected the list of icons. If the user agent tries to use an icon but that icon is determined, upon closer examination, to in fact be inappropriate (e.g. because it uses an unsupported format), then the user agent must try the next-most-appropriate icon as determined by the attributes.

**Note:** User agents are not required to update icons when the list of icons changes, but are encouraged to do so.

There is no default type for resources given by the `icon` keyword. However, for the purposes of determining the type of the resource, user agents must expect the resource to be an image.

The `sizes` attribute gives the sizes of icons for visual media. Its value, if present, is merely advisory. User agents may use the value to decide which icon(s) to use if multiple icons are available.

If specified, the attribute must have a value that is an unordered set of unique space-separated tokens which are ASCII case-insensitive. Each value must be either an ASCII case-insensitive match for the string "any", or a value that consists of two valid non-negative integers that do not have a leading U+0030 DIGIT ZERO (0) character and that are separated by a single U+0078 LATIN SMALL LETTER X or U+0058 LATIN CAPITAL LETTER X character.

The keywords represent icon sizes.

To parse and process the attribute's value, the user agent must first split the attribute's value on spaces, and must then parse each

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- If the keyword doesn't contain exactly one U+0078 LATIN SMALL LETTER X or U+0058 LATIN CAPITAL LETTER X character, then this keyword doesn't represent anything. Abort these steps for that keyword.
- Let *width string* be the string before the "x" or "X".
- Let *height string* be the string after the "x" or "X".
- If either *width string* or *height string* start with a U+0030 DIGIT ZERO (0) character or contain any characters other than ASCII digits, then this keyword doesn't represent anything. Abort these steps for that keyword

- App **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**
- App **/web/20130103030411/http:**
- The dev **Is and a height of *height***

The keyword **//whatwg.org/html**

→ in the linked resource.

In the absence of a `link rel="icon"` element, user agents may instead attempt to fetch and use an icon with the absolute URL obtained by resolving the URL `"/favicon.ico"` against the document's address, as if the page had declared that icon using the `icon` keyword.

The following snippet shows the top part of an application with several icons.

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>lsForums - Inbox</title>
 <link rel="icon" href="favicon.png" sizes="16x16" type="image/png">
 <link rel="icon" href="windows.ico" sizes="32x32 48x48" type="image/vnd.microsoft.icon">
 <link rel="icon" href="mac.icns" sizes="128x128 512x512 8192x8192 32768x32768">
 <link rel="icon" href="iphone.png" sizes="57x57" type="image/png">
 <link rel="icon" href="gnome.svg" sizes="any" type="image/svg+xml">
 <link rel="stylesheet" href="lsforums.css">
 <script src="lsforums.js"></script>
 <meta name="application-name" content="lsForums">
 </head>
 <body>
 ...
 </body>
```

For historical reasons, the `icon` keyword may be preceded by the keyword "shortcut". If the "shortcut" keyword is present, it must be come immediately before the `icon` keyword and the two keywords must be separated by only a single U+0020 SPACE character.

#### 4.12.5.6 Link type "license"

The `license` keyword may be used with `link`, `a`, and `area` elements. This keyword creates a hyperlink.

The `license` keyword indicates that the referenced document provides the copyright license terms under which the main content of the current document is provided.

This specification does not specify how to distinguish between the main content of a document and content that is not deemed to be part of that main content. The distinction should be made clear to the user.

Consider a photo sharing site. A page on that site might describe and show a photograph, and the page might be marked up as follows:

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>Example Pictures: Kissat</title>
 <link rel="stylesheet" href="/style/default">
 </head>
 <body>
 <h1>Kissat</h1>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```

<figcaption>Kissat</figcaption>
</figure>
<p>One of them has six toes!</p>
<p><small>MIT
Licensed</small></p>
<footer>
Home | Photo index
<p><small>© copyright 2009 Exampl Pictures. All Rights Reserved.</small></p>
</footer>
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Synonyms:

e document. In particular not  
. This could be made clearer  
the page copyright in light

use keyword.

#### 4.12.5.7 Link type "nofollow"

The `nofollow` keyword may be used with `a` and `area` elements. This keyword does not create a hyperlink, but annotates any other hyperlinks created by the element (the implied hyperlink, if no other keywords create one).

The `nofollow` keyword indicates that the link is not endorsed by the original author or publisher of the page, or that the link to the referenced document was included primarily because of a commercial relationship between people affiliated with the two pages.

#### 4.12.5.8 Link type "noreferrer"

The `noreferrer` keyword may be used with `a` and `area` elements. This keyword does not create a hyperlink, but annotates any other hyperlinks created by the element (the implied hyperlink, if no other keywords create one).

It indicates that no referrer information is to be leaked when following the link.

If a user agent follows a link defined by an `a` or `area` element that has the `noreferrer` keyword, the user agent must not include a `Referer` (sic) HTTP header (or equivalent for other protocols) in the request.

This keyword also causes the `opener` attribute to remain null if the hyperlink creates a new browsing context.

#### 4.12.5.9 Link type "prefetch"

The `prefetch` keyword may be used with `link`, `a`, and `area` elements. This keyword creates an external resource link.

The `prefetch` keyword indicates that preemptively fetching and caching the specified resource is likely to be beneficial, as it is highly likely that the user will require this resource.

There is no default type for resources given by the `prefetch` keyword.

#### 4.12.5.10 Link type "search"

The `search` keyword may be used with `link`, `a`, and `area` elements. This keyword creates a hyperlink.

The `search` keyword indicates that the referenced document provides an interface specifically for searching the document and its related resources.

**Note:** OpenSearch description documents can be used with `link` elements and the `search` link type to enable user agents to autodiscover search interfaces. [OPENSEARCH]

#### 4.12.5.11 Link type "stylesheet"

The `stylesheet` keyword may be used with `link` elements. This keyword creates an external resource link that contributes to the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the `alternate` keyword is also specified on the `link` element, then the link is an alternative stylesheet; in this case, the `title` attribute must be specified on the `link` element, with a non-empty value.

The default type for resources given by the `stylesheet` keyword is `text/css`.

The appropriate time to obtain the resource is when the external resource link is created or when its element is inserted into a document, whichever happens last. If the resource is an alternative stylesheet then the user agent may defer obtaining the resource until it is part of the preferred style sheet set. [CSSOM]

**Quirk:** If the document has been set to quirks mode, has the same origin as the URL of the external resource, and the Content-Type metadata of'

## Too slow? Try reading the multipage copy of

### 4.12.5.12 the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The tag key [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html):

The tag key [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html):ument.

**Note:** Since ... indicates that the tag applies to the current document, it would be inappropriate to use this keyword in the markup of a tag cloud, which lists the popular tags across a set of pages.

This document is about some gems, and so it is tagged with "http://en.wikipedia.org/wiki/Gemstone" to unambiguously categorise it as applying to the "jewel" kind of gems, and not to, say, the towns in the US, the Ruby package format, or the Swiss locomotive class:

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>My Precious</title>
 </head>
 <body>
 <header><h1>My precious</h1> <p>Summer 2012</p></header>
 <p>Recently I managed to dispose of a red gem that had been bothering me. I now have a much nicer blue sapphire.</p>
 <p>The red gem had been found in a bauxite stone while I was digging out the office level, but nobody was willing to haul it away. The same red gem stayed there for literally years.</p>
 <footer>
 Tags: Gemstone
 </footer>
 </body>
</html>
```

In this document, there are two articles. The "tag" link, however, applies to the whole page (and would do so wherever it was placed, including if it was within the `article` elements).

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>Gem 4/4</title>
 </head>
 <body>
 <article>
 <h1>801: Steinbock</h1>
 <p>The number 801 Gem 4/4 electro-diesel has an ibex and was rebuilt in 2002.</p>
 </article>
 <article>
 <h1>802: Murmeltier</h1>
 <figure>

 <figcaption>The 802 in the 1980s, above Lago Bianco.</figcaption>
 </figure>
 </article>
 </body>
</html>
```

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

&lt;/html&gt;

#### 4.12.5.13 Sequential link types

Some documents form part of a sequence of documents.

A sequence of documents is one where each document can have a *previous sibling* and a *next sibling*. A document with no previous sibling is the start of its sequence, a document with no next sibling is the end of its sequence.

A document

Too slow? Try reading the multipage copy of

**4.12.5.13.1** the spec instead: <https://web.archive.org>

The next logical document: [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

The next logical document: [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html) document that is the next

#### 4.12.5.13.2 Link type "prev"

The `prev` keyword may be used with `link`, `a`, and `area` elements. This keyword creates a hyperlink.

The `prev` keyword indicates that the document is part of a sequence, and that the link is leading to the document that is the previous logical document in the sequence.

**Synonyms:** For historical reasons, user agents must also treat the keyword "previous" like the `prev` keyword.

#### 4.12.5.14 Other link types

Extensions to the predefined set of link types may be registered in the microformats wiki existing-rel-values page. [MFREL]

Anyone is free to edit the microformats wiki existing-rel-values page at any time to add a type. Extension types must be specified with the following information:

##### Keyword

The actual value being defined. The value should not be confusingly similar to any other defined value (e.g. differing only in case).

If the value contains a U+003A COLON character (:), it must also be an absolute URL.

##### Effect on... link

One of the following:

##### Not allowed

The keyword must not be specified on `link` elements.

##### Hyperlink

The keyword may be specified on a `link` element; it creates a hyperlink.

##### External Resource

The keyword may be specified on a `link` element; it creates an external resource link.

##### Effect on... a and area

One of the following:

##### Not allowed

The keyword must not be specified on `a` and `area` elements.

##### Hyperlink

The keyword may be specified on `a` and `area` elements; it creates a hyperlink.

##### External Resource

The keyword may be specified on `a` and `area` elements; it creates an external resource link.

##### Hyperlink Annotation

The keyword may be specified on `a` and `area` elements; it annotates other hyperlinks created by the element.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

A link to a more detailed description of the keyword's semantics and requirements. It could be another page on the Wiki, or a link to an external page.

## Synonyms

A list of other keyword values that have exactly the same processing requirements. Authors should not use the values defined to be synonyms, they are only intended to allow user agents to support legacy content. Anyone may remove synonyms that are not used in practice; only names that need to be processed as synonyms for compatibility with legacy content are to be registered in this way.

### Status



One

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

and is, or soon will be, using

Pro

**/rat** [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

ambiguously defines how to

Disc.

The keyword has received wide peer review and it has been found wanting. Existing pages are using this keyword, but new pages should avoid it. The "brief description" and "specification" entries will give details of what authors should use instead, if anything.

If a keyword is found to be redundant with existing values, it should be removed and listed as a synonym for the existing value.

If a keyword is registered in the "proposed" state for a period of a month or more without being used or specified, then it may be removed from the registry.

If a keyword is added with the "proposed" status and found to be redundant with existing values, it should be removed and listed as a synonym for the existing value. If a keyword is added with the "proposed" status and found to be harmful, then it should be changed to "discontinued" status.

Anyone can change the status at any time, but should only do so in accordance with the definitions above.

Conformance checkers must use the information given on the microformats wiki existing-rel-values page to establish if a value is allowed or not: values defined in this specification or marked as "proposed" or "ratified" must be accepted when used on the elements for which they apply as described in the "Effect on..." field, whereas values marked as "discontinued" or not listed in either this specification or on the aforementioned page must be rejected as invalid. Conformance checkers may cache this information (e.g. for performance reasons or to avoid the use of unreliable network connectivity).

When an author uses a new type not defined by either this specification or the Wiki page, conformance checkers should offer to add the value to the Wiki, with the details described above, with the "proposed" status.

Types defined as extensions in the microformats wiki existing-rel-values page with the status "proposed" or "ratified" may be used with the `rel` attribute on `link`, `a`, and `area` elements in accordance to the "Effect on..." field. [MFREL]

## 4.13 Common idioms without dedicated elements

### 4.13.1 The main part of the content

The main content of a page — not including headers and footers, navigation links, sidebars, advertisements, and so forth — can be marked up in a variety of ways, depending on the needs of the author.

The simplest solution is to not mark up the main content at all, and just leave it as implicit. Another way to think of this is that the `body` elements marks up the main content of the page, and the bits that aren't main content are excluded through the use of more appropriate elements like `aside` and `nav`.

Here is a short Web page marked up along this minimalistic school of thought. The main content is highlighted. Notice how all the *other* content in the `body` is marked up with elements to indicate that it's not part of the main content, in this case `header`, `nav`, and `footer`.

```
<!DOCTYPE HTML>
<html>
 <head>
 <title> My Toys </title>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

<nav>
 <p>Home</p>
 <p>Contact</p>
</nav>
<p>I really like my chained book and my telephone. I'm not such a
fan of my big ball.</p>
<p>Another toy I like is my mirror.</p>
<footer>
 <p>© copyright 2010 by the boy</p>

```



Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If the main content  
would be ap-

n the article element

The code would be:

```

<:
<html>
 <head>
 <title> The Boy Blog: My Toys </title>
 </head>
 <body>
 <header>
 <h1>The Boy Blog</h1>
 </header>
 <nav>
 <p>Home</p>
 <p>Contact</p>
 </nav>
 <article>
 <header>
 <h1>My toys</h1>
 <p>Published August 4th</p>
 </header>
 <p>I really like my chained book and my telephone. I'm not such a
fan of my big ball.</p>
 <p>Another toy I like is my mirror.</p>
 </article>
 <footer>
 <p>© copyright 2010 by the boy</p>
 </footer>
 </body>
</html>

```

If the main content is not an independent unit of content so much as a section of a larger work, for instance a chapter, then the `section` element would be appropriate to mark up the main content of the document.

Here is the same document, case as a chapter in an online book:

```

<!DOCTYPE HTML>
<html>
 <head>
 <title> Chapter 2: My Toys – The Book of the Boy </title>
 </head>
 <body>
 <header>
 <h1>Chapter 2: My Toys</h1>
 </header>
 <nav>
 <p>Front Page</p>
 <p>Table of Contents</p>
 <p>Chapter 1 – Chapter 3</p>

```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
</section>
<footer>
 <p>© copyright 2010 by the boy</p>
</footer>
</body>
</html>
```

If neither `article` nor `section` would be appropriate, but the main content still needs an explicit element, for example for styling purposes, then the `div` element can be used.

This Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
</head>
<body>
 <header>
 <h1>My toys</h1>
 </header>
 <nav>
 <p>Home</p>
 <p>Contact</p>
 </nav>
 <div>
 <p>I really like my chained book and my telephone. I'm not such a fan of my big ball.</p>
 <p>Another toy I like is my mirror.</p>
 </div>
 <footer>
 <p>© copyright 2010 by the boy</p>
 </footer>
</body>
</html>
```

#### 4.13.2 Bread crumb navigation

This specification does not provide a machine-readable way of describing bread-crumb navigation menus. Authors are encouraged to just use a series of links in a paragraph. The `nav` element can be used to mark the section containing these paragraphs as being navigation blocks.

In the following example, the current page can be reached via two paths.

```
<nav>
 <p>
 Main >
 Products >
 Dishwashers >
 <a>Second hand
 </p>
 <p>
 Main >
 Second hand >
 <a>Dishwashers
 </p>
</nav>
```

#### 4.13.3 Tag clouds

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<style>
@media screen, print, handheld, tv {
 /* Should be ignored by non-visual browsers */
 .tag-cloud > li > span { display: none; }
 .tag-cloud > li { display: inline; }
 .tag-cloud-1 { font-size: 0.7em; }
 .tag-cloud-2 { font-size: 0.9em; }
 .tag-cloud-3 { font-size: 1.1em; }
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
 (popular)
```

```
 (rare)
```

```

```

```
<li class="tag-cloud-5">pear (very
```

```
popular)
```

```

```

The actual frequency of each tag is given using the `title` attribute. A CSS style sheet is provided to convert the markup into a cloud of differently-sized words, but for user agents that do not support CSS or are not visual, the markup contains annotations like "(popular)" or "(rare)" to categorize the various tags by frequency, thus enabling all users to benefit from the information.

The `ul` element is used (rather than `ol`) because the order is not particularly important: while the list is in fact ordered alphabetically, it would convey the same information if ordered by, say, the length of the tag.

The `tag` `rel=keyword` is *not* used on these `a` elements because they do not represent tags that apply to the page itself; they are just part of an index listing the tags themselves.

#### 4.13.4 Conversations

This specification does not define a specific element for marking up conversations, meeting minutes, chat transcripts, dialogues in screenplays, instant message logs, and other situations where different players take turns in discourse.

Instead, authors are encouraged to mark up conversations using `p` elements and punctuation. Authors who need to mark the speaker for styling purposes are encouraged to use `span` or `b`. Paragraphs with their text wrapped in the `i` element can be used for marking up stage directions.

This example demonstrates this using an extract from Abbot and Costello's famous sketch, *Who's on first*:

```
<p> Costello: Look, you gotta first baseman?
<p> Abbott: Certainly.
<p> Costello: Who's playing first?
<p> Abbott: That's right.
<p> Costello becomes exasperated.
<p> Costello: When you pay off the first baseman every month, who gets the money?
<p> Abbott: Every dollar of it.
```

The following extract shows how an IM conversation log could be marked up, using the `data` element to provide Unix timestamps for each line. Note that the timestamps are provided in a format that the `time` element does not support, so the `data` element is used instead (namely, Unix `time_t` timestamps). Had the author wished to mark up the data using one of the date and time formats supported by the `time` element, that element could have been used instead of `data`. This could be advantageous as it would allow data analysis tools to detect the timestamps unambiguously, without coordination with the page author.

```
<p> <data value="1319898155">14:22</data> egof I'm not that nerdy, I've only seen
30% of the star trek episodes
<p> <data value="1319898192">14:23</data> kaj if you know what percentage of the
star trek episodes you have seen, you are inarguably nerdy
<p> <data value="1319898200">14:23</data> egof it's unarguably
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

conversation. Another option to consider is describing the conversation in the form of a DOT file, and outputting the result as an SVG image to place in the document. [DOT]

```
<p> Next, you meet a fisherman. You can say one of several greetings:
<dl>
<dt> "Hello there!"
<dd>
<p> He responds with "Hello, how may I help you?"; you can respond with:
<dl>
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```
~you ~p~ he lends you his boat. The conversation ends.
<dt> "A fish." (if you have one)
<dt> "A newspaper." (if you have one)
<dt> "A pebble." (if you have one)
<dd> <p> "No thanks", he replies. Your conversation options at this point are the same as they were after asking to borrow his boat, minus any options you've suggested before.
</dl>
</dd>
</dl>
</dd>
<dt> "Vote for me in the next election!"
<dd> <p> He turns away. The conversation finishes.
<dt> "Sir, are you aware that your fish are running away?"
<dd>
<p> He looks at you skeptically and says "Fish cannot run, sir".
<dl>
<dt> "You got me!"
<dd> <p> The fisherman sighs and the conversation ends.
<dt> "Only kidding."
<dd> <p> "Good one!" he retorts. Your conversation options at this point are the same as those following "Hello there!" above.
<dt> "Oh, then what are they doing?"
<dd> <p> He looks at his fish, giving you an opportunity to steal his boat, which you do. The conversation ends.
</dl>
</dd>

```

In some games, conversations are simpler: each character merely has a fixed set of lines that they say. In this example, a game FAQ/walkthrough lists some of the known possible responses for each character:

```
<section>
<h1>Dialogue</h1>
<p><small>Some characters repeat their lines in order each time you interact with them, others randomly pick from amongst their lines. Those who respond in order have numbered entries in the lists below.</small>
<h2>The Shopkeeper</h2>

How may I help you?
Fresh apples!
A loaf of bread for madam?

<h2>The pilot</h2>
<p>Before the accident:

```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```

I'm about to fly out, sorry!
Ok, I'm not leaving right now, my plane is being cleaned.
Ok, it's not being cleaned, it needs a minor repair first.
Ok, ok, stop bothering me! Truth is, I had a crash.

<h2>Clan Leader</h2>
<p>During the first clan meeting:

 Hey, have you seen my daughter? I bet she's up to something nefarious again...

```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

&lt;/section&gt;

#### 4.13.5 Footnotes

HTML does not have a dedicated mechanism for marking up footnotes. Here are the suggested alternatives.

For short inline annotations, the `title` attribute could be used.

In this example, two parts of a dialogue are annotated with footnote-like content using the `title` attribute.

```

<p> Customer: Hello! I wish to register a complaint. Hello. Miss?
<p> Shopkeeper: Watcha mean, miss?
<p> Customer: Uh, I'm sorry, I have a cold. I wish to make a complaint.
<p> Shopkeeper: Sorry, we're closing for lunch.

```

**Note:** Unfortunately, relying on the `title` attribute is currently discouraged as many user agents do not expose the attribute in an accessible manner as required by this specification (e.g. requiring a pointing device such as a mouse to cause a tooltip to appear, which excludes keyboard-only users and touch-only users, such as anyone with a modern phone or tablet).

**Note:** If the `title` attribute is used, CSS can be used to draw the reader's attention to the elements with the attribute.

For example, the following CSS places a dashed line below elements that have a `title` attribute.

```
[title] { border-bottom: thin dashed; }
```

For longer annotations, the `a` element should be used, pointing to an element later in the document. The convention is that the contents of the link be a number in square brackets.

In this example, a footnote in the dialogue links to a paragraph below the dialogue. The paragraph then reciprocally links back to the dialogue, allowing the user to return to the location of the footnote.

```

<p> Announcer: Number 16: The <i>hand</i>.
<p> Interviewer: Good evening. I have with me in the studio tonight Mr Norman St John Polevaulter, who for the past few years has been contradicting people. Mr Polevaulter, why do you contradict people?
<p> Norman: I don't. ^{[1]}
<p> Interviewer: You told me you did!
<...
<section>

```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

For side notes, longer annotations that apply to entire sections of the text rather than just specific words or sentences, the `aside` element should be used.

In this example, a sidebar is given after a dialogue, giving it some context.

```
<p> Customer: I will not buy this record, it is scratched.
<p> Shopkeeper: I'm sorry?
<p> Customer: I will not buy this record, it is scratched.
<p> Shopkeeper: No no no... this's a baccanist's.
```

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

For figures c

surrounding prose.

In this example, a table has cells with footnotes that are given in prose. A `figure` element is used to give a single legend to the combination of the table and its footnotes.

```
<figure>
 <figcaption>Table 1. Alternative activities for knights.</figcaption>
 <table>
 <tr>
 <th> Activity
 <th> Location
 <th> Cost
 <tr>
 <td> Dance
 <td> Wherever possible
 <td> £0^{1}
 <tr>
 <td> Routines, chorus scenes^{2}
 <td> Undisclosed
 <td> Undisclosed
 <tr>
 <td> Dining^{3}
 <td> Camelot
 <td> Cost of ham, jam, and spam^{4}
 </table>
 <p id="fn1">1. Assumed.</p>
 <p id="fn2">2. Footwork impeccable.</p>
 <p id="fn3">3. Quality described as "well".</p>
 <p id="fn4">4. A lot.</p>
</figure>
```

## 4.14 Disabled elements

An element is said to be **actually disabled** if it falls into one of the following categories:

- `button` elements that are disabled
- `input` elements that are disabled
- `select` elements that are disabled
- `textarea` elements that are disabled
- `optgroup` elements that have a `disabled` attribute
- `option` elements that are disabled

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*pseudo-class.*

## 4.15 Matching HTML elements using selectors

### 4.15.1 Case-sensitivity

The Selectors specification leaves the case-sensitivity of IDs, classes, element names, attribute names, and attribute values to be defined by the user agent.

The unique purpose of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html

Classes from the spec must be treated as case-insensitive for the purposes of selector matching.

Attribute and selector matching must be treated as ASCII case-insensitive for the purposes of selector matching.

Everything else (attribute values on HTML elements, IDs and classes in no-quirks mode and limited-quirks mode, and element names, attribute names, and attribute values in XML documents) must be treated as case-sensitive for the purposes of selector matching.

### 4.15.2 Pseudo-classes

There are a number of dynamic selectors that can be used with HTML. This section defines when these selectors match HTML elements. [SELECTORS] [CSSUI]

**:link****:visited**

All `a` elements that have an `href` attribute, all `area` elements that have an `href` attribute, and all `link` elements that have an `href` attribute, must match one of `:link` and `:visited`.

Other specifications might apply more specific rules regarding how these elements are to match these pseudo-classes, to mitigate some privacy concerns that apply with straightforward implementations of this requirement.

**:active**

The `:active` pseudo-class is defined to match an element “while an element is *being activated* by the user”. For the purposes of defining the `:active` pseudo-class only, an HTML user agent must consider an element as *being activated* if it is:

- An element falling into one of the following categories between the time the user begins to indicate an intent to trigger the element’s activation behavior and either the time the user stops indicating an intent to trigger the element’s activation behavior, or the time the element’s activation behavior has finished running, whichever comes first:

- `a` elements that have an `href` attribute
- `area` elements that have an `href` attribute
- `link` elements that have an `href` attribute
- `button` elements that are not disabled
- `input` elements whose `type` attribute is in the Submit Button, Image Button, Reset Button, or Button state
- `menuitem` elements that do not have a `disabled` attribute
- elements that have their `tabindex` focus flag set

For example, if the user is using a keyboard to push a `button` element by pressing the space bar, the element would match this pseudo-class in between the time that the element received the `keydown` event and the time the element received the `keyup` event.

- An element that the user indicates using a pointing device while that pointing device is in the “down” state (e.g. for a mouse, between the time the mouse button is pressed and the time it is depressed).

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

the purposes of defining the `:hover` pseudo-class only, an HTML user agent must consider an element as being one that the user designates if it is:

- An element that the user indicates using a pointing device.
- An element that has a descendant that the user indicates using a pointing device.
- An element that is the labeled control of a `label` element that is currently matching `:hover`.

Consider in particular a fragment such as:



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

&gt; &lt;/span&gt; &lt;/p&gt;

: (and all its ancestors not with ID "c" will match the > elements match it because > is it through condition 3 (its its descendant is not

#### `:enabled`

The `:enabled` pseudo-class must match any element falling into one of the following categories:

- `a` elements that have an `href` attribute
- `area` elements that have an `href` attribute
- `link` elements that have an `href` attribute
- `button` elements that are not disabled
- `input` elements that are not disabled
- `select` elements that are not disabled
- `textarea` elements that are not disabled
- `optgroup` elements that do not have a `disabled` attribute
- `option` elements that are not disabled
- `menuitem` elements that do not have a `disabled` attribute
- `fieldset` elements that do not have a `disabled` attribute

#### `:disabled`

The `:disabled` pseudo-class must match any element that is actually disabled.

#### `:checked`

The `:checked` pseudo-class must match any element falling into one of the following categories:

- `input` elements whose `type` attribute is in the Checkbox state and whose checkedness state is true
- `input` elements whose `type` attribute is in the Radio Button state and whose checkedness state is true
- `option` elements whose selectedness is true
- `menuitem` elements whose `type` attribute is in the Checkbox state and that have a `checked` attribute
- `menuitem` elements whose `type` attribute is in the Radio state and that have a `checked` attribute

#### `:indeterminate`

The `:indeterminate` pseudo-class must match any element falling into one of the following categories:

- `input` elements whose `type` attribute is in the Checkbox state and whose indeterminate IDL attribute is set to true
- `progress` elements with no `value` content attribute

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- `input` elements whose `type` attribute is in the Submit Button or Image Button state, and that are their form's default button
- `input` elements to which the `checked` attribute applies and that have a `checked` attribute
- `option` elements that have a `selected` attribute

#### `:valid`

The `:valid` pseudo-class must match any element falling into one of the following categories:

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

es for constraint validation

#### `:invalid`

The `:invalid` pseudo-class must match all elements that are candidates for constraint validation but do not satisfy their constraints

#### `:in-range`

The `:in-range` pseudo-class must match all elements that are candidates for constraint validation, have range limitations, and that are neither suffering from an underflow nor suffering from an overflow.

#### `:out-of-range`

The `:out-of-range` pseudo-class must match all elements that are candidates for constraint validation, have range limitations, and that are either suffering from an underflow or suffering from an overflow.

#### `:required`

The `:required` pseudo-class must match any element falling into one of the following categories:

- `input` elements that are *required*
- `select` elements that have a `required` attribute
- `textarea` elements that have a `required` attribute

#### `:optional`

The `:optional` pseudo-class must match any element falling into one of the following categories:

- `input` elements to which the `required` attribute applies that are not *required*
- `select` elements that do not have a `required` attribute
- `textarea` elements that do not have a `required` attribute

#### `:read-only`

#### `:read-write`

The `:read-write` pseudo-class must match any element falling into one of the following categories, which for the purposes of Selectors are thus considered *user-alterable*: [SELECTORS]

- `input` elements to which the `readonly` attribute applies, and that are *mutable* (i.e. that do not have the `readonly` attribute specified and that are not disabled)
- `textarea` elements that do not have a `readonly` attribute, and that are not disabled
- elements that are editing hosts or editable and are neither `input` elements nor `textarea` elements

The `:read-only` pseudo-class must match all other HTML elements.

#### `:dir(ltr)`

The `:dir(ltr)` pseudo-class must match all elements whose directionality is 'ltr'.

[ ]

[Go](#) **NOV JAN FEB**  
◀ **03** ▶  
**2012 2013 2014**

[About this capture](#)

**3,445 captures**  
5 Jun 2004 - 22 Aug 2020

*Note: This specification does not define when an element matches the :focus or :lang() dynamic pseudo-classes, as those are all defined in sufficient detail in a language-agnostic fashion in the Selectors specification.*  
[SELECTORS]



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

### 5.1.1 Overview

*This section is non-normative.*

Sometimes, it is desirable to annotate content with specific machine-readable labels, e.g. to allow generic scripts to provide services that are customised to the page, or to enable content from a variety of cooperating authors to be processed by a single script in a consistent manner.

For this purpose, authors can use the microdata features described in this section. Microdata allows nested groups of name-value pairs to be added to documents, in parallel with the existing content.



## 5.1.2 The Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

*This section* At a high level, property. It is To create an

/web/20130103030411/http://whatwg.org/html

name-value pair is a

To add a property to an item, the `itemprop` attribute is used on one of the item's descendants.

Here there are two items, each of which has the property "name":

```
<div itemscope>
 <p>My name is Elizabeth.</p>
</div>

<div itemscope>
 <p>My name is Daniel.</p>
</div>
```

Markup without microdata attributes has no effect on microdata.

These two examples are exactly equivalent, at a microdata level, as the previous two examples respectively:

```
<div itemscope>
 <p>My name is Elizabeth.</p>
</div>

<section>
 <div itemscope>
 <aside>
 <p>My name is Daniel.</p>
 </aside>
 </div>
</section>
```

Properties generally have values that are strings.

Here the item has three properties:

```
<div itemscope>
 <p>My name is Neil.</p>
 <p>My band is called Four Parts Water.</p>
 <p>I am British.</p>
</div>
```

When a string value is a URL, it is expressed using the `a` element and its `href` attribute, the `img` element and its `src` attribute, or other elements that link to or embed external resources.

In this example, the item has one property, "image", whose value is a URL:

```
<div itemscope>

</div>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<h1 itemscope>
<data itemprop="product-id" value="9678AOU879">The Instigator 2000</data>
</h1>
```

For date- and time-related data, the `time` element and its `datetime` attribute can be used instead.

In this example, the item has one property, "birthday", whose value is a date:

```
<div itemscope>
```

## Too slow? Try reading the multipage copy of

Properties can be declared on the `itemprop` attribute of an `itemscope` element. This example shows how to declare the `name` property.

Items that are associated with the `itemprop` attribute are declared as children of the `itemscope` element.

In this example, the `name` property is declared on the `itemscope` element.

```
<div itemscope>
<p>Name: Amanda</p>
<p>Band: Jazz Band (12 players)</p>
</div>
```

The outer item here has two properties, "name" and "band". The "name" is "Amanda", and the "band" is an item in its own right, with two properties, "name" and "size". The "name" of the band is "Jazz Band", and the "size" is "12".

The outer item in this example is a top-level microdata item.

Properties that are not descendants of the element with the `itemscope` attribute can be associated with the item using the `itemref` attribute. This attribute takes a list of IDs of elements to crawl in addition to crawling the children of the element with the `itemscope` attribute.

This example is the same as the previous one, but all the properties are separated from their items:

```
<div itemscope id="amanda" itemref="a b"></div>
<p id="a">Name: Amanda</p>
<div id="b" itemprop="band" itemscope itemref="c"></div>
<div id="c">
 <p>Band: Jazz Band</p>
 <p>Size: 12 players</p>
</div>
```

This gives the same result as the previous example. The first item has two properties, "name", set to "Amanda", and "band", set to another item. That second item has two further properties, "name", set to "Jazz Band", and "size", set to "12".

An item can have multiple properties with the same name and different values.

This example describes an ice cream, with two flavors:

```
<div itemscope>
<p>Flavors in my favorite ice cream:</p>

 <li itemprop="flavor">Lemon sorbet
 <li itemprop="flavor">Apricot sorbet

</div>
```

This thus results in an item with two properties, both "flavor", having the values "Lemon sorbet" and "Apricot sorbet".

An element introducing a property can also introduce multiple properties at once, to avoid duplication when some of the properties have the same value.

Here we see an item with two properties, "favorite-color" and "favorite-fruit", both set to the value "orange":

```
<div itemscope>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

There is no semantic difference, for instance, between the following two examples:

```
<figure>

 <figcaption>The Castle (1986)
 </figcaption>
</figure>

<meta itemprop="name" content="The Castle">
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Both name forms item. [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

name-value pair with the ~~ut~~ of the document, in the way associated with the

### 5.1.3 Typed items

*This section is non-normative.*

The examples in the previous section show how information could be marked up on a page that doesn't expect its microdata to be reused. Microdata is most useful, though, when it is used in contexts where other authors and readers are able to cooperate to make new uses of the markup.

For this purpose, it is necessary to give each item a type, such as "<http://example.com/person>", or "<http://example.org/cat>", or "<http://band.example.net/>". Types are identified as URLs.

The type for an item is given as the value of an `itemtype` attribute on the same element as the `itemscope` attribute.

Here, the item's type is "<http://example.org/animals#cat>":

```
<section itemscope itemtype="http://example.org/animals#cat">
 <h1 itemprop="name">Hederal</h1>
 <p itemprop="desc">Hederal is a male american domestic shorthair, with a fluffy black fur with white paws and belly.</p>

</section>
```

In this example the "<http://example.org/animals#cat>" item has three properties, a "name" ("Hederal"), a "desc" ("Hederal is..."), and an "img" ("hederal.jpeg").

The type gives the context for the properties, thus selecting a vocabulary: a property named "class" given for an item with the type "<http://census.example/person>" might refer to the economic class of an individual, while a property named "class" given for an item with the type "<http://example.com/school/teacher>" might refer to the classroom a teacher has been assigned. Several types can share a vocabulary. For example, the types "<http://example.org/people/teacher>" and "<http://example.org/people/engineer>" could be defined to use the same vocabulary (though maybe some properties would not be especially useful in both cases, e.g. maybe the "<http://example.org/people/engineer>" type might not typically be used with the "classroom" property). Multiple types defined to use the same vocabulary can be given for a single item by listing the URLs as a space-separated list in the attribute's value. An item cannot be given two types if they do not use the same vocabulary, however.

### 5.1.4 Global identifiers for items

*This section is non-normative.*

Sometimes, an item gives information about a topic that has a global identifier. For example, books can be identified by their ISBN number.

Vocabularies (as identified by the `itemtype` attribute) can be designed such that items get associated with their global identifier in an unambiguous way by expressing the global identifiers as URLs given in an `itemid` attribute.

The exact meaning of the URLs given in `itemid` attributes depends on the vocabulary used.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<dt>Title
<dd itemprop="title">The Reality Dysfunction
<dt>Author
<dd itemprop="author">Peter F. Hamilton
<dt>Publication date
<dd><time itemprop="pubdate" datetime="1996-01-26">26 January 1996</time>
</dl>
```

The "http://vocab.example.net/book" vocabulary in this example would define the itemid attribute takes a urn:  
URL r

## Too slow? Try reading the multipage copy of

**5.1.5 Select the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*This section*

Using microdata to be designed

hers, a vocabulary will need  
ntent re-use easier.

When designing new vocabularies, identifiers can be created either using URLs, or, for properties, as plain words (with no dots or colons). For URLs, conflicts with other vocabularies can be avoided by only using identifiers that correspond to pages that the author has control over.

For instance, if Jon and Adam both write content at example.com, at [http://example.com/~jon/...](http://example.com/~jon/) and [http://example.com/~adam/...](http://example.com/~adam/) respectively, then they could select identifiers of the form "http://example.com/~jon/name" and "http://example.com/~adam/name" respectively.

Properties whose names are just plain words can only be used within the context of the types for which they are intended; properties named using URLs can be reused in items of any type. If an item has no type, and is not part of another item, then if its properties have names that are just plain words, they are not intended to be globally unique, and are instead only intended for limited use. Generally speaking, authors are encouraged to use either properties with globally unique names (URLs) or ensure that their items are typed.

Here, an item is an "http://example.org/animals#cat", and most of the properties have names that are words defined in the context of that type. There are also a few additional properties whose names come from other vocabularies.

```
<section itemscope itemtype="http://example.org/animals#cat">
 <h1 itemprop="name http://example.com/fn">Hedral</h1>
 <p itemprop="desc">Hedral is a male american domestic shorthair, with a fluffy black fur with white paws and belly.</p>

</section>
```

This example has one item with the type "http://example.org/animals#cat" and the following properties:

Property	Value
name	Hedral
http://example.com/fn	Hedral
desc	Hedral is a male american domestic shorthair, with a fluffy black fur with white paws and belly.
http://example.com/color	black
http://example.com/color	white
img	./hedral.jpeg

## 5.1.6 Using the microdata DOM API

*This section is non-normative.*

The microdata becomes even more useful when scripts can use it to expose information to the user, for example offering it in a form that can be used by other applications.

The `document.getItems(typeNames)` method provides access to the top-level microdata items. It returns a `NodeList` containing the items with the specified types, or all types if no argument is specified.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

This sample shows how the `getItems()` method can be used to obtain a list of all the top-level microdata items of a particular type given in the document:

```
var cats = document.getItems("http://example.com/feline");
```

Once an element representing an item has been obtained, its properties can be extracted using the `properties` IDL attribute. This attribute returns an `HTMLPropertiesCollection`, which can be enumerated to go through each element that adds one or more properties to the item. It can also be indexed by name, which will return an object with a list of the elements that add properties with that name.



Each elem

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

"name" property from that

This item.

The `HTMLPr`

The `HTMLPropertiesCollection` object can be used to obtain all the values at once using its `getValues` method, which returns an array of all the values.

In an earlier example, a "http://example.org/animals#cat" item had two "http://example.com/color" values. This script looks up the first such item and then lists all its values.

```
var cat = document.getItems('http://example.org/animals#cat')[0];
var colors = cat.properties['http://example.com/color'].getValues();
var result;
if (colors.length == 0) {
 result = 'Color unknown.';
} else if (colors.length == 1) {
 result = 'Color: ' + colors[0];
} else {
 result = 'Colors:';
 for (var i = 0; i < colors.length; i += 1)
 result += ' ' + colors[i];
}
```

It's also possible to get a list of all the property names using the object's `names` IDL attribute.

This example creates a big list with a nested list for each item on the page, each with all of the property names used in that item.

```
var outer = document.createElement('ul');
var items = document.getItems();
for (var item = 0; item < items.length; item += 1) {
 var itemLi = document.createElement('li');
 var inner = document.createElement('ul');
 for (var name = 0; name < items[item].properties.names.length; name += 1) {
 var propLi = document.createElement('li');
 propLi.appendChild(document.createTextNode(items[item].properties.names[name]));
 inner.appendChild(propLi);
 }
 itemLi.appendChild(inner);
 outer.appendChild(itemLi);
}
document.body.appendChild(outer);
```

If faced with the following from an earlier example:

```
<section itemscope itemtype="http://example.org/animals#cat">
<h1 itemprop="name http://example.com/fn">Hederal</h1>
<p itemprop="desc">Hederal is a male american domestic
shorthair, with a fluffy black fur with white paws and belly.</p>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- - name
  - http://example.com/fn
  - desc
  - http://example.com/color
  - img

(The duplicate occurrence of "http://example.com/color" is not included in the list.)



## 5.2 Encapsulation

### 5.2.1 The `itemscope` attribute

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The microdata specification defines the `itemscope` attribute. Each group of items defined by `itemscope` must support global properties. A property has a global value if it is defined relative to each item in the group.

fied by the item types defined by the item types of the item. own as a property, and each item type has a global value. 1). The names are unordered sets of tokens.

### 5.2.2 Items

Every HTML element may have an `itemscope` attribute specified. The `itemscope` attribute is a boolean attribute.

An element with the `itemscope` attribute specified creates a new **item**, a group of name-value pairs.

Elements with an `itemscope` attribute may have an `itemtype` attribute specified, to give the item types of the item.

The `itemtype` attribute, if specified, must have a value that is an unordered set of unique space-separated tokens that are case-sensitive, each of which is a valid URL that is an absolute URL, and all of which are defined to use the same vocabulary. The attribute's value must have at least one token.

The **item types** of an item are the tokens obtained by splitting the element's `itemtype` attribute's value on spaces. If the `itemtype` attribute is missing or parsing it in this way finds no tokens, the item is said to have no item types.

The item types must all be types defined in applicable specifications and must all be defined to use the same vocabulary.

Except if otherwise specified by that specification, the URLs given as the item types should not be automatically dereferenced.

**Note:** A specification could define that its item type can be dereferenced to provide the user with help information, for example. In fact, vocabulary authors are encouraged to provide useful information at the given URL.

Item types are opaque identifiers, and user agents must not dereference unknown item types, or otherwise deconstruct them, in order to determine how to process items that use them.

The `itemtype` attribute must not be specified on elements that do not have an `itemscope` attribute specified.

An item is said to be a **typed item** when either it has an item type, or it is the value of a property of a typed item. The **relevant types** for a typed item is the item's item types, if it has any, or else is the relevant types of the item for which it is a property's value.

Elements with an `itemscope` attribute and an `itemtype` attribute that references a vocabulary that is defined to support global identifiers for items may also have an `itemid` attribute specified, to give a global identifier for the item, so that it can be related to other items on pages elsewhere on the Web.

The `itemid` attribute, if specified, must have a value that is a valid URL potentially surrounded by spaces.

The **global identifier** of an item is the value of its element's `itemid` attribute, if it has one, resolved relative to the element on which the attribute is specified. If the `itemid` attribute is missing or if resolving it fails, it is said to have no global identifier.

The `itemid` attribute must not be specified on elements that do not have both an `itemscope` attribute and an `itemtype` attribute specified, and must not be specified on elements with an `itemscope` attribute whose `itemtype` attribute specifies a vocabulary that does not support global identifiers for items, as defined by that vocabulary's specification.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Elements with an `itemscope` attribute may have an `itemref` attribute specified, to give a list of additional elements to crawl to find the name-value pairs of the item.

The `itemref` attribute, if specified, must have a value that is an unordered set of unique space-separated tokens that are case-sensitive, consisting of IDs of elements in the same home subtree.

The `itemref` attribute must not be specified on elements that do not have an `itemscope` attribute specified.

**Note:** The `itemref` attribute is not part of the microdata data model. It is merely a syntactic construct to aid authors in adding structure to their documents without changing the document's tree structure. For example, while keeping the

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### `name`

A brief description of the product.

### `scale`

One of "HO", "1", or "Z" (potentially with leading or trailing whitespace), indicating the scale of the product.

### `digital`

If present, one of "Digital", "Delta", or "Systems" (potentially with leading or trailing whitespace) indicating that the product has a digital decoder of the given type.

### `track-type`

For track-specific products, one of "K", "M", "C" (potentially with leading or trailing whitespace) indicating the type of track for which the product is intended.

This vocabulary has four defined item types:

#### <http://md.example.com/loco>

Rolling stock with an engine

#### <http://md.example.com/passengers>

Passenger rolling stock

#### <http://md.example.com/track>

Track pieces

#### <http://md.example.com/lighting>

Equipment with lighting

Each item that uses this vocabulary can be given one or more of these types, depending on what the product is.

Thus, a locomotive might be marked up as:

```
<dl itemscope itemtype="http://md.example.com/loco
 http://md.example.com/lighting">
 <dt>Name:
 <dd itemprop="name">Tank Locomotive (DB 80)
 <dt>Product code:
 <dd itemprop="product-code">33041
 <dt>Scale:
 <dd itemprop="scale">HO
 <dt>Digital:
 <dd itemprop="digital">Delta
</dl>
```

A turnout lantern retrofit kit might be marked up as:

```
<dl itemscope itemtype="http://md.example.com/track
 http://md.example.com/lighting">
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

```
<dd>For retrofitting 2 C Track
turnouts. <meta itemprop="scale" content="HO">
</dl>
```

A passenger car with no lighting might be marked up as:

```
<dl itemscope itemtype="http://md.example.com/passengers">
<dt>Name:
<dd itemprop="name">Express Train Passenger Car (DB Am 203)
```

**Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org**

Grea  
voca  
/web/20130103030411/http:  
//whatwg.org/html

an be taken that results in a  
e.

### 5.2.3 Names: the `itemprop` attribute

Every HTML element may have an `itemprop` attribute specified, if doing so adds one or more properties to one or more items (as defined below).

The `itemprop` attribute, if specified, must have a value that is an unordered set of unique space-separated tokens that are case-sensitive, representing the names of the name-value pairs that it adds. The attribute's value must have at least one token.

Each token must be either:

- If the item is a typed item: a **defined property name** allowed in this situation according to the specification that defines the relevant types for the item, or
- A valid URL that is an absolute URL defined as an item property name allowed in this situation by a vocabulary specification, or
- A valid URL that is an absolute URL, used as a proprietary item property name (i.e. one used by the author for private purposes, not defined in a public specification), or
- If the item is not a typed item: a string that contains no U+002E FULL STOP characters (.) and no U+003A COLON characters (:), used as a proprietary item property name (i.e. one used by the author for private purposes, not defined in a public specification).

Specifications that introduce defined property names must ensure all such property names contain no U+002E FULL STOP characters (.), no U+003A COLON characters (:), and no space characters.

When an element with an `itemprop` attribute adds a property to multiple items, the requirement above regarding the tokens applies for each item individually.

The **property names** of an element are the tokens that the element's `itemprop` attribute is found to contain when its value is split on spaces, with the order preserved but with duplicates removed (leaving only the first occurrence of each name).

Within an item, the properties are unordered with respect to each other, except for properties with the same name, which are ordered in the order they are given by the algorithm that defines the properties of an item.

In the following example, the "a" property has the values "1" and "2", *in that order*, but whether the "a" property comes before the "b" property or not is not important:

```
<div itemscope>
<p itemprop="a">1</p>
<p itemprop="a">2</p>
<p itemprop="b">test</p>
</div>
```

Thus, the following is equivalent:

```
<div itemscope>
<p itemprop="b">test</p>
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
<div itemscope>
<p itemprop="a">1</p>
<p itemprop="b">test</p>
<p itemprop="a">2</p>
</div>
```

And the following:

```
<div id="x">
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 5.2.4 Value

The **property value** of a name-value pair added by an element with an `itemprop` attribute is as given for the first matching case in the following list:

↳ **If the element also has an `itemscope` attribute**

The value is the item created by the element.

↳ **If the element is a `meta` element**

The value is the value of the element's `content` attribute, if any, or the empty string if there is no such attribute.

↳ **If the element is an `audio`, `embed`, `iframe`, `img`, `source`, `track`, or `video` element**

The value is the absolute URL that results from resolving the value of the element's `src` attribute relative to the element at the time the attribute is set, or the empty string if there is no such attribute or if resolving it results in an error.

↳ **If the element is an `a`, `area`, or `link` element**

The value is the absolute URL that results from resolving the value of the element's `href` attribute relative to the element at the time the attribute is set, or the empty string if there is no such attribute or if resolving it results in an error.

↳ **If the element is an `object` element**

The value is the absolute URL that results from resolving the value of the element's `data` attribute relative to the element at the time the attribute is set, or the empty string if there is no such attribute or if resolving it results in an error.

↳ **If the element is a `data` element**

The value is the value of the element's `value` attribute, if it has one, or the empty string otherwise.

↳ **If the element is a `time` element**

The value is the element's `datetime` value.

↳ **Otherwise**

The value is the element's `textContent`.

The **URL property elements** are the `a`, `area`, `audio`, `embed`, `iframe`, `img`, `link`, `object`, `source`, `track`, and `video` elements.

If a property's value, as defined by the property's definition, is an absolute URL, the property must be specified using a URL property element.

**Note: These requirements do not apply just because a property value happens to match the syntax for a URL. They only apply if the property is explicitly defined as taking such a value.**

For example, a book about the first moon landing could be called "mission:moon". A "title" property from a vocabulary that defines a title as being a string would not expect the title to be given in an `a` element, even though it looks like a URL. On the other hand, if there was a (rather narrowly scoped!) vocabulary for "books whose titles look like URLs" which had a "title" property defined to take a URL, then the property *would* expect the title to be given in an `a` element (or one of the other URL property elements), because of the requirement above.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. Let *results*, *memory*, and *pending* be empty lists of elements.
  2. Add the element *root* to *memory*.
  3. Add the child elements of *root*, if any, to *pending*.
  4. If *root* has an `itemref` attribute, split the value of that `itemref` attribute on spaces. For each resulting token *ID*, if there is an element in the home subtree of *root* with the ID *ID*, then add the first such element to *pending*.
  5. *Loop*: If *pending* is empty, jump to the step labeled *end of loop*.
  6. *Rerun*
  7. If *cl*
  8. *Add*
  9. If *cl*
  10. If *cl*
- Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**
11. Return to the step labeled *loop*.
  12. *End of loop*: Sort *results* in tree order.
  13. Return *results*.

A document must not contain any items for which the algorithm to find the properties of an item finds any **microdata errors**.

An item is a **top-level microdata item** if its element does not have an `itemprop` attribute.

All `itemref` attributes in a `Document` must be such that there are no cycles in the graph formed from representing each item in the `Document` as a node in the graph and each property of an item whose value is another item as an edge in the graph connecting those two items.

A document must not contain any elements that have an `itemprop` attribute that would not be found to be a property of any of the items in that document were their properties all to be determined.

In this example, a single license statement is applied to two works, using `itemref` from the items representing the works:

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>Photo gallery</title>
 </head>
 <body>
 <h1>My photos</h1>
 <figure itemscope itemtype="http://n.whatwg.org/work" itemref="licenses">

 <figcaption itemprop="title">The house I found.</figcaption>
 </figure>
 <figure itemscope itemtype="http://n.whatwg.org/work" itemref="licenses">

 <figcaption itemprop="title">The mailbox.</figcaption>
 </figure>
 <footer>
 <p id="licenses">All images licensed under the MIT license.</p>
 </footer>
 </body>
</html>
```

The above results in two items with the type "http://n.whatwg.org/work", one with:

**work**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

<http://www.opensource.org/licenses/mit-license.php>

...and one with:

**work**

images/mailbox.jpeg

**title**

The mailbox.

**license****Too slow? Try reading the multipage copy of****5.2.6 Mic** the spec instead: <https://web.archive.org>Currently, th /web/20130103030411/http:  
with the liter  
such as SV( //whatwg.org/html. This means that attributes  
nents in other namespaces,

Thus, in the following example there is only one item, not two.

```
<p itemscope></p> <!-- this is an item (with no properties and no type) -->
<svg itemscope></svg> <!-- this is not, it's just an svg element with an invalid unknown
attribute -->
```

**5.3 Microdata DOM API***This box is non-normative. Implementation requirements are given below this box.***document.getItems([types])**Returns a `NodeList` of the elements in the `Document` that create items, that are not part of other items, and that are of the types given in the argument, if any are listed.The `types` argument is interpreted as a space-separated list of types.**element.properties**If the element has an `itemscope` attribute, returns an `HTMLPropertiesCollection` object with all the element's properties. Otherwise, an empty `HTMLPropertiesCollection` object.**element.itemValue [= value]**

Returns the element's value.

Can be set, to change the element's value. Setting the value when the element has no `itemprop` attribute or when the element's value is an item throws an `InvalidAccessError` exception.

The `document.getItems(typeNames)` method takes an optional string that contains an unordered set of unique space-separated tokens that are case-sensitive, representing types. When called, the method must return a live `NodeList` object containing all the elements in the document, in tree order, that are each top-level microdata items whose types include all the types specified in the method's argument, having obtained the types by splitting the string on spaces. If there are no tokens specified in the argument, or if the argument is missing, then the method must return a `NodeList` containing all the top-level microdata items in the document. When the method is invoked on a `Document` object again with the same argument, the user agent may return the same object as the object returned by the earlier call. In other cases, a new `NodeList` object must be returned.

The `itemScope` IDL attribute on HTML elements must reflect the `itemscope` content attribute. The `itemType` IDL attribute on HTML elements must reflect the `itemtype` content attribute. The `itemID` IDL attribute on HTML elements must reflect the `itemid` content attribute. The `itemProp` IDL attribute on HTML elements must reflect the `itemprop` content attribute. The `itemRef` IDL attribute on HTML elements must reflect the `itemref` content attribute.

The `properties` IDL attribute on HTML elements must return an `HTMLPropertiesCollection` rooted at the `Document` node, whose filter matches only elements that are the properties of the item created by the element on which the attribute was invoked, while that element is an item, and matches nothing the rest of the time.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**If the element has an `itemscope` attribute**

The attribute must return the element itself on getting and must throw an `InvalidAccessError` exception on setting.

**If the element is a `meta` element**

The attribute must act as it would if it was reflecting the element's `content` `content` attribute.

**If the element is an `audio`, `embed`, `iframe`, `img`, `source`, `track`, or `video` element**

The attribute must act as it would if it was reflecting the element's `src` `content` attribute 

**If the element is a `script` element**

The **Too slow? Try reading the multipage copy of**

**the spec instead: https://web.archive.org**

**/web/20130103030411/http:**

**If the element is a `link` element**

The **//whatwg.org/html**

**If the element is a `time` element**

On getting, if the element has a `datetime` `content` attribute, the IDL attribute must return that `content` attribute's value; otherwise, it must return the element's `textContent`. On setting, the IDL attribute must act as it would if it was reflecting the element's `datetime` `content` attribute.

**Otherwise**

The attribute must act the same as the element's `textContent` attribute.

When the `itemValue` IDL attribute is reflecting a content attribute or acting like the element's `textContent` attribute, the user agent must, on setting, convert the new value to the IDL `DOMString` value before using it according to the mappings described above.

In this example, a script checks to see if a particular element `element` is declaring a particular property, and if it is, it increments a counter:

```
if (element.itemProp.contains('color'))
 count += 1;
```

This script iterates over each of the values of an element's `itemref` attribute, calling a function for each referenced element:

```
for (var index = 0; index < element.itemRef.length; index += 1)
 process(document.getElementById(element.itemRef[index]));
```

## 5.4 Sample microdata vocabularies

The vocabularies in this section are primarily intended to demonstrate how a vocabulary is specified, though they are also usable in their own right.

### 5.4.1 vCard

An item with the item type `http://microformats.org/profile/hcard` represents a person's or organization's contact information.

This vocabulary does not support global identifiers for items.

The following are the type's defined property names. They are based on the vocabulary defined in the vCard 4.0 specification and its extensions, where more information on how to interpret the values can be found. [RFC6350]

**`kind`**

Describes what kind of contact the item represents.

The value must be text that, when compared in a case-sensitive manner, is equal to one of the kind strings.

A single property with the name `kind` may be present within each item with the type `http://microformats.org/profile/hcard`.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Exactly one property with the name `fn` must be present within each item with the type <http://microformats.org/profile/hcard>.

**n**

Gives the structured name of the person or organization.

The value must be an item with zero or more of each of the `family-name`, `given-name`, `additional-name`, `honorific-prefix`, and `honorific-suffix` properties.

Exac<sup>t</sup>

/pr Too slow? Try reading the multipage copy of  
`family-na` the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Given  
The Any of an .

croformats.org

the value of the `n` property

#### **given-name (inside n)**

Gives the given-name of the person.

The value must be text.

Any number of properties with the name `given-name` may be present within the item that forms the value of the `n` property of an item with the type <http://microformats.org/profile/hcard>.

#### **additional-name (inside n)**

Gives the any additional names of the person.

The value must be text.

Any number of properties with the name `additional-name` may be present within the item that forms the value of the `n` property of an item with the type <http://microformats.org/profile/hcard>.

#### **honorific-prefix (inside n)**

Gives the honorific prefix of the person.

The value must be text.

Any number of properties with the name `honorific-prefix` may be present within the item that forms the value of the `n` property of an item with the type <http://microformats.org/profile/hcard>.

#### **honorific-suffix (inside n)**

Gives the honorific suffix of the person.

The value must be text.

Any number of properties with the name `honorific-suffix` may be present within the item that forms the value of the `n` property of an item with the type <http://microformats.org/profile/hcard>.

#### **nickname**

Gives the nickname of the person or organization.

**Note:** The `nickname` is the descriptive name given instead of or in addition to the one belonging to a person, place, or thing. It can also be used to specify a familiar form of a proper name specified by the `fn` or `n` properties.

The value must be text.

Any number of properties with the name `nickname` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### **photo**

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

/profile/hcard.

**bdy**

Gives the birth date of the person or organization.

The value must be a valid date string.

A single property with the name `bdy` may be present within each item with the type <http://microformats.org/profile/hcard>.

**anniversa**

Given the **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

**sex**

Gives the biological sex of the person.

The value must be one of `F`, meaning "female", `M`, meaning "male", `N`, meaning "none or not applicable", `O`, meaning "other", or `U`, meaning "unknown".

A single property with the name `sex` may be present within each item with the type <http://microformats.org/profile/hcard>.

**gender-identity**

Gives the gender identity of the person.

The value must be text.

A single property with the name `gender-identity` may be present within each item with the type <http://microformats.org/profile/hcard>.

**adr**

Gives the delivery address of the person or organization.

The value must be an item with zero or more `type`, `post-office-box`, `extended-address`, `and street-address` properties, and optionally a `locality` property, optionally a `region` property, optionally a `postal-code` property, and optionally a `country-name` property.

If no `type` properties are present within an item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>, then the address type string `work` is implied.

Any number of properties with the name `adr` may be present within each item with the type <http://microformats.org/profile/hcard>.

**type (inside adr)**

Gives the type of delivery address.

The value must be text that, when compared in a case-sensitive manner, is equal to one of the address type strings.

Any number of properties with the name `type` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>, but within each such `adr` property item there must only be one `type` property per distinct value.

**post-office-box (inside adr)**

Gives the post office box component of the delivery address of the person or organization.

The value must be text.

Any number of properties with the name `post-office-box` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The value must be text.

Any number of properties with the name `extended-address` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>.

**Note:** The vCard specification urges authors not to use this field.

#### `street-address` (inside `adr`)



Give

The  
Any  
prop  
The spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

rms the value of an `adr`

#### `locality` (inside `adr`)

Give

The value must be text.

A single property with the name `locality` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>.

#### `region` (inside `adr`)

Gives the region component (e.g. state or province) of the delivery address of the person or organization.

The value must be text.

A single property with the name `region` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>.

#### `postal-code` (inside `adr`)

Gives the postal code component of the delivery address of the person or organization.

The value must be text.

A single property with the name `postal-code` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>.

#### `country-name` (inside `adr`)

Gives the country name component of the delivery address of the person or organization.

The value must be text.

A single property with the name `country-name` may be present within the item that forms the value of an `adr` property of an item with the type <http://microformats.org/profile/hcard>.

#### `tel`

Gives the telephone number of the person or organization.

The value must be either text that can be interpreted as a telephone number as defined in the CCITT specifications E.163 and X.121, or an item with zero or more `type` properties and exactly one `value` property. [E163] [X121]

If no `type` properties are present within an item that forms the value of a `tel` property of an item with the type <http://microformats.org/profile/hcard>, or if the value of such a `tel` property is text, then the telephone type string `voice` is implied.

Any number of properties with the name `tel` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `type` (inside `tel`)

Gives the type of telephone number.

The value must be text that, when compared in a case-sensitive manner, is equal to one of the telephone type strings.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

Gives the actual telephone number of the person or organization.

The value must be text that can be interpreted as a telephone number as defined in the CCITT specifications E.163 and X.121. [E163] [X121]

Exactly one property with the name `value` must be present within the item that forms the value of a `tel` property of an item with the type <http://microformats.org/profile/hcard>.

#### `email`

Give

The **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

<http://microformats.org>

Any

/pr

#### `impp`

Give

//whatwg.org/html

anization.

#### `lang`

Gives a language understood by the person or organization.

The value must be a valid BCP 47 language tag. [BCP47].

Any number of properties with the name `lang` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `tz`

Gives the time zone of the person or organization.

The value must be text and must match the following syntax:

1. Either a U+002B PLUS SIGN character (+) or a U+002D HYPHEN-MINUS character (-).
2. A valid non-negative integer that is exactly two digits long and that represents a number in the range 00..23.
3. A U+003A COLON character (:).
4. A valid non-negative integer that is exactly two digits long and that represents a number in the range 00..59.

Any number of properties with the name `tz` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `geo`

Gives the geographical position of the person or organization.

The value must be text and must match the following syntax:

1. Optionally, either a U+002B PLUS SIGN character (+) or a U+002D HYPHEN-MINUS character (-).
2. One or more ASCII digits.
3. Optionally\*, a U+002E FULL STOP character (.) followed by one or more ASCII digits.
4. A U+003B SEMICOLON character (;).
5. Optionally, either a U+002B PLUS SIGN character (+) or a U+002D HYPHEN-MINUS character (-).
6. One or more ASCII digits.
7. Optionally\*, a U+002E FULL STOP character (.) followed by one or more ASCII digits.

The optional components marked with an asterisk (\*) should be included, and should have six digits each.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Any number of properties with the name `geo` may be present within each item with the type <http://microformats.org/profile/hcard>.

**title**

Gives the job title, functional position or function of the person or organization.

The value must be text.

Any number of properties with the name `title` may be present within each item with the type <http://microformats.org/profile/hcard>.

**role**

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

Given the URL [/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html):

Any number of properties with the name `role` may be present within each item with the type <http://microformats.org/profile/hcard>.

**logo**

Gives the logo of the person or organization.

The value must be an absolute URL.

Any number of properties with the name `logo` may be present within each item with the type <http://microformats.org/profile/hcard>.

**agent**

Gives the contact information of another person who will act on behalf of the person or organization.

The value must be either an item with the type <http://microformats.org/profile/hcard>, or an absolute URL, or text.

Any number of properties with the name `agent` may be present within each item with the type <http://microformats.org/profile/hcard>.

**org**

Gives the name and units of the organization.

The value must be either text or an item with one `organization-name` property and zero or more `organization-unit` properties.

Any number of properties with the name `org` may be present within each item with the type <http://microformats.org/profile/hcard>.

**organization-name (inside org)**

Gives the name of the organization.

The value must be text.

Exactly one property with the name `organization-name` must be present within the item that forms the value of an `org` property of an item with the type <http://microformats.org/profile/hcard>.

**organization-unit (inside org)**

Gives the name of the organization unit.

The value must be text.

Any number of properties with the name `organization-unit` may be present within the item that forms the value of the `org` property of an item with the type <http://microformats.org/profile/hcard>.

**member**

Gives a URL that represents a member of the group.

The value must be an absolute URL.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Gives a relationship to another entity.

The value must be an item with one `url` property and one `rel` properties.

Any number of properties with the name `related` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `url` (inside `related`)

Gives the URL for the related entity.



The

Exa  
iter  
`rel` (inside  
Give  
The

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

related property of an

relationship strings.

Exactly one property with the name `rel` must be present within the item that forms the value of an `related` property of an item with the type <http://microformats.org/profile/hcard>.

#### `categories`

Gives the name of a category or tag that the person or organization could be classified as.

The value must be text.

Any number of properties with the name `categories` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `note`

Gives supplemental information or a comment about the person or organization.

The value must be text.

Any number of properties with the name `note` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `rev`

Gives the revision date and time of the contact information.

The value must be text that is a valid global date and time string.

**Note:** The value distinguishes the current revision of the information for other renditions of the information.

Any number of properties with the name `rev` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `sound`

Gives a sound file relating to the person or organization.

The value must be an absolute URL.

Any number of properties with the name `sound` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `uid`

Gives a globally unique identifier corresponding to the person or organization.

The value must be text.

A single property with the name `uid` may be present within each item with the type <http://microformats.org/profile/hcard>.

#### `url`

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

/profile/hcard.

The **kind strings** are:

***individual***

Indicates a single entity (e.g. a person).

***group***

Indicates multiple entities (e.g. a mailing list).



***org***

Indi **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

***location***

Indi **the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The ***address***:

***home***

Indicates a delivery address for a residence.

***work***

Indicates a delivery address for a place of work.

The **telephone type strings** are:

***home***

Indicates a residential number.

***work***

Indicates a telephone number for a place of work.

***text***

Indicates that the telephone number supports text messages (SMS).

***voice***

Indicates a voice telephone number.

***fax***

Indicates a facsimile telephone number.

***cell***

Indicates a cellular telephone number.

***video***

Indicates a video conferencing telephone number.

***pager***

Indicates a paging device telephone number.

***textphone***

Indicates a telecommunication device for people with hearing or speech difficulties.

The **relationship strings** are:

***emergency***

An emergency contact.

***agent***

Another entity that acts on behalf of this entity.

***contact***

***acquaintance***

***friend***

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*neighbor**child**parent**sibling**spouse**kin**muse**crush**date**sweetheart**me*

Has



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http:](https://web.archive.org/web/20130103030411/http://www.whatwg.org/)

#### 5.4.1.1 Co //whatwg.org/html

Given a list of .

vCard data represented by

**those nodes** (only the first vCard is returned):

1. If none of the nodes in *nodes* are items with the item type `http://microformats.org/profile/hcard`, then there is no vCard. Abort the algorithm, returning nothing.
2. Let *node* be the first node in *nodes* that is an item with the item type `http://microformats.org/profile/hcard`.
3. Let *output* be an empty string.
4. Add a vCard line with the type "BEGIN" and the value "VCARD" to *output*.
5. Add a vCard line with the type "PROFILE" and the value "VCARD" to *output*.
6. Add a vCard line with the type "VERSION" and the value "4.0" to *output*.
7. Add a vCard line with the type "SOURCE" and the result of escaping the vCard text string that is the document's address as the value to *output*.
8. If the *title* element is not null, add a vCard line with the type "NAME" and with the result of escaping the vCard text string obtained from the *textContent* of the *title* element as the value to *output*.
9. Let *sex* be the empty string.
10. Let *gender-identity* be the empty string.
11. For each element *element* that is a property of the item *node*: for each name *name* in *element*'s property names, run the following substeps:
  1. Let *parameters* be an empty set of name-value pairs.
  2. Run the appropriate set of substeps from the following list. The steps will set a variable *value*, which is used in the next step.
 

If the property's value is an item *subitem* and *name* is n

    1. Let *value* be the empty string.
    2. Append to *value* the result of collecting the first vCard subproperty named *family-name* in *subitem*.
    3. Append a U+003B SEMICOLON character (;) to *value*.
    4. Append to *value* the result of collecting the first vCard subproperty named *given-name* in *subitem*.
    5. Append a U+003B SEMICOLON character (;) to *value*.
    6. Append to *value* the result of collecting the first vCard subproperty named *additional-name* in *subitem*.
    7. Append a U+003B SEMICOLON character (;) to *value*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

10. Append to *value* the result of collecting the first vCard subproperty named `honorific-suffix` in *subitem*.

**If the property's value is an item *subitem* and *name* is `adr`**

1. Let *value* be the empty string.
2. Append to *value* the result of collecting vCard subproperties named `post-office-box` in *subitem*.
3. Append a U+003B SEMICOLON character (;) to *value*.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`ended-address` in *subitem*.`set-address` in *subitem*.`ad locality` in *subitem*.

9. Append a U+003B SEMICOLON character (;) to *value*.
10. Append to *value* the result of collecting the first vCard subproperty named `region` in *subitem*.
11. Append a U+003B SEMICOLON character (;) to *value*.
12. Append to *value* the result of collecting the first vCard subproperty named `postal-code` in *subitem*.
13. Append a U+003B SEMICOLON character (;) to *value*.
14. Append to *value* the result of collecting the first vCard subproperty named `country-name` in *subitem*.
15. If there is a property named `type` in *subitem*, and the first such property has a value that is not an item and whose value consists only of alphanumeric ASCII characters, then add a parameter named "TYPE" whose value is the value of that property to *parameters*.

**If the property's value is an item *subitem* and *name* is `org`**

1. Let *value* be the empty string.
2. Append to *value* the result of collecting the first vCard subproperty named `organization-name` in *subitem*.
3. For each property named `organization-unit` in *subitem*, run the following steps:
  1. If the value of the property is an item, then skip this property.
  2. Append a U+003B SEMICOLON character (;) to *value*.
  3. Append the result of escaping the vCard text string given by the value of the property to *value*.

**If the property's value is an item *subitem* with the item type <http://microformats.org/profile/hcard> and *name* is `related`**

1. Let *value* be the empty string.
2. If there is a property named `url` in *subitem*, and its element is a URL property element, then append the result of escaping the vCard text string given by the value of the first such property to *value*, and add a parameter with the name "VALUE" and the value "URI" to *parameters*.
3. If there is a property named `rel` in *subitem*, and the first such property has a value that is not an item and whose value consists only of alphanumeric ASCII characters, then add a parameter named "RELATION" whose value is the value of that property to *parameters*.

**If the property's value is an item and *name* is none of the above**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**If the property's value is not an item and its *name* is `sex`**If this is the first such property to be found, set `sex` to the property's value.**If the property's value is not an item and its *name* is `gender-identity`**If this is the first such property to be found, set `gender-identity` to the property's value.**Otherwise (the property's value is not an item)**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

→ name "VALUE" and the

te string, add a parameter

tring, add a parameter with

5. Prefix every U+005C REVERSE SOLIDUS character (\) in *value* with another U+005C REVERSE SOLIDUS character (\).
6. Prefix every U+002C COMMA character (,) in *value* with a U+005C REVERSE SOLIDUS character (\).
7. Unless *name* is `geo`, prefix every U+003B SEMICOLON character (;) in *value* with a U+005C REVERSE SOLIDUS character (\).
8. Replace every U+000D CARRIAGE RETURN U+000A LINE FEED character pair (CRLF) in *value* with a U+005C REVERSE SOLIDUS character (\) followed by a U+006E LATIN SMALL LETTER N character (n).
9. Replace every remaining U+000D CARRIAGE RETURN (CR) or U+000A LINE FEED (LF) character in *value* with a U+005C REVERSE SOLIDUS character (\) followed by a U+006E LATIN SMALL LETTER N character (n).

3. Add a vCard line with the type *name*, the parameters *parameters*, and the value *value* to *output*.12. If either of `sex` or `gender-identity` have a value that is not the empty string, add a vCard line with the type "GENDER" and the value consisting of the concatenation of `sex`, a U+003B SEMICOLON character (;), and `gender-identity` to *output*.13. Add a vCard line with the type "END" and the value "VCARD" to *output*.

When the above algorithm says that the user agent is to **add a vCard line** consisting of a type *type*, optionally some parameters, and a value *value* to a string *output*, it must run the following steps:

1. Let *line* be an empty string.
2. Append *type*, converted to ASCII uppercase, to *line*.
3. If there are any parameters, then for each parameter, in the order that they were added, run these substeps:
  1. Append a U+003B SEMICOLON character (;) to *line*.
  2. Append the parameter's name to *line*.
  3. Append a U+003D EQUALS SIGN character (=) to *line*.
  4. Append the parameter's value to *line*.
4. Append a U+003A COLON character (:) to *line*.
5. Append *value* to *line*.
6. Let *maximum length* be 75.
7. If and while *line* is longer than *maximum length* Unicode code points long, run the following substeps:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4. Append a U+000A LINE FEED character (LF) to *output*.

5. Append a U+0020 SPACE character to *output*.

6. Let *maximum length* be 74.

8. Append (what remains of) *line* to *output*.

9. Append a U+000D CARRIAGE RETURN character (CR) to *output*.

10. App

When the st  
user agent r  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. Let
2. For

2. If this is not the first property named *subname* in *subitem* (ignoring any that were skipped by the previous step), then append a U+002C COMMA character (,) to *value*.

3. Append the result of escaping the vCard text string given by the value of the property to *value*.

3. Return *value*.

When the steps above require the user agent to obtain the result of **collecting the first vCard subproperty** named *subname* in *subitem*, the user agent must run the following steps:

1. If there are no properties named *subname* in *subitem*, then abort these substeps, returning the empty string.
2. If the value of the first property named *subname* in *subitem* is an item, then abort these substeps, returning the empty string.
3. Return the result of escaping the vCard text string given by the value of the first property named *subname* in *subitem*.

When the above algorithms say the user agent is to **escape the vCard text string** *value*, the user agent must use the following steps:

1. Prefix every U+005C REVERSE SOLIDUS character (\) in *value* with another U+005C REVERSE SOLIDUS character (\).
2. Prefix every U+002C COMMA character (,) in *value* with a U+005C REVERSE SOLIDUS character (\).
3. Prefix every U+003B SEMICOLON character (;) in *value* with a U+005C REVERSE SOLIDUS character (\).
4. Replace every U+000D CARRIAGE RETURN U+000A LINE FEED character pair (CRLF) in *value* with a U+005C REVERSE SOLIDUS character (\) followed by a U+006E LATIN SMALL LETTER N character (n).
5. Replace every remaining U+000D CARRIAGE RETURN (CR) or U+000A LINE FEED (LF) character in *value* with a U+005C REVERSE SOLIDUS character (\) followed by a U+006E LATIN SMALL LETTER N character (n).
6. Return the mutated *value*.

**Note:** This algorithm can generate invalid vCard output, if the input does not conform to the rules described for the <http://microformats.org/profile/hcard> item type and defined property names.

#### 5.4.1.2 Examples

This section is non-normative.

Here is a long example vCard for a fictional character called "Jack Bauer":

```
<section id="jack" itemscope itemtype="http://microformats.org/profile/hcard">
 <h1 itemprop="fn">

 Jack
 Bauer

```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
</p>
<p>

 10201 W. Pico Blvd.

 Los Angeles,
 CA
 90064

 United States


```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```
>e">work
.">I'm on Wikipedia
so you can leave a message on my user talk page.
Jack Bauer Facts
<li itemprop="email">j.bauer@la.ctu.gov.invalid
<li itemprop="tel" itemscope>
 +1 (310) 555 3781
 <meta itemprop="type" content="cell">mobile phone

<ins datetime="2008-07-20 21:00:00+01:00">
 <meta itemprop="rev" content="2008-07-20 21:00:00+01:00">
 <p itemprop="tel" itemscope>Update!
 My new home phone number is
 01632 960 123.</p>
</ins>
</section>
```

The odd line wrapping is needed because newlines are meaningful in microdata: newlines would be preserved in a conversion to, for example, the vCard format.

This example shows a site's contact details (using the `address` element) containing an address with two street components:

```
<address itemscope itemtype="http://microformats.org/profile/hcard">
 <strong itemprop="fn">Alfred
 Person

 1600 Amphitheatre Parkway

 Building 43, Second Floor

 Mountain View,
 CA 94043

</address>
```

The vCard vocabulary can be used to just mark up people's names:

```


 George Washington

```

This creates a single item with a two name-value pairs, one with the name "fn" and the value "George Washington", and the other with the name "n" and a second item as its value, the second item having the two name-value pairs "given-name" and "family-name" with the values "George" and "Washington" respectively. This is defined to map to the following vCard:

BEGIN:VCARD

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

END:VCARD

### 5.4.2 vEvent

An item with the item type <http://microformats.org/profile/hcalendar#vevent> represents an event.

This vocabulary does not support global identifiers for items.

The following items may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>, where more information can be found in the [HTML5 calendar specification](#), where

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

**Note: Complete** [The spec cannot express a complete vEvent](#)  
**attach** [Gives the URL of the attached file.](#)  
The value must be an absolute URL.

Any number of properties with the name `attach` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### categories

Gives the name of a category or tag that the event could be classified as.

The value must be text.

Any number of properties with the name `categories` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### class

Gives the access classification of the information regarding the event.

The value must be text with one of the following values:

- public
- private
- confidential

**⚠ Warning! This is merely advisory and cannot be considered a confidentiality measure.**

A single property with the name `class` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### comment

Gives a comment regarding the event.

The value must be text.

Any number of properties with the name `comment` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### description

Gives a detailed description of the event.

The value must be text.

A single property with the name `description` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### geo

Gives the geographical position of the event.

The value must be text and must match the following syntax:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4. A U+003B SEMICOLON character (;).
5. Optionally, either a U+002B PLUS SIGN character (+) or a U+002D HYPHEN-MINUS character (-).
6. One or more ASCII digits.
7. Optionally\*, a U+002E FULL STOP character (.) followed by one or more ASCII digits.

The optional components marked with an asterisk (\*) should be included, and should have six digits each.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

, in decimal degrees.  
or negative real  
or as a positive or

A single property with the name `location` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>:

<http://microformats.org/profile>

/hc //whatwg.org/html

#### `location`

Gives the location of the event.

The value must be text.

A single property with the name `location` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### `resources`

Gives a resource that will be needed for the event.

The value must be text.

Any number of properties with the name `resources` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### `status`

Gives the confirmation status of the event.

The value must be text with one of the following values:

- tentative
- confirmed
- cancelled

A single property with the name `status` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### `summary`

Gives a short summary of the event.

The value must be text.

User agents should replace U+000A LINE FEED (LF) characters in the value by U+0020 SPACE characters when using the value.

A single property with the name `summary` may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

#### `dtend`

Gives the date and time by which the event ends.

If the property with the name `dtend` is present within an item with the type <http://microformats.org/profile/hcalendar#vevent> that has a property with the name `dtstart` whose value is a valid date string, then the value of the property with the name `dtend` must be text that is a valid date string also. Otherwise, the value of the property must be text that is a valid global date and time string.

In either case, the value be later in time than the value of the `dtstart` property of the same item.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

/profile/hcalendar#vevent, so long as that <http://microformats.org/profile/hcalendar#vevent> does not have a property with the name duration.

### **dtstart**

Gives the date and time at which the event starts.

The value must be text that is either a valid date string or a valid global date and time string.

Exactly one property with the name dtstart must be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

**duration** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Given

The

The

A si

nce

/profile/ncalendar#vevent, so long as that <http://microformats.org/profile/ncalendar#vevent> does not have a property with the name dtend.

### **transp**

Gives whether the event is to be considered as consuming time on a calendar, for the purpose of free-busy time searches.

The value must be text with one of the following values:

- opaque
- transparent

A single property with the name transp may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

### **contact**

Gives the contact information for the event.

The value must be text.

Any number of properties with the name contact may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

### **url**

Gives a URL for the event.

The value must be an absolute URL.

A single property with the name url may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

### **uid**

Gives a globally unique identifier corresponding to the event.

The value must be text.

A single property with the name uid may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

### **exdate**

Gives a date and time at which the event does not occur despite the recurrence rules.

The value must be text that is either a valid date string or a valid global date and time string.

Any number of properties with the name exdate may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.

### **exrule**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

<http://microformats.org/profile/hcalendar#vevent>.**rdate**

Gives a date and time at which the event recurs.

The value must be text that is one of the following:

- A valid date string.
- A valid global date and time string.



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

by a second valid global

by a valid vevent duration

Any  
/pr<http://microformats.org>**rrule**

Gives a rule for finding dates and times at which the event occurs.

The value must be text that matches the RECUR value type defined in the iCalendar specification. [RFC2445]

Any number of properties with the name rrule may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.**created**

Gives the date and time at which the event information was first created in a calendaring system.

The value must be text that is a valid global date and time string.

A single property with the name created may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.**last-modified**

Gives the date and time at which the event information was last modified in a calendaring system.

The value must be text that is a valid global date and time string.

A single property with the name last-modified may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.**sequence**

Gives a revision number for the event information.

The value must be text that is a valid non-negative integer.

A single property with the name sequence may be present within each item with the type <http://microformats.org/profile/hcalendar#vevent>.A string is a **valid vevent duration string** if it matches the following pattern:

1. A U+0050 LATIN CAPITAL LETTER P character (P).
2. One of the following:
  - A valid non-negative integer followed by a U+0057 LATIN CAPITAL LETTER W character (W). The integer represents a duration of that number of weeks.
  - At least one, and possibly both in this order, of the following:
    1. A valid non-negative integer followed by a U+0044 LATIN CAPITAL LETTER D character (D). The integer represents a duration of that number of days.
    2. A U+0054 LATIN CAPITAL LETTER T character (T) followed by any one of the following, or the first and second of the following in that order, or the second and third of the following in that order, or all three of the following in this order:
      1. A valid non-negative integer followed by a U+0048 LATIN CAPITAL LETTER H character (H). The integer represents a duration of that number of hours.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

#### 5.4.2.1 Conversion to iCalendar

Given a list of nodes *nodes* in a Document, a user agent must run the following algorithm to **extract any vEvent data represented by those nodes**:

1. If none of the nodes in *nodes* are items with the type `http://microformats.org/profile/hcalendar#vevent`, then there is no vEvent data. Abort the algorithm, returning nothing.
2. Let *output* be an empty string
3. Add **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html** representing the user agent to *output*
4. Add *outp*
5. Add */web/20130103030411/http://whatwg.org/html*
6. For each item *node* in *nodes*:
  - run the following steps:

1. Add an iCalendar line with the type "BEGIN" and the value "VEVENT" to *output*.
2. Add an iCalendar line with the type "DTSTAMP" and a value consisting of an iCalendar DATE-TIME string representing the current date and time, with the annotation "VALUE=DATE-TIME", to *output*. [RFC2445]
3. For each element *element* that is a property of the item *node*: for each name *name* in *element*'s property names, run the appropriate set of substeps from the following list:

**If the property's value is an item**

Skip the property.

**If the property is dtend**

**If the property is dtstart**

**If the property is exdate**

**If the property is rdate**

**If the property is created**

**If the property is last-modified**

Let *value* be the result of stripping all U+002D HYPHEN-MINUS (-) and U+003A COLON (:) characters from the property's value.

If the property's value is a valid date string then add an iCalendar line with the type *name* and the value *value* to *output*, with the annotation "VALUE=DATE".

Otherwise, if the property's value is a valid global date and time string then add an iCalendar line with the type *name* and the value *value* to *output*, with the annotation "VALUE=DATE-TIME".

Otherwise skip the property.

**Otherwise**

Add an iCalendar line with the type *name* and the property's value to *output*.

4. Add an iCalendar line with the type "END" and the value "VEVENT" to *output*.

7. Add an iCalendar line with the type "END" and the value "VCALENDAR" to *output*.

When the above algorithm says that the user agent is to **add an iCalendar line** consisting of a type *type*, a value *value*, and optionally an annotation, to a string *output*, it must run the following steps:

1. Let *line* be an empty string.
2. Append *type*, converted to ASCII uppercase, to *line*.
3. If there is an annotation:

1. Append a U+003B SEMICOLON character (;) to *line*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

6. Prefix every U+002C COMMA character (,) in *value* with a U+005C REVERSE SOLIDUS character (\).
7. Prefix every U+003B SEMICOLON character (;) in *value* with a U+005C REVERSE SOLIDUS character (\).
8. Replace every U+000D CARRIAGE RETURN U+000A LINE FEED character pair (CRLF) in *value* with a U+005C REVERSE SOLIDUS character (\) followed by a U+006E LATIN SMALL LETTER N character (n).
9. Replace every remaining U+000D CARRIAGE RETURN (CR) or U+000A LINE FEED (LF) character in *value* with a U+005C REVERSE SOLIDUS character (\) followed by a U+006E LATIN SMALL LETTER N character (n).

10. App

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

11. Let

12. If ar

ps:

3. Append a U+000D CARRIAGE RETURN character (CR) to *output*.
4. Append a U+000A LINE FEED character (LF) to *output*.
5. Append a U+0020 SPACE character to *output*.
6. Let *maximum length* be 74.

13. Append (what remains of) *line* to *output*.14. Append a U+000D CARRIAGE RETURN character (CR) to *output*.15. Append a U+000A LINE FEED character (LF) to *output*.

**Note: This algorithm can generate invalid iCalendar output, if the input does not conform to the rules described for the <http://microformats.org/profile/hcalendar#vevent> item type and defined property names.**

#### 5.4.2.2 Examples

This section is non-normative.

Here is an example of a page that uses the vEvent vocabulary to mark up an event:

```
<body itemscope itemtype="http://microformats.org/profile/hcalendar#vevent">
...
<h1 itemprop="summary">Bluesday Tuesday: Money Road</h1>
...
<time itemprop="dtstart" datetime="2009-05-05T19:00:00Z">May 5th @ 7pm</time>
(until <time itemprop="dtend" datetime="2009-05-05T21:00:00Z">9pm</time>)
...
<a href="http://livebrum.co.uk/2009/05/05/bluesday-tuesday-money-road"
 rel="bookmark" itemprop="url">Link to this page
...
<p>Location: The RoadHouse</p>
...
<p><input type="button" value="Add to Calendar"
 onclick="location = getCalendar(this)"></p>
...
<meta itemprop="description" content="via livebrum.co.uk">
</body>
```

The "getCalendar ()" method could look like this:

```
function getCalendar(node) {
 // This function assumes the content is valid.
 // It is not a compliant implementation of the algorithm for extracting vEvent data.
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```

 return;
 }
 var stamp = new Date();
 var stampString = '' + stamp.getUTCFullYear() + (stamp.getUTCMonth() + 1) +
 stamp.getUTCDate() + 'T' +
 stamp.getUTCHours() + stamp.getUTCMinutes() +
 stamp.getUTCSeconds() + 'Z';
 var calendar = 'BEGIN:VCALENDAR\r\nPRODID:HTML\r\nVERSION:2.0\r\nBEGIN:VEVENT
\r\nDTSTAMP:' + stampString + '\r\n':

```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```

 parameters = ';VALUE=DATE-TIME';
 } else {
 value = value.replace(/\n/g, '\\n');
 value = value.replace(/;/g, '\\;');
 value = value.replace(//g, '\\,');
 value = value.replace(/\n/g, '\\n');
 }
 for (var nameIndex = 0; nameIndex < prop.itemProp.length; nameIndex += 1) {
 var name = prop.itemProp[nameIndex];
 if (!name.match(/:/) && !name.match(/\./))
 calendar += name.toUpperCase() + parameters + ':' + value + '\r\n';
 }
}
calendar += 'END:VEVENT\r\nEND:VCALENDAR\r\n';
return 'data:text/calendar;component=event,' + encodeURI(calendar);
}

```

The same page could offer some markup, such as the following, for copy-and-pasting into blogs:

```

<div itemscope itemtype="http://microformats.org/profile/hcalendar#event">
<p>I'm going to
<strong itemprop="summary">Bluesday Tuesday: Money Road,
<time itemprop="dtstart" datetime="2009-05-05T19:00:00Z">May 5th at 7pm</time>
to <time itemprop="dtend" datetime="2009-05-05T21:00:00Z">9pm</time>,
at The RoadHouse!</p>
<p><a href="http://livebrum.co.uk/2009/05/05/bluesday-tuesday-money-road"
 itemprop="url">See this event on livebrum.co.uk.</p>
<meta itemprop="description" content="via livebrum.co.uk">
</div>

```

### 5.4.3 Licensing works

An item with the item type `http://n.whatwg.org/work` represents a work (e.g. an article, an image, a video, a song, etc). This type is primarily intended to allow authors to include licensing information for works.

The following are the type's defined property names.

#### `work`

Identifies the work being described.

The value must be an absolute URL.

Exactly one property with the name `work` must be present within each item with the type `http://n.whatwg.org/work`.

#### `title`

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Gives the name or contact information of one of the authors or creators of the work.

The value must be either an item with the type `http://microformats.org/profile/hcard`, or text.

Any number of properties with the name `author` may be present within each item with the type `http://n.whatwg.org/work`.

#### **license**

Identifies one of the licenses under which the work is available.



The  
Any  
/wc

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

<http://n.whatwg.org>

#### **5.4.3.1 Ex**

This section

This example shows an embedded image entitled *My Pond*, licensed under the Creative Commons Attribution-Share Alike 3.0 United States License and the MIT license simultaneously.

```
<figure itemscope itemtype="http://n.whatwg.org/work">

 <figcaption>
 <p><cite itemprop="title">My Pond</cite></p>
 <p><small>Licensed under the <a itemprop="license"
 href="http://creativecommons.org/licenses/by-sa/3.0/us/">Creative
 Commons Attribution-Share Alike 3.0 United States License
 and the <a itemprop="license"
 href="http://www.opensource.org/licenses/mit-license.php">MIT
 license.</small>
 </figcaption>
 </figure>
```

## 5.5 Converting HTML to other formats

### 5.5.1 JSON

Given a list of nodes `nodes` in a Document, a user agent must run the following algorithm to **extract the microdata from those nodes into a JSON form**:

1. Let `result` be an empty object.
2. Let `items` be an empty array.
3. For each `node` in `nodes`, check if the element is a top-level microdata item, and if it is then get the object for that element and add it to `items`.
4. Add an entry to `result` called "items" whose value is the array `items`.
5. Return the result of serializing `result` to JSON in the shortest possible way (meaning no whitespace between tokens, no unnecessary zero digits in numbers, and only using Unicode escapes in strings for characters that do not have a dedicated escape sequence), and with a lowercase "e" used, when appropriate, in the representation of any numbers. [JSON]

**Note:** This algorithm returns an object with a single property that is an array, instead of just returning an array, so that it is possible to extend the algorithm in the future if necessary.

When the user agent is to **get the object** for an item `item`, optionally with a list of elements `memory`, it must run the following substeps:

1. Let `result` be an empty object.
2. Add `item` to `memory`.

3,445 captures

5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB  
◀ 03 ▶  
2012 2013 2014

About this capture

5. Let *properties* be an empty object.
6. For each element *element* that has one or more property names and is one of the properties of the item *item*, in the order those elements are given by the algorithm that returns the properties of an item, run the following substeps:
  1. Let *value* be the property value of *element*.
  2. If *value* is an item, then: If *value* is in *memory*, then let *value* be the string "ERROR". Otherwise, get the object for *value*, passing a copy of *memory*, and then replace *value* with the object returned from those steps.

- Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

*properties* whose value is an

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## 6.1 Browsing contexts

A **browsing context** is an environment in which `Document` objects are presented to the user.

**Note:** A tab or window in a Web browser typically contains a browsing context, as does an `iframe` or `frameset`.

Each browsing context has a corresponding `WindowProxy` object.



A browsing context will present. context is the `createDoc`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

presented, is presenting, or `Document`'s browsing `ed` using an API such as

Each `Docur` context's ac

everything to the browsing

**Note:** In general, there is a 1-to-1 mapping from the `window` object to the `Document` object. There are two exceptions. First, a `window` can be reused for the presentation of a second `Document` in the same browsing context, such that the mapping is then 1-to-2. This occurs when a browsing context is navigated from the initial `about:blank` Document to another, with replacement enabled. Second, a `Document` can end up being reused for several `window` objects when the `document.open()` method is used, such that the mapping is then many-to-1.

**Note:** A `Document` does not necessarily have a browsing context associated with it. In particular, data mining tools are likely to never instantiate browsing contexts.

A browsing context can have a **creator browsing context**, the browsing context that was responsible for its creation. If a browsing context has a parent browsing context, then that is its creator browsing context. Otherwise, if the browsing context has an opener browsing context, then that is its creator browsing context. Otherwise, the browsing context has no creator browsing context.

If a browsing context *A* has a creator browsing context, then the `Document` that was the active document of that creator browsing context at the time *A* was created is the **creator Document**.

When a browsing context is first created, it must be created with a single `Document` in its session history, whose address is `about:blank`, which is marked as being an HTML document, whose character encoding is UTF-8, and which is both ready for post-load tasks and completely loaded immediately. The `Document` must have a single child `html` node, which itself has a single child `body` node. As soon as this `Document` is created, the user agent must implement the sandboxing for it.

**Note:** If the browsing context is created specifically to be immediately navigated, then that initial navigation will have replacement enabled.

The origin and effective script origin of the `about:blank` Document are set when the `Document` is created. If the new browsing context has a creator browsing context, then the origin of the `about:blank` Document is an alias to the origin of the creator `Document` and the effective script origin of the `about:blank` Document is initially an alias to the effective script origin of the creator `Document`. Otherwise, the origin of the `about:blank` Document is a globally unique identifier assigned when the new browsing context is created and the effective script origin of the `about:blank` Document is initially an alias to its origin.

### 6.1.1 Nested browsing contexts

Certain elements (for example, `iframe` elements) can instantiate further browsing contexts. These are called **nested browsing contexts**. If a browsing context *P* has a `Document` *D* with an element *E* that nests another browsing context *C* inside it, then *C* is said to be **nested through** *D*, and *E* is said to be the **browsing context container** of *C*. If the browsing context container element *E* is in the `Document` *D*, then *P* is said to be the **parent browsing context** of *C* and *C* is said to be a **child browsing context** of *P*. Otherwise, the nested browsing context *C* has no parent browsing context.

A browsing context *A* is said to be an **ancestor** of a browsing context *B* if there exists a browsing context *A'* that is a child browsing context of *A* and that is itself an ancestor of *B*, or if there is a browsing context *P* that is a child browsing context of *A* and that is the parent browsing context of *B*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The list of the descendant browsing contexts of a Document  $d$  is the (ordered) list returned by the following algorithm:

1. Let  $list$  be an empty list.
2. For each child browsing context of  $d$  that is nested through an element that is in the Document  $d$ , in the tree order of the elements nesting those browsing contexts, run these substeps:
  1. Append that child browsing context to the list  $list$ .
  2. Append the list of the descendant browsing contexts of the active document of  $\times$  child browsing context to the list

### 3. Ret **Too slow? Try reading the multipage copy of**

A Document level browsi  
Because the context. Use //whatwg.org/html themselves fu

browsing context is a top-itself fully active.

ent in their parent browsing 1 Documents that are not

A nested browsing context can have a seamless browsing context flag set, if it is embedded through an `iframe` element with a `seamless` attribute.

A nested browsing context can be put into a **delaying load events mode**. This is used when it is navigated, to delay the load event of the browsing context container `iframe` element before the new Document is created.

The **document family** of a browsing context consists of the union of all the Document objects in that browsing context's session history and the document families of all those Document objects. The document family of a Document object consists of the union of all the document families of the browsing contexts that are nested through the Document object.

#### 6.1.1.1 Navigating nested browsing contexts in the DOM

This box is non-normative. Implementation requirements are given below this box.

##### `window.top`

Returns the WindowProxy for the top-level browsing context.

##### `window.parent`

Returns the WindowProxy for the parent browsing context.

##### `window.frameElement`

Returns the Element for the browsing context container.

Returns null if there isn't one.

Throws a `SecurityError` exception in cross-origin situations.

The `top` IDL attribute on the Window object of a Document in a browsing context  $b$  must return the WindowProxy object of its top-level browsing context (which would be its own WindowProxy object if it was a top-level browsing context itself), if it has one, or its own WindowProxy object otherwise (e.g. if it was a detached nested browsing context).

The `parent` IDL attribute on the Window object of a Document in a browsing context  $b$  must return the WindowProxy object of the parent browsing context, if there is one (i.e. if  $b$  is a child browsing context), or the WindowProxy object of the browsing context  $b$  itself, otherwise (i.e. if it is a top-level browsing context or a detached nested browsing context).

The `frameElement` IDL attribute on the Window object of a Document  $d$ , on getting, must run the following algorithm:

1. If  $d$  is not a Document in a nested browsing context, return null and abort these steps.
2. If the browsing context container's Document does not have the same effective script origin as the entry script, then throw a `SecurityError` exception.
3. Otherwise, return the browsing context container for  $b$ .

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

An auxiliary browsing context has an **opener browsing context**, which is the browsing context from which the auxiliary browsing context was created.

### 6.1.2.1 Navigating auxiliary browsing contexts in the DOM

The `opener` IDL attribute on the `Window` object, on getting, must return the `WindowProxy` object of the browsing context from which the current browsing context was created (its opener browsing context), if there is one, if it is still available, and if the current browsing context has not *disowned its opener*; otherwise, it must return null. On setting, if the new value is null then the current browsing context must *disown its opener* if the new value is anything else then the user agent must ignore the `newValue`.

**6.1.3 Sec** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**6.1.4 Sec**

A browsing context A is **allowed to navigate** a second browsing context B if one of the following conditions is true:

- Either the origin of the active document of A is the same as the origin of the active document of B, or
- The browsing context A is a nested browsing context with a top-level browsing context, and its top-level browsing context is B, or
- The browsing context B is an auxiliary browsing context and A is allowed to navigate B's opener browsing context, or
- The browsing context B is not a top-level browsing context, but there exists an ancestor browsing context of B whose active document has the same origin as the active document of A (possibly in fact being A itself).

An element has a **browsing context scope origin** if its `Document`'s browsing context is a top-level browsing context or if all of its `Document`'s ancestor browsing contexts all have active documents whose origin are the same origin as the element's `Document`'s origin. If an element has a browsing context scope origin, then its value is the origin of the element's `Document`.

### 6.1.5 Groupings of browsing contexts

Each browsing context is defined as having a list of one or more **directly reachable browsing contexts**. These are:

- The browsing context itself.
- All the browsing context's child browsing contexts.
- The browsing context's parent browsing context.
- All the browsing contexts that have the browsing context as their opener browsing context.
- The browsing context's opener browsing context.

The transitive closure of all the browsing contexts that are directly reachable browsing contexts forms a **unit of related browsing contexts**.

Each unit of related browsing contexts is then further divided into the smallest number of groups such that every member of each group has an active document with an effective script origin that, through appropriate manipulation of the `document.domain` attribute, could be made to be the same as other members of the group, but could not be made the same as members of any other group. Each such group is a **unit of related similar-origin browsing contexts**.

**Note:** There is also at most one event loop per unit of related similar-origin browsing contexts (though several units of related similar-origin browsing contexts can have a shared event loop).

### 6.1.6 Browsing context names

Browsing contexts can have a **browsing context name**. By default, a browsing context has no name (its name is not set).

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

These values have different meanings based on whether the page is sandboxed or not, as summarized in the following (non-normative) table. In this table, "current" means the browsing context that the link or script is in, "parent" means the parent browsing context of the one the link or script is in, "master" means the nearest ancestor browsing context of the one the link or script is in that is not itself in a seamless iframe, "top" means the top-level browsing context of the one the link or script is in, "new" means a new top-level browsing context or auxiliary browsing context is to be created, subject to various user preferences and user agent policies, "none" means that nothing will happen, and "maybe new" means the same as "new" if the "allow-popups" keyword is also specified on the `sandbox` attribute (or if the user overrode the sandboxing), and the same as "none" otherwise.

Keywo...

Too slow? Try reading the multipage copy of the spec instead: [https://web.archive.org/web/20130103030411/http://whatwg.org/html/\\_blank](https://web.archive.org/web/20130103030411/http://whatwg.org/html/_blank)

						<code>sandbox="allow-top-navigation" seamless=""</code>
none specified	form submission	current	current	current	current	master
none specified	<code>window.open</code>	current	parent	none	parent	maybe new†
empty string		parent	parent	none	parent	master
<code>_blank</code>		parent	parent	none	parent	maybe new
<code>_self</code>	current	current	current	current	current	current
<code>_parent</code> if there isn't a parent	current	current	current	current	current	current
<code>_parent</code> if parent is also top	parent/top	parent/top	none	none	parent/top	parent/top
<code>_parent</code> if there is one and it's not top	parent	parent	none	none	parent	none
<code>_top</code> if top is current	current	current	current	current	current	current
<code>_top</code> if top is not current	top	top	none	none	top	top
name that doesn't exist	new	new	maybe new	maybe new	maybe new	maybe new
name that exists and is a descendant	specified descendant	specified descendant	specified descendant	specified descendant	specified descendant	specified descendant
name that exists and is current	current	current	current	current	current	current
name that exists and is an ancestor that is top	specified ancestor	specified ancestor	none	none	specified ancestor/top	specified ancestor/top
name that exists and is an ancestor that is not top	specified ancestor	specified ancestor	none	none	none	none

† This case is only possible if the `sandbox` attribute also allows scripts.

An algorithm is **allowed to show a popup** if, in the task in which the algorithm is running, either:

- an activation behavior is currently being processed whose `click` event was trusted, or
- the event listener for a trusted `click` event is being handled.

The rules for choosing a browsing context given a browsing context name are as follows. The rules assume that they are being applied in the context of a browsing context, as part of the execution of a task.

1. If the given browsing context name is the empty string or `_self`, then the chosen browsing context must be the current one.  
If the given browsing context name is `_self`, then this is an explicit self-navigation override, which overrides the behavior of the seamless browsing context flag set by the `seamless` attribute on `iframe` elements.
2. If the given browsing context name is `_parent`, then the chosen browsing context must be the `parent` browsing context of the current one, unless there isn't one, in which case the chosen browsing context must be the current browsing context.
3. If the given browsing context name is `_top`, then the chosen browsing context must be the top-level browsing context of the current one, if there is one, or else the current browsing context.
4. If the given browsing context name is not `_blank` and there exists a browsing context whose name is the same as the given browsing context name, and the current browsing context is allowed to navigate that browsing context, and the user agent determines that the two browsing contexts are related enough that it is ok if they reach each other, then that browsing context must be the chosen one. If there are multiple matching browsing contexts, the user agent should select one in some arbitrary consistent manner, such as the most recently opened, most recently focused, or more closely related.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- ↳ If the algorithm is not allowed to show a popup and the user agent has been configured to not show popups (i.e. the user agent has a "popup blocker" enabled)

There is no chosen browsing context. The user agent may inform the user that a popup has been blocked.

- ↳ If the current browsing context's active document's active sandboxing flag set has the sandboxed auxiliary navigation browsing context flag set.

Typically, there is no chosen browsing context.



Existing top-level browsing must be the chosen one (the behaviour (if the user agent to be used) must be that

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*# the document that is*

- ↳ A new top-level browsing context, and the browsing context is being requested as part of following a hyperlink whose link types include the `noreferrer` keyword

A new top-level browsing context must be created. If the given browsing context name is not `_blank`, then the new top-level browsing context's name must be the given browsing context name (otherwise, it has no name). The chosen browsing context must be this new browsing context. The creation of such a browsing context is a new start for session storage.

*Note: If it is immediately navigated, then the navigation will be done with replacement enabled.*

- ↳ If the user agent has been configured such that in this instance it will create a new browsing context, and the `noreferrer` keyword doesn't apply

A new auxiliary browsing context must be created, with the opener browsing context being the current one. If the given browsing context name is not `_blank`, then the new auxiliary browsing context's name must be the given browsing context name (otherwise, it has no name). The chosen browsing context must be this new browsing context.

*Note: If it is immediately navigated, then the navigation will be done with replacement enabled.*

- ↳ If the user agent has been configured such that in this instance it will reuse the current browsing context

The chosen browsing context is the current browsing context.

- ↳ If the user agent has been configured such that in this instance it will not find a browsing context

There must not be a chosen browsing context.

User agent implementors are encouraged to provide a way for users to configure the user agent to always reuse the current browsing context.

If the current browsing context's active document's active sandboxing flag set has the sandboxed navigation browsing context flag set and chosen browsing context picked above, if any, is a new browsing context (whether top-level or auxiliary), then all the flags that are set in the current browsing context's active document's active sandboxing flag set when the new browsing context is created must be set in the new browsing context's popup sandboxing flag set, and the current browsing context must be set as the new browsing context's one permitted sandboxed navigator.

## 6.2 The Window object

```
[NamedPropertiesObject]
/*sealed*/ interface Window : EventTarget {
 // the current browsing context
 [Unforgeable] readonly attribute WindowProxy window;
 [Replaceable] readonly attribute WindowProxy self;
 [Unforgeable] readonly attribute Document document;
 attribute DOMString name;
```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
[Replaceable] readonly attribute BarProp scrollbars;
[Replaceable] readonly attribute BarProp statusbar;
[Replaceable] readonly attribute BarProp toolbar;
 attribute DOMString status;
void close();
void stop();
void focus();
void blur();

// Too slow? Try reading the multipage copy of
[R] the spec instead: https://web.archive.org
[U] /web/20130103030411/http:
[re] //whatwg.org/html
[re] featu ional DOMString
[re] getter WindowProxy (unsigned long index);
[re] getter object (DOMString name);

// the user agent
readonly attribute Navigator navigator;
readonly attribute External external;
readonly attribute ApplicationCache applicationCache;

// user prompts
void alert(DOMString message);
boolean confirm(DOMString message);
DOMString? prompt(DOMString message, optional DOMString default);
void print();
any showModalDialog(DOMString url, optional any argument);

void postMessage(any message, DOMString targetOrigin, optional sequence<Transferable>
transfer);
// event handler IDL attributes
 attribute EventHandler onabort;
 attribute EventHandler onafterprint;
 attribute EventHandler onbeforeprint;
 attribute EventHandler onbeforeunload;
 attribute EventHandler onblur;
 attribute EventHandler oncancel;
 attribute EventHandler oncanplay;
 attribute EventHandler oncanplaythrough;
 attribute EventHandler onchange;
 attribute EventHandler onclick;
 attribute EventHandler onclose;
 attribute EventHandler oncontextmenu;
 attribute EventHandler oncuechange;
 attribute EventHandler ondblclick;
 attribute EventHandler ondrag;
 attribute EventHandler ondragend;
 attribute EventHandler ondragenter;
 attribute EventHandler ondragleave;
 attribute EventHandler ondragover;
 attribute EventHandler ondragstart;
 attribute EventHandler ondrop;
 attribute EventHandler ondurationchange;
 attribute EventHandler onemptied;
 attribute EventHandler onended;
 attribute OnErrorEventHandler onerror;
 attribute EventHandler onfocus;
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
attribute EventHandler onkeydown;
attribute EventHandler onkeypress;
attribute EventHandler onkeyup;
attribute EventHandler onload;
attribute EventHandler onloadeddata;
attribute EventHandler onloadedmetadata;
attribute EventHandler onloadstart;
attribute EventHandler onmessage;
attribute EventHandler onmousedown;
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
attribute EventHandler onplaying;
attribute EventHandler onpagehide;
attribute EventHandler onpageshow;
attribute EventHandler onpopstate;
attribute EventHandler onprogress;
attribute EventHandler onratechange;
attribute EventHandler onreset;
attribute EventHandler onresize;
attribute EventHandler onscroll;
attribute EventHandler onseeked;
attribute EventHandler onseeking;
attribute EventHandler onselect;
attribute EventHandler onshow;
attribute EventHandler onsort;
attribute EventHandler onstalled;
attribute EventHandler onstorage;
attribute EventHandler onsubmit;
attribute EventHandler onsuspend;
attribute EventHandler ontimeupdate;
attribute EventHandler onunload;
attribute EventHandler onvolumechange;
attribute EventHandler onwaiting;
};
```

This box is non-normative. Implementation requirements are given below this box.

**window.window****window.frames****window.self**

These attributes all return `window`.

**window.document**

Returns the active document.

**document.defaultView**

Returns the `Window` object of the active document.

The `Window` interface must only be exposed if the JavaScript global environment is a document environment.

The `window`, `frames`, and `self` IDL attributes must all return the `Window` object's browsing context's `WindowProxy` object.

The `document` IDL attribute must return the `Document` object of the `Window` object's `Document`'s browsing context's active

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

For historical reasons, Window objects must also have a writable, configurable, non-enumerable property named `HTMLDocument` whose value is the `Document` interface object.

### 6.2.1 Security

User agents must throw a `SecurityError` exception whenever any properties of a `Window` object are accessed by scripts whose effective script origin is not the same as the `Window` object's `Document`'s effective script origin, with the following exceptions:

- The `top` attribute
  - The `parent` attribute
  - The dynamic nested browsing context properties
- Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

When a script whose effective script origin is not the same as the `Window` object's `Document`'s effective script origin attempts to access that `Window` object's methods or attributes, the user agent must act as if any changes to the `Window` object's properties, getters, setters, etc, were not present.

For members that return objects (including function objects), each distinct effective script origin that is not the same as the `Window` object's `Document`'s effective script origin must be provided with a separate set of objects. These objects must have the prototype chain appropriate for the script for which the objects are created (not those that would be appropriate for scripts whose script's global object is the `Window` object in question).

For instance, if two frames containing `Documents` from different origins access the same `Window` object's `postMessage()` method, they will get distinct objects that are not equal.

### 6.2.2 APIs for creating and navigating browsing contexts by name

This box is non-normative. Implementation requirements are given below this box.

`window = window.open( [ url [, target [, features [, replace ]]]])`

Opens a window to show `url` (defaults to `about:blank`), and returns it. The `target` argument gives the name of the new window. If a window exists with that name already, it is reused. The `replace` attribute, if true, means that whatever page is currently open in that window will be removed from the window's session history. The `features` argument is ignored.

`window.name [= value]`

Returns the name of the window.

Can be set, to change the name.

`window.close()`

Closes the window.

`window.stop()`

Cancels the document load.

The `open()` method on `Window` objects provides a mechanism for navigating an existing browsing context or opening and navigating an auxiliary browsing context.

The method has four arguments, though they are all optional.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

name or keyword. If fewer than two arguments are provided, then the `target` argument defaults to the value "`_blank`".

The third argument, `features`, has no defined effect and is mentioned for historical reasons only. User agents may interpret this argument as instructions to set the size and position of the browsing context, but are encouraged to instead ignore the argument entirely.

The fourth argument, `replace`, specifies whether or not the new page will replace the page currently loaded in the browsing context, when `target` identifies an existing browsing context (as opposed to leaving the current page in the browsing context's session history). When three or fewer arguments are provided, `replace` defaults to false.

When the m  
browsing co  
context in wl  
may instead

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

For e  
user  
so, h  
tab.

/web/20130103030411/http:  
//whatwg.org/html

re rules for choosing a context of the script as the browsing context to navigate

ab. If a user clicks in that in an iframe, but, while doing context to instead target a new

If applying the rules for choosing a browsing context given a browsing context name using the `target` argument would result in there not being a chosen browsing context, then throw an `InvalidAccessError` exception and abort these steps.

Otherwise, if `url` is not "about:blank", the user agent must navigate the selected browsing context to the absolute URL obtained from resolving `url` earlier. If the `replace` is true or if the browsing context was just created as part of the rules for choosing a browsing context given a browsing context name, then replacement must be enabled. The navigation must be done with the browsing context of the entry script as the source browsing context. If the resolve a URL algorithm failed, then the user agent may either instead navigate to an inline error page, using the same replacement behavior and source browsing context behavior as described earlier in this paragraph; or treat the `url` as "about:blank", acting as described in the next paragraph.

If `url` is "about:blank", the user agent must instead queue a task to fire a simple event named `load` at the selected browsing context's `Window` object, but with its `target` set to the selected browsing context's `Window` object's `Document` object (and the `currentTarget` set to the `Window` object).

The method must return the `WindowProxy` object of the browsing context that was navigated, or null if no browsing context was navigated.

The `name` attribute of the `Window` object must, on getting, return the current name of the browsing context, and, on setting, set the name of the browsing context to the new value.

**Note:** The name gets reset when the browsing context is navigated to another domain.

The `close()` method on `Window` objects should, if the corresponding browsing context A is script-closable and the browsing context of the script that invokes the method is allowed to navigate the browsing context A, close the browsing context A.

A browsing context is **script-closable** if it is an auxiliary browsing context that was created by a script (as opposed to by an action of the user), or if it is a browsing context whose session history contains only one `Document`.

The `stop()` method on `Window` objects should, if there is an existing attempt to navigate the browsing context and that attempt is not currently running the unload a document algorithm, cancel that navigation; then, it must abort the active document of the browsing context of the `Window` object on which it was invoked.

### 6.2.3 Accessing other browsing contexts

This box is non-normative. Implementation requirements are given below this box.

`window.length`

Returns the number of child browsing contexts.

`window[index]`

Returns the indicated child browsing context.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

returned by the `length` IDL attribute. If `n` is zero then there are no supported property indices.

To determine the value of an indexed property `index` of a `Window` object, the user agent must return the `WindowProxy` object of the `index`th child browsing context of the `Document` that is nested through an element that is in the `Document`, sorted in the tree order of the elements nesting those browsing contexts.

These properties are the **dynamic nested browsing context properties**.



#### 6.2.4 Narr

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ts are given below this box.

The `Window` interface has the following properties:

of:

- the browsing context name of any child browsing context of the active document whose name is not the empty string,
- the value of the `name` content attribute for all `a`, `applet`, `area`, `embed`, `form`, `frameset`, `img`, and `object` elements in the active document that have a `name` content attribute, and
- the value of the `id` content attribute of any HTML element in the active document with an `id` content attribute.

To determine the value of a named property `name` when the `Window` object is indexed for property retrieval, the user agent must return the value obtained using the following steps:

1. Let `objects` be the list of named objects with the name `name` in the active document.

**Note: There will be at least one such object, by definition.**

2. If `objects` contains a nested browsing context, then return the `WindowProxy` object of the nested browsing context corresponding to the first browsing context container in tree order whose browsing context is in `objects`, and abort these steps.
3. Otherwise, if `elements` has only one element, return that element and abort these steps.
4. Otherwise return an `HTMLCollection` rooted at the `Document` node, whose filter matches only named objects with the name `name`. (By definition, these will all be elements.)

**Named objects** with the name `name`, for the purposes of the above algorithm, are those that are either:

- child browsing contexts of the active document whose name is `name`,
- `a`, `applet`, `area`, `embed`, `form`, `frameset`, `img`, or `object` elements that have a `name` content attribute whose value is `name`, or
- HTML elements that have an `id` content attribute whose value is `name`.

#### 6.2.5 Garbage collection and browsing contexts

A browsing context has a strong reference to each of its `Documents` and its `WindowProxy` object, and the user agent itself has a strong reference to its top-level browsing contexts.

A `Document` has a strong reference to its `Window` object.

**Note: A `Window` object has a strong reference to its `Document` object through its `document` attribute. Thus, references from other scripts to either of those objects will keep both alive. Similarly, both `Document` and `Window` objects have implied strong references to the `WindowProxy` object.**

Each script has a strong reference to its browsing context and its document.

When a browsing context is to **discard a `Document`**, the user agent must run the following steps:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. Abort the Document.
4. Remove any tasks associated with the Document in any task source, without running those tasks.
5. Discard all the child browsing contexts of the Document.
6. Lose the strong reference from the Document's browsing context to the Document.

**Note:** Whenever a Document object is discarded, it is also removed from the list of the worker's Documents of each worker with its browsing context.

When a browser closes a document, the document's browsing context must be severed, and well. User agents must close their WindowProxy objects when they are force-closed by the user.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### 6.2.6 Closing browsing contexts

When the user agent is required to **close a browsing context**, it must run the following steps:

1. Let *specified browsing context* be the browsing context being closed.
2. Prompt to unload the active document of the *specified browsing context*. If the user refused to allow the document to be unloaded, then abort these steps.
3. Unload the active document of the *specified browsing context* with the *recycle* parameter set to false.
4. Remove the *specified browsing context* from the user interface (e.g. close or hide its tab in a tabbed browser).
5. Discard the *specified browsing context*.

User agents should offer users the ability to arbitrarily close any top-level browsing context.

### 6.2.7 Browser interface elements

To allow Web pages to integrate with Web browsers, certain Web browser interface elements are exposed in a limited way to scripts in Web pages.

Each interface element is represented by a `BarProp` object:

```
interface BarProp {
 attribute boolean visible;
};
```

*This box is non-normative. Implementation requirements are given below this box.*

`window.locationbar.visible`

Returns true if the location bar is visible; otherwise, returns false.

`window.menubar.visible`

Returns true if the menu bar is visible; otherwise, returns false.

`window.personalbar.visible`

Returns true if the personal bar is visible; otherwise, returns false.

`window.scrollbars.visible`

Returns true if the scroll bars are visible; otherwise, returns false.

`window.statusbar.visible`

Returns true if the status bar is visible; otherwise, returns false.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

The **visible** attribute, on getting, must return either true or a value determined by the user agent to most accurately represent the visibility state of the user interface element that the object represents, as described below. On setting, the new value must be discarded.

The following **BarProp** objects exist for each **Document** object in a browsing context. Some of the user interface elements represented by these objects might have no equivalent in some user agents; for those user agents, except when otherwise specified, the object must act as if it was present and visible (i.e. its **visible** attribute must return true).

#### **The location bar BarProp object**

Represents the user interface element that contains the location bar. If the user agent has no such user interface element, then the object may act as if the corresponding user interface element was absent (i.e. its **visible** attribute may return false).

**locationbar** attribute must return the location bar **BarProp** object.

**menubar** attribute must return the menu bar **BarProp** object.

**personalbar** attribute must return the personal bar **BarProp** object.

**scrollbars** attribute must return the scrollbar **BarProp** object.

**statusbar** attribute must return the status bar **BarProp** object.

**toolbar** attribute must return the toolbar **BarProp** object.

#### **The scrollba.**

Represents the user interface element that contains a scrolling mechanism, or some similar interface concept.

#### **The status bar BarProp object**

Represents a user interface element found immediately below or after the document, as appropriate for the user's media. If the user agent has no such user interface element, then the object may act as if the corresponding user interface element was absent (i.e. its **visible** attribute may return false).

#### **The toolbar BarProp object**

Represents the user interface element found immediately above or before the document, as appropriate for the user's media. If the user agent has no such user interface element, then the object may act as if the corresponding user interface element was absent (i.e. its **visible** attribute may return false).

The **locationbar** attribute must return the location bar **BarProp** object.

The **menubar** attribute must return the menu bar **BarProp** object.

The **personalbar** attribute must return the personal bar **BarProp** object.

The **scrollbars** attribute must return the scrollbar **BarProp** object.

The **statusbar** attribute must return the status bar **BarProp** object.

The **toolbar** attribute must return the toolbar **BarProp** object.

For historical reasons, the **status** attribute on the **Window** object must return an empty string on getting, and do nothing on setting.

### **6.2.8 The WindowProxy object**

As mentioned earlier, each browsing context has a **WindowProxy** object. This object is unusual in that all operations that would be performed on it must be performed on the **Window** object of the browsing context's active document instead. It is thus indistinguishable from that **Window** object in every way until the browsing context is navigated.

There is no **WindowProxy** interface object.

**Note:** The **WindowProxy** object allows scripts to act as if each browsing context had a single **Window** object, while still keeping separate **Window** objects for each **Document**.

In the following example, the variable **x** is set to the **WindowProxy** object returned by the **window** accessor on the global object. All of the expressions following the assignment return true, because in every respect, the **WindowProxy** object acts like the underlying **Window** object.

```
var x = window;
x instanceof Window; // true
x === this; // true
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Note:** The extra data could include the certificate of the site when using encrypted connections, to ensure that if the site's secure certificate changes, the origin is considered to change as well.

An origin or effective script origin can be defined as an **alias** to another origin or effective script origin. The value of the origin or effective script origin is then the value of the origin or effective script origin to which it is an alias.

These characteristics are defined as follows:

#### For URLs

The

For Document

Too slow? Try reading the multipage copy of

[ORIGIN]

- ↳ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

tag set

The origin is an alias to the origin of the script or that `javascript: URL`.

The effective script origin is initially an alias to the origin of the Document.

- ↳ If a Document was served over the network and has an address that uses a URL scheme with a server-based naming authority

The origin is an alias to the origin of the Document's address.

The effective script origin is initially an alias to the origin of the Document.

- ↳ If a Document was generated from a data: URL that was returned as the location of an HTTP redirect (or equivalent in other protocols)

The origin is an alias to the origin of the URL that redirected to the data: URL.

The effective script origin is initially an alias to the origin of the Document.

- ↳ If a Document was generated from a data: URL found in another Document or in a script

The origin is an alias to the origin of the Document or script that initiated the navigation to that URL.

The effective script origin is initially an alias to the effective script origin of the Document or script that initiated the navigation to that URL.

- ↳ If a Document has the address "about:blank"

The origin and effective script origin of the Document are those it was assigned when its browsing context was created.

- ↳ If a Document is an iframe srcdoc document

The origin of the Document is an alias to the origin of the Document's browsing context's browsing context container's Document.

The effective script origin is initially an alias to the effective script origin of the Document's browsing context's browsing context container's Document.

- ↳ If a Document was obtained in some other manner (e.g. a data: URL typed in by the user, a Document created using the `createDocument()` API, etc)

The origin is a globally unique identifier assigned when the Document is created.

The effective script origin is initially an alias to the origin of the Document.

**Note:** The effective script origin of a Document can be manipulated using the `document.domain` IDL attribute.

#### For images

- ↳ If an image is the image of an `img` element and its image data is CORS-cross-origin

The origin is a globally unique identifier assigned when the image is created.

- ↳ If an image is the image of an `img` element and its image data is CORS-same-origin

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

↪ If the media data is CORS-cross-origin

The origin is a globally unique identifier assigned when the image is created.

↪ If the media data is CORS-same-origin

The origin is an alias to the origin of the media element's Document.

Media elements do not have an effective script origin.

**For fonts**

The origin of a downloadable Web font is an alias to the origin of the absolute URL used to obtain the font (after any redirects). [CSSF]

The **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

font is being used.

**For scripts** /web/20130103030411/http:

The owner is the Document of the browser window.

↪

The owner is the Document to which the script element belongs.

↪ If a script is in an event handler content attribute

The owner is the Document to which the attribute node belongs.

↪ If a script is a function or other code reference created by another script

The owner is the script that created it.

↪ If a script is a javascript: URL that was returned as the location of an HTTP redirect (or equivalent in other protocols)

The owner is the URL that redirected to the javascript: URL.

↪ If a script is a javascript: URL in an attribute

The owner is the Document of the element on which the attribute is found.

↪ If a script is a javascript: URL in a style sheet

The owner is the URL of the style sheet.

↪ If a script is a javascript: URL to which a browsing context is being navigated, the URL having been provided by the user (e.g. by using a bookmarklet)

The owner is the Document of the browsing context's active document.

↪ If a script is a javascript: URL to which a browsing context is being navigated, the URL having been declared in markup

The owner is the Document of the element (e.g. an a or area element) that declared the URL.

↪ If a script is a javascript: URL to which a browsing context is being navigated, the URL having been provided by script

The owner is the script that provided the URL.

The origin of the script is then an alias to the origin of the owner, and the effective script origin of the script is an alias to the effective script origin of the owner.

Other specifications can override the above definitions by themselves specifying the origin of a particular URL, Document, image, media element, font, or script.

The **Unicode serialization of an origin** is the string obtained by applying the following algorithm to the given origin:

1. If the origin in question is not a scheme/host/port tuple, then return the literal string "null" and abort these steps.
2. Otherwise, let *result* be the scheme part of the origin tuple.
3. Append the string ": //" to *result*.
4. Apply the IDNA ToUnicode algorithm to each component of the host part of the origin tuple, and append the results — each component, in the same order, separated by U+002E FULL STOP characters (.) — to *result*. [RFC3490]
5. If the port part of the origin tuple gives a port that is different from the default port for the protocol given by the scheme part of the origin tuple, then append a U+003A COLON character (:) and the given port, in base ten, to *result*.
6. Return *result*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

3. Append the string ": //" to *result*.
4. Apply the IDNA ToASCII algorithm the host part of the origin tuple, with both the AllowUnassigned and UseSTD3ASCIIRules flags set, and append the results *result*.

If ToASCII fails to convert one of the components of the string, e.g. because it is too long or because it contains invalid characters, then return the empty string and abort these steps. [RFC3490]

5. If the port part of the origin tuple gives a port that is different from the default port for the protocol given by the scheme part of the origin, then return the empty string and abort these steps.

## 6. Ret **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

Two origins:

1. Let `/web/20130103030411/http:`

2. If `A` `//whatwg.org/html`

3. Otherwise,

4. If `A` and `B` have scheme components that are not identical, return false.

5. If `A` and `B` have host components that are not identical, return false.

6. If `A` and `B` have port components that are not identical, return false.

7. If either `A` or `B` have additional data, but that data is not identical for both, return false.

8. Return true.

### 6.3.1 Relaxing the same-origin restriction

*This box is non-normative. Implementation requirements are given below this box.*

`document.domain [= domain]`

Returns the current domain used for security checks.

Can be set to a value that removes subdomains, to change the effective script origin to allow pages on other subdomains of the same domain (if they do the same thing) to access each other.

The `domain` attribute on `Document` objects must be initialized to the document's domain, if it has one, and the empty string otherwise. If the value is an IPv6 address, then the square brackets from the host portion of the `<host>` component must be omitted from the attribute's value.

On getting, the attribute must return its current value, unless the `Document` has no browsing context, in which case it must return the empty string.

On setting, the user agent must run the following algorithm:

1. If the `Document` has no browsing context, throw a `SecurityError` exception and abort these steps.
2. If the new value is an IP address, let `new value` be the new value. Otherwise, apply the IDNA ToASCII algorithm to the new value, with both the AllowUnassigned and UseSTD3ASCIIRules flags set, and let `new value` be the result of the ToASCII algorithm.

If ToASCII fails to convert one of the components of the string, e.g. because it is too long or because it contains invalid characters, then throw a `SecurityError` exception and abort these steps. [RFC3490]

3. If `new value` is not exactly equal to the current value of the `document.domain` attribute, then run these substeps:

1. If the current value is an IP address, throw a `SecurityError` exception and abort these steps.

2. If `new value`, prefixed by a U+002E FULL STOP (.), does not exactly match the end of the current value, throw a `SecurityError` exception and abort these steps.

3,445 captures  
5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB  
◀ 03 ▶  
2012 2013 2014

About this capture

4. Release the storage mutex.
5. Set the attribute's value to *new value*.
6. If the effective script origin of the Document is an alias, set it to the value of the effective script origin (essentially de-aliasing the effective script origin).
7. If *new value* is not the empty string, then run these substeps:



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The domain then the doc /web/20130103030411/http://whatwg.org/html

*Note: T*

a value that, for the tier value).

/host/port tuple. If it isn't,

such others' DOMs.

**⚠Warning! Do not use the document.domain attribute when using shared hosting. If an untrusted third party is able to host an HTTP server at the same IP address but on a different port, then the same-origin protection that normally protects two different sites on the same host will fail, as the ports are ignored when comparing origins after the document.domain attribute has been used.**

## 6.4 Sandboxing

A **sandboxing flag set** is a set of zero or more of the following flags, which are used to restrict the abilities that potentially untrusted resources have:

### The sandboxed navigation browsing context flag

This flag prevents content from navigating browsing contexts other than the sandboxed browsing context itself (or browsing contexts further nested inside it), auxiliary browsing contexts (which are protected by the sandboxed auxiliary navigation browsing context flag defined next), and the top-level browsing context (which is protected by the sandboxed top-level navigation browsing context flag defined below).

If the sandboxed auxiliary navigation browsing context flag is not set, then in certain cases the restrictions nonetheless allow popups (new top-level browsing contexts) to be opened. These browsing contexts always have **one permitted sandboxed navigator**, set when the browsing context is created, which allows the browsing context that created them to actually navigate them. (Otherwise, the sandboxed navigation browsing context flag would prevent them from being navigated even if they were opened.)

### The sandboxed auxiliary navigation browsing context flag

This flag prevents content from creating new auxiliary browsing contexts, e.g. using the target attribute, the window.open() method, or the showModalDialog() method.

### The sandboxed top-level navigation browsing context flag

This flag prevents content from navigating their top-level browsing context.

When the sandboxed top-level navigation browsing context flag is *not* set, content can navigate its top-level browsing context, but other browsing contexts are still protected by the sandboxed navigation browsing context flag and possibly the sandboxed auxiliary navigation browsing context flag.

### The sandboxed plugins browsing context flag

This flag prevents content from instantiating plugins, whether using the embed element, the object element, the applet element, or through navigation of a nested browsing context, unless those plugins can be secured.

### The sandboxed seamless iframes flag

This flag prevents content from using the seamless attribute on descendant iframe elements.

**Note: This prevents a page inserted using the allow-same-origin keyword from using a CSS-selector-based method of probing the DOM of other pages on the same site (in particular, pages that contain user-sensitive information).**

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**The `sandboxed forms` browsing context flag**

This flag blocks form submission.

**The `sandboxed pointer lock` browsing context flag**

This flag disables the Pointer Lock API. [POINTERLOCK]

**The `sandboxed scripts` browsing context flag**

This flag blocks script execution.



**The `sandbox`** **Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html/fullscreen>

**The `sandbox`** **/web/20130103030411/http://whatwg.org/html/fullscreen flag,** it illustrates the following steps.

1. Split *input* on spaces, to obtain *tokens*.

2. Let *output* be empty.

3. Add the following flags to *output*:

- The sandboxed navigation browsing context flag.
- The sandboxed auxiliary navigation browsing context flag, unless *tokens* contains the `allow-popups` keyword.
- The sandboxed top-level navigation browsing context flag, unless *tokens* contains the `allow-top-navigation` keyword.
- The sandboxed plugins browsing context flag.
- The sandboxed seamless iframes flag.
- The sandboxed origin browsing context flag, unless the *tokens* contains the `allow-same-origin` keyword.

**The `allow-same-origin` keyword is intended for two cases.**

**First, it can be used to allow content from the same site to be sandboxed to disable scripting, while still allowing access to the DOM of the sandboxed content.**

**Second, it can be used to embed content from a third-party site, sandboxed to prevent that site from opening pop-up windows, etc, without preventing the embedded page from communicating back to its originating site, using the database APIs to store data, etc.**

- The sandboxed forms browsing context flag, unless *tokens* contains the `allow-forms` keyword.
- The sandboxed pointer lock browsing context flag, unless *tokens* contains the `allow-pointer-lock` keyword.
- The sandboxed scripts browsing context flag, unless *tokens* contains the `allow-scripts` keyword.
- The sandboxed automatic features browsing context flag, unless *tokens* contains the `allow-scripts` keyword (defined above).

**Note: This flag is relaxed by the same keyword as scripts, because when scripts are enabled these features are trivially possible anyway, and it would be unfortunate to force authors to use script to do them when sandboxed rather than allowing them to use the declarative features.**

- The sandboxed fullscreen browsing context flag, unless the `allow fullscreen` flag was passed to the parse a sandboxing directive flag.

Every top-level browsing context has a **popup sandboxing flag set**, which is a sandboxing flag set. When a browsing context is

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Every Document has an **active sandboxing flag set**, which is a sandboxing flag set. When the Document is created, its active sandboxing flag set must be empty. It is populated by the navigation algorithm.

Every resource that is obtained by the navigation algorithm has a **forced sandboxing flag set**, which is a sandboxing flag set. A resource by default has no flags set in its forced sandboxing flag set, but other specifications can define that certain flags are set.

**Note:** In particular, the forced sandboxing flag set is used by the Content Security Policy specification. [CSP]



When a user navigates to a document, the browser creates a new browsing context. This browsing context has an active sandboxing flag set with the same flags as the document's forced sandboxing flag set. The active sandboxing flag set is initially empty. It is populated by the navigation algorithm.

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- If the Document's browsing context is a top-level browsing context, then: the flags set on the Document's active sandboxing flag set.
- If the Document's browsing context is a nested browsing context, then: the flags set on the Document's active sandboxing flag set.
- The flags set on the Document's resource's forced sandboxing flag set, if it has one.

When a user navigates to a document, the browser creates a new browsing context. This browsing context has an active sandboxing flag set with the same flags as the document's forced sandboxing flag set. The active sandboxing flag set is initially empty. It is populated by the navigation algorithm.

If the Document's browsing context is a nested browsing context, then: the flags set on the Document's active sandboxing flag set.

The flags set on the Document's resource's forced sandboxing flag set, if it has one.

## 6.5 Session history and navigation

### 6.5.1 The session history of browsing contexts

The sequence of Documents in a browsing context is its **session history**.

History objects provide a representation of the pages in the session history of browsing contexts. Each browsing context, including nested browsing contexts, has a distinct session history.

Each Document object in a browsing context's session history is associated with a unique instance of the History object, although they all must model the same underlying session history.

The **history** attribute of the Window interface must return the object implementing the History interface for that Window object's Document.

History objects represent their browsing context's session history as a flat list of session history entries. Each **session history entry** consists of a URL and optionally a state object, and may in addition have a title, a Document object, form data, a scroll position, and other information associated with it.

**Note:** This does not imply that the user interface need be linear. See the notes below.

**Note:** Titles associated with session history entries need not have any relation with the current title of the Document. The title of a session history entry is intended to explain the state of the document at that point, so that the user can navigate the document's history.

URLs without associated state objects are added to the session history as the user (or script) navigates from page to page.

A **state object** is an object representing a user interface state.

Pages can add state objects to the session history. These are then returned to the script when the user (or script) goes back in the history, thus enabling authors to use the "navigation" metaphor even in one-page applications.

**State objects are intended to be used for two main purposes:** first, storing a prepared description of the state in the URL so that in the simple case an author doesn't have to do the parsing (though one would still need the parsing for handling URLs passed around by users, so it's only a minor optimization), and second, so that the author can store state that one wouldn't store in the URL because it only applies to the current Document instance and it would have to be reconstructed if a new Document were opened.

An example of the latter would be something like keeping track of the precise coordinate from which a pop-up div

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

At any point, one of the entries in the session history is the **current entry**. This is the entry representing the active document of the browsing context. Which entry is the current entry is changed by the algorithms defined in this specification, e.g. during session history traversal.

**Note:** The current entry is usually an entry for the location of the Document. However, it can also be one of the entries for state objects added to the history by that document.

An entry with persisted user state is one that also has user-agent defined state. This specification does not specify what kind of state can be stored.

For ε Too slow? Try reading the multipage copy of

**Note:** L of the ε the use the spec instead: https://web.archive.org /web/20130103030411/http:

Entries that //whatwg.org/html

irectionality (the value story traversal when

en they were added.

Contiguous entries that differ just by fragment identifier also share the same Document.

**Note:** All entries that share the same Document (and that are therefore merely different states of one particular document) are contiguous by definition.

Each Document in a browsing context can also have a latest entry. This is the entry or that Document that was most recently traversed to. When a Document is created, it initially has no latest entry.

User agents may discard the Document objects of entries other than the current entry that are not referenced from any script, reloading the pages afresh when the user or script navigates back to such pages. This specification does not specify when user agents should discard Document objects and when they should cache them.

Entries that have had their Document objects discarded must, for the purposes of the algorithms given below, act as if they had not. When the user or script navigates back or forwards to a page which has no in-memory DOM objects, any other entries that shared the same Document object with it must share the new object as well.

## 6.5.2 The History interface

```
interface History {
 readonly attribute long length;
 readonly attribute any state;
 void go(optional long delta);
 void back();
 void forward();
 void pushState(any data, DOMString title, optional DOMString url);
 void replaceState(any data, DOMString title, optional DOMString url);
};
```

This box is non-normative. Implementation requirements are given below this box.

`window.history.length`

Returns the number of entries in the joint session history.

`window.history.state`

Returns the current state object.

`window.history.go([delta])`

Goes back or forward the specified number of steps in the joint session history.

A zero delta will reload the current page.

If the delta is out of range, does nothing.

`window.history.back()`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Goes forward one step in the joint session history.

If there is no next page, does nothing.

`window.history.pushState(data, title [, url ])`

Pushes the given data onto the session history, with the given title, and, if provided, the given URL.

`window.history.replaceState(data, title [, url ])`

Updates the current entry in the session history to have the given data, title, and, if provided, URL.

The joint session history consists of all the fully active current entries in their respective contexts.

The current document context's session history consists of all the entries in its session history.

Entries in the session history are numbered consecutively starting from 0.

All these browsing contexts by definition share an event loop; there is always a well-defined sequential order in which their session histories had their entries added.) Each entry has an index; the earliest entry has index 0, and the subsequent entries are numbered with consecutively increasing integers (1, 2, 3, etc).

The `length` attribute of the `History` interface must return the number of entries in the joint session history.

The actual entries are not accessible from script.

The `state` attribute of the `History` interface must return the last value it was set to by the user agent. Initially, its value must be null.

When the `go(delta)` method is invoked, if the argument to the method was omitted or has the value zero, the user agent must act as if the `location.reload()` method was called instead. Otherwise, the user agent must traverse the history by a delta whose value is the value of the method's argument.

When the `back()` method is invoked, the user agent must traverse the history by a delta -1.

When the `forward()` method is invoked, the user agent must traverse the history by a delta +1.

To **traverse the history by a delta *delta***, the user agent must queue a task to run the following steps. The task source for the queued task is the history traversal task source.

1. Let *delta* be the argument to the method.
2. If the index of the current entry of the joint session history plus *delta* is less than zero or greater than or equal to the number of items in the joint session history, then abort these steps.
3. If the `Document`'s `unload` a document algorithm is currently running, abort these steps.
4. If there is an ongoing attempt to navigate the browsing context that has not yet matured (i.e. it has not passed the point of making its `Document` the active document), then cancel that attempt to navigate the browsing context.
5. Let *specified entry* be the entry in the joint session history whose index is the sum of *delta* and the index of the current entry of the joint session history.
6. Let *specified browsing context* be the browsing context of the *specified entry*.
7. If the *specified browsing context*'s active document is not the same `Document` as the `Document` of the *specified entry*, then run these substeps:
  1. Fully exit fullscreen.
  2. Prompt to unload the active document of the *specified browsing context*. If the user refused to allow the document to be unloaded, then abort these steps.
  3. Unload the active document of the *specified browsing context* with the `recycle` parameter set to false.
8. Traverse the history of the *specified browsing context* to the *specified entry*.

When the user navigates through a browsing context, e.g. using a browser's back and forward buttons, the user agent must traverse

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `replaceState(data, title, url)` method updates the state object, title, and optionally the URL of the current entry in the history.

When either of these methods is invoked, the user agent must run the following steps:

1. Let `cloned data` be a structured clone of the specified `data`. If this throws an exception, then rethrow that exception and abort these steps.

2. If a third argument is specified, run these substeps:

[REDACTED]

**Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)**

JRLs differ other than the  
and abort these steps.

document, and either the  
) or a `SecurityError`  
(pages on the same origin.)

5. Let `new URL` be the resulting absolute URL.

For the purposes of the comparisons in the above substeps, the `<path>` and `<query>` components can only be the same if the URLs are both hierarchical URLs.

3. If a third argument is *not* specified, then let `new URL` be the URL of the current entry.

4. If the method invoked was the `pushState()` method:

1. Remove all the entries in the browsing context's session history after the current entry. If the current entry is the last entry in the session history, then no entries are removed.

**Note: This doesn't necessarily have to affect the user agent's user interface.**

2. Remove any tasks queued by the history traversal task source that are associated with any `Document` objects in the top-level browsing context's document family.

3. If appropriate, update the current entry to reflect any state that the user agent wishes to persist. The entry is then said to be an entry with persisted user state.

4. Add a state object entry to the session history, after the current entry, with `cloned data` as the state object, the given `title` as the title, and `new URL` as the URL of the entry.

5. Update the current entry to be this newly added entry.

Otherwise, if the method invoked was the `replaceState()` method:

1. Update the current entry in the session history so that `cloned data` is the entry's new state object, the given `title` is the new title, and `new URL` is the entry's new URL.

5. If the current entry in the session history represents a non-GET request (e.g. it was the result of a POST submission) then update it to instead represent a GET request (or equivalent).

6. Set the document's address to `new URL`.

**Note: Since this is neither a navigation of the browsing context nor a history traversal, it does not cause a hashchange event to be fired.**

7. Set `history.state` to a structured clone of `cloned data`.

8. Let the latest entry of the `Document` of the current entry be the current entry.

**Note: The title is purely advisory. User agents might use the title in the user interface.**

User agents may limit the number of state objects added to the session history per page. If a page hits the UA-defined limit, user agents must remove the entry immediately after the first entry for that `Document` object in the session history after having added the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

A static page implementing the `x=5` position in such a game could look like the following:

```
<!DOCTYPE HTML>
<!-- this is http://example.com/line?x=5 -->
<title>Line Game - 5</title>
<p>You are at coordinate 5 on the line.</p>
<p>
Advance to 6 or
retreat to 4?
</p>
<script>
var currentPage = 5; // prefilled by server
function go(d) {
 setupPage(currentPage + d);
 history.pushState(currentPage, document.title, '?x=' + currentPage);
}
onpopstate = function(event) {
 setupPage(event.state);
}
function setupPage(page) {
 currentPage = page;
 document.title = 'Line Game - ' + currentPage;
 document.getElementById('coord').textContent = currentPage;
 document.links[0].href = '?x=' + (currentPage+1);
 document.links[0].textContent = 'Advance to ' + (currentPage+1);
 document.links[1].href = '?x=' + (currentPage-1);
 document.links[1].textContent = 'retreat to ' + (currentPage-1);
}
</script>
```

In systems without script, this still works like the previous example. However, users that *do* have script support can now navigate much faster, since there is no network access for the same experience. Furthermore, contrary to the experience the user would have with just a naïve script-based approach, bookmarking and navigating the session history still work.

In the example above, the `data` argument to the `pushState()` method is the same information as would be sent to the server, but in a more convenient form, so that the script doesn't have to parse the URL each time the user navigates.

Applications might not use the same title for a session history entry as the value of the document's `title` element at that time. For example, here is a simple page that shows a block in the `title` element. Clearly, when navigating backwards to a previous state the user does not go back in time, and therefore it would be inappropriate to put the time in the session history title.

```
<!DOCTYPE HTML>
<TITLE>Line</TITLE>
<SCRIPT>
setInterval(function () { document.title = 'Line - ' + new Date(); }, 1000);
var i = 1;
function inc() {
 set(i+1);
 history.pushState(i, 'Line - ' + i);
}
function set(newI) {
 i = newI;
 document.forms.F.I.value = newI;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

&lt;/FORM&gt;

### 6.5.3 The Location interface

Each `Document` object in a browsing context's session history is associated with a unique instance of a `Location` object.

*This box is non-normative. Implementation requirements are given below this box.*

`document`

`window` Too slow? Try reading the multipage copy of  
<sub>Ref</sub> the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

`Call`

The `location` attribute of the `Window` interface must return the `Location` object for that `Window` object's `Document`.  
 object, if it is in a browsing context, and `null`.

The `location` attribute of the `Window` interface must return the `Location` object for that `Window` object's `Document`.

`Location` objects provide a representation of the address of the active document of their `Document`'s browsing context, and allow the current entry of the browsing context's session history to be changed, by adding or replacing entries in the `history` object.

```
[Unforgeable] interface Location {
 stringifier attribute DOMString href;
 void assign(DOMString url);
 void replace(DOMString url);
 void reload();

 // URL decomposition IDL attributes
 attribute DOMString protocol;
 attribute DOMString host;
 attribute DOMString hostname;
 attribute DOMString port;
 attribute DOMString pathname;
 attribute DOMString search;
 attribute DOMString hash;
};
```

*This box is non-normative. Implementation requirements are given below this box.*

`location.href [= value]`

Returns the current page's location.

Can be set, to navigate to another page.

`location.assign(url)`

Navigates to the given page.

`location.replace(url)`

Removes the current page from the session history and navigates to the given page.

`location.reload()`

Reloads the current page.

The `relevant Document` is the `Location` object's associated `Document` object's browsing context's active document.

The `href` attribute must return the address of the `relevant Document`, as an absolute URL.

On setting, if the `Location` object's `relevant Document` has completely loaded, then the user agent must act as if the `assign()`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Document, and that was the `about:blank` Document created when the browsing context was created, then the navigation must be done with replacement enabled.

When the `replace(url)` method is invoked, the UA must resolve the argument, relative to the entry script's base URL, and if that is successful, navigate the browsing context to the specified `url` with replacement enabled.

Navigation for the `assign()` and `replace()` methods must be done with the browsing context of the script that invoked the method as the source browsing context.

If the resolving

SyntaxErr

Too slow? Try reading the multipage copy of

When the re

the spec instead: <https://web.archive.org>

↳ If the cl

/web/20130103030411/http:

↳ If the b

//whatwg.org/html

↳ If the browsing context's active document has its reload override flag set

Perform an overridden reload.

↳ Otherwise

Navigate the browsing context to the document's address with replacement enabled. The source browsing context must be the browsing context being navigated.

When a user requests that the active document of a browsing context be reloaded through a user interface element, the user agent should navigate the browsing context to the same resource as that Document, with replacement enabled. In the case of non-idempotent methods (e.g. HTTP POST), the user agent should prompt the user to confirm the operation first, since otherwise transactions (e.g. purchases or database modifications) could be repeated. User agents may allow the user to explicitly override any caches when reloading. If browsing context's active document's reload override flag is set, then the user agent may instead perform an overridden reload rather than the navigation described in this paragraph.

The `Location` interface also has the complement of URL decomposition IDL attributes, `protocol`, `host`, `port`, `hostname`, `pathname`, `search`, and `hash`. These must follow the rules given for URL decomposition IDL attributes, with the input being the address of the *relevant Document*, as an absolute URL (same as the `href` attribute), and the common setter action being the same as setting the `href` attribute to the new output value.

### 6.5.3.1 Security

User agents must throw a `SecurityError` exception whenever any properties of a `Location` object are accessed by scripts whose effective script origin is not the same as the `Location` object's associated Document's browsing context's active document's effective script origin, with the following exceptions:

- The `href` setter, if the script is running in a browsing context that is allowed to navigate the browsing context with which the `Location` object is associated
- The `replace()` method, if the script is running in a browsing context that is allowed to navigate the browsing context with which the `Location` object is associated

When a script whose effective script origin is not the same as the `Location` object's associated Document's effective script origin attempts to access that `Location` object's methods or attributes, the user agent must act as if any changes to the `Location` object's properties, getters, setters, etc, were not present.

For members that return objects (including function objects), each distinct effective script origin that is not the same as the `Location` object's Document's effective script origin must be provided with a separate set of objects. These objects must have the prototype chain appropriate for the script for which the objects are created (not those that would be appropriate for scripts whose script's global object is the `Location` object's Document's `Window` object).

### 6.5.4 Implementation notes for session history

*This section is non-normative.*

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Similarly, a page containing two `iframes` has a `history` object distinct from the `iframes`' `history` objects, despite the fact that typical Web browsers present the user with just one "Back" button, with a session history that interleaves the navigation of the two inner frames and the outer page.

**Security:** It is suggested that to avoid letting a page "hijack" the history navigation facilities of a UA by abusing `pushState()`, the UA provide the user with a way to jump back to the previous page (rather than just going back to the previous state). For example, the back button could have a drop down showing just the pages in the session history, and not showing any of the states. Similarly, an aural browser could have two "back" commands, one that goes back to the previous state, and one that jumps straight back to the previous page.

In addition, to the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**6.6 Browsing contexts**

### 6.6.1 Navigating across documents

Certain actions cause the browsing context to *navigate* to a new resource. Navigation always involves **source browsing context**, which is the browsing context which was responsible for starting the navigation.

For example, following a hyperlink, form submission, and the `window.open()` and `location.assign()` methods can all cause a browsing context to navigate.

A user agent may provide various ways for the user to explicitly cause a browsing context to navigate, in addition to those defined in this specification.

When a browsing context is **navigated** to a new resource, the user agent must run the following steps:

1. Release the storage mutex.
2. If the source browsing context is not the same as the browsing context being navigated, and the source browsing context is not one of the ancestor browsing contexts of the browsing context being navigated, and the browsing context being navigated is not a top-level browsing context, and the source browsing context's active document's active sandboxing flag set has its sandboxed navigation browsing context flag set, then abort these steps.

Otherwise, if the browsing context being navigated is a top-level browsing context, and is one of the ancestor browsing contexts of the source browsing context, and the source browsing context's Document's active sandboxing flag set has its sandboxed top-level navigation browsing context flag set, then abort these steps.

Otherwise, if the browsing context being navigated is a top-level browsing context, and is not one of the ancestor browsing contexts of the source browsing context, and the source browsing context's Document's active sandboxing flag set has its sandboxed navigation browsing context flag set, and the source browsing context is not the one permitted sandboxed navigator of the browsing context being navigated, then abort these steps.

In all of these cases, the user agent may additionally offer to open the new resource in a new top-level browsing context or in the top-level browsing context of the source browsing context, at the user's option, in which case the user agent must navigate that designated top-level browsing context to the new resource as if the user had requested it independently.

**Note: Doing so, however, can be dangerous, as it means that the user is overriding the author's explicit request to sandbox the content.**

3. If the source browsing context is the same as the browsing context being navigated, and this browsing context has its seamless browsing context flag set, and the browsing context being navigated was not chosen using an **explicit self-navigation override**, then find the nearest ancestor browsing context that does not have its seamless browsing context flag set, and continue these steps as if *that* browsing context was the one that was going to be navigated instead.
4. If there is a preexisting attempt to navigate the browsing context, and the source browsing context is the same as the browsing context being navigated, and that attempt is currently running the unload a document algorithm, and the origin of the URL of the resource being loaded in that navigation is not the same origin as the origin of the URL of the resource being loaded in *this* navigation, then abort these steps without affecting the preexisting attempt to navigate the browsing context.
5. If a task queued by the traverse the history by a delta algorithm is running the unload a document algorithm for the active

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

7. Let `gone async` be false.

8. *Fragment identifiers*: If the absolute URL of the new resource is the same as the address of the active document of the browsing context being navigated, ignoring any `<fragment>` components of those URLs, and the new resource is to be fetched using HTTP GET or equivalent, and the absolute URL of the new resource has a `<fragment>` component (even if it is empty), then navigate to that fragment identifier and abort these steps.

9. If `gone async` is false, cancel any preexisting but not yet mature attempt to navigate the browsing context, including canceling any instances of the fetch algorithm started by those attempts. If one of those attempts already created a new Document object has already created a new Document object, and are history entries, and are

ther Too slow? Try reading the multipage copy of

10. If the req with /web/20130103030411/http://web.archive.org

e.g. ignoring the navigation of these steps and proceed

11. If go abo //whatwg.org/html

iment to be unloaded, then

If this instance of the navigation algorithm gets canceled while this step is running, the prompt to unload a document algorithm must nonetheless be run to completion.

12. If `gone async` is false, abort the active document of the browsing context.

13. If the new resource is to be handled by displaying some sort of inline content, e.g. an error message because the specified scheme is not one of the supported protocols, or an inline prompt to allow the user to select a registered handler for the given scheme, then display the inline content and abort these steps.

**Note:** In the case of a registered handler being used, the algorithm will be reinvoked with a new URL to handle the request.

14. If the resource has already been obtained (e.g. because it is being used to populate an `object` element's new child browsing context), then skip this step.

Otherwise:

If the new resource is to be fetched using HTTP GET or equivalent, and there are relevant application caches that are identified by a URL with the same origin as the URL in question, and that have this URL as one of their entries, excluding entries marked as foreign, and whose mode is fast, and the user agent is not in a mode where it will avoid using application caches then get the resource from the most appropriate application cache of those that match.

For example, imagine an HTML page with an associated application cache displaying an image and a form, where the image is also used by several other application caches. If the user right-clicks on the image and chooses "View Image", then the user agent could decide to show the image from any of those caches, but it is likely that the most useful cache for the user would be the one that was used for the aforementioned HTML page. On the other hand, if the user submits the form, and the form does a POST submission, then the user agent will not use an application cache at all; the submission will be made to the network.

Otherwise, fetch the new resource, with the `manual redirect flag` set.

If the resource is being fetched using a method other than one equivalent to HTTP's GET, or, if the navigation algorithm was invoked as a result of the form submission algorithm, then the fetching algorithm must be invoked from the origin of the active document of the source browsing context, if any.

If the browsing context being navigated is a child browsing context for an `iframe` or `object` element, then the fetching algorithm must be invoked from the `iframe` or `object` element's browsing context scope origin, if it has one.

If the browsing context is a nested browsing context, then in the time between the fetch algorithm being started by this step, and either the creation of a `Document` object or the canceling of the fetch or navigation algorithms, the browsing context must be put in the delaying `load events` mode.

15. If `gone async` is false, the user agents must return to whatever algorithm invoked the navigation steps and must continue running these steps asynchronously.

16. Let `gone async` be true.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

So for instance, if the original URL was "`http://example.com/#!sample`" and "`http://example.com/`" is found to redirect to "`https://example.com/`", the URL of the new resource will be "`https://example.com/#!sample`".

Otherwise, if fetching the resource results in a redirect but the URL of the target of the redirect does not have the same origin as the original resource and the resource is being obtained using a method that is neither the POST method nor a safe method (in HTTP terms), then abort these steps. The user agent may indicate to the user that the navigation has been aborted for security reasons.



18. Wait until the navigation attempt has completed, and if the navigation attempt failed (e.g. the server returned a 4xx or 5xx status code or equivalent, or there was a DNS error), then:
19. If the navigation attempt failed, then if the user agent has a fallback resource specified for the fallback namespace in question, let `candidate` be the fallback resource. If multiple application caches match, the user agent must use the fallback of the most appropriate application cache of those that match.

estion is empty. During this attempt.

For equivalent, and its URL is a previously cached resource's URL and that is a previous step, and the navigation attempt failed (e.g. the server returned a 4xx or 5xx status code or equivalent, or there was a DNS error), then:

20. If the resource was not fetched from an application cache, and was to be fetched using HTTP GET or equivalent, and there are relevant application caches that are identified by a URL with the same origin as the URL in question, and that have this URL as one of their entries, excluding entries marked as foreign, and whose mode is prefer-online, and the user didn't cancel the navigation attempt during the earlier step, and the navigation attempt failed (e.g. the server returned a 4xx or 5xx status code or equivalent, or there was a DNS error), then:

Let `candidate` be the resource identified by the URL in question from the most appropriate application cache of those that match.

If `candidate` is not marked as foreign, then the user agent must discard the failed load and instead continue along these steps using `candidate` as the resource. The user agent may indicate to the user that the original page load failed, and that the page used was a previously cached resource.

21. *Resource handling*: If the resource's out-of-band metadata (e.g. HTTP headers), not counting any type information (such as the Content-Type HTTP header), requires some sort of processing that will not affect the browsing context, then perform that processing and abort these steps.

#### **Such processing might be triggered by, amongst other things, the following:**

- **HTTP status codes** (e.g. 204 No Content or 205 Reset Content)
- **Network errors** (e.g. the network interface being unavailable)
- **Cryptographic protocol failures** (e.g. an incorrect TLS certificate)

Responses with HTTP Content-Disposition headers specifying the attachment disposition type must be handled as a download.

HTTP 401 responses that do not include a challenge recognized by the user agent must be processed as if they had no challenge, e.g. rendering the entity body as if the response had been 200 OK.

User agents may show the entity body of an HTTP 401 response even when the response does include a recognized challenge, with the option to login being included in a non-modal fashion, to enable the information provided by the server to be used by the user before authenticating. Similarly, user agents should allow the user to authenticate (in a non-modal fashion) against authentication challenges included in other responses such as HTTP 200 OK responses, effectively allowing resources to present HTTP login forms without requiring their use.

22. Let `type` be the sniffed type of the resource.
23. If the user agent has been configured to process resources of the given `type` using some mechanism other than rendering the content in a browsing context, then skip this step. Otherwise, if the `type` is one of the following types, jump to the appropriate

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

- ↳ "application/xml"
- ↳ "text/xml"
- ↳ "image/svg+xml"
- ↳ "application/xhtml+xml"
- ↳ Any other type ending in "+xml" that is not an explicitly supported XML type

Follow the steps given in the XML document section. If that section determines that the content is *not* to be displayed as a generic XML document, then proceed to the next step in this overall set of steps. Otherwise, abort these steps.



- ↳ Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
- ↳ A type that will use an external application to render the content in the browsing context

Follow the steps given in the plugin section, and abort these steps.

An **explicitly supported XML type** is one for which the user agent is configured to use an external application to render the content (either a plugin rendering directly in the browsing context, or a separate application), or one for which the user agent has dedicated processing rules (e.g. a Web browser with a built-in Atom feed viewer would be said to explicitly support the application/atom+xml MIME type), or one for which the user agent has a dedicated handler (e.g. one registered using registerContentHandler()).

**Setting the document's address:** If there is no **override URL**, then any `Document` created by these steps must have its address set to the URL that was originally to be fetched, ignoring any other data that was used to obtain the resource (e.g. the entity body in the case of a POST submission is not part of the document's address, nor is the URL of the fallback resource in the case of the original load having failed and that URL having been found to match a fallback namespace). However, if there *is* an override URL, then any `Document` created by these steps must have its address set to that URL instead.

**Note:** An override URL is set when dereferencing a javascript: URL.

**Creating a new Document object:** when a `Document` is created as part of the above steps, the user agent must additionally run the following algorithm as part of creating the new object:

1. Create a new `Window` object, and associate it with the `Document`, with one exception: if the browsing context's only entry in its session history is the `about:blank` `Document` that was added when the browsing context was created, and navigation is occurring with replacement enabled, and that `Document` has the same origin as the new `Document`, then use the `Window` object of that `Document` instead, and change the `document` attribute of the `Window` object to point to the new `Document`.
2. Implement the sandboxing for the `Document`.
3. If the active sandboxing flag set of the `Document`'s browsing context or any of its ancestor browsing contexts (if any) have the sandboxed fullscreen browsing context flag set, then skip this step.

If the `Document`'s browsing context has a browsing context container and it is either not an `iframe` element, or it does not have the `allowfullscreen` attribute specified, or its `Document` does not have the `fullscreen` enabled flag set, then also skip this step.

Otherwise, set the `Document`'s `fullscreen` enabled flag.

24. **Non-document content:** If, given `type`, the new resource is to be handled by displaying some sort of inline content, e.g. a native rendering of the content, an error message because the specified type is not supported, or an inline prompt to allow the user to select a registered handler for the given type, then display the inline content and abort these steps.

**Note:** In the case of a registered handler being used, the algorithm will be reinvoked with a new URL to handle the request.

25. Otherwise, the document's `type` is such that the resource will not affect the browsing context, e.g. because the resource is to

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**history with the new page.** When a user agent is required to do this, it must queue a task (associated with the `Document` object of the current entry, not the new one) to run the following steps:

1. Unload the `Document` object of the current entry, with the `recycle` parameter set to false.

If this instance of the navigation algorithm is canceled while this step is running the unload a document algorithm, then the unload a document algorithm must be allowed to run to completion, but this instance of the navigation algorithm must not run beyond this step. (In particular, for instance, the cancelation of this algorithm does not abort any event dispatch or script execution occurring as part of unloading the document or its descendants.)

2. If th

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

erenced the same document

and can never happen  
ion history entry thatno longer has a Document object)**Otherwise**

1. Remove all the entries in the browsing context's session history after the current entry. If the current entry is the last entry in the session history, then no entries are removed.

**Note:** This doesn't necessarily have to affect the user agent's user interface.

2. Remove any tasks queued by the history traversal task source that are associated with any `Document` objects in the top-level browsing context's document family.
3. Append a new entry at the end of the `History` object representing the new resource and its `Document` object and related state.
4. Traverse the history to the new entry. If the navigation was initiated with replacement enabled, then the traversal must itself be initiated with replacement enabled.

3. The navigation algorithm has now **matured**.

4. *Fragment identifier loop:* Spin the event loop for a user-agent-defined amount of time, as desired by the user agent implementor. (This is intended to allow the user agent to optimize the user experience in the face of performance concerns.)

5. If the `Document` object has no parser, or its parser has stopped parsing, or the user agent has reason to believe the user is no longer interested in scrolling to the fragment identifier, then abort these steps.

6. Scroll to the fragment identifier given in the document's address. If this fails to find an indicated part of the document, then return to the *fragment identifier loop* step.

The task source for this task is the networking task source.

### 6.6.2 Page load processing model for HTML files

When an HTML document is to be loaded in a browsing context, the user agent must queue a task to create a `Document` object, mark it as being an HTML document, set its content type to "text/html", create an HTML parser, and associate it with the document. Each task that the networking task source places on the task queue while the fetching algorithm runs must then fill the parser's input byte stream with the fetched bytes and cause the HTML parser to perform the appropriate processing of the input stream.

**Note:** The input byte stream converts bytes into characters for use in the tokenizer. This process relies, in part, on character encoding information found in the real Content-Type metadata of the resource; the "sniffed type" is not used for this purpose.

When no more bytes are available, the user agent must queue a task for the parser to process the implied EOF character, which eventually causes a `load` event to be fired.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

The task source for the two tasks mentioned in this section must be the networking task source.

### 6.6.3 Page load processing model for XML files

When faced with displaying an XML file inline, user agents must first create a `Document` object, following the requirements of the XML and Namespaces in XML recommendations, RFC 3023, DOM, and other relevant specifications. [XML] [XMLNS] [RFC3023] [DOM]

The actual HTTP headers and other metadata, not the headers as mutated or implied by the `allexmlms` given in this specification, are the ones that character er

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

If the root el manifest v resolve the \ with the rest Document. element is in. in the Document.

ment with an attribute t, the user agent must cache selection algorithm in the newly-created n as soon as the root with no manifest, and passing

**Note:** Because the processing of the `manifest` attribute happens only once the root element is parsed, any URLs referenced by processing instructions before the root element (such as `<?xml-stylesheet?>` and `<?xbl?>`) will be fetched from the network and cannot be cached.

User agents may examine the namespace of the root `Element` node of this `Document` object to perform namespace-based dispatch to alternative processing tools, e.g. determining that the content is actually a syndication feed and passing it to a feed handler. If such processing is to take place, abort the steps in this section, and jump to the next step (labeled "non-document content") in the navigate steps above.

Otherwise, then, with the newly created `Document`, the user agents must update the session history with the new page. User agents may do this before the complete document has been parsed (thus achieving *incremental rendering*), and must do this before any scripts are to be executed.

Error messages from the parse process (e.g. XML namespace well-formedness errors) may be reported inline by mutating the `Document`.

### 6.6.4 Page load processing model for text files

When a plain text document is to be loaded in a browsing context, the user agent must queue a task to create a `Document` object, mark it as being an HTML document, set its content type to "`text/plain`", create an HTML parser, associate it with the document, act as if the tokenizer had emitted a start tag token with the tag name "`pre`" followed by a single U+000A LINE FEED (LF) character, and switch the HTML parser's tokenizer to the PLAINTEXT state. Each task that the networking task source places on the task queue while the fetching algorithm runs must then fill the parser's input byte stream with the fetched bytes and cause the HTML parser to perform the appropriate processing of the input stream.

The rules for how to convert the bytes of the plain text document into actual characters, and the rules for actually rendering the text to the user, are defined in RFC 2046, RFC 3676, and subsequent versions thereof. [RFC2046] [RFC3676]

The document's character encoding must be set to the character encoding used to decode the document.

Upon creation of the `Document` object, the user agent must run the application cache selection algorithm with no manifest, and passing in the newly-created `Document`.

When no more bytes are available, the user agent must queue a task for the parser to process the implied EOF character, which eventually causes a `load` event to be fired.

After creating the `Document` object, but potentially before the page has finished parsing, the user agent must update the session history with the new page.

User agents may add content to the `head` element of the `Document`, e.g. linking to a style sheet or an XBL binding, providing script, giving the document a title, etc.

**Note:** In particular, if the user agent supports the `Format=Flowed` feature of RFC 3676 then the user agent would

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

### 6.6.5 Page load processing model for multipart/x-mixed-replace resources

When a resource with the type `multipart/x-mixed-replace` is to be loaded in a browsing context, the user agent must parse the resource using the rules for multipart types. [RFC2046]

For each body part obtained from the resource, the user agent must run a new instance of the navigate algorithm, starting from the *resource handling* step, using the new body part as the resource being navigated, with replacement enabled if a previous body part from the same resource resulted in a `Document` object being created, and otherwise using the setup as the navigate attempt that caused this `s`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

s, the user agent must act ed.

**Note:** T /[web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

### 6.6.6 Page

When an image, video, or audio resource is to be loaded in a browsing context, the user agent should create a `Document` object, mark it as being an HTML document, set its content type to the sniffed MIME type of the resource (*type* in the navigate algorithm), append an `html` element to the `Document`, append a `head` element and a `body` element to the `html` element, append an element *host element* for the media, as described below, to the `body` element, and set the appropriate attribute of the element *host element*, as described below, to the address of the image, video, or audio resource.

The element *host element* to create for the media is the element given in the table below in the second cell of the row whose first cell describes the media. The appropriate attribute to set is the one given by the third cell in that same row.

Type of media	Element for the media	Appropriate attribute
Image	<code>img</code>	<code>src</code>
Video	<code>video</code>	<code>src</code>
Audio	<code>audio</code>	<code>src</code>

Then, the user agent must act as if it had stopped parsing.

Upon creation of the `Document` object, the user agent must run the application cache selection algorithm with no manifest, and passing in the newly-created `Document`.

After creating the `Document` object, but potentially before the page has finished fully loading, the user agent must update the session history with the new page.

User agents may add content to the `head` element of the `Document`, or attributes to the element *host element*, e.g. to link to a style sheet or an XBL binding, to provide a script, to give the document a `title`, to make the media autoplay, etc.

### 6.6.7 Page load processing model for content that uses plugins

When a resource that requires an external resource to be rendered is to be loaded in a browsing context, the user agent should create a `Document` object, mark it as being an HTML document, set its content type to the sniffed MIME type of the resource (*type* in the navigate algorithm), append an `html` element to the `Document`, append a `head` element and a `body` element to the `html` element, append an `embed` to the `body` element, and set the `src` attribute of the `embed` element to the address of the resource.

Then, the user agent must act as if it had stopped parsing.

Upon creation of the `Document` object, the user agent must run the application cache selection algorithm with no manifest, and passing in the newly-created `Document`.

After creating the `Document` object, but potentially before the page has finished fully loading, the user agent must update the session history with the new page.

User agents may add content to the `head` element of the `Document`, or attributes to the `embed` element, e.g. to link to a style sheet or an XBL binding, or to give the document a `title`.

**Note:** If the `Document`'s active sandboxing flag set has its sandboxed plugins browsing context flag set, the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the user agent is to display a user agent page inline in a browsing context, the user agent should create a `Document` object, mark it as being an HTML document, set its content type to "text/html", and then either associate that `Document` with a custom rendering that is not rendered using the normal `Document` rendering rules, or mutate that `Document` until it represents the content the user agent wants to render.

Once the page has been set up, the user agent must act as if it had stopped parsing.

Upon creation of the `Document` object, the user agent must run the application cache selection algorithm with no manifest, passing in the newly-created `Document`.

After creatin Too slow? Try reading the multipage copy of gent must update the session hist the spec instead: <https://web.archive.org>

## 6.6.9 Nav /web/20130103030411/http:

When a use //whatwg.org/html

to run the following steps:

1. Remove any entries from the session history. If the current entry is the last entry in the session history, then no entries are removed.

**Note: This doesn't necessarily have to affect the user agent's user interface.**

2. Remove any tasks queued by the history traversal task source that are associated with any `Document` objects in the top-level browsing context's document family.
3. Append a new entry at the end of the `History` object representing the new resource and its `Document` object and related state. Its URL must be set to the address to which the user agent was navigating. The title must be left unset.
4. Traverse the history to the new entry. This will scroll to the fragment identifier given in what is now the document's address.

**Note: If the scrolling fails because the relevant ID has not yet been parsed, then the original navigation algorithm will take care of the scrolling instead, as the last few steps of its update the session history with the new page algorithm.**

When the user agent is required to **scroll to the fragment identifier**, it must either change the scrolling position of the document using the scroll an element into view algorithm defined in the CSSOM View specification, with the `align to top` flag set, or perform some other action, such that the indicated part of the document is brought to the user's attention. If there is no indicated part, or if the indicated part is not being rendered, then the user agent must not scroll anywhere. [CSSOMVIEW]

**The indicated part of the document** is the one that the fragment identifier, if any, identifies. The semantics of the fragment identifier in terms of mapping it to a specific DOM Node is defined by the specification that defines the MIME type used by the `Document` (for example, the processing of fragment identifiers for XML MIME types is the responsibility of RFC3023). [RFC3023]

For HTML documents (and HTML MIME types), the following processing model must be followed to determine what the indicated part of the document is.

1. Parse the URL, and let `fragid` be the <fragment> component of the URL.
2. If `fragid` is the empty string, then the indicated part of the document is the top of the document; stop the algorithm here.
3. Let `decoded fragid` be the result of expanding any sequences of percent-encoded octets in `fragid` that are valid UTF-8 sequences into Unicode characters as defined by UTF-8. If any percent-encoded octets in that string are not valid UTF-8 sequences (e.g. they expand to surrogate code points), then skip this step and the next one.
4. If this step was not skipped and there is an element in the DOM that has an ID exactly equal to `decoded fragid`, then the first such element in tree order is the indicated part of the document; stop the algorithm here.
5. If there is an element in the DOM that has a `name` attribute whose value is exactly equal to `fragid` (*not decoded fragid*), then the first such element in tree order is the indicated part of the document; stop the algorithm here.
6. If `fragid` is an ASCII case-insensitive match for the string `top`, then the indicated part of the document is the top of the document; stop the algorithm here.
7. Otherwise, there is no indicated part of the document.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

## 6.6.10 History traversal

When a user agent is required to **traverse the history** to a *specified entry*, optionally with replacement enabled, the user agent must act as follows.

**Note:** This algorithm is not just invoked when explicitly going back or forwards in the session history — it is also invoked in other situations, for example when navigating a browsing context, as part of updating the session history with the new page.

1. If the

perf  
corr  
usir  
repl

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the location for that entry to its "traverse" algorithm to be navigation must be done an never happen with

2. If the

retu

et its title to the value

3. If app.

object to reflect any state that

the user agent wishes to persist. The entry is then said to be an entry with persisted user state.

4. If the *specified entry* has a different Document object than the current entry then the following substeps:

1. If the origin of the Document of the *specified entry* is not the same as the origin of the Document of the current entry, then run the following sub-sub-steps:

1. The current browsing context name must be stored with all the entries in the history that are associated with Document objects with the same origin as the active document *and* that are contiguous with the current entry.

2. If the browsing context is a top-level browsing context, but not an auxiliary browsing context, then the browsing context's browsing context name must be unset.

2. Make the *specified entry*'s Document object the active document of the browsing context.

3. If the *specified entry* has a browsing context name stored with it, then run the following sub-sub-steps:

1. Set the browsing context's browsing context name to the name stored with the specified entry.

2. Clear any browsing context names stored with all entries in the history that are associated with Document objects with the same origin as the new active document and that are contiguous with the specified entry.

4. If the *specified entry*'s Document has any form controls whose autofill field name is "off", invoke the reset algorithm of each of those elements.

5. If the current document readiness of the *specified entry*'s Document is "complete", queue a task to run the following sub-sub-steps:

1. If the Document's page showing flag is true, then abort this task (i.e. don't fire the event below).

2. Set the Document's page showing flag to true.

3. Fire a trusted event with the name pageshow at the Window object of that Document, but with its target set to the Document object (and the currentTarget set to the Window object), using the PageTransitionEvent interface, with the persisted attribute initialized to true. This event must not bubble, must not be cancelable, and has no default action.

5. Set the document's address to the URL of the *specified entry*.

6. If the *specified entry* has a URL whose fragment identifier differs from that of the current entry's when compared in a case-sensitive manner, and the two share the same Document object, then let hash changed be true, and let old URL be the URL of the current entry and new URL be the URL of the *specified entry*. Otherwise, let hash changed be false.

7. If the traversal was initiated with **replacement enabled**, remove the entry immediately before the *specified entry* in the session history.

8. If the *specified entry* is not an entry with persisted user state, but its URL has a fragment identifier, scroll to the fragment identifier.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*attribute is in either the Text State or the Search State, if the persisted state includes the directionality of user input in such controls.*

10. If the entry is a state object entry, let `state` be a structured clone of that state object. Otherwise, let `state` be null.
11. Set `history.state` to `state`.
12. Let `state changed` be true if the Document of the specified entry has a latest entry, and that entry is not the specified entry; otherwise let it be false.
13. Let `t`
14. If `st` Pop can the `URL`
15. If `h` the `URL`
16. The current entry is now the specified entry.

The task source for the tasks mentioned above is the DOM manipulation task source.

#### 6.6.10.1 Event definitions

The `popstate` event is fired in certain cases when navigating to a session history entry.

```
[Constructor(DOMString type, optional PopStateEventInit eventInitDict)]
interface PopStateEvent : Event {
 readonly attribute any state;
};

dictionary PopStateEventInit : EventInit {
 any state;
};
```

*This box is non-normative. Implementation requirements are given below this box.*

`event.state`

Returns a copy of the information that was provided to `pushState()` or `replaceState()`.

The `state` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the context information for the event, or null, if the state represented is the initial state of the Document.

The `hashchange` event is fired when navigating to a session history entry whose URL differs from that of the previous one only in the fragment identifier.

```
[Constructor(DOMString type, optional HashChangeEventInit eventInitDict)]
interface HashChangeEvent : Event {
 readonly attribute DOMString oldURL;
 readonly attribute DOMString newURL;
};

dictionary HashChangeEventInit : EventInit {
 DOMString oldURL;
 DOMString newURL;
};
```

*This box is non-normative. Implementation requirements are given below this box.*

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Returns the URL of the session history entry that is now current.

The `oldURL` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents context information for the event, specifically the URL of the session history entry that was traversed from.

The `newURL` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents context information for the event, specifically the URL of the session history entry that was traversed to.

The `page` history entry  
they never r **Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org**

traversing from a session  
consistent manner (e.g. that

```
[Con
inte
re
//whatwg.org/html
};

dictionary PageTransitionEventInit : EventInit {
 boolean persisted;
};
```

st) ]

This box is non-normative. Implementation requirements are given below this box.

**`event.persisted`**Returns false if the page is newly being loaded (and the `load` event will fire). Otherwise, returns true.

The `persisted` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to false. It represents the context information for the event.

### 6.6.11 Unloading documents

A Document has a **salvageable** state, which must initially be true, a **fired unload** flag, which must initially be false, and a **page showing** flag, which must initially be false.

Event loops have a **termination nesting level** counter, which must initially be zero.

When a user agent is to **prompt to unload a document**, it must run the following steps.

1. Increase the event loop's termination nesting level by one.
2. Increase the Document's ignore-opens-during-unload counter by one.
3. Let `event` be a new trusted `BeforeUnloadEvent` event object with the name `beforeunload`, which does not bubble but is cancelable.
4. *Dispatch:* Dispatch `event` at the Document's Window object.
5. Decrease the event loop's termination nesting level by one.
6. Release the storage mutex.
7. If any event listeners were triggered by the earlier *dispatch* step, then set the Document's **salvageable** state to false.
8. If the `returnValue` attribute of the `event` object is not the empty string, or if the event was canceled, then the user agent should ask the user to confirm that they wish to unload the document.

The prompt shown by the user agent may include the string of the `returnValue` attribute, or some leading subset thereof. (A user agent may want to truncate the string to 1024 characters for display, for instance.)

The user agent must pause while waiting for the user's response.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

10. Let `descendants` be the list of the descendant browsing contexts of the Document.11. If `descendants` is not an empty list, then for each browsing context `b` in `descendants` run the following substeps:1. Prompt to unload the active document of the browsing context `b`. If the user refused to allow the document to be unloaded, then the user implicitly also refused to allow *this* document to be unloaded; jump to the step labeled `end`.2. If `salvageable` state of the active document of the browsing context `b` is false, then set the `salvageable` state of *this* document to false also.12. *End:*

When a user either true or false

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

argument, `recycle`, which is implemented by `Document.open()` method.)

1. Incr

<https://web.archive.org/web/20130103030411/http://whatwg.org/html>

2. Incr

<https://web.archive.org/web/20130103030411/http://whatwg.org/html>

3. If the

even and don't rerun the unloading document visibility change steps).

.e. skip firing the `pagehide`

4. Set the Document's page showing flag to false.

5. Fire a trusted event with the name `pagehide` at the `Window` object of the Document, but with its `target` set to the `Document` object (and the `currentTarget` set to the `Window` object), using the `PageTransitionEvent` interface, with the `persisted` attribute initialized to true. This event must not bubble, must not be cancelable, and has no default action.6. Run any **unloading document visibility change steps** for Document that are defined by other applicable specifications.**Note: This is specifically intended for use by the Page Visibility specification. [PAGEVIS]**7. *Unload event:* If the Document's fired `unload` flag is false, fire a simple event named `unload` at the Document's `Window` object.

8. Decrease the event loop's termination nesting level by one.

9. Release the storage mutex.

10. If any event listeners were triggered by the earlier `unload` event step, then set the Document object's `salvageable` state to false and set the Document's fired `unload` flag to true.

11. Run any unloading document cleanup steps for Document that are defined by this specification and other applicable specifications.

12. If this algorithm was invoked by another instance of the "unload a document" algorithm (i.e. by the steps below that invoke this algorithm for all descendant browsing contexts), then jump to the step labeled `end`.13. Let `descendants` be the list of the descendant browsing contexts of the Document.14. If `descendants` is not an empty list, then for each browsing context `b` in `descendants` run the following substeps:1. Unload the active document of the browsing context `b` with the `recycle` parameter set to false.2. If `salvageable` state of the active document of the browsing context `b` is false, then set the `salvageable` state of *this* document to false also.15. If `salvageable` and `recycle` are both false, then the Document's browsing context must discard the Document.16. *End:* Decrease the Document's ignore-opens-during-unload counter by one.This specification defines the following **unloading document cleanup steps**. Other specifications can define more.1. Make disappear any `WebSocket` objects that were created by the `WebSocket()` constructor whose global object is the Document's `Window` object.If this affected any `WebSocket` objects, the set Document's `salvageable` state to false.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

### 6.6.11.1 Event definition

```
interface BeforeUnloadEvent : Event {
 attribute DOMString returnValue;
};
```

*This box is non-normative. Implementation requirements are given below this box.*

**event . . .**

**Re**l Too slow? Try reading the multipage copy of  
**Ca**l the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**Note:** [T](#) <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `returnValue` attribute represents the message to show the user. When the event is created, the attribute must be set to the empty string. On getting, it must return the last value it was set to. On setting, the attribute must be set to the new value.

### 6.6.12 Aborting a document load

If a `Document` is **aborted**, the user agent must run the following steps:

1. Abort the active documents of every child browsing context. If this results in any of those `Document` objects having their `salvageable` state set to false, then set this `Document`'s `salvageable` state to false also.
2. Cancel any instances of the fetch algorithm in the context of this `Document`, discarding any tasks queued for them, and discarding any further data received from the network for them. If this resulted in any instances of the fetch algorithm being canceled or any queued tasks or any network data getting discarded, then set the `Document`'s `salvageable` state to false.
3. If the `Document` has an active parser, then abort that parser and set the `Document`'s `salvageable` state to false.

User agents may allow users to explicitly invoke the abort a document algorithm for a `Document`. If the user does so, then, if that `Document` is an active document, the user agent should queue a task to fire a simple event named `abort` at that `Document`'s `Window` object before invoking the abort algorithm.

## 6.7 Offline Web applications

### 6.7.1 Introduction

*This section is non-normative.*

In order to enable users to continue interacting with Web applications and documents even when their network connection is unavailable — for instance, because they are traveling outside of their ISP's coverage area — authors can provide a manifest which lists the files that are needed for the Web application to work offline and which causes the user's browser to keep a copy of the files for use offline.

To illustrate this, consider a simple clock applet consisting of an HTML page "`clock.html`", a CSS style sheet "`clock.css`", and a JavaScript script "`clock.js`".

Before adding the manifest, these three files might look like this:

```
<!-- clock.html -->
<!DOCTYPE HTML>
<html>
 <head>
 <title>Clock</title>
 <script src="clock.js"></script>
 <link rel="stylesheet" href="clock.css">
 </head>
```

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

```
/* clock.css */
output { font: 2em sans-serif; }

/* clock.js */
setInterval(function () {
 document.getElementById('clock').value = new Date();
}, 1000);
```

If the user tries to open the "clock.html" page while offline, though, the user agent (unless it happens to have it still in the local cache) will fail.

The author suggests reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/cACHE-clock.html>

With a small

pplication:

```
<!-- clock.html -->
<!DOCTYPE HTML>
<html manifest="clock.appcache">
<head>
 <title>Clock</title>
 <script src="clock.js"></script>
 <link rel="stylesheet" href="clock.css">
</head>
<body>
 <p>The time is: <output id="clock"></output></p>
</body>
</html>
```

Now, if the user goes to the page, the browser will cache the files and make them available even when the user is offline.

**Note:** Authors are encouraged to include the main page in the manifest also, but in practice the page that referenced the manifest is automatically cached even if it isn't explicitly mentioned.

**Note:** With the exception of "no-store" directive, HTTP cache headers and restrictions on caching pages served over TLS (encrypted, using https:) are overridden by manifests. Thus, pages will not expire from an application cache before the user agent has updated it, and even applications served over TLS can be made to work offline.

[View this example online.](#)

### 6.7.1.1 Supporting offline caching for legacy applications

*This section is non-normative.*

The application cache feature works best if the application logic is separate from the application and user data, with the logic (markup, scripts, style sheets, images, etc) listed in the manifest and stored in the application cache, with a finite number of static HTML pages for the application, and with the application and user data stored in Web Storage or a client-side Indexed Database, updated dynamically using Web Sockets, XMLHttpRequest, server-sent events, or some other similar mechanism.

This model results in a fast experience for the user: the application immediately loads, and fresh data is obtained as fast as the network will allow it (possibly while stale data shows).

Legacy applications, however, tend to be designed so that the user data and the logic are mixed together in the HTML, with each operation resulting in a new HTML page from the server.

For example, consider a news application. The typical architecture of such an application, when not using the application cache feature, is that the user fetches the main page, and the server returns a dynamically-generated page with the current headlines and the user interface logic mixed together.

A news application designed for the application cache feature, however, would instead have the main page just consist of the

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

While there is no way to make the legacy model work as fast as the separated model, it can at least be retrofitted for offline use using the prefer-online application cache mode. To do so, list all the static resources used by the HTML page you want to have work offline in an application cache manifest, use the `manifest` attribute to select that manifest from the HTML file, and then add the following line at the bottom of the manifest:

```
SETTINGS:
prefer-online
NETWORK:
*
```



This causes used as an manifest to l

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

he application cache to be sources not listed in the

### 6.7.1.2 Ev /web/20130103030411/http:

*This section* //whatwg.org/html

When the user uses by fetching a copy of the manifest and, if the manifest has changed since the user agent last saw it, redownloading all the resources it mentions and caching them anew.

As this is going on, a number of events get fired on the `ApplicationCache` object to keep the script updated as to the state of the cache update, so that the user can be notified appropriately. The events are as follows:

Event name	Interface	Fired when...	Next events
<code>checking</code>	Event	The user agent is checking for an update, or attempting to download the manifest for the first time. <b>This is always the first event in the sequence.</b>	<code>noupdate</code> , <code>downloading</code> , <code>obsolete</code> , <code>error</code>
<code>noupdate</code>	Event	The manifest hadn't changed.	Last event in sequence.
<code>downloading</code>	Event	The user agent has found an update and is fetching it, or is downloading the resources listed by the manifest for the first time.	<code>progress</code> , <code>error</code> , <code>cached</code> , <code>updateready</code>
<code>progress</code>	ProgressEvent	The user agent is downloading resources listed by the manifest. The event object's <code>total</code> attribute returns the total number of files to be downloaded. The event object's <code>loaded</code> attribute returns the number of files processed so far.	<code>progress</code> , <code>error</code> , <code>cached</code> , <code>updateready</code>
<code>cached</code>	Event	The resources listed in the manifest have been downloaded, and the application is now cached.	Last event in sequence.
<code>updateready</code>	Event	The resources listed in the manifest have been newly redownloaded, and the script can use <code>swapCache()</code> to switch to the new cache.	Last event in sequence.
<code>obsolete</code>	Event	The manifest was found to have become a 404 or 410 page, so the application cache is being deleted.	Last event in sequence.
<code>error</code>	Event	The manifest was a 404 or 410 page, so the attempt to cache the application has been aborted. The manifest hadn't changed, but the page referencing the manifest failed to download properly. A fatal error occurred while fetching the resources listed in the manifest. The manifest changed while the update was being run.	Last event in sequence. The user agent will try fetching the files again momentarily.

These events are cancelable; their default action is for the user agent to show download progress information. If the page shows its own update UI, canceling the events will prevent the user agent from showing redundant progress information.

### 6.7.2 Application caches

An **application cache** is a set of cached resources consisting of:

- One or more resources (including their out-of-band metadata, such as HTTP headers, if any), identified by URLs, each falling into one (or more) of the following categories:

#### Master entries

**Note:** These are documents that were added to the cache because a browsing context was navigated to that document and the document indicated that this was its cache, using the `manifest` attribute.

#### The manifest

**Note:** This is the resource corresponding to the URL that was given in a master entry's `html`

Go NOV JAN FEB  
 03  
 2012 2013 2014

**3,445 captures**  
 5 Jun 2004 - 22 Aug 2020

Note: These are the resources that were listed in the cache's manifest in an explicit section.



About this capture

### Fallback entries

Note: These are the resources that were listed in the cache's manifest in a fallback section.

Explicit entries and Fallback entries can be marked as **foreign**, which means that they have a `manifest` attribute but that it doesn't point at this cache's manifest.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- Zero or more URLs that form the **online whitelist namespaces**.

Note: These are used as prefix match patterns, and declare URLs for which the user agent will ignore the application cache, instead fetching them normally (i.e. from the network or local HTTP cache as appropriate).

- An **online whitelist wildcard flag**, which is either `open` or `blocking`.

Note: The `open` state indicates that any URL not listed as cached is to be implicitly treated as being in the online whitelist namespaces; the `blocking` state indicates that URLs not listed explicitly in the manifest are to be treated as unavailable.

- A **cache mode flag**, which is either in the `fast` state or the `prefer-online` state.

Each application cache has a **completeness flag**, which is either `complete` or `incomplete`.

An **application cache group** is a group of application caches, identified by the absolute URL of a resource manifest which is used to populate the caches in the group.

An application cache is **newer** than another if it was created after the other (in other words, application caches in an application cache group have a chronological order).

Only the newest application cache in an application cache group can have its completeness flag set to `incomplete`; the others are always all `complete`.

Each application cache group has an **update status**, which is one of the following: `idle`, `checking`, `downloading`.

A **relevant application cache** is an application cache that is the newest in its group to be `complete`.

Each application cache group has a **list of pending master entries**. Each entry in this list consists of a resource and a corresponding `Document` object. It is used during the application cache download process to ensure that new master entries are cached even if the application cache download process was already running for their application cache group when they were loaded.

An application cache group can be marked as **obsolete**, meaning that it must be ignored when looking at what application cache groups exist.

A **cache host** is a `Document` or a `SharedWorkerGlobalScope` object. A cache host can be associated with an application cache.

A `Document` initially is not associated with an application cache, but can become associated with one early during the page load process, when steps in the parser and in the navigation sections cause cache selection to occur.

A `SharedWorkerGlobalScope` can be associated with an application cache when it is created.

Each cache host has an associated `ApplicationCache` object.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- which application cache was being used to display the resource from which the user decided to look at the new resource, and
- which application cache the user prefers.

A URL **matches a fallback namespace** if there exists a relevant application cache whose manifest's URL has the same origin as the URL in question, and that has a fallback namespace that is a prefix match for the URL being examined. If multiple fallback namespaces match the same URL, the longest one is the one that matches. A URL looking for a fallback namespace can match more than one application cache at a time, but only matches one namespace in each cache.



If a n  
fallb  
agen  
a ma  
Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

sources/images is a  
/cat.png, then the user  
: contains a namespace with

### 6.7.3 The `//whatwg.org/html`

#### 6.7.3.1 Son.

*This section is non-normative.*

This example manifest requires two images and a style sheet to be cached and whitelists a CGI script.

```
CACHE MANIFEST
the above line is required

this is a comment
there can be as many of these anywhere in the file
they are all ignored
comments can have spaces before them
but must be alone on the line

blank lines are ignored too

these are files that need to be cached they can either be listed
first, or a "CACHE:" header could be put before them, as is done
lower down.
images/sound-icon.png
images/background.png
note that each file has to be put on its own line

here is a file for the online whitelist -- it isn't cached, and
references to this file will bypass the cache, always hitting the
network (or trying to, if the user is offline).
NETWORK:
comm.cgi

here is another set of files to cache, this time just the CSS file.
CACHE:
style/default.css
```

It could equally well be written as follows:

```
CACHE MANIFEST
NETWORK:
comm.cgi
CACHE:
style/default.css
images/sound-icon.png
images/background.png
```

Offline application cache manifests can use absolute paths or even absolute URLs:

```
CACHE MANIFEST
```

Go

NOV JAN FEB

03

2012 2013 2014



## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

<http://img.example.com/logo.png>  
<http://img.example.com/check.png>  
<http://img.example.com/cross.png>

The following manifest defines a catch-all error page that is displayed for any page on the site while the user is offline. It also specifies that the online whitelist wildcard flag is *open*, meaning that accesses to resources on other sites will not be blocked. (Resources on the same site are already not blocked because of the catch-all fallback namespace.)

So long as all pages on the site reference this manifest, they will get cached locally as they are fetched, so that subsequent hits to the ↗

from **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

ching semantics, however.

### 6.7.3.2 Writing cache manifests

Manifests must be served using the `text/cache-manifest` MIME type. All resources served using the `text/cache-manifest` MIME type must follow the syntax of application cache manifests, as described in this section.

An application cache manifest is a text file, whose text is encoded using UTF-8. Data in application cache manifests is line-based. Newlines must be represented by U+000A LINE FEED (LF) characters, U+000D CARRIAGE RETURN (CR) characters, or U+000D CARRIAGE RETURN (CR) U+000A LINE FEED (LF) pairs. [RFC3629]

**Note:** This is a willful violation of RFC 2046, which requires all `text/*` types to only allow CRLF line breaks. This requirement, however, is outdated; the use of CR, LF, and CRLF line breaks is commonly supported and indeed sometimes CRLF is not supported by text editors. [RFC2046]

The first line of an application cache manifest must consist of the string "CACHE", a single U+0020 SPACE character, the string "MANIFEST", and either a U+0020 SPACE character, a U+0009 CHARACTER TABULATION (tab) character, a U+000A LINE FEED (LF) character, or a U+000D CARRIAGE RETURN (CR) character. The first line may optionally be preceded by a U+FEFF BYTE ORDER MARK (BOM) character. If any other text is found on the first line, it is ignored.

Subsequent lines, if any, must all be one of the following:

#### A blank line

Blank lines must consist of zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters only.

#### A comment

Comment lines must consist of zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters, followed by a single U+0023 NUMBER SIGN character (#), followed by zero or more characters other than U+000A LINE FEED (LF) and U+000D CARRIAGE RETURN (CR) characters.

**Note:** Comments must be on a line on their own. If they were to be included on a line with a URL, the "#" would be mistaken for part of a fragment identifier.

#### A section header

Section headers change the current section. There are four possible section headers:

##### CACHE :

Switches to the **explicit section**.

##### FALLBACK :

Switches to the **fallback section**.

##### NETWORK :

Switches to the **online whitelist section**.

##### SETTINGS :

Switches to the **settings section**.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

### Data for the current section

The format that data lines must take depends on the current section.

When the current section is the explicit section, data lines must consist of zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters, a valid URL identifying a resource other than the manifest itself, and then zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters.

When the current section is the fallback section, data lines must consist of zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters, a valid URL identifying a resource other than the manifest itself, one or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters, a valid URL identifying a resource other than the manifest itself, one or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters.

Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org>

/web/20130103030411/http:

//whatwg.org/html

Where the current section is the settings section, data lines must consist of zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters, a setting, and then zero or more U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters.

Currently only one **setting** is defined:

#### The cache mode setting

This consists of the string "prefer-online". It sets the cache mode to prefer-online. (The cache mode defaults to fast.)

Within a settings section, each setting must occur no more than once.

Manifests may contain sections more than once. Sections may be empty.

URLs that are to be fallback pages associated with fallback namespaces, and those namespaces themselves, must be given in fallback sections, with the namespace being the first URL of the data line, and the corresponding fallback page being the second URL. All the other pages to be cached must be listed in explicit sections.

Fallback namespaces and fallback entries must have the same origin as the manifest itself.

A fallback namespace must not be listed more than once.

Namespaces that the user agent is to put into the online whitelist must all be specified in online whitelist sections. (This is needed for any URL that the page is intending to use to communicate back to the server.) To specify that all URLs are automatically whitelisted in this way, a U+002A ASTERISK character (\*) may be specified as one of the URLs.

Authors should not include namespaces in the online whitelist for which another namespace in the online whitelist is a prefix match.

Relative URLs must be given relative to the manifest's own URL. All URLs in the manifest must have the same <scheme> as the manifest itself (either explicitly or implicitly, through the use of relative URLs).

URLs in manifests must not have fragment identifiers (i.e. the U+0023 NUMBER SIGN character isn't allowed in URLs in manifests).

Fallback namespaces and namespaces in the online whitelist are matched by prefix match.

#### 6.7.3.3 Parsing cache manifests

When a user agent is to **parse a manifest**, it means that the user agent must run the following steps:

1. The user agent must decode the byte stream corresponding with the manifest to be parsed as UTF-8, with error handling.
2. Let *base URL* be the absolute URL representing the manifest.
3. Let *explicit URLs* be an initially empty list of absolute URLs for explicit entries.
4. Let *fallback URLs* be an initially empty mapping of fallback namespaces to absolute URLs for fallback entries.
5. Let *online whitelist namespaces* be an initially empty list of absolute URLs for an online whitelist.
6. Let *online whitelist wildcard flag* be *blocking*.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

10. If *position* is pointing at a U+FEFF BYTE ORDER MARK (BOM) character, then advance *position* to the next character.
11. If the characters starting from *position* are "CACHE", followed by a U+0020 SPACE character, followed by "MANIFEST", then advance *position* to the next character after those. Otherwise, this isn't a cache manifest; abort this algorithm with a failure while checking for the magic signature.
12. If the character at *position* is neither a U+0020 SPACE character, a U+0009 CHARACTER TABULATION (tab) character, U+000A LINE FEED (LF) character, nor a U+000D CARRIAGE RETURN (CR) character, then this isn't a cache manifest; abort this algorithm with a failure while checking for the magic signature.
13. This is ignored.
14. Collect the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
15. Let *start* be the sequence of characters that are not U+0020 SPACE or U+0009 CHARACTER TABULATION (tab) characters.
16. Now, collect a sequence of characters that are not U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN (CR) characters, and let the result be *line*.
17. Drop any trailing U+0020 SPACE and U+0009 CHARACTER TABULATION (tab) characters at the end of *line*.
18. If *line* is the empty string, then jump back to the step labeled "start of line".
19. If the first character in *line* is a U+0023 NUMBER SIGN character (#), then jump back to the step labeled "start of line".
20. If *line* equals "CACHE:" (the word "CACHE" followed by a U+003A COLON character (:)), then set *mode* to "explicit" and jump back to the step labeled "start of line".
21. If *line* equals "FALLBACK:" (the word "FALLBACK" followed by a U+003A COLON character (:)), then set *mode* to "fallback" and jump back to the step labeled "start of line".
22. If *line* equals "NETWORK:" (the word "NETWORK" followed by a U+003A COLON character (:)), then set *mode* to "online whitelist" and jump back to the step labeled "start of line".
23. If *line* equals "SETTINGS:" (the word "SETTINGS" followed by a U+003A COLON character (:)), then set *mode* to "settings" and jump back to the step labeled "start of line".
24. If *line* ends with a U+003A COLON character (:), then set *mode* to "unknown" and jump back to the step labeled "start of line".
25. This is either a data line or it is syntactically incorrect.
26. Let *position* be a pointer into *line*, initially pointing at the start of the string.
27. Let *tokens* be a list of strings, initially empty.
28. While *position* doesn't point past the end of *line*:
  1. Let *current token* be an empty string.
  2. While *position* doesn't point past the end of *line* and the character at *position* is neither a U+0020 SPACE nor a U+0009 CHARACTER TABULATION (tab) character, add the character at *position* to *current token* and advance *position* to the next character in *input*.
  3. Add *current token* to the *tokens* list.
  4. While *position* doesn't point past the end of *line* and the character at *position* is either a U+0020 SPACE or a U+0009 CHARACTER TABULATION (tab) character, advance *position* to the next character in *input*.
29. Process *tokens* as follows:
  1. If *mode* is "explicit"
    1. Resolve the first item in *tokens*, relative to *base URL*; ignore the rest.
    2. If this fails, then jump back to the step labeled "start of line".

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Add the resulting absolute URL to the *explicit URLs*.

↳ If *mode* is "fallback"

Let *part one* be the first token in *tokens*, and let *part two* be the second token in *tokens*.

Resolve *part one* and *part two*, relative to *base URL*.

If either fails, then jump back to the step labeled "start of line".

If the absolute URL corresponding to either *part one* or *part two* does not the same origin as the manifest's

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

g as a fallback namespace,

oping as a fallback ntry.

↳ If *mode* is "online whitelist"

If the first item in *tokens* is a U+002A ASTERISK character (\*), then set *online whitelist wildcard flag* to *open* and jump back to the step labeled "start of line".

Otherwise, resolve the first item in *tokens*, relative to *base URL*; ignore the rest.

If this fails, then jump back to the step labeled "start of line".

If the resulting absolute URL has a different <scheme> component than the manifest's URL (compared in an ASCII case-insensitive manner), then jump back to the step labeled "start of line".

Drop the <fragment> component of the resulting absolute URL, if it has one.

Add the resulting absolute URL to the *online whitelist namespaces*.

↳ If *mode* is "settings"

If *tokens* contains a single token, and that token is a case-sensitive match for the string "prefer-online", then set *cache mode flag* to *prefer-online* and jump back to the step labeled "start of line".

Otherwise, the line is an unsupported setting: do nothing; the line is ignored.

↳ If *mode* is "unknown"

Do nothing. The line is ignored.

31. Jump back to the step labeled "start of line". (That step jumps to the next, and last, step when the end of the file is reached.)

32. Return the *explicit URLs* list, the *fallback URLs* mapping, the *online whitelist namespaces*, the *online whitelist wildcard flag*, and the *cache mode flag*.

**The resource that declares the manifest (with the `manifest` attribute) will always get taken from the cache, whether it is listed in the cache or not, even if it is listed in an online whitelist namespace.**

**If a resource is listed in the explicit section or as a fallback entry in the fallback section, the resource will always be taken from the cache, regardless of any other matching entries in the fallback namespaces or online whitelist namespaces.**

**When a fallback namespace and an online whitelist namespace overlap, the online whitelist namespace has priority.**

**The online whitelist wildcard flag is applied last, only for URLs that match neither the online whitelist namespace nor the fallback namespace and that are not listed in the explicit section.**

#### 6.7.4 Downloading or updating an application cache

When the user agent is required (by other parts of this specification) to start the **application cache download process** for an absolute URL purported to identify a manifest, or for an application cache group, potentially given a particular cache host, and potentially given a master resource, the user agent must run the steps below. These steps are always run asynchronously, in parallel with the event loop

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

goal of this is to allow Web applications to provide more seamless update mechanisms, hiding from the user the mechanics of the application cache mechanism. User agents may display user interfaces independent of this, but are encouraged to not show prominent update progress notifications for applications that cancel the relevant events.

**Note:** These events are delayed until after the `load` event has fired.

The application cache download process steps are as follows:

1. Optionally, wait until the permission to start the application cache download process has been obtained from the user and until the user explicitly opts-in to pass this point. (This step is in highly privacy-sensitive environments.)
2. At or before the `load` event, let *manifest URL* be that absolute URL.

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

Let *manifest URL* be that absolute URL.

If there is no application cache group identified by *manifest URL*, then create a new application cache group identified by *manifest URL*. Initially, it has no application caches. One will be created later in this algorithm.

#### ↳ If these steps were invoked with an application cache group

Let *manifest URL* be the absolute URL of the manifest used to identify the application cache group to be updated.

If that application cache group is obsolete, then abort this instance of the application cache download process. This can happen if another instance of this algorithm found the manifest to be 404 or 410 while this algorithm was waiting in the first step above.

2. Let *cache group* be the application cache group identified by *manifest URL*.
3. If these steps were invoked with a master resource, then add the resource, along with the resource's Document, to *cache group*'s list of pending master entries.
4. If these steps were invoked with a cache host, and the status of *cache group* is *checking* or *downloading*, then queue a post-load task to fire a simple event named *checking* that is cancelable at the `ApplicationCache` singleton of that cache host. The default action of this event must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent is checking to see if it can download the application.
5. If these steps were invoked with a cache host, and the status of *cache group* is *downloading*, then also queue a post-load task to fire a simple event named *downloading* that is cancelable at the `ApplicationCache` singleton of that cache host. The default action of this event must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user the application is being downloaded.
6. If the status of the *cache group* is either *checking* or *downloading*, then abort this instance of the application cache download process, as an update is already in progress.
7. Set the status of *cache group* to *checking*.
8. For each cache host associated with an application cache in *cache group*, queue a post-load task to fire a simple event that is cancelable named *checking* at the `ApplicationCache` singleton of the cache host. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent is checking for the availability of updates.

**Note:** The remainder of the steps run asynchronously.

If *cache group* already has an application cache in it, then this is an **upgrade attempt**. Otherwise, this is a **cache attempt**.

3. If this is a cache attempt, then this algorithm was invoked with a cache host; queue a post-load task to fire a simple event named *checking* that is cancelable at the `ApplicationCache` singleton of that cache host. The default action of this event must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent is checking to see if it can download the application.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Parse *manifest* according to the rules for parsing manifests, obtaining a list of explicit entries, fallback entries and the fallback namespaces that map to them, entries for the online whitelist, and values for the online whitelist wildcard flag and the cache mode flag.

**Note:** The MIME type of the resource is ignored — it is assumed to be `text/cache-manifest`. In the future, if new manifest formats are supported, the different types will probably be distinguished on the basis of the file signatures (for the current format, that is the "CACHE MANIFEST" string at the top of the file).

5. If *fetching the manifest* fails due to a 404 or 410 response or equivalent, then run these steps:

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

4. For each entry in *cache group*'s list of pending master entries, create a task to fire a simple event that is cancelable named `error` (not obsolete!) at the `ApplicationCache` singleton of the `Document` for this entry, if there still is one, and append it to *task list*. The default action of this event must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent failed to save the application for offline use.
5. If *cache group* has an application cache whose completeness flag is *incomplete*, then discard that application cache.
6. If appropriate, remove any user interface indicating that an update for this cache is in progress.
7. Let the status of *cache group* be *idle*.
8. For each task in *task list*, queue that task as a post-load task.
9. Abort the application cache download process.
6. Otherwise, if *fetching the manifest* fails in some other way (e.g. the server returns another 4xx or 5xx response or equivalent, or there is a DNS error, or the connection times out, or the user cancels the download, or the parser for manifests fails when checking the magic signature), or if the server returned a redirect, then run the cache failure steps. [HTTP]
7. If this is an upgrade attempt and the newly downloaded *manifest* is byte-for-byte identical to the manifest found in the newest application cache in *cache group*, or the server reported it as "304 Not Modified" or equivalent, then run these substeps:

1. Let *cache* be the newest application cache in *cache group*.
2. Let *task list* be an empty list of tasks.
3. For each entry in *cache group*'s list of pending master entries, wait for the resource for this entry to have either completely downloaded or failed.

If the download failed (e.g. the server returns a 4xx or 5xx response or equivalent, or there is a DNS error, the connection times out, or the user cancels the download), or if the resource is labeled with the "no-store" cache directive, then create a task to fire a simple event that is cancelable named `error` at the `ApplicationCache` singleton of the `Document` for this entry, if there still is one, and append it to *task list*. The default action of this event must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent failed to save the application for offline use.

Otherwise, associate the `Document` for this entry with *cache*; store the resource for this entry in *cache*, if it isn't already there, and categorize its entry as a master entry. If the resource's URL has a <fragment> component, it must be removed from the entry in *cache* (application caches never include fragment identifiers).

4. For each cache host associated with an application cache in *cache group*, create a task to fire a simple event that is cancelable named `noupdate` at the `ApplicationCache` singleton of the cache host, and append it to *task list*. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the application is up to date.
5. Empty *cache group*'s list of pending master entries.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

9. Abort the application cache download process.
8. Let *new cache* be a newly created application cache in *cache group*. Set its completeness flag to *incomplete*.
9. For each entry in *cache group*'s list of pending master entries, associate the *Document* for this entry with *new cache*.
10. Set the status of *cache group* to *downloading*.
11. For each cache host associated with an application cache in *cache group*, queue a post-load task to fire a simple event that is cancelable, at the ApplicationCache singleton of the cache host. The default action of these events must be to inform the user that a new version of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
12. Let *Add* /[web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html) with "explicit entry".
13. Add *Add* //[whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html) with "fallback entry".
14. Add *If this* . . . *cache in cache group* whose completeness flag is *complete* to *file list*, each flagged with "master entry".
15. If any URL is in *file list* more than once, then merge the entries into one entry for that URL, that entry having all the flags that the original entries had.
16. For each URL in *file list*, run the following steps. These steps may be run in parallel for two or more of the URLs at a time. If, while running these steps, the ApplicationCache object's *abort()* method sends a signal to this instance of the application cache download process algorithm, then run the cache failure steps instead.
  1. If the resource URL being processed was flagged as neither an "explicit entry" nor a "fallback entry", then the user agent may skip this URL.

**Note:** This is intended to allow user agents to expire resources not listed in the manifest from the cache. Generally, implementors are urged to use an approach that expires lesser-used resources first.

2. For each cache host associated with an application cache in *cache group*, queue a post-load task to fire a trusted event with the name *progress*, which does not bubble, which is cancelable, and which uses the ProgressEvent interface, at the ApplicationCache singleton of the cache host. The *lengthComputable* attribute must be set to true, the *total* attribute must be set to the number of files in *file list*, and the *loaded* attribute must be set to the number of files in *file list* that have been either downloaded or skipped so far. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that a file is being downloaded in preparation for updating the application. [XHR]
3. Fetch the resource, from the origin of the URL *manifest URL*, with the *synchronous flag* set and the *manual redirect flag* set. If this is an upgrade attempt, then use the newest application cache in *cache group* as an HTTP cache, and honor HTTP caching semantics (such as expiration, ETags, and so forth) with respect to that cache. User agents may also have other caches in place that are also honored.

**Note:** If the resource in question is already being downloaded for other reasons then the existing download process can sometimes be used for the purposes of this step, as defined by the fetching algorithm.

An example of a resource that might already be being downloaded is a large image on a Web page that is being seen for the first time. The image would get downloaded to satisfy the *img* element on the page, as well as being listed in the cache manifest. According to the rules for fetching that image only need be downloaded once, and it can be used both for the cache and for the rendered Web page.

4. If the previous step fails (e.g. the server returns a 4xx or 5xx response or equivalent, or there is a DNS error, or the connection times out, or the user cancels the download), or if the server returned a redirect, or if the resource is labeled with the "no-store" cache directive, then run the first appropriate step from the following list: [HTTP]

#### ↳ If the URL being processed was flagged as an "explicit entry" or a "fallback entry"

If these steps are being run in parallel for any other URLs in *file list*, then abort these steps for those other URLs. Run the cache failure steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*these situations are bad.*

↳ If the error was a 404 or 410 HTTP response or equivalent

↳ If the resource was labeled with the "no-store" cache directive

Skip this resource. It is dropped from the cache.

↳ Otherwise

Copy the resource and its metadata from the newest application cache in *cache group* whose completeness flag is *complete*, and act as if that was the fetched ~~resource~~, ignoring the resource

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

taking it possible for  
ever, without errors,

*...e a file to be expired or  
otherwise not cached are ignored for the purposes of the application cache download process.*

5. Otherwise, the fetching succeeded. Store the resource in the *new cache*.

If the user agent is not able to store the resource (e.g. because of quota restrictions), the user agent may prompt the user or try to resolve the problem in some other manner (e.g. automatically pruning content in other caches). If the problem cannot be resolved, the user agent must run the cache failure steps.

6. If the URL being processed was flagged as an "explicit entry" in *file list*, then categorize the entry as an explicit entry.

7. If the URL being processed was flagged as a "fallback entry" in *file list*, then categorize the entry as a fallback entry.

8. If the URL being processed was flagged as an "master entry" in *file list*, then categorize the entry as a master entry.

9. As an optimization, if the resource is an HTML or XML file whose root element is an `html` element with a `manifest` attribute whose value doesn't match the manifest URL of the application cache being processed, then the user agent should mark the entry as being foreign.

18. For each cache host associated with an application cache in *cache group*, queue a post-load task to fire a trusted event with the name `progress`, which does not bubble, which is cancelable, and which uses the `ProgressEvent` interface, at the `ApplicationCache` singleton of the cache host. The `lengthComputable` attribute must be set to true, the `total` and the `loaded` attributes must be set to the number of files in *file list*. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that all the files have been downloaded. [XHR]

19. Store the list of fallback namespaces, and the URLs of the fallback entries that they map to, in *new cache*.

20. Store the URLs that form the new online whitelist in *new cache*.

21. Store the value of the new online whitelist wildcard flag in *new cache*.

22. Store the value of the new cache mode flag in *new cache*.

23. For each entry in *cache group*'s list of pending master entries, wait for the resource for this entry to have either completely downloaded or failed.

If the download failed (e.g. the server returns a 4xx or 5xx response or equivalent, or there is a DNS error, the connection times out, or the user cancels the download), or if the resource is labeled with the "no-store" cache directive, then run these substeps:

1. Unassociate the `Document` for this entry from *new cache*.

2. Queue a post-load task to fire a simple event that is cancelable named `error` at the `ApplicationCache` singleton of the `Document` for this entry, if there still is one. The default action of this event must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent failed to save the application for offline use.

3. If this is a cache attempt and this entry is the last entry in *cache group*'s list of pending master entries, then run these

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

---

Go

**NOV JAN FEB**

◀ 03 ▶

2012 2013 2014



▼ About this capture

3. Abort the application cache download process.
  4. Otherwise, remove this entry from *cache group*'s list of pending master entries.

Otherwise, store the resource for this entry in *new cache*, if it isn't already there, and categorize its entry as a master entry.

24. Fetch the resource from *manifest URL* again, with the *synchronous flag* set, and let *second manifest* be that resource. HTTP caching semantics should again be honored for this request.

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

- 25. If the  
not  
the
- 26. Other

27. Set the completeness flag of *new cache* to *complete*.

28. Let *task list* be an empty list of tasks.

29. If this is a cache attempt, then for each cache host associated with an application cache in `cache group`, create a task to fire a simple event that is cancelable named `cached` at the `ApplicationCache` singleton of the cache host, and append it to `task list`. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the application has been cached and that they can now use it offline.

Otherwise, it is an upgrade attempt. For each cache host associated with an application cache in `cache group`, create a task to fire a simple event that is cancelable named `updateReady` at the `ApplicationCache` singleton of the cache host, and append it to `task list`. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that a new version is available and that they can activate it by reloading the page.

30. If appropriate, remove any user interface indicating that an update for this cache is in progress.

31. Set the update status of *cache group* to *idle*

32. For each task in *task list*, queue that task as a post-load task.

The **cache failure steps** are as follows:

1. Let `task list` be an empty list of tasks.
  2. For each entry in `cache group`'s list of pending master entries, run the following further substeps. These steps may be run in parallel for two or more entries at a time.
    1. Wait for the resource for this entry to have either completely downloaded or failed.
    2. Unassociate the `Document` for this entry from its application cache, if it has one.
    3. Create a task to fire a simple event that is cancelable named `error` at the `ApplicationCache` singleton of the `Document` for this entry, if there still is one, and append it to `task list`. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent failed to save the application for offline use.
  3. For each cache host still associated with an application cache in `cache group`, create a task to fire a simple event that is cancelable named `error` at the `ApplicationCache` singleton of the cache host, and append it to `task list`. The default action of these events must be, if the user agent shows caching progress, the display of some sort of user interface indicating to the user that the user agent failed to save the application for offline use.
  4. Empty `cache group`'s list of pending master entries.
  5. If `cache group` has an application cache whose completeness flag is `incomplete`, then discard that application cache.
  6. If appropriate, remove any user interface indicating that an update for this cache is in progress.
  7. Let the status of `cache group` be `idle`.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Attempts to fetch resources as part of the application cache download process may be done with cache-defeating semantics, to avoid problems with stale or inconsistent intermediary caches.

User agents may invoke the application cache download process, in the background, for any application cache group, at any time (with no cache host). This allows user agents to keep caches primed and to update caches even before the user visits a site.

Each Document has a list of pending application cache download process tasks that is used to delay events fired by the algorithm above until the document has been loaded.

When the step **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html** is reached, the user agent must run the following steps:

- If target's Document is not yet available, queue a task to add the target to the document's list of pending application cache download process tasks.
- Otherwise, add the target to the document's list of pending application cache download process tasks.

The task source for these tasks is the networking task source.

### 6.7.5 The application cache selection algorithm

When the **application cache selection algorithm** algorithm is invoked with a Document *document* and optionally a manifest URL *manifest URL*, the user agent must run the first applicable set of steps from the following list:

↪ If there is a *manifest URL*, and *document* was loaded from an application cache, and the URL of the manifest of that cache's application cache group is not the same as *manifest URL*

Mark the entry for the resource from which *document* was taken in the application cache from which it was loaded as foreign.

Restart the current navigation from the top of the navigation algorithm, undoing any changes that were made as part of the initial load (changes can be avoided by ensuring that the step to update the session history with the new page is only ever completed after this application cache selection algorithm is run, though this is not required).

**Note:** The navigation will not result in the same resource being loaded, because "foreign" entries are never picked during navigation.

User agents may notify the user of the inconsistency between the cache manifest and the document's own metadata, to aid in application development.

↪ If *document* was loaded from an application cache, and that application cache still exists (it is not now obsolete)

Associate *document* with the application cache from which it was loaded. Invoke, in the background, the application cache download process for that application cache's application cache group, with *document* as the cache host.

↪ If *document* was loaded using HTTP GET or equivalent, and, there is a *manifest URL*, and *manifest URL* has the same origin as *document*

Invoke, in the background, the application cache download process for *manifest URL*, with *document* as the cache host and with the resource from which *document* was parsed as the master resource.

If there are relevant application caches that are identified by a URL with the same origin as the URL of *document*, and that have this URL as one of their entries, excluding entries marked as foreign, then the user agent should use the most appropriate application cache of those that match as an HTTP cache for any subresource loads. User agents may also have other caches in place that are also honored.

↪ Otherwise

The Document is not associated with any application cache.

If there was a *manifest URL*, the user agent may report to the user that it was ignored, to aid in application development.

### 6.7.6 Changes to the networking model

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

component than the application cache's manifest, then fetch the resource normally and abort these steps.

2. If the resource's URL is a master entry, the manifest, an explicit entry, or a fallback entry in the application cache, then get the resource from the cache (instead of fetching it), and abort these steps.
3. If there is an entry in the application cache's online whitelist that has the same origin as the resource's URL and that is a prefix match for the resource's URL, then fetch the resource normally and abort these steps.
4. If the resource's URL has the same origin as the manifest's URL, and there is a fallback namespace *f* in the application cache that is a prefix match for the resource's IRI, then:

Fetc  
or 5  
fron  
the spec instead: <https://web.archive.org>  
5. If th  
/web/20130103030411/http:  
6. Fail  
//whatwg.org/html

: of a captive portal), or a 4xx  
ownload), then instead get,  
t these steps.

d abort these steps.

*Note: The user agent will always fail to load (at least, after the application cache has been primed the first time), making the testing of offline applications simpler.*

### 6.7.7 Expiring application caches

As a general rule, user agents should not expire application caches, except on request from the user, or after having been left unused for an extended period of time.

Application caches and cookies have similar implications with respect to privacy (e.g. if the site can identify the user when providing the cache, it can store data in the cache that can be used for cookie resurrection). Implementors are therefore encouraged to expose application caches in a manner related to HTTP cookies, allowing caches to be expunged together with cookies and other origin-specific data.

For example, a user agent could have a "delete site-specific data" feature that clears all cookies, application caches, local storage, databases, etc, from an origin all at once.

### 6.7.8 Disk space

User agents should consider applying constraints on disk usage of application caches, and care should be taken to ensure that the restrictions cannot be easily worked around using subdomains.

User agents should allow users to see how much space each domain is using, and may offer the user the ability to delete specific application caches.

*Note: How quotas are presented to the user is not defined by this specification. User agents are encouraged to provide features such as allowing a user to indicate that certain sites are trusted to use more than the default quota, e.g. by asynchronously presenting a user interface while a cache is being updated, or by having an explicit whitelist in the user agent's configuration interface.*

### 6.7.9 Application cache API

```
interface ApplicationCache : EventTarget {

 // update status
 const unsigned short UNCACHED = 0;
 const unsigned short IDLE = 1;
 const unsigned short CHECKING = 2;
 const unsigned short DOWNLOADING = 3;
 const unsigned short UPDATEREADY = 4;
 const unsigned short OBSOLETE = 5;
 readonly attribute unsigned short status;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
// events
 attribute EventHandler onchecking;
 attribute EventHandler onerror;
 attribute EventHandler onnoupdate;
 attribute EventHandler ondownloading;
 attribute EventHandler onprogress;
 attribute EventHandler onupdateready;
 attribute EventHandler oncached;
```



**}; Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>**

**cache:** /web/20130103030411/http:  
(In //whatwg.org/html

Comments are given below this box.

**cache =** --

t Window.

(In a shared worker.) Returns the ApplicationCache object that applies to the current shared worker.

**cache . status**

Returns the current status of the application cache, as given by the constants defined below.

**cache . update()**

Invokes the application cache download process.

Throws an `InvalidStateError` exception if there is no application cache to update.

Calling this method is not usually necessary, as user agents will generally take care of updating application caches automatically.

The method can be useful in situations such as long-lived applications. For example, a Web mail application might stay open in a browser tab for weeks at a time. Such an application could want to test for updates each day.

**cache . abort()**

Cancels the application cache download process.

This method is intended to be used by Web application showing their own caching progress UI, in case the user wants to stop the update (e.g. because bandwidth is limited).

**cache . swapCache()**Switches to the most recent application cache, if there is a newer one. If there isn't, throws an `InvalidStateError` exception.

This does not cause previously-loaded resources to be reloaded; for example, images do not suddenly get reloaded and style sheets and scripts do not get reparsed or reevaluated. The only change is that subsequent requests for cached resources will obtain the newer copies.

The `updateready` event will fire before this method can be called. Once it fires, the Web application can, at its leisure, call this method to switch the underlying cache to the one with the more recent updates. To make proper use of this, applications have to be able to bring the new features into play; for example, reloading scripts to enable new features.An easier alternative to `swapCache()` is just to reload the entire page at a time suitable for the user, using `location.reload()`.

There is a one-to-one mapping from cache hosts to `ApplicationCache` objects. The `applicationCache` attribute on `Window` objects must return the `ApplicationCache` object associated with the `Window` object's active document. The `applicationCache` attribute on `SharedWorkerGlobalScope` objects must return the `ApplicationCache` object associated with the worker.

**Note:** A `Window` or `SharedWorkerGlobalScope` object has an associated `ApplicationCache` object even if that cache host has no actual application cache.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**IDLE** (numeric value 1)

The `ApplicationCache` object's cache host is associated with an application cache whose application cache group's update status is *idle*, and that application cache is the newest cache in its application cache group, and the application cache group is not marked as obsolete.

**CHECKING** (numeric value 2)

The `ApplicationCache` object's cache host is associated with an application cache whose application cache group's update status is *idle*.

**DOWNLOADING** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

**UPDATEREADY** /web/20130103030411/http:

The `ApplicationCache` object's cache host is associated with an application cache whose application cache group's update status is *idle*.

**OBSOLETE** (numeric value 5)

The `ApplicationCache` object's cache host is associated with an application cache whose application cache group is marked as obsolete.

If the `update()` method is invoked, the user agent must invoke the application cache download process, in the background, for the application cache group of the application cache with which the `ApplicationCache` object's cache host is associated, but without giving that cache host to the algorithm. If there is no such application cache, or if its application cache group is marked as obsolete, then the method must throw an `InvalidStateError` exception instead.

If the `abort()` method is invoked, the user agent must send a signal to the current application cache download process for the application cache group of the application cache with which the `ApplicationCache` object's cache host is associated, if any. If there is no such application cache, or it does not have a current application cache download process, then do nothing.

If the `swapCache()` method is invoked, the user agent must run the following steps:

1. Check that `ApplicationCache` object's cache host is associated with an application cache. If it is not, then throw an `InvalidStateError` exception and abort these steps.
2. Let `cache` be the application cache with which the `ApplicationCache` object's cache host is associated. (By definition, this is the same as the one that was found in the previous step.)
3. If `cache`'s application cache group is marked as obsolete, then unassociate the `ApplicationCache` object's cache host from `cache` and abort these steps. (Resources will now load from the network instead of the cache.)
4. Check that there is an application cache in the same application cache group as `cache` whose completeness flag is *complete* and that is newer than `cache`. If there is not, then throw an `InvalidStateError` exception and abort these steps.
5. Let `new cache` be the newest application cache in the same application cache group as `cache` whose completeness flag is *complete*.
6. Unassociate the `ApplicationCache` object's cache host from `cache` and instead associate it with `new cache`.

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by all objects implementing the `ApplicationCache` interface:

Event handler	Event handler event type
<code>onchecking</code>	<code>checking</code>
<code>onerror</code>	<code>error</code>
<code>onnoupdate</code>	<code>noupdate</code>
<code>ondownloading</code>	<code>downloading</code>
<code>onprogress</code>	<code>progress</code>
<code>onupdateready</code>	<code>updateready</code>
<code>oncached</code>	<code>cached</code>

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

## 6.7.10 Browser state

```
[NoInterfaceObject]
interface NavigatorOnLine {
 readonly attribute boolean onLine;
};
```

This box is non-normative. Implementation requirements are given below this box.

**window**

Re<sup>l</sup> Too slow? Try reading the multipage copy of the user agent might be  
on<sup>l</sup> the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>  
The<sup>l</sup> The [navigator](https://web.archive.org/web/20130103030411/http://whatwg.org/html) script request → user follows links or when a

When the value that would be returned by the `navigator.onLine` attribute of a Window or WorkerGlobalScope changes from true to false, the user agent must queue a task to fire a simple event named `offline` at the Window or WorkerGlobalScope object.

On the other hand, when the value that would be returned by the `navigator.onLine` attribute of a Window or WorkerGlobalScope changes from false to true, the user agent must queue a task to fire a simple event named `online` at the Window or WorkerGlobalScope object.

The task source for these tasks is the networking task source.

**Note:** This attribute is inherently unreliable. A computer can be connected to a network without having Internet access.

In this example, an indicator is updated as the browser goes online and offline.

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>Online status</title>
 <script>
 function updateIndicator() {
 document.getElementById('indicator').textContent = navigator.onLine ? 'online' :
 'offline';
 }
 </script>
 </head>
 <body onload="updateIndicator()" ononline="updateIndicator()"
 onoffline="updateIndicator()">
 <p>The network is: (state unknown)
 </body>
</html>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

## 7.1.1 Introduction

Various mechanisms can cause author-provided executable code to run in the context of a document. These mechanisms include, but are probably not limited to:

- Processing of `script` elements.
- Processing of inline `javascript:` URLs (e.g. the `src` attribute of `img` elements, or an `@import` rule in a CSS style element block).
- Event attribute
- Processing of event handler content
- Pro Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

### 7.1.2 **Ena** /web/20130103030411/http:

**Scripting is** [//whatwg.org/html](http://whatwg.org/html)

- The user agent supports scripting.
- The user has not disabled scripting for this browsing context at this time. (User agents may provide users with the option to disable scripting globally, or in a finer-grained manner, e.g. on a per-origin basis.)
- The browsing context's active document's active sandboxing flag set does not have its sandboxed scripts browsing context flag set.

**Scripting is disabled** in a browsing context when any of the above conditions are false (i.e. when scripting is not enabled).



**Scripting is enabled** for a `node` if the `Document` object of the node (the node itself, if it is itself a `Document` object) has an associated browsing context, and scripting is enabled in that browsing context.

**Scripting is disabled** for a node if there is no such browsing context, or if scripting is disabled in that browsing context.

### 7.1.3 Processing model

#### 7.1.3.1 Definitions

This specification describes three kinds of JavaScript global environments: the **document environment**, the **dedicated worker environment**, and the **shared worker environment**. The dedicated worker environment and the shared worker environment are both types of **worker environments**.

Except where otherwise specified, a JavaScript global environment is a document environment.

A `script` has:

##### A *script execution environment*

The characteristics of the script execution environment depend on the language, and are not defined by this specification.

In JavaScript, the script execution environment consists of the interpreter, the stack of *execution contexts*, the *global code* and *function code* and the `Function` objects resulting, and so forth.

##### A *list of code entry-points*

Each code entry-point represents a block of executable code that the script exposes to other scripts and to the user agent.

Each `Function` object in a JavaScript script execution environment has a corresponding code entry-point, for instance.

The main program code of the script, if any, is the **initial code entry-point**. Typically, the code corresponding to this entry-point is executed immediately after the script is parsed.

In JavaScript, this corresponds to the execution context of the global code.

##### A relationship with the *script's global object*

An object that provides the APIs that the code can use.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

If the script's global object is a `Window` object, then in JavaScript, the `ThisBinding` of the global execution context for this script must be the `Window` object's `WindowProxy` object, rather than the global object. [ECMA262]

**Note:** This is a willful violation of the JavaScript specification current at the time of writing (ECMAScript edition 5, as defined in section 10.4.1.1 Initial Global Execution Context, step 3). The JavaScript specification requires that the `this` keyword in the global scope return the global object, but this is not compatible with the security design prevalent in implementations as specified herein. [ECMA262]



#### A relationsh<sup>o</sup>

A b<sup>l</sup> Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

e of the new browsin

#### A relationsl<sup>o</sup>

A D<sup>l</sup> <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

For example, the address of the script's document is used to set the address of any `Document` elements created using `createDocument()`.

#### The script's referrer source

Either a `Document` (specifically, the script's document), or a URL, which is used by some APIs to determine what value to use for the `Referer` (sic) header in calls to the fetching algorithm.

#### A URL character encoding

A character encoding, set when the script is created, used to encode URLs. If the character encoding is set from another source, e.g. a document's character encoding, then the script's URL character encoding must follow the source, so that if the source's changes, so does the script's.

#### A base URL

A URL, set when the script is created, used to resolve relative URLs. If the base URL is set from another source, e.g. a document base URL, then the script's base URL must follow the source, so that if the source's changes, so does the script's.

#### Optionally, a muted errors flag

A flag which, if set, means that error information will not be provided for errors in this script (used to mute errors for cross-origin scripts, since that can leak private information).

### 7.1.3.2 Calling scripts

When a user agent is to **jump to a code entry-point** for a script, for example to invoke an event listener defined in that script, the user agent must run the following steps:

1. If the script's global object is a `Window` object whose `Document` object is not fully active, then abort these steps without doing anything. The callback is not run.
2. Set the entry script to be the script being invoked.
3. Make the script execution environment for the script execute the code for the given code entry-point.
4. Set the entry script back to whatever it was when this algorithm started (possibly nothing).
5. If there is no longer an entry script, run the global script clean-up jobs.
6. Perform a microtask checkpoint. (If this runs scripts, it will result in this algorithm being invoked reentrantly.)

This algorithm is not invoked by one script directly calling another, but it can be invoked reentrantly in an indirect manner, e.g. if a script dispatches an event which has event listeners registered.

Each unit of related similar-origin browsing contexts can have an **entry script** which is used to obtain, amongst other things, the script's base URL to resolve relative URLs used in scripts running in that unit of related similar-origin browsing contexts. Initially, there is no entry script. It is changed by the jump to a code entry-point algorithm above.

Each unit of related similar-origin browsing contexts has a **running mutation observers** flag, which must initially be false. It is used to

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

When the user agent is to **run the global script clean-up jobs**, the user agent must perform each of the jobs in the global script clean-up jobs list and then empty the list.

### 7.1.3.3 Creating scripts

When the specification says that a script is to be **created**, given some script source, a script source URL, its scripting language, a global object, a browsing context, a document, a referrer source, a URL character encoding, a base URL, and optionally a *muted errors* flag, the user agent must run the following steps:

1. If `script` is **too slow**, then
 

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

The script did nothing but return `null`.
2. Set `script` to a new `Script` object created using the `script` settings passed to this algorithm.
3. Parse the `script` settings, determining the following values:
  - `lang`: the scripting language for the script
  - `sou`: the URL character encoding for the script
  - `base`: the base URL for the script
  - `script`: the script's global object
  - `context`: the script's browsing context
  - `document`: the script's document
  - `referrer`: the script's referrer source
  - `encoding`: the script's URL character encoding
  - `base`: the script's base URL
4. Set up the script's global object, the script's browsing context, the script's document, the script's referrer source, the script's URL character encoding, and the script's base URL from the settings passed to this algorithm.
5. If the *muted errors* flag was set, then set the script's muted errors flag also.
6. If all the steps above succeeded (in particular, if the script was compiled successfully), Jump to the script's *initial code entry-point*.

Otherwise, report the error for the script, with the problematic position (line number and column number), in the script's origin, using the `onerror` event handler of the script's global object. If the error is still *not handled* after this, then the error may be reported to the user.

When the user agent is to **create an impotent script**, given some script source and URL, its scripting language, and a browsing context, the user agent must create a script, using the given script source, URL, and scripting language, using a new empty object as the global object, and using the given browsing context as the browsing context. The referrer source, URL character encoding, and base URL for the resulting script are not important as no APIs are exposed to the script.

When the specification says that a script is to be **created from a node node**, given some script source, its URL, its scripting language, and optionally a *muted errors* flag, the user agent must create a script, using the given script source, URL, and scripting language, the script settings determined from the node `node`, and, if the *muted errors* flag was set in the call to this algorithm, the *muted errors* flag.

The **script settings determined from the node node** are computed as follows:

1. Let `document` be the `Document` of `node` (or `node` itself if it is a `Document`).
2. The global object is the `Window` object of `document`.
3. The browsing context is the browsing context of `document`.
4. The document is `document`.
5. The referrer source is `document`.
6. The URL character encoding is the character encoding of `document`. (This is a reference, not a copy.)
7. The base URL is the base URL of `document`. (This is a reference, not a copy.)

### 7.1.3.4 Killing scripts

User agents may impose resource limitations on scripts, for example CPU quotas, memory limits, total execution time limits, or bandwidth limitations. When a script exceeds a limit, the user agent may either throw a `QuotaExceededError` exception, abort the script without an exception, prompt the user, or throttle script execution.

For example, the following script never terminates. A user agent could, after waiting for a few seconds, prompt the user to either terminate the script or let it continue.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

`window.alert()` API) or because of a script's actions (e.g. because it has exceeded a time limit).

If scripting is disabled while a script is executing, the script should be terminated immediately.

User agents may allow users to specifically disable scripts just for the purposes of closing a browsing context.

For example, the prompt mentioned in the example above could also offer the user with a mechanism to just close the page entirely, without running any `unload` event handlers.



#### 7.1.3.5 Runtime script errors

When the `onerror` event handler one:

↳ If the value of `v`:

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

and `col` to 0.

3. The callback must be invoked with four arguments: `message`, `location`, `line`, and `col`.

If the callback returns true, then the error is *handled*. Otherwise, the error is *not handled*.

Any uncaught exceptions thrown or errors caused by this function may be reported to the user immediately after the error that the function was called for; the report an error algorithm must not be used to handle exceptions thrown or errors caused by this function.

↳ Otherwise

The error is *not handled*.

##### 7.1.3.5.1 Runtime script errors in documents

Whenever an uncaught runtime script error occurs in one of the scripts associated with a `Document`, the user agent must report the error for the relevant script, with the problematic position (line number and column number) in the resource containing the script, using the `onerror` event handler of the script's global object. If the error is still *not handled* after this, then the error may be reported to the user.

The resource containing the script will typically be the file from which the `Document` was parsed, e.g. for inline `script` elements or event handler content attributes; or the JavaScript file that the script was in, for external scripts. Even for dynamically-generated scripts, user agents are strongly encouraged to attempt to keep track of the original source of a script. For example, if an external script uses the `document.write()` API to insert an inline `script` element during parsing, the URL of the resource containing the script would ideally be reported as being the external script, and the line number might ideally be reported as the line with the `document.write()` call or where the string passed to that call was first constructed. Naturally, implementing this can be somewhat non-trivial.

User agents are similarly encouraged to keep careful track of the original line numbers, even in the face of `document.write()` calls mutating the document as it is parsed, or event handler content attributes spanning multiple lines.

## 7.1.4 Event loops

### 7.1.4.1 Definitions

To coordinate events, user interaction, scripts, rendering, networking, and so forth, user agents must use **event loops** as described in this section.

There must be at least one event loop per user agent, and at most one event loop per unit of related similar-origin browsing contexts.

**Note:** When there is more than one event loop for a unit of related browsing contexts, complications arise when a browsing context in that group is navigated such that it switches from one unit of related similar-origin browsing contexts to another. This specification does not currently describe how to handle these complications.

An event loop always has at least one browsing context. If an event loop's browsing contexts all go away, then the event loop goes away as well. A browsing context always has an event loop coordinating its activities.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

**Note:** Not all events are dispatched using the task queue, many are dispatched synchronously during other tasks.

## Parsing

The HTML parser tokenizing one or more bytes, and then processing any resulting tokens, is typically a task.

## Callbacks

Calling a callback asynchronously is a task.



## Using a res...

When all c... of the resource once some or

## Reacting to

Son doc is inserted into the

/web/20130103030411/http://whatwg.org/html

When a user

from one particular **task source** (e.g. the callbacks generated by timers, the events fired for mouse movements, the tasks queued for the parser) must always be added to the same task queue, but tasks from different task sources may be placed in different task queues.

For example, a user agent could have one task queue for mouse and key events (the user interaction task source), and another for everything else. The user agent could then give keyboard and mouse events preference over other tasks three quarters of the time, keeping the interface responsive but not starving other task queues, and never processing events from any one task source out of order.

Each task that is queued onto a task queue of an event loop defined by this specification is associated with a `Document`; if the task was queued in the context of an element, then it is the element's `Document`; if the task was queued in the context of a browsing context, then it is the browsing context's active document at the time the task was queued; if the task was queued by or for a script then the document is the script's document.

A user agent may have one **storage mutex**. This mutex is used to control access to shared state like cookies. At any one point, the storage mutex is either free, or owned by a particular event loop or instance of the fetching algorithm.

If a user agent does not implement a storage mutex, it is exempt from implementing the requirements that require it to acquire or release it.

**Note:** User agent implementors have to make a choice between two evils. On the one hand, not implementing the storage mutex means that there is a risk of data corruption: a site could, for instance, try to read a cookie, increment its value, then write it back out, using the new value of the cookie as a unique identifier for the session; if the site does this twice in two different browser windows at the same time, it might end up using the same "unique" identifier for both sessions, with potentially disastrous effects. On the other hand, implementing the storage mutex has potentially serious performance implications: whenever a site uses Web Storage or cookies, all other sites that try to use Web Storage or cookies are blocked until the first site finishes.

Whenever a script calls into a plugin, and whenever a plugin calls into a script, the user agent must release the storage mutex.

### 7.1.4.2 Processing model

An event loop must continually run through the following steps for as long as it exists:

1. Run the oldest task on one of the event loop's task queues, if any, ignoring tasks whose associated `Documents` are not fully active. The user agent may pick any task queue.
2. If the storage mutex is now owned by the event loop, release it so that it is once again free.
3. If a task was run in the first step above, remove that task from its task queue.
4. Perform a microtask checkpoint.
5. Provide a stable state.
6. If necessary, update the rendering or user interface of any `Document` or browsing context to reflect the current state.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. Let the running mutation observers flag be true.
2. Sort the tables with pending sorts.
3. Invoke `MutationObserver` objects for the unit of related similar-origin browsing contexts to which the script's browsing context belongs.

**Note:** This will typically invoke scripted callbacks, which calls the jump to a code entry-point algorithm, which calls this perform a microtask checkpoint alorithm again. which is why use the running mutation

4. Let **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

When the user agent must **spin the event loop** until a condition *goal* is met, then the user agent must **run the following steps**:

**Note:** A **stable state**, then the user agent must **run the following steps**:

**detectable from**

**another synchronous section, and thus synchronous sections can be run in any order.**

**Note:** Steps in synchronous sections are marked with .

When an algorithm says to **spin the event loop** until a condition *goal* is met, the user agent must run the following steps:

1. Let *task source* be the task source of the currently running task.
2. Stop the currently running task, allowing the event loop to resume, but continue these steps asynchronously.

**Note:** This causes the event loop to move on to the second step of its processing model (defined above).

3. Wait until the condition *goal* is met.
4. Queue a task to continue running these steps, using the task source *task source*. Wait until this task runs before continuing these steps.
5. Return to the caller.

Some of the algorithms in this specification, for historical reasons, require the user agent to **pause** while running a task until a condition *goal* is met. This means running the following steps:

1. If any asynchronously-running algorithms are awaiting a stable state, then run their synchronous section and then resume running their asynchronous algorithm. (See the event loop processing model definition above for details.)
2. If necessary, update the rendering or user interface of any `Document` or browsing context to reflect the current state.
3. Wait until the condition *goal* is met. While a user agent has a paused task, the corresponding event loop must not run further tasks, and any script in the currently running task must block. User agents should remain responsive to user input while paused, however, albeit in a reduced capacity since the event loop will not be doing anything.

When a user agent is to **obtain the storage mutex** as part of running a task, it must run through the following steps:

1. If the storage mutex is already owned by this task's event loop, then abort these steps.
2. Otherwise, pause until the storage mutex can be taken by the event loop.
3. Take ownership of the storage mutex.

#### 7.1.4.3 Generic task sources

The following task sources are used by a number of mostly unrelated features in this and other specifications.

##### The **DOM manipulation task source**

This task source is used for features that react to DOM manipulations, such as things that happen asynchronously when an

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Asynchronous events sent in response to user input (e.g. `click` events) must be fired using tasks queued with the user interaction task source. [DOMEVENTS]

#### The networking task source

This task source is used for features that trigger in response to network activity.

#### The history traversal task source

This task source is used to queue calls to `history.back()` and similar APIs.



### 7.1.5 The Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

When a URI

1. Let `/web/20130103030411/http://whatwg.org/html`
2. Use `//whatwg.org/html`

If a `script` element is present in the document, and if any of its `src` attributes has a value that is a URL, then let `script` be the URL. Otherwise, let `script` be `null`.

Let `result` be void.

OS:

`script: URLs. [JSURL]`

context for that navigation,

If a browsing context is being navigated to a `javascript: URL`, and the active document of that browsing context has the same origin as the script given by that URL

Let `address` be the address of the active document of the browsing context being navigated.

If `address` is `about:blank`, and the browsing context being navigated has a creator browsing context, then let `address` be the address of the creator Document instead.

Create a script from the `Document` node of the active document, using the aforementioned script source, the URL of the resource where the `javascript: URL`, was found, and assuming the scripting language is JavaScript.

Let `result` be the return value of the `initial code entry-point` of this script. If an exception was thrown, let `result` be void instead. (The result will be void also if scripting is disabled.)

When it comes time to set the document's address in the navigation algorithm, use `address` as the override URL.

#### Otherwise

Let `result` be void.

3. If the result of executing the script is void (there is no return value), then the URL must be treated in a manner equivalent to an HTTP resource with an HTTP 204 No Content response.

Otherwise, the URL must be treated in a manner equivalent to an HTTP resource with a 200 OK response whose Content-Type metadata is `text/html` and whose response body is the return value converted to a string value.

**Note: Certain contexts, in particular `img` elements, ignore the Content-Type metadata.**

So for example a `javascript: URL` for a `src` attribute of an `img` element would be evaluated in the context of an empty object as soon as the attribute is set; it would then be sniffed to determine the image type and decoded as an image.

A `javascript: URL` in an `href` attribute of an `a` element would only be evaluated when the link was followed.

The `src` attribute of an `iframe` element would be evaluated in the context of the `iframe`'s own browsing context; once evaluated, its return value (if it was not void) would replace that browsing context's document, thus changing the variables visible in that browsing context.

## 7.1.6 Events

### 7.1.6.1 Event handlers

Many objects can have **event handlers** specified. These act as non-capture event listeners for the object on which they are specified. [DOM]

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The first way, common to all event handlers, is as an event handler IDL attribute.

The second way is as an event handler content attribute. Event handlers on HTML elements and some of the event handlers on Window objects are exposed in this way.

**Event handler IDL attributes**, on setting, must set the corresponding event handler to their new value, and on getting, must return whatever the current value of the corresponding event handler is (possibly null).

If an event handler IDL attribute exposes an event handler of an object that doesn't exist, it must always return null on getting and must do nothing or

**Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

*Note: This note applies to the `Window` object.*

*Note: Content attributes are not yet supported by the browser's JavaScript engine. Instead, use the `MessageEvent` attribute of the `MessageEvent` interface.*

On getting, event handler IDL attributes must return the value of their corresponding event handlers, except when the value is an internal error value, in which case the user agent must set the corresponding event handler to null, and then throw an exception corresponding to the error condition.

**Event handler content attributes**, when specified, must contain valid JavaScript code which, when parsed, would match the `FunctionBody` production after automatic semicolon insertion. [ECMA262]

When an event handler content attribute is set, if the element is owned by a `Document` that is in a browsing context, and scripting is enabled for that browsing context, the user agent must run the following steps to create a script after setting the content attribute to its new value:

1. Set the corresponding event handler to null.
2. Set up a script execution environment for JavaScript.
3. Let `body` be the event handler content attribute's new value.
4. If `body` is not parsable as `FunctionBody` or if parsing detects an *early error* then set the event handler content attribute to an error as defined below, and abort these steps.

*Note: `FunctionBody` is defined in ECMAScript edition 5 section 13 Function Definition. Early error is defined in ECMAScript edition 5 section 16 Errors. [ECMA262]*

5. If `body` begins with a Directive Prologue that contains a Use Strict Directive then let `strict` be true, otherwise let `strict` be false.

*Note: The terms "Directive Prologue" and "Use Strict Directive" are defined in ECMAScript edition 5 section 14.1 Directive Prologues and the Use Strict Directive. [ECMA262]*

6. Using the script execution environment created above, create a function object (as defined in ECMAScript edition 5 section 13.2 Creating Function Objects), with:

#### Parameter list `FormalParameterList`

↳ If the attribute is the `onerror` attribute of the `Window` object

Let the function have four arguments, named `event`, `source`, `lineno`, and `column`.

↳ Otherwise

Let the function have a single argument called `event`.

#### Function body `FunctionBody`

The result of parsing `body` above.

#### Lexical Environment Scope

1. Let `Scope` be the result of `NewObjectEnvironment`(the element's `Document`, the *global environment*).
2. If the element has a form owner, let `Scope` be the result of `NewObjectEnvironment`(the element's form owner, `Scope`).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Boolean flag Strict**The value of *strict*.

Let this new function be the only entry in the script's list of code entry-points.

7. Set up the script's global object, the script's browsing context, the script's document, the script's referrer source, the script's URL character encoding, and the script's base URL from the script settings determined from the node on which the attribute is being set.

8. Set  $\mathbf{t}^{\mathbf{f}}$ 

When a user  
the correspond  
where the element  
**Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org**

r, the user agent must set  
of the URL of the resource  
the error occurred.

When an event handler  $H$  is set to a non-null value, the user agent must append an event listener to the list of event listeners associated with  $T$  with *type* set to the **event handler event type** corresponding to  $H$ , *capture* set to false, and *listener* set to the event handler processing algorithm defined below. [DOM]

andler to null.

**Note:** *V* context, ..

*is not in a browsing*

When an event handler  $H$  of an element or object  $T$  implementing the `EventTarget` interface is first set to a non-null value, the user agent must append an event listener to the list of event listeners associated with  $T$  with *type* set to the **event handler event type** corresponding to  $H$ , *capture* set to false, and *listener* set to the event handler processing algorithm defined below. [DOM]

**Note:** *The listener is emphatically not the event handler itself. Every event handler ends up registering the same listener, the algorithm defined below, which takes care of invoking the right callback, and processing the callback's return value.*

**Note:** *This only happens the first time the event handler's value is set. Since listeners are called in the order they were registered, the order of event listeners for a particular event type will always be first the event listeners registered with `addEventListener()` before the first time the event handler was set to a non-null value, then the callback to which it is currently set, if any, and finally the event listeners registered with `addEventListener()` after the first time the event handler was set to a non-null value.*

This example demonstrates the order in which event listeners are invoked. If the button in this example is clicked by the user, the page will show four alerts, with the text "ONE", "TWO", "THREE", and "FOUR" respectively.

```
<button id="test">Start Demo</button>
<script>
var button = document.getElementById('test');
button.addEventListener('click', function () { alert('ONE') }, false);
button.setAttribute('onclick', "alert('NOT CALLED')"); // event handler listener is registered here
button.addEventListener('click', function () { alert('THREE') }, false);
button.onclick = function () { alert('TWO') };
button.addEventListener('click', function () { alert('FOUR') }, false);
</script>
```

**Note:** *The interfaces implemented by the event object do not influence whether an event handler is triggered or not.*

**The event handler processing algorithm** for an event handler  $H$  and an `Event` object  $E$  is as follows:

1. If  $H$ 's value is null, then abort these steps.
2. If  $H$ 's value is an internal error value, then set the event handler to null and then report the error for the appropriate script and with the appropriate position (line number and column number), as established when the error was detected, using the `onerror` event handler of the `Window` object of that `Document`, and abort these steps.
3. Let *callback* be  $H$ 's value, the callback that the event handler was last set to.
4. Invoke *callback* with one argument, the value of which is the `Event` object  $E$ , with the `callback this` value set to  $E$ 's `currentTarget`. Let the return value be *return value*. [WEBIDL]

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

↳ If the Event object *E* is a BeforeUnloadEvent object

If *return value* is a string, and the Event object *E*'s `returnValue` attribute's value is the empty string, then set the `returnValue` attribute's value to *return value*.

↳ Otherwise

If *return value* is a boolean with the value false, then cancel the event.

The `EventHandler` interface represents a callback used for event handlers. It is represented in IDL as follows:

[Treat call type] Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*Note: It*

For e

```
<body onload="alert(this)" onclick="alert(this)">
...leads to an alert saying "[object Window]" when the document is loaded, and an alert saying "[object HTMLBodyElement]" whenever the user clicks something in the page.
```

**Note: The return value of the function affects whether the event is canceled or not: as described above, if the return value is false, the event is canceled (except for mouseover events, where the return value has to be true to cancel the event). With beforeunload events, the value is instead used to determine the message to show the user.**

For historical reasons, the `onerror` handler has different arguments:

```
[TreatNonCallableAsNull]
callback OnErrorEventHandlerNonNull = any ((Event or DOMString) event, DOMString source,
unsigned long lineno, unsigned long column);
typedef OnErrorEventHandlerNonNull? OnErrorHandler;
```

### 7.1.6.2 Event handlers on elements, Document objects, and Window objects

The following are the event handlers (and their corresponding event handler event types) that must be supported by all HTML elements, as both content attributes and IDL attributes, and on `Document` and `Window` objects, as IDL attributes.

Event handler	Event handler event type
<code>onabort</code>	<code>abort</code>
<code>oncancel</code>	<code>cancel</code>
<code>oncanplay</code>	<code>canplay</code>
<code>oncanplaythrough</code>	<code>canplaythrough</code>
<code>onchange</code>	<code>change</code>
<code>onclick</code>	<code>click</code>
<code>onclose</code>	<code>close</code>
<code>oncontextmenu</code>	<code>contextmenu</code>
<code>oncuechange</code>	<code>cuechange</code>
<code>ondblclick</code>	<code>dblclick</code>
<code>ondrag</code>	<code>drag</code>
<code>ondragend</code>	<code>dragend</code>
<code>ondragenter</code>	<code>dragenter</code>
<code>ondragleave</code>	<code>dragleave</code>
<code>ondragover</code>	<code>dragover</code>
<code>ondragstart</code>	<code>dragstart</code>
<code>ondrop</code>	<code>drop</code>
<code>ondurationchange</code>	<code>durationchange</code>

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

<code>oninvalid</code>	invalid
<code>onkeydown</code>	keydown
<code>onkeypress</code>	keypress
<code>onkeyup</code>	keyup
<code>onloadeddata</code>	loadeddata
<code>onloadedmetadata</code>	loadedmetadata
<code>onloadstart</code>	loadstart
<code>onmousedown</code>	mousedown
<code>onmousemove</code>	
<code>onmouseout</code>	
<code>onmouseove</code>	
<code>onmouseup</code>	
<code>onmousewheel</code>	
<code>onpause</code>	
<code>onplay</code>	
<code>onplaying</code>	
<code>onprogress</code>	progress
<code>onratechange</code>	ratechange
<code>onreset</code>	reset
<code>onseeked</code>	seeked
<code>onseeking</code>	seeking
<code>onselect</code>	select
<code>onshow</code>	show
<code>onsort</code>	sort
<code>onstalled</code>	stalled
<code>onsubmit</code>	submit
<code>onsuspend</code>	suspend
<code>ontimeupdate</code>	timeupdate
<code>onvolumechange</code>	volumechange
<code>onwaiting</code>	waiting

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



The following are the event handlers (and their corresponding event handler event types) that must be supported by all HTML elements other than `body` and `frameset`, as both content attributes and IDL attributes, and on `Document` objects, as IDL attributes:

Event handler	Event handler event type
<code>onblur</code>	blur
<code>onerror</code>	error
<code>onfocus</code>	focus
<code>onload</code>	load
<code>onscroll</code>	scroll

The following are the event handlers (and their corresponding event handler event types) that must be supported by `Window` objects, as IDL attributes on the `Window` object, and with corresponding content attributes and IDL attributes exposed on the `body` and `frameset` elements:

Event handler	Event handler event type
<code>onafterprint</code>	afterprint
<code>onbeforeprint</code>	beforeprint
<code>onbeforeunload</code>	beforeunload
<code>onblur</code>	blur
<code>onerror</code>	error
<code>onfocus</code>	focus
<code>onfullscreenchange</code>	fullscreenchange
<code>onfullscreenerror</code>	fullscreenerror

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

onoffline	offline
ononline	online
onpagehide	pagehide
onpageshow	pageshow
onpopstate	popstate
onresize	resize
onscroll	scroll
onstorage	storage
onunload	



Too slow? Try reading the multipage copy of

**Note:** Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://www.whatwg.org/html>

The following are IDL attributes defined as IDL attributes:  
<https://web.archive.org/web/20130103030411/http://www.whatwg.org/html>

ported on Document objects

Event han...	
onreadystatechange	readystatechange

### 7.1.6.3 Event firing

Certain operations and methods are defined as firing events on elements. For example, the `click()` method on the `HTMLElement` interface is defined as firing a `click` event on the element. [DOMEVENTS]

**Firing a simple event named e** means that a trusted event with the name `e`, which does not bubble (except where otherwise stated) and is not cancelable (except where otherwise stated), and which uses the `Event` interface, must be created and dispatched at the given target.

**Firing a synthetic mouse event named e** means that an event with the name `e`, which is trusted (except where otherwise stated), does not bubble (except where otherwise stated), is not cancelable (except where otherwise stated), and which uses the `MouseEvent` interface, must be created and dispatched at the given target. The event object must have its `screenX`, `screenY`, `clientX`, `clientY`, and `button` attributes initialized to 0, its `ctrlKey`, `shiftKey`, `altKey`, and `metaKey` attributes initialized according to the current state of the key input device, if any (false for any keys that are not available), its `detail` attribute initialized to 1, and its `relatedTarget` attribute initialized to null. The `getModifierState()` method on the object must return values appropriately describing the state of the key input device at the time the event is created.

**Firing a click event** means firing a synthetic mouse event named `click`, which bubbles and is cancelable.

The default action of these events is to do nothing except where otherwise stated.

### 7.1.6.4 Events and the Window object

When an event is dispatched at a DOM node in a `Document` in a browsing context, if the event is not a `load` event, the user agent must act as if, for the purposes of event dispatching, the `Window` object is the parent of the `Document` object. [DOM]

## 7.2 Base64 utility methods

The `atob()` and `btoa()` methods allow authors to transform content to and from the base64 encoding.

```
[NoInterfaceObject]
interface WindowBase64 {
 DOMString btoa(DOMString btoa);
 DOMString atob(DOMString atob);
};

Window implements WindowBase64;
```

**Note:** In these APIs, for mnemonic purposes, the "b" can be considered to stand for "binary", and the "a" for "ASCII". In practice, though, for primarily historical reasons, both the input and output of these functions are Unicode strings.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

returns.

Throws an `InvalidCharacterError` exception if the input string contains any out-of-range characters.

```
result = window.atob(data)
```

Takes the input data, in the form of a Unicode string containing base64-encoded binary data, decodes it, and returns a string consisting of characters in the range U+0000 to U+00FF, each representing a binary byte with values 0x00 to 0xFF respectively, corresponding to that binary data.

Throws an `InvalidCharacterError` exception if the input string is not valid base64.

## Too slow? Try reading the multipage copy of

**Note: This is a note about the Worker API.** See the [spec instead](#): <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

[e \(part of Web\)](#)

The `btoa()` code point is `//whatwg.org/html`. The eight-bit sequence of octets, and return the result. [RFC4648]

contains any character whose octet whose *n*th octet is mapped by the base64 algorithm to that

The `atob()` method must run the following steps to parse the string passed in the method's first argument:

1. Let *input* be the string being parsed.
2. Let *position* be a pointer into *input*, initially pointing at the start of the string.
3. Remove all space characters from *input*.
4. If the length of *input* divides by 4 leaving no remainder, then: if *input* ends with one or two U+003D EQUALS SIGN (=) characters, remove them from *input*.
5. If the length of *input* divides by 4 leaving a remainder of 1, throw an `InvalidCharacterError` exception and abort these steps.
6. If *input* contains a character that is not in the following list of characters and character ranges, throw an `InvalidCharacterError` exception and abort these steps:
  - o U+002B PLUS SIGN (+)
  - o U+002F SOLIDUS (/)
  - o Alphanumeric ASCII characters
7. Let *output* be a string, initially empty.
8. Let *buffer* be a buffer that can have bits appended to it, initially empty.
9. While *position* does not point past the end of *input*, run these substeps:
  1. Find the character pointed to by *position* in the first column of the following table. Let *n* be the number given in the second cell of the same row.

Go NOV JAN FEB  
 ◀ 03 ▶  
 2012 2013 2014 ▾ About this capture

[3,445 captures](#)  
 5 Jun 2004 - 22 Aug 2020

```
H: 7
I: 8
J: 9
K: 10
L: 11
M: 12
N: 13
O: 14
P: 15
Q: 16
R: 17
S: 18
T: 19
U: 20
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
. -->
k: 36
l: 37
m: 38
n: 39
o: 40
p: 41
q: 42
r: 43
s: 44
t: 45
u: 46
v: 47
w: 48
x: 49
y: 50
z: 51
0: 52
1: 53
2: 54
3: 55
4: 56
5: 57
6: 58
7: 59
8: 60
9: 61
+: 62
!: 63
```

2. Append to *buffer* the six bits corresponding to *number*, most significant bit first.
3. If *buffer* has accumulated 24 bits, interpret them as three 8-bit big-endian numbers. Append the three characters with code points equal to those numbers to *output*, in the same order, and then empty *buffer*.
4. Advance *position* by one character.
10. If *buffer* is not empty, it contains either 12 or 18 bits. If it contains 12 bits, discard the last four and interpret the remaining eight as an 8-bit big-endian number. If it contains 18 bits, discard the last two and interpret the remaining 16 as two 8-bit big-endian numbers. Append the one or two characters with code points equal to those one or two numbers to *output*, in the same order.

**Note:** The discarded bits mean that, for instance, `atob("YQ")` and `atob("YR")` both return "a".

11. Return *output*.

**Note:** Some base64 encoders add newlines or other whitespace to their output. The `atob()` method throws an exception if its input contains characters other than those described by the regular expression bracket expression `[+/-=0-9A-Za-z]`, so other characters need to be removed before `atob()` is used for decoding.

### 7.3 Timers

The `setTimeout()` and `setInterval()` methods allow authors to schedule timer-based callbacks.

[NoInterfaceObject]

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
long setInterval(DOMString handler, optional long timeout, any... arguments);
void clearInterval(long handle);
};

Window implements WindowTimers;
```

This box is non-normative. Implementation requirements are given below this box.

`handle = window.setTimeout( handler [, timeout [, arguments...]] )`

Schedules a timeout through to the `handler`.

`handle` Too slow? Try reading the multipage copy of

`Sci` the spec instead: <https://web.archive.org>

`window` /web/20130103030411/http:

`Cal`

`handle` //whatwg.org/html

Schedules a timeout to run `handler` every `timeout` milliseconds. Any `arguments` are passed straight through to the `handler`.

`handle = window.setInterval( code [, timeout] )`

Schedules a timeout to compile and run `code` every `timeout` milliseconds.

`window.clearInterval( handle )`

Cancels the timeout set with `setInterval()` identified by `handle`.

**Note:** This API does not guarantee that timers will run exactly on schedule. Delays due to CPU load, other tasks, etc, are to be expected.

**Note:** The `WindowTimers` interface adds to the `Window` interface and the `WorkerUtils` interface (part of Web Workers).

Each object that implements the `WindowTimers` interface has a **list of active timers**. Each entry in this lists is identified by a number, which must be unique within the list for the lifetime of the object that implements the `WindowTimers` interface.

The `setTimeout()` method must run the following steps:

1. Let `handle` be a user-agent-defined integer that is greater than zero that will identify the timeout to be set by this call in the list of active timers.
2. Add an entry to the list of active timers for `handle`.
3. Get the timed task `handle` in the list of active timers, and let `task` be the result. This algorithm uses the first argument to the method (`handler`) and, if there are any, the third and subsequent arguments to the method (`arguments`), to establish precisely what `task` does.
4. Let `timeout` be the second argument to the method, or zero if the argument was omitted.
5. If the currently running task is a task that was created by the `setTimeout()` method, and `timeout` is less than 4, then increase `timeout` to 4.
6. Return `handle`, and then continue running this algorithm asynchronously.
7. If the method context is a `Window` object, wait until the `Document` associated with the method context has been fully active for a further `timeout` milliseconds (not necessarily consecutively).

Otherwise, if the method context is a `WorkerUtils` object, wait until `timeout` milliseconds have passed with the worker not suspended (not necessarily consecutively).

Otherwise, act as described in the specification that defines that the `WindowTimers` interface is implemented by some other object.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

passed as the first argument, happens in the algorithms defined in Web IDL, before this algorithm is invoked.

So for example, the following rather silly code will result in the log containing "ONE TWO":

```
var log = '';
function logger(s) { log += s + ' '; }

setTimeout({ toString: function () {
 setTimeout("logger('ONE')", 100);
 return "logger('TWO')";
}}
```



9. Opt  
Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*power usage of the timers is reduced; owing the processor to*

10. Queue the task *task*.

**Note:** Once the task has been processed, it is safe to remove the entry for handle from the list of active timers (there is no way for the entry's existence to be detected past this point, so it does not technically matter one way or the other).

The `setInterval()` method must run the following steps:

1. Let *handle* be a user-agent-defined integer that is greater than zero that will identify the timeout to be set by this call in the list of active timers.
2. Add an entry to the list of active timers for *handle*.
3. Get the timed task *handle* in the list of active timers, and let *task* be the result. This algorithm uses the first argument to the method (*handler*) and, if there are any, the third and subsequent arguments to the method (*arguments*), to establish precisely what *task* does.
4. Let *timeout* be the second argument to the method, or zero if the argument was omitted.
5. If *timeout* is less than 4, then increase *timeout* to 4.
6. Return *handle*, and then continue running this algorithm asynchronously.
7. *Wait:* If the method context is a `Window` object, wait until the `Document` associated with the method context has been fully active for a further *interval* milliseconds (not necessarily consecutively).  
Otherwise, if the method context is a `WorkerUtils` object, wait until *interval* milliseconds have passed with the worker not suspended (not necessarily consecutively).  
Otherwise, act as described in the specification that defines that the `WindowTimers` interface is implemented by some other object.
8. Optionally, wait a further user-agent defined length of time.

**Note:** This is intended to allow user agents to pad timeouts as needed to optimise the power usage of the device. For example, some processors have a low-power mode where the granularity of timers is reduced; on such platforms, user agents can slow timers down to fit this schedule instead of requiring the processor to use the more accurate mode with its associated higher power usage.

9. Queue the *task* task.

10. Return to the step labeled *wait*.

The `clearTimeout()` and `clearInterval()` methods must clear the entry identified as *handle* from the list of active timers of the `WindowTimers` object on which the method was invoked, where *handle* is the argument passed to the method, if any. (If *handle* does

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

running is implemented (a `Window` or `WorkerUtils` object).

When the above methods are invoked and try to **get the timed task handle** in list *list*, they must run the following steps:

1. If the first argument to the invoked method is a `Function`, then return a task that runs the following substeps, and then abort these steps:

1. If the entry for *handle* in *list* has been cleared, then abort this task's substeps.
2. Call the `Function`. Use the third and subsequent arguments to the invoked *m* (if any) as the arguments for *on*. [ECMA262]

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

*ed with the this  
ie code was running in*

Oth

//whatwg.org/html

2. Let *s*

3. Let *script language* be JavaScript.

4. If the method context is a `Window` object, let *global object* be the method context, let *browsing context* be the browsing context with which *global object* is associated, let *document* and *referrer source* be the `Document` associated with *global object*, let *character encoding* be the character encoding of the `Document` associated with *global object* (this is a reference, not a copy), and let *base URL* be the base URL of the `Document` associated with *global object* (this is a reference, not a copy).

Otherwise, if the method context is a `WorkerUtils` object, let *global object*, *browsing context*, *document*, *referrer source*, *character encoding*, and *base URL* be the script's global object, script's browsing context, script's document, script's referrer source, script's URL character encoding, and script's base URL (respectively) of the script that the run a worker algorithm created when it created the method context.

Otherwise, act as described in the specification that defines that the `WindowTimers` interface is implemented by some other object.

5. Return a task that checks if the entry for *handle* in *list* has been cleared, and if it has not, creates a script using *script source* as the script source, the URL where *script source* can be found, *scripting language* as the scripting language, *global object* as the global object, *browsing context* as the browsing context, *document* as the document, *referrer source* as the referrer source, *character encoding* as the URL character encoding, and *base URL* as the base URL.

The task source for these tasks is the **timer task source**.

## 7.4 User prompts

### 7.4.1 Simple dialogs

This box is non-normative. Implementation requirements are given below this box.

`window.alert(message)`

Displays a modal alert with the given message, and waits for the user to dismiss it.

A call to the `navigator.yieldForStorageUpdates()` method is implied when this method is invoked.

`result = window.confirm(message)`

Displays a modal OK/Cancel prompt with the given message, waits for the user to dismiss it, and returns true if the user clicks OK and false if the user clicks Cancel.

A call to the `navigator.yieldForStorageUpdates()` method is implied when this method is invoked.

`result = window.prompt(message [, default])`

Displays a modal text field prompt with the given message, waits for the user to dismiss it, and returns the value that the user entered. If the user cancels the prompt, then returns null instead. If the second argument is present, then the given value is used as a default.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. If the event loop's termination nesting level is non-zero, optionally abort these steps.
2. Release the storage mutex.
3. Optionally, abort these steps. (For example, the user agent might give the user the option to ignore all alerts, and would thus abort at this step whenever the method was invoked.)
4. Show the given *message* to the user.
5. Optio



The `config` **Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. If th the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
2. Rel /web/20130103030411/http:/
3. Opt pror /whatwg.org/html ↗ option to ignore all

4. Show the given *message* to the user, and ask the user to respond with a positive or negative response.
5. Pause until the user responds either positively or negatively.
6. If the user responded positively, return true; otherwise, the user responded negatively: return false.

The `prompt(message, default)` method, when invoked, must run the following steps:

1. If the event loop's termination nesting level is non-zero, optionally abort these steps, returning null.
2. Release the storage mutex.
3. Optionally, return null and abort these steps. (For example, the user agent might give the user the option to ignore all prompts, and would thus abort at this step whenever the method was invoked.)
4. Show the given *message* to the user, and ask the user to either respond with a string value or abort. The second argument is optional. If the second argument (*default*) is present, then the response must be defaulted to the value given by *default*.
5. Pause while waiting for the user's response.
6. If the user aborts, then return null; otherwise, return the string that the user responded with.

#### 7.4.2 Printing

*This box is non-normative. Implementation requirements are given below this box.*

`window.print()`

Prompts the user to print the page.

A call to the `navigator.yieldForStorageUpdates()` method is implied when this method is invoked.

When the `print()` method is invoked, if the `Document` is ready for post-load tasks, then the user agent must synchronously run the printing steps. Otherwise, the user agent must only set the `print when loaded` flag on the `Document`.

User agents should also run the printing steps whenever the user asks for the opportunity to obtain a physical form (e.g. printed copy), or the representation of a physical form (e.g. PDF copy), of a document.

The **printing steps** are as follows:

1. The user agent may display a message to the user or abort these steps (or both).
  - || For instance, a kiosk browser could silently ignore any invocations of the `print()` method.
  - || For instance, a browser on a mobile device could detect that there are no printers in the vicinity and display a message saying so before continuing to offer a "save to PDF" option.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

3. The user agent must release the storage mutex.
4. The user agent should offer the user the opportunity to obtain a physical form (or the representation of a physical form) of the document. The user agent may wait for the user to either accept or decline before returning; if so, the user agent must pause while the method is waiting. Even if the user agent doesn't wait at this point, the user agent must use the state of the relevant documents as they are at this point in the algorithm if and when it eventually creates the alternate form.
5. The user agent must fire a simple event named `afterprint` at the `Window` object of the `Document` that is being printed, as well as ~~any nested browsing contexts in it~~.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

#### 7.4.3 Dial

<https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ell as showing post-printing  
1, the script could

nts are given below this box.

**`result = window.showModalDialog(url [, argument]`**

Prompts the user with the given page, waits for that page to close, and returns the return value.

A call to the `navigator.yieldForStorageUpdates()` method is implied when this method is invoked.

The `showModalDialog(url, argument)` method, when invoked, must cause the user agent to run the following steps:

1. Resolve `url` relative to the entry script's base URL.  
If this fails, then throw a `SyntaxError` exception and abort these steps.
2. If the event loop's termination nesting level is non-zero, optionally abort these steps, returning the empty string.
3. Release the storage mutex.
4. If the user agent is configured such that this invocation of `showModalDialog()` is somehow disabled, then return the empty string and abort these steps.

**Note:** User agents are expected to disable this method in certain cases to avoid user annoyance (e.g. as part of their popup blocker feature). For instance, a user agent could require that a site be white-listed before enabling this method, or the user agent could be configured to only allow one modal dialog at a time.

5. If the active sandboxing flag set of the active document of the browsing context of the script that invoked the method has its sandboxed auxiliary navigation browsing context flag set, then return the empty string and abort these steps.
6. Let the *list of background browsing contexts* be a list of all the browsing contexts that:
  - o are part of the same unit of related browsing contexts as the browsing context of the `Window` object on which the `showModalDialog()` method was called, and that
  - o have an active document whose origin is the same as the origin of the script that called the `showModalDialog()` method at the time the method was called,
 ...as well as any browsing contexts that are nested inside any of the browsing contexts matching those conditions.
7. Disable the user interface for all the browsing contexts in the *list of background browsing contexts*. This should prevent the user from navigating those browsing contexts, causing events to be sent to those browsing context, or editing any content in those browsing contexts. However, it does not prevent those browsing contexts from receiving events from sources other than the user, from running scripts, from running animations, and so forth.
8. Create a new auxiliary browsing context, with the opener browsing context being the browsing context of the `Window` object on which the `showModalDialog()` method was called. The new auxiliary browsing context has no name.

**Note:** This browsing context's `Document`'s `Window` objects all implement the `WindowModal` interface.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

was omitted.

11. Let the dialog arguments' origin be the origin of the script that called the `showModalDialog()` method.
12. Navigate the new browsing context to the absolute URL that resulted from resolving *url* earlier, with replacement enabled, and with the browsing context of the script that invoked the method as the source browsing context.
13. Spin the event loop until the new browsing context is closed. (The user agent must allow the user to indicate that the browsing context is to be closed.)

14. Reer

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

**Note: V  
be part**

xts.

) implement the

ents, appear to actually  
t () method).

```
[NoInterfaceObject] interface WindowModal {
 readonly attribute any dialogArguments;
 attribute DOMString returnValue;
};
```

This box is non-normative. Implementation requirements are given below this box.

**window.dialogArguments**Returns the *argument* argument that was passed to the `showModalDialog()` method.**window.returnValue [= value]**

Returns the current return value for the window.

Can be set, to change the value that will be returned by the `showModalDialog()` method.

Such browsing contexts have associated **dialog arguments**, which are stored along with the **dialog arguments' origin**. These values are set by the `showModalDialog()` method in the algorithm above, when the browsing context is created, based on the arguments provided to the method.

The **dialogArguments** IDL attribute, on getting, must check whether its browsing context's active document's origin is the same as the dialog arguments' origin. If it is, then the browsing context's dialog arguments must be returned unchanged. Otherwise, if the dialog arguments are an object, then the empty string must be returned, and if the dialog arguments are not an object, then the stringification of the dialog arguments must be returned.

These browsing contexts also have an associated **return value**. The return value of a browsing context must be initialized to the empty string when the browsing context is created.

The **returnValue** IDL attribute, on getting, must return the return value of its browsing context, and on setting, must set the return value to the given new value.

**Note:** The `window.close()` method can be used to close the browsing context.

## 7.5 System state and capabilities

### 7.5.1 The Navigator object

The **navigator** attribute of the **Window** interface must return an instance of the **Navigator** interface, which represents the identity and state of the user agent (the client), and allows Web pages to register themselves as potential protocol and content handlers:

```
interface Navigator {
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

```
Navigator implements NavigatorContentUtils;
Navigator implements NavigatorStorageUtils;
```

These interfaces are defined separately so that other specifications can re-use parts of the `Navigator` interface.

#### 7.5.1.1 Client identification

```
[NoInterfaceObject]
interface Navigator {
 // ...
}
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

In certain cases, Web authors are forced to work around.

This section defines a collection of attributes that can be used to determine, from script, the kind of user agent in use, in order to work around these issues.

Client detection should always be limited to detecting known current versions; future versions and unknown versions should always be assumed to be fully compliant.

*This box is non-normative. Implementation requirements are given below this box.*

##### `window.navigator.appName`

Returns the name of the browser.

##### `window.navigator.appVersion`

Returns the version of the browser.

##### `window.navigator.platform`

Returns the name of the platform.

##### `window.navigator.userAgent`

Returns the complete User-Agent header.

##### `appName`

Must return either the string "Netscape" or the full name of the browser, e.g. "Mellblom Browzernator".

##### `appVersion`

Must return either the string "4.0" or a string representing the version of the browser in detail, e.g. "1.0 (VMS; en-US) Mellblomenator/9000".

##### `platform`

Must return either the empty string or a string representing the platform on which the browser is executing, e.g. "MacIntel", "Win32", "FreeBSD i386", "WebTV OS".

##### `userAgent`

Must return the string used for the value of the "User-Agent" header in HTTP requests, or the empty string if no such header is ever sent.

**⚠Warning!** Any information in this API that varies from user to user can be used to profile the user. In fact, if enough such information is available, a user can actually be uniquely identified. For this reason, user agent implementors are strongly urged to include as little information in this API as possible.

#### 7.5.1.2 Language preferences



Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

This box is non-normative. Implementation requirements are given below this box.

`window.navigator.language`

Returns a language tag representing the user's preferred language.

`language`

Must

**⚠ Warning!**  
**profile or icon explicitly in**  
**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

to user can be used to unless the user has

**7.5.1.3 Cu**

```
[NoIn]
interface NavigatorContentUtils {
 // content handler registration
 void registerProtocolHandler(DOMString scheme, DOMString url, DOMString title);
 void registerContentHandler(DOMString mimeType, DOMString url, DOMString title);
 DOMString isProtocolHandlerRegistered(DOMString scheme, DOMString url);
 DOMString isContentHandlerRegistered(DOMString mimeType, DOMString url);
 void unregisterProtocolHandler(DOMString scheme, DOMString url);
 void unregisterContentHandler(DOMString mimeType, DOMString url);
};
```

The `registerProtocolHandler()` method allows Web sites to register themselves as possible handlers for particular schemes. For example, an online telephone messaging service could register itself as a handler of the `sms:` scheme, so that if the user clicks on such a link, he is given the opportunity to use that Web site. Analogously, the `registerContentHandler()` method allows Web sites to register themselves as possible handlers for content in a particular MIME type. For example, the same online telephone messaging service could register itself as a handler for `text/vcard` files, so that if the user has no native application capable of handling vCards, his Web browser can instead suggest he use that site to view contact information stored on vCards that he opens. [RFC5724] RFC6350

This box is non-normative. Implementation requirements are given below this box.

`window.navigator.registerProtocolHandler(scheme, url, title)``window.navigator.registerContentHandler(mimeType, url, title)`

Registers a handler for the given scheme or content type, at the given URL, with the given title.

The string "%s" in the URL is used as a placeholder for where to put the URL of the content to be handled.

Throws a `SecurityError` exception if the user agent blocks the registration (this might happen if trying to register as a handler for "http", for instance).

Throws a `SyntaxError` if the "%s" string is missing in the URL.

User agents may, within the constraints described in this section, do whatever they like when the methods are called. A UA could, for instance, prompt the user and offer the user the opportunity to add the site to a shortlist of handlers, or make the handlers his default, or cancel the request. UAs could provide such a UI through modal UI or through a non-modal transient notification interface. UAs could also simply silently collect the information, providing it only when relevant to the user.

User agents should keep track of which sites have registered handlers (even if the user has declined such registrations) so that the user is not repeatedly prompted with the same request.

The arguments to the methods have the following meanings and corresponding implementation requirements. The requirements that involve throwing exceptions must be processed in the order given below, stopping at the first exception thrown. (So the exceptions for the first argument take precedence over the exceptions for the second argument.)

**`scheme (registerProtocolHandler() only)`**

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the `registerProtocolHandler()` method is invoked with a scheme that is neither a whitelisted scheme nor a scheme whose value starts with the substring "web+" and otherwise contains only lowercase ASCII letters, and whose length is at least five characters (including the "web+" prefix), the user agent must throw a `SecurityError` exception.

The following schemes are the **whitelisted schemes**:

- irc
- geo
- mailto
- magnet



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- [Archive](#)

**Note:** This list can be changed. If there are schemes that should be added, please send feedback.

**Note:** This list excludes any schemes that could reasonably be expected to be supported inline, e.g. in an `iframe`, such as `http` or (more theoretically) `gopher`. If those were supported, they could potentially be used in man-in-the-middle attacks, by replacing pages that have frames with such content with content under the control of the protocol handler. If the user agent has native support for the schemes, this could further be used for cookie-theft attacks.

#### `mimeType (registerContentHandler() only)`

A MIME type, such as `model/vnd.flatland.3dml` or `application/vnd.google-earth.kml+xml`. The MIME type must be compared in an ASCII case-insensitive manner by user agents for the purposes of comparing with MIME types of documents that they consider against the list of registered handlers.

User agents must compare the given values only to the MIME type/subtype parts of content types, not to the complete type including parameters. Thus, if `mimeType` values passed to this method include characters such as commas or whitespace, or include MIME parameters, then the handler being registered will never be used.

**Note:** The type is compared to the MIME type used by the user agent after the sniffing algorithms have been applied.

If the `registerContentHandler()` method is invoked with a MIME type that is in the type blacklist or that the user agent has deemed a privileged type, the user agent must throw a `SecurityError` exception.

The following MIME types are in the **type blacklist**:

- `application/x-www-form-urlencoded`
- `application/xhtml+xml`
- `application/xml`
- `image/gif`
- `image/jpeg`
- `image/png`
- `image/svg+xml`
- `multipart/x-mixed-replace`
- `text/cache-manifest`
- `text/css`
- `text/html`
- `text/ping`
- `text/plain`
- `text/xml`
- All types that the user agent supports displaying natively in a browsing context during navigation, except for `application/rss+xml` and `application/atom+xml`

**Note:** This list can be changed. If there are MIME types that should be added, please send feedback.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

User agents must throw a `SyntaxError` if resolving the `url` argument relative to the entry script's base URL, is not successful.

**Note:** *The resulting absolute URL would by definition not be a valid URL as it would include the string "%s" which is not a valid component in a URL.*

User agents must throw a `SecurityError` exception if the resulting absolute URL has an origin that differs from the origin of the entry script.



of the URL.

The spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

With the `rel="noopener"` attribute, the user agent must not resolve the resulting URL, but instead use the context to the resulting URL using the `getURL()` method (or equivalent for non-`HTMLDocument` objects).

To get the escaped version of the absolute URL of the content in question, the user agent must replace every character in that absolute URL that doesn't match the `<query>` production defined in RFC 3986 by the percent-encoded form of that character. [RFC3986]

If the user had visited a site at `http://example.com/` that made the following call:

```
navigator.registerContentHandler('application/x-soup', 'soup?url=%s', 'SoupWeb™')
```

...and then, much later, while visiting `http://www.example.net/`, clicked on a link such as:

```
Download our Chicken Kiwi soup!
```

...then, assuming this `chickenkiwi.soup` file was served with the MIME type `application/x-soup`, the UA might navigate to the following URL:

```
http://example.com/soup?url=http://www.example.net/chickenk%C3%AFwi.soup
```

This site could then fetch the `chickenkiwi.soup` file and do whatever it is that it does with soup (synthesize it and ship it to the user, or whatever).

### title

A descriptive title of the handler, which the UA might use to remind the user what the site in question is.

This section does not define how the pages registered by these methods are used, beyond the requirements on how to process the `url` value (see above). To some extent, the processing model for navigating across documents defines some cases where these methods are relevant, but in general UAs may use this information wherever they would otherwise consider handing content to native plugins or helper applications.

UAs must not use registered content handlers to handle content that was returned as part of a non-GET transaction (or rather, as part of any non-idempotent transaction), as the remote site would not be able to fetch the same data.

In addition to the registration methods, there are also methods for determining if particular handlers have been registered, and for unregistering handlers.

*This box is non-normative. Implementation requirements are given below this box.*

```
state = window.navigator.isProtocolHandlerRegistered(scheme, url)
state = window.navigator.isContentHandlerRegistered(mimeType, url)
```

Returns one of the following strings describing the state of the handler given by the arguments:

`new`

Indicates that no attempt has been made to register the given handler (or that the handler has been unregistered). It would be appropriate to promote the availability of the handler or to just automatically register the handler.

`registered`

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Indicates that the given handler has been created but was rejected. Trying to register the handler again may prompt the user again.

```
state = window.navigator.unregisterProtocolHandler(scheme, url)
state = window.navigator.unregisterContentHandler(mimeType, url)
```

Unregisters the handler given by the arguments.

The `isProtocolHandlerRegistered()` method must return the handler state string that most closely describes the current state of the handler described by the two arguments to the method, where the first argument gives the scheme and the second gives the string used to build the URL of the page that will handle the requests.

The first argument must be compared to the MIME types for which custom content handlers are registered in an ASCII case-insensitive manner to find the relevant handlers.

The second argument must be preprocessed as described below, and if that is successful, must then be matched against the proto-URLs of the relevant handlers to find the described handler.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

lv describes the current state of the second gives the string an ASCII case-attached against the



The `isContentHandlerRegistered()` method must return the handler state string that most closely describes the current state of the handler described by the two arguments to the method, where the first argument gives the MIME type and the second gives the string used to build the URL of the page that will handle the requests.

The first argument must be compared to the MIME types for which custom content handlers are registered in an ASCII case-insensitive manner to find the relevant handlers.



The second argument must be preprocessed as described below, and if that is successful, must then be matched against the proto-URLs of the relevant handlers to find the described handler.

The **handler state strings** are the following strings. Each string describes several situations, as given by the following list.

### `new`

The described handler has never been registered for the given scheme or type.

The described handler was once registered for the given scheme or type, but the site has since unregistered it. If the handler were to be reregistered, the user would be notified accordingly.

The described handler was once registered for the given scheme or type, but the site has since unregistered it, but the user has indicated that the site is to be blocked from registering the type again, so the user agent would ignore further registration attempts.

### `registered`

An attempt was made to register the described handler for the given scheme or type, but the user has not yet been notified, and the user agent would ignore further registration attempts. (Maybe the user agent batches registration requests to display them when the user requests to be notified about them, and the user has not yet requested that the user agent notify it of the previous registration attempt.)

The described handler is registered for the given scheme or type (maybe, or maybe not, as the default handler).

The described handler is permanently blocked from being (re)registered. (Maybe the user marked the registration attempt as spam, or blocked the site for other reasons.)

### `declined`

An attempt was made to register the described handler for the given scheme or type, but the user has not yet been notified; however, the user might be notified if another registration attempt were to be made. (Maybe the last registration attempt was made while the page was in the background and the user closed the page without looking at it, and the user agent requires confirmation for this registration attempt.)

An attempt was made to register the described handler for the given scheme or type, but the user has not yet responded.

An attempt was made to register the described handler for the given scheme or type, but the user declined the offer. The user has not indicated that the handler is to be permanently blocked, however, so another attempt to register the described handler might result in the user being prompted again.

The described handler was once registered for the given scheme or type, but the user has since removed it. The user has not indicated that the handler is to be permanently blocked, however, so another attempt to register the described handler might result in the user being prompted again.

The `unregisterProtocolHandler()` method must unregister the handler described by the two arguments to the method, where the first argument gives the scheme and the second gives the string used to build the URL of the page that will handle the requests.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

The `unregisterContentHandler()` method must unregister the handler described by the two arguments to the method, where the first argument gives the MIME type and the second gives the string used to build the URL of the page that will handle the requests.

The first argument must be compared to the MIME types for which custom content handlers are registered in an ASCII case-insensitive manner to find the relevant handlers.

The second argument must be preprocessed as described below, and if that is successful, must then be matched against the proto-URLs of the relevant handlers to find the described handler.



## The second Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

1. If the `url` argument is a relative URL, then throw a `SecurityError` exception, aborting the method.
2. Resolve the `url` argument to an absolute URL.
3. If the `url` argument is a relative URL, then throw a `SecurityError` exception, aborting the method.
4. Return the resulting absolute URL as the result of preprocessing the argument.

### 7.5.1.3.1 Security and privacy

These mechanisms can introduce a number of concerns, in particular privacy concerns.

**Hijacking all Web usage.** User agents should not allow schemes that are key to its normal operation, such as `http` or `https`, to be rerouted through third-party sites. This would allow a user's activities to be trivially tracked, and would allow user information, even in secure connections, to be collected.

**Hijacking defaults.** User agents are strongly urged to not automatically change any defaults, as this could lead the user to send data to remote hosts that the user is not expecting. New handlers registering themselves should never automatically cause those sites to be used.

**Registration spamming.** User agents should consider the possibility that a site will attempt to register a large number of handlers, possibly from multiple domains (e.g. by redirecting through a series of pages each on a different domain, and each registering a handler for `video/mpeg` — analogous practices abusing other Web browser features have been used by pornography Web sites for many years). User agents should gracefully handle such hostile attempts, protecting the user.

**Misleading titles.** User agents should not rely wholly on the `title` argument to the methods when presenting the registered handlers to the user, since sites could easily lie. For example, a site `hostile.example.net` could claim that it was registering the "Cuddly Bear Happy Content Handler". User agents should therefore use the handler's domain in any UI along with any title.

**Hostile handler metadata.** User agents should protect against typical attacks against strings embedded in their interface, for example ensuring that markup or escape characters in such strings are not executed, that null bytes are properly handled, that over-long strings do not cause crashes or buffer overruns, and so forth.

**Leaking Intranet URLs.** The mechanism described in this section can result in secret Intranet URLs being leaked, in the following manner:

1. The user registers a third-party content handler as the default handler for a content type.
2. The user then browses his corporate Intranet site and accesses a document that uses that content type.
3. The user agent contacts the third party and hands the third party the URL to the Intranet content.

No actual confidential file data is leaked in this manner, but the URLs themselves could contain confidential information. For example, the URL could be `http://www.corp.example.com/upcoming-aquisitions/the-sample-company.egf`, which might tell the third party that Example Corporation is intending to merge with The Sample Company. Implementors might wish to consider allowing administrators to disable this feature for certain subdomains, content types, or schemes.

**Leaking secure URLs.** User agents should not send HTTPS URLs to third-party sites registered as content handlers without the user's informed consent, for the same reason that user agents sometimes avoid sending `Referer` (sic) HTTP headers from secure sites to third-party sites.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Interface interference.** User agents should be prepared to handle intentionally long arguments to the methods. For example, if the user interface exposed consists of an "accept" button and a "deny" button, with the "accept" binding containing the name of the handler, it's important that a long name not cause the "deny" button to be pushed off the screen.

**Fingerprinting users.** Since a site can detect if it has attempted to register a particular handler or not, whether or not the user responds, the mechanism can be used to store data. User agents are therefore strongly urged to treat registrations in the same manner as cookies: clearing cookies for a site should also clear all registrations for that site, and disabling cookies for a site should also disable registrations.



### 7.5.1.3.2 Sa Too slow? Try reading the multipage copy of This section the spec instead: <https://web.archive.org>

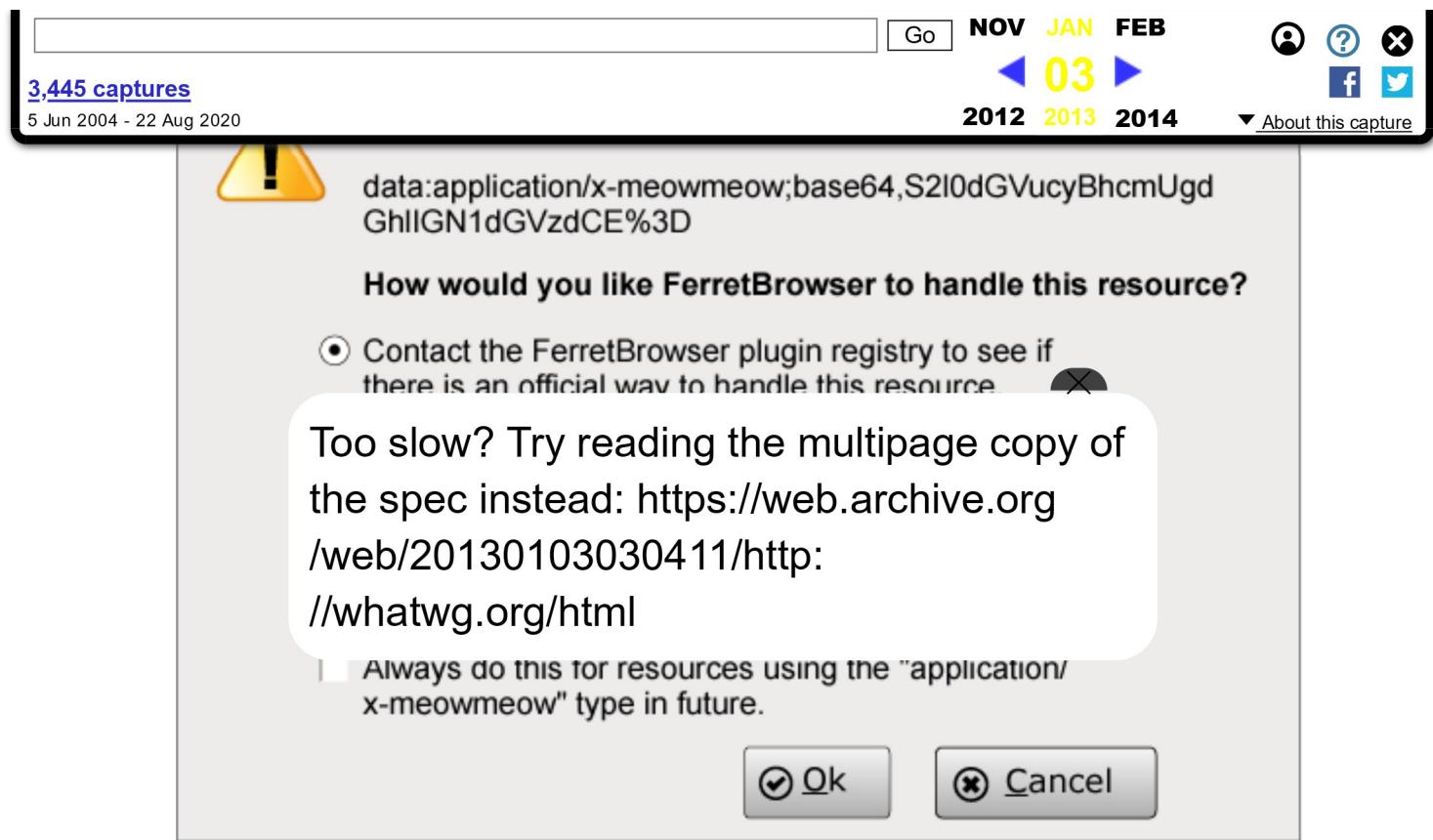
A simple `imp` /[web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)



In this dialog box, "Kittens at work" is the title of the page that invoked the method, "http://kittens.example.org/" is the URL of that page, "application/x-meowmeow" is the string that was passed to the `registerContentHandler()` method as its first argument ( `mimeType`), "http://kittens.example.org/?show=%s" was the second argument (`url`), and "Kittens-at-work displayer" was the third argument (`title`).

If the user clicks the Cancel button, then nothing further happens. If the user clicks the "Trust" button, then the handler is remembered.

When the user then attempts to fetch a URL that uses the "application/x-meowmeow" MIME type, then it might display a dialog as follows:



In this dialog, the third option is the one that was primed by the site registering itself earlier.

If the user does select that option, then the browser, in accordance with the requirements described in the previous two sections, will redirect the user to "http://kittens.example.org/?show=data%3Aapplication/x-meowmeow;base64,S2l0dGVucyBhcmUgdGhlIGN1dGVzdCE%253D".

The `registerProtocolHandler()` method would work equivalently, but for schemes instead of unknown content types.

#### 7.5.1.4 Manually releasing the storage mutex

```
[NoInterfaceObject]
interface NavigatorStorageUtils {
 void yieldForStorageUpdates();
}
```

*This box is non-normative. Implementation requirements are given below this box.*

`window.navigator.yieldForStorageUpdates()`

If a script uses the `document.cookie` API, or the `localStorage` API, the browser will block other scripts from accessing cookies or storage until the first script finishes.

Calling the `navigator.yieldForStorageUpdates()` method tells the user agent to unlock any other scripts that may be blocked, even though the script hasn't returned.

Values of cookies and items in the `Storage` objects of `localStorage` attributes can change after calling this method, whence its name.

The `yieldForStorageUpdates()` method, when invoked, must, if the storage mutex is owned by the event loop of the task that resulted in the method being called, release the storage mutex so that it is once again free. Otherwise, it must do nothing.

#### 7.5.2 The External interface

The `external` attribute of the `Window` interface must return an instance of the `External` interface. The same object must be returned each time.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

This box is non-normative. Implementation requirements are given below this box.

```
window.external.AddSearchProvider(url)
```

Adds the search engine described by the OpenSearch description document at *url*. [OPENSEARCH]

The OpenSearch description document has to be on the same server as the script that calls this method.

```
installed = window.external.IsSearchProviderInstalled(url)
```

Return value: A Boolean value indicating whether the user agent has installed the search engine described by the OpenSearch description document at *url*.

- 0 Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
- 1 e.
- 2 //whatwg.org/html

The *url* is compared to the URLs of the results pages of the installed search engines using a prefix match. Only results pages on the same domain as the script that calls this method are checked.

**Note:** Another way of exposing search engines using OpenSearch description documents is using a `link` element with the `search` link type.

The `AddSearchProvider()` method, when invoked, must run the following steps:

1. Optionally, abort these steps. User agents may implement the method as a stub method that never does anything, or may arbitrarily ignore invocations with particular arguments for security, privacy, or usability reasons.
2. Resolve the value of the method's first argument relative to the entry script's base URL.
3. If this fails, abort these steps.
4. Process the resulting absolute URL as the URL to an OpenSearch description document. [OPENSEARCH]

The `IsSearchProviderInstalled()` method, when invoked, must run the following steps:

1. Optionally, return 0 and abort these steps. User agents may implement the method as a stub method that never returns a non-zero value, or may arbitrarily ignore invocations with particular arguments for security, privacy, or usability reasons.
2. If the origin of the entry script is an opaque identifier (i.e. it has no host component), then return 0 and abort these steps.
3. Let *host1* be the host component of the origin of the entry script.
4. Resolve the `scriptURL` argument relative to the entry script's base URL.
5. If this fails, return 0 and abort these steps.
6. Let *host2* be the <host> component of the resulting absolute URL.
7. If the longest suffix in the Public Suffix List that matches the end of *host1* is different than the longest suffix in the Public Suffix List that matches the end of *host2*, then return 0 and abort these steps. [PSL]



If the next domain component of *host1* and *host2* after their common suffix are not the same, then return 0 and abort these steps.

Domain labels must be compared after applying the IDNA ToASCII algorithm to them, with both the AllowUnassigned and UseSTD3ASCIIRules flags set, in an ASCII case-insensitive manner. [RFC3490]

8. Let *search engines* be the list of search engines known by the user agent and made available to the user by the user agent for which the resulting absolute URL is a prefix match of the search engine's URL, if any. For search engines registered using OpenSearch description documents, the URL of the search engine corresponds to the URL given in a `Url` element whose `rel`

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

return 2 and abort these steps.

11. Return 1.

## 7.6 Images

```

interface ImageBitmap {
 // opaque object
};

call
type
 Too slow? Try reading the multipage copy of
 the spec instead: https://web.archive.org
 /web/20130103030411/http:
 //whatwg.org/html

 CanvasRenderingContext2D or
 ImageBitmap) ImageBitmapSource;

[NoInterfaceObject]
interface ImageBitmapFactories {
 void createImageBitmap(ImageBitmapSource image, ImageBitmapCallback _callback, optional
 long sx, long sy, long sw, long sh);
};

Window implements ImageBitmapFactories;
WorkerUtils implements ImageBitmapFactories;

```

An `ImageBitmap` object represents a bitmap image that can be painted to a canvas without undue latency.

**Note:** The exact judgement of what is undue latency of this is left up to the implementer, but in general if making use of the bitmap requires network I/O, or even local disk I/O, then the latency is probably undue; whereas if it only requires a blocking read from a GPU or system RAM, the latency is probably acceptable.

This box is non-normative. Implementation requirements are given below this box.

`Window.createImageBitmap(image, callback [, sx, sy, sw, sh ])`

Takes `image`, which can be an `img` element, `video`, or `canvas` element, a `Blob` object, an `ImageData` object, a `CanvasRenderingContext2D` object, or another `ImageBitmap` object, and asynchronously calls `callback` with a new `ImageBitmap` as its argument when it has created one.

If no `ImageBitmap` object can be constructed, for example because the provided `image` data is not actually an image, then the `callback` is invoked with null as the value instead.

If `sx`, `sy`, `sw`, and `sh` arguments are provided, the source image is cropped to the given pixels, with any pixels missing in the original replaced by transparent black. These coordinates are in the source image's pixel coordinate space, *not* in CSS pixels.

Throws an `InvalidStateError` exception if the source image is not in a valid state (e.g. an `img` element that hasn't finished loading, or a `CanvasRenderingContext2D` object whose bitmap data has zero length along one or both dimensions). Throws a `SecurityError` exception if the script is not allowed to access the image data of the source image (e.g. a `video` that is CORS-cross-origin, or a `canvas` being drawn on by a script in a worker from another origin).

An `ImageBitmap` object always has associated bitmap data, with a width and a height. However, it is possible for this data to be corrupted. If an `ImageBitmap` object's media data can be decoded without errors, it is said to be **fully decodable**.

An `ImageBitmap` object can be obtained from a variety of different objects, using the `createImageBitmap()` method. This method takes two arguments, `image` and `callback`. When invoked, the method must act as follows:

If `image` is an `img` element

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

SecurityError exception and abort these steps.

4. If the `img` element's media data is not a bitmap (e.g. it's a vector graphic), then throw an `InvalidStateError` exception and abort these steps.

5. Create a new `ImageBitmap` object.

6. Let the `ImageBitmap` object's bitmap data be a copy of the `img` element's media data, cropped to the source rectangle. If this is an animated image, the `ImageBitmap` object's bitmap data must only be taken from the first frame of the image.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If `image` is a `video` element:

and abort these steps.

3. If the origin of the `video` element is not the same origin as the entry script's origin, then throw a `SecurityError` exception and abort these steps.

4. If the `video` element's `readyState` attribute is either `HAVE NOTHING` or `HAVE_METADATA`, then throw an `InvalidStateError` exception and abort these steps.

5. Create a new `ImageBitmap` object.

6. Let the `ImageBitmap` object's bitmap data be a copy of the frame at the current playback position, at the media resource's intrinsic width and intrinsic height (i.e. after any aspect-ratio correction has been applied), cropped to the source rectangle.

7. Return, but continue running these steps asynchronously.

8. Queue a task to invoke `callback` with the new `ImageBitmap` object as its argument.

If `image` is a `canvas` element

1. If either the `sw` or `sh` arguments are specified but zero, throw an `IndexSizeError` exception.

2. If the `canvas` element's bitmap data does not have its origin-clean flag set, then throw an `InvalidStateError` exception and abort these steps.

3. If the `canvas` element's bitmap has either a horizontal dimension or a vertical dimension equal to zero, then throw an `InvalidStateError` exception and abort these steps.

4. Create a new `ImageBitmap` object.

5. Let the `ImageBitmap` object's bitmap data be a copy of the `canvas` element's bitmap data, cropped to the source rectangle.

6. Return, but continue running these steps asynchronously.

7. Queue a task to invoke `callback` with the new `ImageBitmap` object as its argument.

If `image` is a `Blob` object

1. If either the `sw` or `sh` arguments are specified but zero, throw an `IndexSizeError` exception.

2. If the `Blob` object has been neutered through the `close` method, then throw an `InvalidStateError` exception and abort these steps.

3. Return, but continue running these steps asynchronously.

4. Read the `Blob` object's data. If an error occurs during reading of the object, then queue a task to invoke `callback` with `null` as the argument, and abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

argument, and abort these steps.

7. Create a new `ImageBitmap` object.
8. Let the `ImageBitmap` object's bitmap data be the image data read from the `Blob` object, cropped to the source rectangle. If this is an animated image, the `ImageBitmap` object's bitmap data must only be taken from the animation's poster frame, or, if there is no poster frame, the first frame of the animation.
9. Queue a task to invoke `callback` with the new `ImageBitmap` object as its argument.

**If `image` is**

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

5. Queue a task to invoke `callback` with the new `ImageBitmap` object as its argument.

**If `image` is a `CanvasRenderingContext2D` object**

1. If either the `sw` or `sh` arguments are specified but zero, throw an `IndexSizeError` exception.
2. If the `CanvasRenderingContext2D` object's scratch bitmap does not have its origin-clean flag set, then throw an `InvalidStateError` exception and abort these steps.
3. If the `CanvasRenderingContext2D` object's scratch bitmap has either a horizontal dimension or a vertical dimension equal to zero, then throw an `InvalidStateError` exception and abort these steps.
4. Create a new `ImageBitmap` object.
5. Let the `ImageBitmap` object's bitmap data be a copy of the `CanvasRenderingContext2D` object's scratch bitmap, cropped to the source rectangle.
6. Return, but continue running these steps asynchronously.
7. Queue a task to invoke `callback` with the new `ImageBitmap` object as its argument.

**If `image` is an `ImageBitmap` object**

1. If either the `sw` or `sh` arguments are specified but zero, throw an `IndexSizeError` exception.
2. Create a new `ImageBitmap` object.
3. Let the `ImageBitmap` object's bitmap data be a copy of the `image` argument's bitmap data, cropped to the source rectangle.
4. Return, but continue running these steps asynchronously.
5. Queue a task to invoke `callback` with the new `ImageBitmap` object as its argument.

When the steps above require that the user agent **crop bitmap data to the source rectangle**, the user agent must run the following steps:

1. Let `input` be the image data being cropped.
2. If the `sx`, `sy`, `sw`, and `sh` arguments are omitted, return `input`.
3. Place `input` on an infinite transparent black grid plane, positioned so that its top left corner is at the origin of the plane, with the `x`-coordinate increasing to the right, and the `y`-coordinate increasing down, and with each pixel in the `input` image data occupying a cell on the plane's grid.
4. Let `output` be the rectangle on the plane denoted by the rectangle whose corners are the four points  $(sx, sy)$ ,  $(sx+sw, sy)$ ,  $(sx+sw, sy+sh)$ ,  $(sx, sy+sh)$ .

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

5. Let *output*'s pixel density be *input*'s pixel density.6. Return *output*.

Using this API, a sprite sheet can be precut and prepared:

```
var sprites = {};
function loadMySprites(loaderCallback) {
 var pending = 0;
 function getSpriteAdder(name) {
 image.src = 'mysprites.png';
 image.onload = function () {
 createImageBitmap(image, getSpriteAdder('woman'), 0, 0, 40, 40);
 createImageBitmap(image, getSpriteAdder('man'), 40, 0, 40, 40);
 createImageBitmap(image, getSpriteAdder('tree'), 80, 0, 40, 40);
 createImageBitmap(image, getSpriteAdder('hut'), 0, 40, 40, 40);
 createImageBitmap(image, getSpriteAdder('apple'), 40, 40, 40, 40);
 createImageBitmap(image, getSpriteAdder('snake'), 80, 40, 40, 40);
 };
 }

 function runDemo() {
 var canvas = document.querySelector('canvas#demo');
 var context = canvas.getContext('2d');
 context.drawImage(sprites.tree, 30, 10);
 context.drawImage(sprites.snake, 70, 10);
 }

 loadMySprites(runDemo);
}
```



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

All HTML elements may have the `hidden` content attribute set. The `hidden` attribute is a boolean attribute. When specified on an element, it indicates that the element is not yet, or is no longer, directly relevant to the page's current state, or that it is being used to declare content to be reused by other parts of the page as opposed to being directly accessed by the user. User agents should not render elements that have the `hidden` attribute specified.

In the following skeletal example, the attribute is used to hide the Web game's main screen until the user logs in:

```
<h1>The Example Game</h1>
<section id="login">
 <h2>Login</h2>
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

```
}
```

`</script>`
`</section>`
`<section id="game" hidden>`
`...`
`</section>`

--&gt;

The `hidden` attribute must not be used to hide content that could legitimately be shown in another presentation. For example, it is incorrect to use `hidden` to hide panels in a tabbed dialog, because the tabbed interface is merely a kind of overflow presentation — one could equally well just show all the form controls in one big page with a scrollbar. It is similarly incorrect to use this attribute to hide content just from one presentation — if something is marked `hidden`, it is hidden from all presentations, including, for instance, screen readers.

Elements that are not themselves `hidden` must not hyperlink to elements that are `hidden`. The `for` attributes of `label` and `output` elements that are not themselves `hidden` must similarly not refer to elements that are `hidden`. In both cases, such references would cause user confusion.

Elements and scripts may, however, refer to elements that are `hidden` in other contexts.

For example, it would be incorrect to use the `href` attribute to link to a section marked with the `hidden` attribute. If the content is not applicable or relevant, then there is no reason to link to it.

It would be fine, however, to use the ARIA `aria-describedby` attribute to refer to descriptions that are themselves `hidden`. While hiding the descriptions implies that they are not useful alone, they could be written in such a way that they are useful in the specific context of being referenced from the images that they describe.

Similarly, a `canvas` element with the `hidden` attribute could be used by a scripted graphics engine as an off-screen buffer, and a form control could refer to a `hidden` `form` element using its `form` attribute.

Elements in a section hidden by the `hidden` attribute are still active, e.g. scripts and form controls in such sections still execute and submit respectively. Only their presentation to the user changes.

The `hidden` IDL attribute must reflect the content attribute of the same name.

## 8.2 Inert subtrees

A subtree of a `Document` can be marked as `inert`. When a node or one of its ancestors is `inert`, then the user agent must act as if the element was absent for the purposes of targeting user interaction events, may ignore the node for the purposes of text search user interfaces (commonly known as "find in page"), and may prevent the user from selecting text in that node. User agents should allow the user to override the restrictions on search and text selection, however.

For example, consider a page that consists of just a single inert paragraph positioned in the middle of a `body`. If a user moves their pointing device from the `body` over to the inert paragraph and clicks on the paragraph, no `mouseover` event would be fired, and the `mousemove` and `click` events would be fired on the `body` element rather than the paragraph.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

by this paragraph can additionally be marked inert through other means; being part of a modal dialog does not "protect" a node from being marked inert.)

Only one element at a time can mark a Document as being blocked by a modal dialog. When a new dialog is made to block a Document, the previous element, if any, stops blocking the Document.

**Note:** The dialog element's showModal() method makes use of this mechanism.



### 8.2.1 The

The inert; Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

When an ele /web/20130103030411/http://whatwg.org/html

**Note:** E parts o that no

ert.

to clearly mark what  
is worth remembering  
ers on small devices or

with magnifiers, and even users just using particularly small windows might not be able to see the active part of a page and may get frustrated if inert sections are not obviously inert. For individual controls, the disabled attribute is probably more appropriate.

## 8.3 Activation

This box is non-normative. Implementation requirements are given below this box.

`element.click()`

Acts as if the element was clicked.

The click() method must run synthetic click activation steps on the element.

## 8.4 Focus

When an element is focused, key events received by the document must be targeted at that element. There may be no element focused; when no element is focused, key events received by the document must be targeted at the body element, if there is one, or else at the Document's root element, if there is one. If there is no root element, key events must not be fired.

User agents may track focus for each browsing context or Document individually, or may support only one focused element per top-level browsing context — user agents should follow platform conventions in this regard.

Which elements within a top-level browsing context currently have focus must be independent of whether or not the top-level browsing context itself has the system focus.

When a child browsing context is focused, its browsing context container must also have focus.

**Note:** When an element is focused, the element matches the CSS :focus pseudo-class.

### 8.4.1 Sequential focus navigation and the tabindex attribute

The tabindex content attribute allows authors to control whether an element is supposed to be focusable, whether it is supposed to be reachable using sequential focus navigation, and what is to be the relative order of the element for the purposes of sequential focus navigation. The name "tab index" comes from the common use of the "tab" key to navigate through the focusable elements. The term "tabbing" refers to moving forward through the focusable elements that can be reached using sequential focus navigation.

The tabindex attribute, if specified, must have a value that is a valid integer.

Each element can have a tabindex focus flag set, as defined below. This flag is a factor that contributes towards determining whether an element is focusable, as described in the next section.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Modulo platform conventions, it is suggested that for the following elements, the tabindex focus flag be set:

- a elements that have an href attribute
- link elements that have an href attribute
- button elements
- input elements whose type attribute are not in the Hidden state



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

user to begin a drag

- Editing hosts
- Browsing context containers
- Sorting interface th elements

**Note:** One valid reason to ignore the platform conventions and always allow an element to be focused (by setting its tabindex focus flag) would be if the user's only mechanism for activating an element is through a keyboard action that triggers the focused element.

#### If the value is a negative integer

The user agent must set the element's tabindex focus flag, but should not allow the element to be reached using sequential focus navigation.

**Note:** One valid reason to ignore the requirement that sequential focus navigation not allow the author to lead to the element would be if the user's only mechanism for moving the focus is sequential focus navigation. For instance, a keyboard-only user would be unable to click on a text field with a negative tabindex, so that user's user agent would be well justified in allowing the user to tab to the control regardless.

#### If the value is a zero

The user agent must set the element's tabindex focus flag, should allow the element to be reached using sequential focus navigation, and should follow platform conventions to determine the element's relative order.

#### If the value is greater than zero

The user agent must set the element's tabindex focus flag, should allow the element to be reached using sequential focus navigation, and should place the element in the sequential focus navigation order so that it is:

- before any focusable element whose tabindex attribute has been omitted or whose value, when parsed, returns an error,
- before any focusable element whose tabindex attribute has a value equal to or less than zero,
- after any element whose tabindex attribute has a value greater than zero but less than the value of the tabindex attribute on the element,
- after any element whose tabindex attribute has a value equal to the value of the tabindex attribute on the element but that is earlier in the document in tree order than the element,
- before any element whose tabindex attribute has a value equal to the value of the tabindex attribute on the element but that is later in the document in tree order than the element, and
- before any element whose tabindex attribute has a value greater than the value of the tabindex attribute on the element.

An element that has its tabindex focus flag set but does not otherwise have an activation behavior defined has an activation behavior that does nothing.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

focusable and -1 for elements that are not focusable.

### 8.4.2 Focus management

An element is **focusable** if all of the following conditions are met:

- The element's tabindex focus flag is set.
- The element is either being rendered or is a descendant of a canvas element that represents embedded content.

• Neil Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

In addition, e.g. a play button, spin control, single area

ents of media elements (e.g. <input type=number>) ding element is disabled. (A pages on a page.)

The user agent may also make part of a details element's rendering focusable, to enable the element to be opened or closed using keyboard input. However, this is distinct from the details or summary element being focusable.

Notwithstanding the above, user agents may make *any* element or part of an element focusable, especially to aid with accessibility or to better match platform conventions.

The **focusing steps** for an element are as follows:

1. If the element is not in a Document, or if the element's Document has no browsing context, or if the element's Document's browsing context has no top-level browsing context, or if the element is not focusable, or if the element is already focused, then abort these steps.
2. If focusing the element will remove the focus from another element, then run the unfocusing steps for that element.
3. Make the element the currently focused element in its top-level browsing context.

Some elements, most notably area, can correspond to more than one distinct focusable area. If a particular area was indicated when the element was focused, then that is the area that must get focus; otherwise, e.g. when using the focus() method, the first such region in tree order is the one that must be focused.

4. The user agent may apply relevant platform-specific conventions for focusing widgets.

**Note:** For example, some platforms select the contents of a text field when that field is focused.

5. Fire a simple event named focus at the element.

User agents must synchronously run the focusing steps for an element whenever the user moves the focus to a focusable element.

The **unfocusing steps** for an element are as follows:

1. If the element is an input element, and the change event applies to the element, and the element does not have a defined activation behavior, and the user has changed the element's value or its list of selected files while the control was focused without committing that change, then fire a simple event that bubbles named change at the element.
2. Unfocus the element.
3. Fire a simple event named blur at the element.

When an element that is focused stops being a focusable element, or stops being focused without another element being explicitly focused in its stead, the user agent should synchronously run the unfocusing steps for the affected element only.

For example, this might happen because the element is removed from its Document, or has a hidden attribute added. It would also happen to an input element when the element gets disabled.

### 8.4.3 Document-level focus APIs

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

`document.hasFocus()`

Returns true if the document has focus; otherwise, returns false.

`window.focus()`

Focuses the window. Use of this method is discouraged. Allow the user to control window focus instead.

`window.blur()`

Unfocuses the window. Use of this method is discouraged. Allow the user to control window focus instead.



The `activeDocument` is **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

sed. If no element in the

*Note: V  
example  
browser* /web/20130103030411/http://whatwg.org/html

*1, by definition. For  
h focus in the parent*

The `hasFocus()` method on `Document` objects must return true if the `Document`'s browsing context is focused, and all its ancestor browsing contexts are also focused, and the top-level browsing context has the *system focus*. If the `Document` has no browsing context or if its browsing context has no top-level browsing context, then the method will always return false.

The `focus()` method on the `Window` object, when invoked, provides a hint to the user agent that the script believes the user might be interested in the contents of the browsing context of the `Window` object on which the method was invoked.

User agents are encouraged to have this `focus()` method trigger some kind of notification.

The `blur()` method on the `Window` object, when invoked, provides a hint to the user agent that the script believes the user probably is not currently interested in the contents of the browsing context of the `Window` object on which the method was invoked, but that the contents might become interesting again in the future.

User agents are encouraged to ignore calls to this `blur()` method entirely.

*Note: Historically the `focus()` and `blur()` methods actually affected the system focus, but hostile sites widely abuse this behavior to the user's detriment.*

#### 8.4.4 Element-level focus APIs

This box is non-normative. Implementation requirements are given below this box.

`element.focus()`

Focuses the element.

`element.blur()`

Unfocuses the element. Use of this method is discouraged. Focus another element instead.

Do not use this method to hide the focus ring if you find the focus ring unsightly. Instead, use a CSS rule to override the 'outline' property. Be aware, however, that if an alternative focusing style isn't made available instead, the page will be significantly less usable for people who primarily navigate pages using a keyboard, or those with reduced vision who use focus outlines to help them navigate the page.

For example, to hide the outline from links and instead use a yellow background, you could use:

```
:link:focus, :visited:focus { outline: none; background: yellow; color: black; }
```

The `focus()` method, when invoked, must run the following algorithm:

1. If the element is marked as *locked for focus*, then abort these steps.
2. Mark the element as **locked for focus**.
3. Run the focusing steps for the element.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

For example, if the `blur()` method is unwisely being used to remove the focus ring for aesthetics reasons, the page would become unusable by keyboard users. Ignoring calls to this method would thus allow keyboard users to interact with the page.

## 8.5 Assigning keyboard shortcuts

### 8.5.1 Introduction

*This section is non-normative.*



Each element

The exact set

exist on the

attribute as:

In order to e

alternatives

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ng the `accesskey` attribute.

keyboard shortcuts already provided in the `accesskey`

or can provide a number of

Each alternative consists of a single character, such as a letter or digit.

User agents can provide users with a list of the keyboard shortcuts, but authors are encouraged to do so also. The `accessKeyLabel` IDL attribute returns a string representing the actual key combination assigned by the user agent.

In this example, an author has provided a button that can be invoked using a shortcut key. To support full keyboards, the author has provided "C" as a possible key. To support devices equipped only with numeric keypads, the author has provided "1" as another possibly key.

```
<input type=button value=Collect onclick="collect()" accesskey="C 1" id=c>
```

To tell the user what the shortcut key is, the author has this script here opted to explicitly add the key combination to the button's label:

```
function addShortcutKeyLabel(button) {
 if (button.accessKeyLabel != '')
 button.value += ' (' + button.accessKeyLabel + ')';
}
addShortcutKeyLabel(document.getElementById('c'));
```

Browsers on different platforms will show different labels, even for the same key combination, based on the convention prevalent on that platform. For example, if the key combination is the Control key, the Shift key, and the letter C, a Windows browser might display "Ctrl+Shift+C", whereas a Mac browser might display "^T C", while an Emacs browser might just display "C-C". Similarly, if the key combination is the Alt key and the Escape key, Windows might use "Alt+Esc", Mac might use "¬Ø", and an Emacs browser might use "M-ESC" or "ESC ESC".

In general, therefore, it is unwise to attempt to parse the value returned from the `accessKeyLabel` IDL attribute.

### 8.5.2 The `accesskey` attribute

All HTML elements may have the `accesskey` content attribute set. The `accesskey` attribute's value is used by the user agent as a guide for creating a keyboard shortcut that activates or focuses the element.

If specified, the value must be an ordered set of unique space-separated tokens that are case-sensitive, each of which must be exactly one Unicode code point in length.

In the following example, a variety of links are given with access keys so that keyboard users familiar with the site can more quickly navigate to the relevant pages:

```
<nav>
<p>
 Activities |
 Technical Reports |
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

&lt;/nav&gt;

In the following example, the search field is given two possible access keys, "s" and "0" (in that order). A user agent on a device with a full keyboard might pick `Ctrl+Alt+S` as the shortcut key, while a user agent on a small device with just a numeric keypad might pick just the plain unadorned key 0:

```
<form action="/search">
<label>Search: <input type="search" name="q" accesskey="s 0"></label>
<input type="submit">
<
```



In the adve  
adv  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the button's label to

```
button.value += ' (' + button.accessKeyLabel + ')';
}
var inputs = document.getElementsByTagName('input');
for (var i = 0; i < inputs.length; i += 1) {
 if (inputs[i].type == "submit")
 labelButton(inputs[i]);
}
</script>
```

On one user agent, the button's label might become "Compose (⌘N)". On another, it might become "Compose (Alt+⇧+1)". If the user agent doesn't assign a key, it will be just "Compose". The exact string depends on what the assigned access key is, and on how the user agent represents that key combination.

### 8.5.3 Processing model

An element's **assigned access key** is a key combination derived from the element's `accesskey` content attribute. Initially, an element must not have an assigned access key.

Whenever an element's `accesskey` attribute is set, changed, or removed, the user agent must update the element's assigned access key by running the following steps:

1. If the element has no `accesskey` attribute, then skip to the *fallback* step below.
2. Otherwise, split the attribute's value on spaces, and let *keys* be the resulting tokens.
3. For each value in *keys* in turn, in the order the tokens appeared in the attribute's value, run the following substeps:
  1. If the value is not a string exactly one Unicode code point in length, then skip the remainder of these steps for this value.
  2. If the value does not correspond to a key on the system's keyboard, then skip the remainder of these steps for this value.
  3. If the user agent can find a mix of zero or more modifier keys that, combined with the key that corresponds to the value given in the attribute, can be used as the access key, then the user agent may assign that combination of keys as the element's assigned access key and abort these steps.
4. *Fallback*: Optionally, the user agent may assign a key combination of its choosing as the element's assigned access key and then abort these steps.
5. If this step is reached, the element has no assigned access key.

Once a user agent has selected and assigned an access key for an element, the user agent should not change the element's assigned access key unless the `accesskey` content attribute is changed or the element is moved to another Document.

When the user presses the key combination corresponding to the assigned access key for an element, if the element defines a

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Note: User agents might expose elements that have an `accessKey` attribute in other ways as well, e.g. in a menu displayed in response to a specific key combination.

The `accessKey` IDL attribute must reflect the `accesskey` content attribute.

The `accessKeyLabel` IDL attribute must return a string that represents the element's assigned access key, if any. If the element does not have one, then the IDL attribute must return the empty string.



## 8.6 Editi Too slow? Try reading the multipage copy of

### 8.6.1 Mai the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `contentEditable` string and the `inherit` state, `//whatwg.org/html`

The `true` state indicates that the element is not editable.

te

and `false`. The empty

, there is a third state, the

, if its parent is. The `false`

This box is non-normative. Implementation requirements are given below this box.

`element.contentEditable [= value]`

Returns "true", "false", or "inherit", based on the state of the `contenteditable` attribute.

Can be set, to change that state.

Throws a `SyntaxError` exception if the new value isn't one of those strings.

`element.isContentEditable`

Returns true if the element is editable; otherwise, returns false.

The `contentEditable` IDL attribute, on getting, must return the string "true" if the content attribute is set to the true state, "false" if the content attribute is set to the false state, and "inherit" otherwise. On setting, if the new value is an ASCII case-insensitive match for the string "inherit" then the content attribute must be removed, if the new value is an ASCII case-insensitive match for the string "true" then the content attribute must be set to the string "true", if the new value is an ASCII case-insensitive match for the string "false" then the content attribute must be set to the string "false", and otherwise the attribute setter must throw a `SyntaxError` exception.

The `isContentEditable` IDL attribute, on getting, must return true if the element is either an editing host or editable, and false otherwise.

### 8.6.2 Making entire documents editable: The `designMode` IDL attribute

Documents have a `designMode`, which can be either enabled or disabled.

This box is non-normative. Implementation requirements are given below this box.

`document.designMode [= value]`

Returns "on" if the document is editable, and "off" if it isn't.

Can be set, to change the document's current state.

The `designMode` IDL attribute on the `Document` object takes two values, "on" and "off". When it is set, the new value must be compared in an ASCII case-insensitive manner to these two values. If it matches the "on" value, then `designMode` must be enabled, and if it matches the "off" value, then `designMode` must be disabled. Other values must be ignored.

When `designMode` is enabled, the IDL attribute must return the value "on", and when it is disabled, it must return the value "off".

The last state set must persist until the document is destroyed or the state is changed. Initially, documents must have their

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Authors are encouraged to set the 'white-space' property on editing hosts and on markup that was originally created through these editing mechanisms to the value 'pre-wrap'. Default HTML whitespace handling is not well suited to WYSIWYG editing, and line wrapping will not work correctly in some corner cases if 'white-space' is left at its default value.

As an example of problems that occur if the default 'normal' value is used instead, consider the case of the user typing "yellow \_ ball", with two spaces (here represented by "\_") between the words. With the editing rules in place for the default value of 'white-space' ('normal'), the resulting markup will either consist of "yellow&nbsp; ball" or "yellow &nbsp;ball"; i.e., there will be a non-breaking space between the two words in addition to the regular space. This is necessary because the 'normal' value for 'white-space' requires adjacent regular spaces to be collapsed together. X

In the "yellow \_ ball" under "In the spec instead: https://web.archive.org /web/20130103030411/http://whatwg.org/html

When and s

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org /web/20130103030411/http://whatwg.org/html**

breaking space) even though the line, would have visible

r spaces between the words, > rendering.

#### 8.6.4 Editing

The definition of the terms **active range**, **editing host**, and **editable**, the user interface requirements of elements that are editing hosts or editable, the `execCommand()`, `queryCommandEnabled()`, `queryCommandIndeterm()`, `queryCommandState()`, `queryCommandSupported()`, and `queryCommandValue()` methods, text selections, and the **delete the selection** algorithm are defined in the HTML Editing APIs specification. The interaction of editing and the undo/redo features in user agents is defined by the UndoManager and DOM Transaction specification. [EDITING] [UNDO]

#### 8.6.5 Spelling and grammar checking

User agents can support the checking of spelling and grammar of editable text, either in form controls (such as the value of `textarea` elements), or in elements in an editing host (e.g. using `contenteditable`).

For each element, user agents must establish a **default behavior**, either through defaults or through preferences expressed by the user. There are three possible default behaviors for each element:

##### **true-by-default**

The element will be checked for spelling and grammar if its contents are editable.

##### **false-by-default**

The element will never be checked for spelling and grammar.

##### **inherit-by-default**

The element's default behavior is the same as its parent element's. Elements that have no parent element cannot have this as their default behavior.

The `spellcheck` attribute is an enumerated attribute whose keywords are the empty string, `true` and `false`. The empty string and the `true` keyword map to the `true` state. The `false` keyword maps to the `false` state. In addition, there is a third state, the `default` state, which is the *missing value default* (and the *invalid value default*).

**Note:** The true state indicates that the element is to have its spelling and grammar checked. The default state indicates that the element is to act according to a default behavior, possibly based on the parent element's own spellcheck state, as defined below. The false state indicates that the element is not to be checked.

This box is non-normative. Implementation requirements are given below this box.

**`element.spellcheck [= value]`**

Returns true if the element is to have its spelling and grammar checked; otherwise, returns false.

Can be set, to override the default and set the `spellcheck` content attribute.

**`element.forceSpellCheck()`**

Forces the user agent to report spelling and grammar errors on the element (if checking is enabled), even if the user has never focused the element. (If the method is not invoked, user agents can hide errors in text that wasn't just entered by the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

elements's `spellcheck` content attribute is in the `default` state and the element's default behavior is true-by-default, or if the element's `spellcheck` content attribute is in the `default` state and the element's default behavior is inherit-by-default and the element's parent element's `spellcheck` IDL attribute would return true; otherwise, if none of those conditions applies, then the attribute must instead return false.

**Note:** The `spellcheck` IDL attribute is not affected by user preferences that override the `spellcheck` content attribute, and therefore might not reflect the actual spellchecking state.

On setting, if the new value is true, then the element's `spellcheck` content attribute must be set to the literal string "true", otherwise it must be set to

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

User agents

- The `and` and `Text` nodes that are children of editing hosts or editable elements.
- The `//whatwg.org/html` element.
- Text in attributes of editable elements.
- Text in `Text` nodes that are children of editing hosts or editable elements.

For text that is part of a `Text` node, the element with which the text is associated is the element that is the immediate parent of the first character of the word, sentence, or other piece of text. For text in attributes, it is the attribute's element. For the values of `input` and `textarea` elements, it is the element itself.

To determine if a word, sentence, or other piece of text in an applicable element (as defined above) is to have spelling- and grammar-checking enabled, the UA must use the following algorithm:

1. If the user has disabled the checking for this text, then the checking is disabled.
2. Otherwise, if the user has forced the checking for this text to always be enabled, then the checking is enabled.
3. Otherwise, if the element with which the text is associated has a `spellcheck` content attribute, then: if that attribute is in the `true` state, then checking is enabled; otherwise, if that attribute is in the `false` state, then checking is disabled.
4. Otherwise, if there is an ancestor element with a `spellcheck` content attribute that is not in the `default` state, then: if the nearest such ancestor's `spellcheck` content attribute is in the `true` state, then checking is enabled; otherwise, checking is disabled.
5. Otherwise, if the element's default behavior is true-by-default, then checking is enabled.
6. Otherwise, if the element's default behavior is false-by-default, then checking is disabled.
7. Otherwise, if the element's parent element has its checking enabled, then checking is enabled.
8. Otherwise, checking is disabled.

If the checking is enabled for a word/sentence/text, the user agent should indicate spelling and grammar errors in that text. User agents should take into account the other semantics given in the document when suggesting spelling and grammar corrections. User agents may use the language of the element to determine what spelling and grammar rules to use, or may use the user's preferred language settings. UAs should use `input` element attributes such as `pattern` to ensure that the resulting value is valid, where possible.

If checking is disabled, the user agent should not indicate spelling or grammar errors for that text.

Even when checking is enabled, user agents may opt to not report spelling or grammar errors in text that the user agent deems the user has no interest in having checked (e.g. text that was already present when the page was loaded, or that the user did not type, or text in controls that the user has not focused). The `forceSpellCheck()` method, when invoked on an element, must override this behavior, forcing the user agent to consider all spelling and grammar errors in text in that element for which checking is enabled to be of interest to the user.

The element with ID "a" in the following example would be the one used to determine if the word "Hello" is checked for spelling errors. In this example, it would not be.

```
<div contenteditable="true">
Hello!
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<p spellcheck="true">
 <label>Name: <input spellcheck=" false" id="b"></label>
</p>
```

**Note:** This specification does not define the user interface for spelling and grammar checkers. A user agent could offer on-demand checking, could perform continuous checking while the checking is enabled, or could use other interfaces.



## 8.7 Drag Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

This section This specific /web/20130103030411/http:

On a visual series of mou

vent that is followed by a

When using an input modality other than a pointing device, users would probably have to explicitly indicate their intention to perform a drag-and-drop operation, stating what they wish to drag and where they wish to drop it, respectively.

However it is implemented, drag-and-drop operations must have a starting point (e.g. where the mouse was clicked, or the start of the selection or element that was selected for the drag), may have any number of intermediate steps (elements that the mouse moves over during a drag, or elements that the user picks as possible drop points as he cycles through possibilities), and must either have an end point (the element above which the mouse button was released, or the element that was finally selected), or be canceled. The end point must be the last element selected as a possible drop point before the drop occurs (so if the operation is not canceled, there must be at least one element in the middle step).

### 8.7.1 Introduction

*This section is non-normative.*

To make an element draggable is simple: give the element a `draggable` attribute, and set an event listener for `dragstart` that stores the data being dragged.

The event handler typically needs to check that it's not a text selection that is being dragged, and then needs to store data into the `DataTransfer` object and set the allowed effects (copy, move, link, or some combination).

For example:

```
<p>What fruits do you like?</p>
<ol ondragstart="dragStartHandler(event)">
 <li draggable="true" data-value="fruit-apple">Apples
 <li draggable="true" data-value="fruit-orange">Oranges
 <li draggable="true" data-value="fruit-pear">Pears

<script>
 var internalDNDType = 'text/x-example'; // set this to something specific to your site
 function dragStartHandler(event) {
 if (event.target instanceof HTMLElement) {
 // use the element's data-value="" attribute as the value to be moving:
 event.dataTransfer.setData(internalDNDType, event.target.dataset.value);
 event.dataTransfer.effectAllowed = 'move'; // only allow moves
 } else {
 event.preventDefault(); // don't allow selection to be dragged
 }
 }
</script>
```

To accept a drop, the drop target has to have a `dropzone` attribute and listen to the `drop` event.

The value of the `dropzone` attribute specifies what kind of data to accept (e.g. "string:text/plain" to accept any text strings, or

Go

NOV JAN FEB

03



2012 2013 2014

About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*not the drop target is to accept the drop, and the dragover event (to specify what feedback is to be shown to the user).*

The `drop` event allows the actual drop to be performed. This event needs to be canceled, so that the `dropEffect` attribute's value can be used by the source (otherwise it's reset).

For example:

```
<p>Drop your favorite fruits below:</p>
<ol dropzone="move" string:text/x-example" ondrop="dropHandler(event)">
<!--
speci Too slow? Try reading the multipage copy of

<script>
var func /web/20130103030411/http:
ve ve //whatwg.org/html
if
 li.textContent = 'Apples';
} else if (data == 'fruit-orange') {
 li.textContent = 'Oranges';
} else if (data == 'fruit-pear') {
 li.textContent = 'Pears';
} else {
 li.textContent = 'Unknown Fruit';
}
event.target.appendChild(li);
}
</script>
```

to your site

To remove the original element (the one that was dragged) from the display, the `dragend` event can be used.

For our example here, that means updating the original markup to handle that event:

```
<p>What fruits do you like?</p>
<ol ondragstart="dragStartHandler(event)" ondragend="dragEndHandler(event)">
 ...as before...

<script>
 function dragStartHandler(event) {
 // ...as before...
 }
 function dragEndHandler(event) {
 // remove the dragged element
 event.target.parentNode.removeChild(event.target);
 }
</script>
```

### 8.7.2 The drag data store

The data that underlies a drag-and-drop operation, known as the **drag data store**, consists of the following information:

- A **drag data store item list**, which is a list of items representing the dragged data, each consisting of the following information:

#### *The drag data item kind*

The kind of data:

#### *Plain Unicode string*

Text.

#### *File*

Binary data with a file name.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Note:** Strings that contain space characters cannot be used with the dropzone attribute, so authors are encouraged to use only MIME types or custom strings (without spaces).

There is a limit of one *Plain Unicode string* item per item type string.

#### The actual data

A Unicode or binary string, in some cases with a file name (itself a Unicode string), as per the drag data item kind.

The drag data store item list is ordered in the order that the items were added to the list, most recently added last.

- The **Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>
- A d l eedback. drag data store bitmap and

#### Read/write mode

For the `dragstart` event. New data can be added to the drag data store.

#### Read-only mode

For the `drop` event. The list of items representing dragged data can be read, including the data. No new data can be added.

#### Protected mode

For all other events. The formats and kinds in the drag data store list of items representing dragged data can be enumerated, but the data itself is unavailable and no new data can be added.

- A **drag data store allowed effects state**, which is a string.

When a drag data store is **created**, it must be initialized such that its drag data store item list is empty, it has no drag data store default feedback, it has no drag data store bitmap and drag data store hot spot coordinate, its drag data store mode is protected mode, and its drag data store allowed effects state is the string "uninitialized".

### 8.7.3 The DataTransfer interface

DataTransfer objects are used to expose the drag data store that underlies a drag-and-drop operation.

```
interface DataTransfer {
 attribute DOMString dropEffect;
 attribute DOMString effectAllowed;

 readonly attribute DataTransferItemList items;

 void setDragImage(Element image, long x, long y);

 /* old interface */
 readonly attribute DOMString[] types;
 DOMString getData(DOMString format);
 void setData(DOMString format, DOMString data);
 void clearData(optional DOMString format);
 readonly attribute FileList files;
};
```

This box is non-normative. Implementation requirements are given below this box.

#### `dataTransfer.dropEffect [= value]`

Returns the kind of operation that is currently selected. If the kind of operation isn't one of those that is allowed by the `effectAllowed` attribute, then the operation will fail.

Can be set, to change the selected operation.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Can be set (during the dragstart event), to change the allowed operations.

The possible values are "none", "copy", "copyLink", "copyMove", "link", "linkMove", "move", "all", and "uninitialized".

#### `dataTransfer.items`

Returns a `DataTransferItemList` object, with the drag data.

#### `dataTransfer.setDragImage(element, x, y)`

User

`dataTr` Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org>

are being dragged, then

`one /web/20130103030411/http:`

`data = (`

`Re /whatwg.org/html`



k.

#### `dataTransfer.setData(format, data)`

Adds the specified data.

#### `dataTransfer.clearData( [ format ] )`

Removes the data of the specified formats. Removes all data if the argument is omitted.

#### `dataTransfer.files`

Returns a `FileList` of the files being dragged, if any.

`DataTransfer` objects are used during the drag-and-drop events, and are only valid while those events are being fired.

A `DataTransfer` object is associated with a drag data store while it is valid.

The `dropEffect` attribute controls the drag-and-drop feedback that the user is given during a drag-and-drop operation. When the `DataTransfer` object is created, the `dropEffect` attribute is set to a string value. On getting, it must return its current value. On setting, if the new value is one of "none", "copy", "link", or "move", then the attribute's current value must be set to the new value. Other values must be ignored.

The `effectAllowed` attribute is used in the drag-and-drop processing model to initialize the `dropEffect` attribute during the `dragenter` and `dragover` events. When the `DataTransfer` object is created, the `effectAllowed` attribute is set to a string value. On getting, it must return its current value. On setting, if drag data store's mode is the read/write mode and the new value is one of "none", "copy", "copyLink", "copyMove", "link", "linkMove", "move", "all", or "uninitialized", then the attribute's current value must be set to the new value. Otherwise it must be left unchanged.

The `items` attribute must return a `DataTransferItemList` object associated with the `DataTransfer` object. The same object must be returned each time.

The `setDragImage(element, x, y)` method must run the following steps:

1. If the `DataTransfer` object is no longer associated with a drag data store, abort these steps. Nothing happens.
2. If the drag data store's mode is not the read/write mode, abort these steps. Nothing happens.
3. If the `element` argument is an `img` element, then set the drag data store bitmap to the element's image (at its intrinsic size); otherwise, set the drag data store bitmap to an image generated from the given element (the exact mechanism for doing so is not currently specified).
4. Set the drag data store hot spot coordinate to the given `x, y` coordinate.

The `types` attribute must return a live read only array giving the strings that the following steps would produce. The same object must be returned each time.

1. Start with an empty list  $L$ .

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4. If there are any items in the drag data store item list whose kind is *File*, then add an entry to the list *L* consisting of the string "Files". (This value can be distinguished from the other values because it is not lowercase.)

5. The strings produced by these steps are those in the list *L*.

The `getData(format)` method must run the following steps:

1. If the `DataTransfer` object is no longer associated with a drag data store, return the empty string and abort these steps.
2. If the `format` argument is not one of "text", "url", or "uri-list", return the empty string and abort these steps.
3. Let **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**
4. Let `fo` be the first argument, converted to ASCII lowercase.
5. If `fo` is "/web/20130103030411/http:"
6. If `fo` is "//whatwg.org/html"
7. If there is no item in the drag data store item list whose kind is *Plain Unicode string* and whose type string is equal to `format`, return the empty string and abort these steps.
8. Let `result` be the data of the item in the drag data store item list whose kind is *Plain Unicode string* and whose type string is equal to `format`.
9. If `convert-to-URL` is true, then parse `result` as appropriate for `text/uri-list` data, and then set `result` to the first URL from the list, if any, or the empty string otherwise. [RFC2483]
10. Return `result`.

The `setData(format, data)` method must run the following steps:

1. If the `DataTransfer` object is no longer associated with a drag data store, abort these steps. Nothing happens.
2. If the drag data store's mode is not the read/write mode, abort these steps. Nothing happens.
3. Let `format` be the first argument, converted to ASCII lowercase.
4. If `format` equals "text", change it to "text/plain".  
If `format` equals "url", change it to "text/uri-list".
5. Remove the item in the drag data store item list whose kind is *Plain Unicode string* and whose type string is equal to `format`, if there is one.
6. Add an item to the drag data store item list whose kind is *Plain Unicode string*, whose type string is equal to `format`, and whose data is the string given by the method's second argument.

The `clearData()` method must run the following steps:

1. If the `DataTransfer` object is no longer associated with a drag data store, abort these steps. Nothing happens.
2. If the drag data store's mode is not the read/write mode, abort these steps. Nothing happens.
3. If the method was called with no arguments, remove each item in the drag data store item list whose kind is *Plain Unicode string*, and abort these steps.
4. Let `format` be the first argument, converted to ASCII lowercase.
5. If `format` equals "text", change it to "text/plain".  
If `format` equals "url", change it to "text/uri-list".
6. Remove the item in the drag data store item list whose kind is *Plain Unicode string* and whose type string is equal to `format`, if there is one.

**Note:** The `clearData()` method does not affect whether any files were included in the drag, so the `types` attribute's list might still not be empty after calling `clearData()` (it would still contain the "Files" string if any files were

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

File object must be used each time.

1. Start with an empty list  $L$ .
2. If the DataTransfer object is no longer associated with a drag data store, the `FileList` is empty. Abort these steps; return the empty list  $L$ .
3. If the drag data store's mode is the protected mode, abort these steps; return the empty list  $L$ .
4. For each item in the drag data store item list whose kind is `File`, add the item's data (the `File`, in particular its name and content).

5. The **Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**Note:** [7](#) **8.7.3.1 Th** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Each `DataTransferItemList`

```
interface DataTransferItemList {
 readonly attribute unsigned long length;
 getter DataTransferItem (unsigned long index);
 deleter void (unsigned long index);
 void clear();

 DataTransferItem? add(DOMString data, DOMString type);
 DataTransferItem? add(File data);
};
```

*This box is non-normative. Implementation requirements are given below this box.*

**items . length**

Returns the number of items in the drag data store.

**items[index]**

Returns the `DataTransferItem` object representing the  $index$ th entry in the drag data store.

**delete items[index]**

Removes the  $index$ th entry in the drag data store.

**items . clear()**

Removes all the entries in the drag data store.

**items . add(data)****items . add(data, type)**

Adds a new entry for the given data to the drag data store. If the data is plain text then a `type` string has to be provided also.

While the `DataTransferItemList` object's `DataTransfer` object is associated with a drag data store, the `DataTransferItemList` object's *mode* is the same as the drag data store mode. When the `DataTransferItemList` object's `DataTransfer` object is *not* associated with a drag data store, the `DataTransferItemList` object's *mode* is the *disabled mode*. The drag data store referenced in this section (which is used only when the `DataTransferItemList` object is not in the *disabled mode*) is the drag data store with which the `DataTransferItemList` object's `DataTransfer` object is associated.

The `length` attribute must return zero if the object is in the *disabled mode*; otherwise it must return the number of items in the drag data store item list.

When a `DataTransferItemList` object is not in the *disabled mode*, its supported property indices are the numbers in the range 0 ..  $n$ -1, where  $n$  is the number of items in the drag data store item list.

To determine the value of an indexed property  $i$  of a `DataTransferItemList` object, the user agent must return a

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. If the `DataTransferItemList` object is not in the *read/write mode*, throw an `InvalidStateError` exception and abort these steps.

2. Remove the *i*th item from the drag data store.

The `clear` method, if the `DataTransferItemList` object is in the *read/write mode*, must remove all the items from the drag data store. Otherwise, it must do nothing.

The `add()` method must run the following steps:



1. If the `Too slow? Try reading the multipage copy of` steps.

2. Just the spec instead: <https://web.archive.org>

↳ /web/20130103030411/http://whatwg.org/html

*string* and whose type string  
, then throw a

Otherwise, add an item to the drag data store item list whose kind is *Plain Unicode string*, whose type string is equal to the value of the method's second argument, converted to ASCII lowercase, and whose data is the string given by the method's first argument.

↳ If the first argument to the method is a `File`

Add an item to the drag data store item list whose kind is `File`, whose type string is the `type` of the `File`, converted to ASCII lowercase, and whose data is the same as the `File`'s data.

3. Determine the value of the indexed property corresponding to the newly added item, and return that value (a newly created `DataTransferItem` object).

### 8.7.3.2 The `DataTransferItem` interface

Each `DataTransferItem` object is associated with a `DataTransfer` object.

```
interface DataTransferItem {
 readonly attribute DOMString kind;
 readonly attribute DOMString type;
 void getAsString(FunctionStringCallback? _callback);
 File? getAsFile();
};

[Callback, NoInterfaceObject]
interface FunctionStringCallback {
 void handleEvent(DOMString data);
};
```

This box is non-normative. Implementation requirements are given below this box.

`item.kind`

Returns the drag data item kind, one of: "string", "file".

`item.type`

Returns the drag data item type string.

`item.getAsString(callback)`

Invokes the `callback` with the string data as the argument, if the drag data item kind is *Plain Unicode string*.

`file = item.getAsFile()`

Returns a `File` object, if the drag data item kind is `File`.

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

the `DataTransferItem` object's mode is the *disabled mode*. The drag data store referenced in this section (which is used only when the `DataTransferItem` object is not in the *disabled mode*) is the drag data store with which the `DataTransferItem` object's `DataTransfer` object is associated.

The `kind` attribute must return the empty string if the `DataTransferItem` object is in the *disabled mode*; otherwise it must return the string given in the cell from the second column of the following table from the row whose cell in the first column contains the drag data item kind of the item represented by the `DataTransferItem` object:

Kind	String
Plain Unicode str	

`File` Too slow? Try reading the multipage copy of

The `type` attribute of the drag data item

The `getAsFile` method

1. If the

2. If the `DataTransferItem` object is not in the *read/write mode* or the *read-only mode*, abort these steps. The callback is never invoked.

3. If the drag data item kind is not *Plain Unicode string*, abort these steps. The callback is never invoked.

4. Otherwise, queue a task to invoke `callback`, passing the actual data of the item represented by the `DataTransferItem` object as the argument.

The `getAsFile()` method must run the following steps:

1. If the `DataTransferItem` object is not in the *read/write mode* or the *read-only mode*, return null and abort these steps.

2. If the drag data item kind is not `File`, then return null and abort these steps.

3. Return a new `File` object representing the actual data of the item represented by the `DataTransferItem` object.

### 8.7.4 The `DragEvent` interface

The drag-and-drop processing model involves several events. They all use the `DragEvent` interface.

```
[Constructor(DOMString type, optional DragEventInit eventInitDict)]
interface DragEvent : MouseEvent {
 readonly attribute DataTransfer? dataTransfer;
};

dictionary DragEventInit : MouseEventInit {
 DataTransfer? dataTransfer;
};
```

This box is non-normative. Implementation requirements are given below this box.

`event.dataTransfer`

Returns the `DataTransfer` object for the event.

The `dataTransfer` attribute of the `DragEvent` interface must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the context information for the event.

When a user agent is required to fire a DND event named `e` at an element, using a particular drag data store, the user agent must run the following steps:

1. If `e` is `dragstart`, set the drag data store mode to the *read/write mode*.

If `e` is `drop`, set the drag data store mode to the *read-only mode*.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

drag operation if `e` is `drop` or `dragend`; and to a value based on the `effectAllowed` attribute's value and to the drag-and-drop source, as given by the following table, otherwise (i.e. if `e` is `dragenter` or `dragover`):

<code>effectAllowed</code>	<code>dropEffect</code>
"none"	"none"
"copy"	"copy"
"copyLink"	"copy", or, if appropriate, "link"
"copyMove"	"copy", or, if appropriate, "move"
"all"	"link" or "move"
"link"	"link"
"move"	"move"
"unspecified"	"copy" or "link"
"unrestricted"	"link" or "move"
"any"	"copy" or "move"
Any	"link" or "move"

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Where the table above provides **possibly appropriate alternatives**, user agents may instead use the listed alternative values if platform conventions dictate that the user has requested those alternate effects.

For example, Windows platform conventions are such that dragging while holding the "alt" key indicates a preference for linking the data, rather than moving or copying it. Therefore, on a Windows system, if "link" is an option according to the table above while the "alt" key is depressed, the user agent could select that instead of "copy" or "move".

5. Create a trusted `DragEvent` object and initialize it to have the given name `e`, to bubble, to be cancelable unless `e` is `dragleave` or `dragend`, and to have the `detail` attribute initialized to zero, the mouse and key attributes initialized according to the state of the input devices as they would be for user interaction events, the `relatedTarget` attribute initialized to null, and the `dataTransfer` attribute initialized to `dataTransfer`, the `DataTransfer` object created above.

If there is no relevant pointing device, the object must have its `screenX`, `screenY`, `clientX`, `clientY`, and `button` attributes set to 0.

6. Dispatch the newly created `DragEvent` object at the specified target element.
7. Set the drag data store allowed effects state to the current value of `dataTransfer`'s `effectAllowed` attribute. (It can only have changed value if `e` is `dragstart`.)
8. Set the drag data store mode back to the protected mode if it was changed in the first step.
9. Break the association between `dataTransfer` and the drag data store.

### 8.7.5 Drag-and-drop processing model

When the user attempts to begin a drag operation, the user agent must run the following steps. User agents must act as if these steps were run even if the drag actually started in another document or application and the user agent was not aware that the drag was occurring until it intersected with a document under the user agent's purview.

1. Determine what is being dragged, as follows:

If the drag operation was invoked on a selection, then it is the selection that is being dragged.

Otherwise, if the drag operation was invoked on a `Document`, it is the first element, going up the ancestor chain, starting at the node that the user tried to drag, that has the IDL attribute `draggable` set to true. If there is no such element, then nothing is being dragged; abort these steps, the drag-and-drop operation is never started.

Otherwise, the drag operation was invoked outside the user agent's purview. What is being dragged is defined by the document or application where the drag was started.

**Note:** `img` elements and `a` elements with an `href` attribute have their `draggable` attribute set to true by default.

2. Create a drag data store. All the DND events fired subsequently by the steps in this section must use this drag data store.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Otherwise, if it is an element that is being dragged, then the source node is the element that is being dragged.

Otherwise, the source node is part of another document or application. When this specification requires that an event be dispatched at the source node in this case, the user agent must instead follow the platform-specific conventions relevant to that situation.

**Note:** *Multiple events are fired on the source node during the course of the drag-and-drop operation.*



4. Dete

If it  
cor

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

y node that is partially or

Oth

/web/20130103030411/http:

5. If it

//whatwg.org/html

roperties set as follows:

The

"text/plain"

#### The drag data item kind

Plain Unicode string

#### The actual data

The text of the selection

Otherwise, if any files are being dragged, then add one item per file to the drag data store item list, with their properties set as follows:

#### The drag data item type string

The MIME type of the file, if known, or "application/octet-stream" otherwise.

#### The drag data item kind

File

#### The actual data

The file's contents and name.

**Note:** *Dragging files can currently only happen from outside a browsing context, for example from a file system manager application.*

If the drag initiated outside of the application, the user agent must add items to the drag data store item list as appropriate for the data being dragged, honoring platform conventions where appropriate; however, if the platform conventions do not use MIME types to label dragged data, the user agent must make a best-effort attempt to map the types to MIME types, and, in any case, all the drag data item type strings must be converted to ASCII lowercase.

User agents may also add one or more items representing the selection or dragged element(s) in other forms, e.g. as HTML.

6. If the list of dragged nodes is not empty, then extract the microdata from those nodes into a JSON form, and add one item to the drag data store item list, with its properties set as follows:

#### The drag data item type string

application/microdata+json

#### The drag data item kind

Plain Unicode string

#### The actual data

The resulting JSON string.

7. Run the following substeps:

1. Let *urls* be an empty list of absolute URLs.

2. For each *node* in the list of dragged nodes:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Add to and the result of resolving the element's BFC content attribute relative to the element.

▼ About this capture

3. If *urls* is still empty, abort these substeps.
4. Let *url string* be the result of concatenating the strings in *urls*, in the order they were added, separated by a U+000D CARRIAGE RETURN U+000A LINE FEED character pair (CRLF).
5. Add one item to the drag data store item list, with its properties set as follows:

**The drag data item type string**



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

8. Upon selection, if the drag began outside the user agent, then the element's rendering would be used; if the drag began outside the user agent, then the platform conventions for determining the drag feedback should be used.

9. Fire a DND event named `dragstart` at the source node.

If the event is canceled, then the drag-and-drop operation should not occur; abort these steps.

**Note:** Since events with no event listeners registered are, almost by definition, never canceled, drag-and-drop is always available to the user if the author does not specifically prevent it.

10. Initiate the drag-and-drop operation in a manner consistent with platform conventions, and as described below.

The drag-and-drop feedback must be generated from the first of the following sources that is available:

1. The drag data store bitmap, if any. In this case, the drag data store hot spot coordinate should be used as hints for where to put the cursor relative to the resulting image. The values are expressed as distances in CSS pixels from the left side and from the top side of the image respectively. [CSS]
2. The drag data store default feedback.

From the moment that the user agent is to **initiate the drag-and-drop operation**, until the end of the drag-and-drop operation, device input events (e.g. mouse and keyboard events) must be suppressed.

During the drag operation, the element directly indicated by the user as the drop target is called the **immediate user selection**. (Only elements can be selected by the user; other nodes must not be made available as drop targets.) However, the immediate user selection is not necessarily the **current target element**, which is the element currently selected for the drop part of the drag-and-drop operation.

The immediate user selection changes as the user selects different elements (either by pointing at them with a pointing device, or by selecting them in some other way). The current target element changes when the immediate user selection changes, based on the results of event listeners in the document, as described below.

Both the current target element and the immediate user selection can be null, which means no target element is selected. They can also both be elements in other (DOM-based) documents, or other (non-Web) programs altogether. (For example, a user could drag text to a word-processor.) The current target element is initially null.

In addition, there is also a **current drag operation**, which can take on the values "none", "copy", "link", and "move". Initially, it has the value "none". It is updated by the user agent as described in the steps below.

User agents must, as soon as the drag operation is initiated and every 350ms ( $\pm 200$ ms) thereafter for as long as the drag operation is ongoing, queue a task to perform the following steps in sequence:

1. If the user agent is still performing the previous iteration of the sequence (if any) when the next iteration becomes due, abort these steps for this iteration (effectively "skipping missed frames" of the drag-and-drop operation).
2. Fire a DND event named `drag` event at the source node. If this event is canceled, the user agent must set the current drag operation to "none" (no drag operation).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

element as follows:

## ↳ If the new immediate user selection is null

Set the current target element to null also.

## ↳ If the new immediate user selection is in a non-DOM document or application

Set the current target element to the immediate user selection.

## ↳ Otherwise



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

user selection.

`input` element whose type attribute is in the Text state) or an editing host or editable element, and the drag data store item list has an item with the drag data item type string "`text/plain`" and the drag data item kind *Plain Unicode string*

Set the current target element to the immediate user selection anyway.

↳ If the immediate user selection is an element with a `dropzone` attribute that matches the drag data store

Set the current target element to the immediate user selection anyway.

↳ If the immediate user selection is an element that itself has an ancestor element with a `dropzone` attribute that matches the drag data storeLet *new target* be the nearest (deepest) such ancestor element.If the immediate user selection is *new target*, then leave the current target element unchanged.Otherwise, fire a DND event named `dragenter` at *new target*. Then, set the current target element to *new target*, regardless of whether that event was canceled or not.

## ↳ If the immediate user selection is the body element

Leave the current target element unchanged.

## ↳ Otherwise

Fire a DND event named `dragenter` at the body element, if there is one, or at the `Document` object, if not. Then, set the current target element to the body element, regardless of whether that event was canceled or not.

2. If the previous step caused the current target element to change, and if the previous target element was not null or a part of a non-DOM document, then fire a DND event named `dragleave` at the previous target element.

3. If the current target element is a DOM element, then fire a DND event named `dragover` at this current target element.

If the `dragover` event is not canceled, run the appropriate step from the following list:↳ If the current target element is a text field (e.g. `textarea`, or an `input` element whose `type` attribute is in the Text state) or an editing host or editable element, and the drag data store item list has an item with the drag data item type string "`text/plain`" and the drag data item kind *Plain Unicode string*

Set the current drag operation to either "copy" or "move", as appropriate given the platform conventions.

↳ If the current target element is an element with a `dropzone` attribute that matches the drag data store and specifies an operationSet the current drag operation to the operation specified by the `dropzone` attribute of the current target element.↳ If the current target element is an element with a `dropzone` attribute that matches the drag data store and does not specify an operation

Set the current drag operation to "copy".

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

the event dispatch finished, as per the following table:

effectAllowed	dropEffect	Drag operation
"uninitialized", "copy", "copyLink", "copyMove", or "all"	"copy"	"copy"
"uninitialized", "link", "copyLink", "linkMove", or "all"	"link"	"link"
"uninitialized", "move", "copyMove", "linkMove", or "all"	"move"	"move"
Any other case	"none"	

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

"none" | No operation allowed, dropping here will cancel the drag-and-drop operation.

4. Otherwise, if the user ended the drag-and-drop operation (e.g. by releasing the mouse button in a mouse-driven drag-and-drop interface), or if the `drag` event was canceled, then this will be the last iteration. Run the following steps, then stop the drag-and-drop operation:

1. If the current drag operation is "none" (no drag operation), or, if the user ended the drag-and-drop operation by canceling it (e.g. by hitting the `Escape` key), or if the current target element is null, then the drag operation failed. Run these substeps:

1. Let `dropped` be false.

2. If the current target element is a DOM element, fire a DND event named `dragleave` at it; otherwise, if it is not null, use platform-specific conventions for drag cancellation.

Otherwise, the drag operation might be a success; run these substeps:

1. Let `dropped` be true.

2. If the current target element is a DOM element, fire a DND event named `drop` at it; otherwise, use platform-specific conventions for indicating a drop.

3. If the event is canceled, set the current drag operation to the value of the `dropEffect` attribute of the `DragEvent` object's `dataTransfer` object as it stood after the event dispatch finished.

Otherwise, the event is not canceled; perform the event's default action, which depends on the exact target as follows:

↳ If the current target element is a text field (e.g. `textarea`, or an `input` element whose `type` attribute is in the Text state) or an editing host or editable element, and the drag data store item list has an item with the drag data item type string "text/plain" and the drag data item kind *Plain Unicode string*

Insert the actual data of the first item in the drag data store item list to have a drag data item type string of "text/plain" and a drag data item kind that is *Plain Unicode string* into the text field or editing host or editable element in a manner consistent with platform-specific conventions (e.g. inserting it at the current mouse cursor position, or inserting it at the end of the field).

↳ Otherwise

Reset the current drag operation to "none".

2. Fire a DND event named `dragend` at the source node.

3. Run the appropriate steps from the following list as the default action of the `dragend` event:

↳ If `dropped` is true, the current target element is a text field (see below), the current drag operation is "move", and the source of the drag-and-drop operation is a selection in the DOM that is entirely contained within an editing host

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

↳ If `dropped` is `false` or if the current drag operation is "none"

The drag was canceled. If the platform conventions dictate that this be represented to the user (e.g. by animating the dragged selection going back to the source of the drag-and-drop operation), then do so.

↳ Otherwise

The event has no default action.

For the purposes of this step, a `text field` is a `textarea` element or an `input` element whose `type` attribute is in one

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*Note: U  
example  
that the*

*regions. For  
ie to scroll the page so*

*Note: T  
describ  
are involve*

*rents are fired as  
f how many documents*

### 8.7.6 Events summary

This section is non-normative.

The following events are involved in the drag-and-drop model.

Event Name	Target	Cancelable?	Drag data store mode	<code>dropEffect</code>	Default Action
<code>dragstart</code>	Source node	✓ Cancelable	Read/write mode	"none"	Initiate the drag-and-drop operation
<code>drag</code>	Source node	✓ Cancelable	Protected mode	"none"	Continue the drag-and-drop operation
<code>dragenter</code>	Immediate user selection or the body element	✓ Cancelable	Protected mode	Based on <code>effectAllowed</code> value	Reject immediate user selection as potential target element
<code>dragleave</code>	Previous target element	—	Protected mode	"none"	None
<code>dragover</code>	Current target element	✓ Cancelable	Protected mode	Based on <code>effectAllowed</code> value	Reset the current drag operation to "none"
<code>drop</code>	Current target element	✓ Cancelable	Read-only mode	Current drag operation	Varies
<code>dragend</code>	Source node	—	Protected mode	Current drag operation	Varies

Not shown in the above table: all these events bubble, and the `effectAllowed` attribute always has the value it had after the `dragstart` event, defaulting to "uninitialized" in the `dragstart` event.

### 8.7.7 The `draggable` attribute

All HTML elements may have the `draggable` content attribute set. The `draggable` attribute is an enumerated attribute. It has three states. The first state is `true` and it has the keyword `true`. The second state is `false` and it has the keyword `false`. The third state is `auto`; it has no keywords but it is the *missing value default*.

The `true` state means the element is draggable; the `false` state means that it is not. The `auto` state uses the default behavior of the user agent.

An element with a `draggable` attribute should also have a `title` attribute that names the element for the purpose of non-visual interactions.

*This box is non-normative. Implementation requirements are given below this box.*

`element.draggable [= value]`

Returns true if the element is draggable; otherwise, returns false.

Can be set, to override the default and set the `draggable` content attribute.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Otherwise, if the element's `draggable` content attribute has the state `false`, the `draggable` IDL attribute must return `false`.

Otherwise, the element's `draggable` content attribute has the state `auto`. If the element is an `img` element, or, if the element is an `a` element with an `href` content attribute, the `draggable` IDL attribute must return `true`.

Otherwise, the `draggable` DOM must return `false`.

If the `draggable` IDL attribute is set to the value `false`, the `draggable` content attribute must be set to the literal value `false`. If the `draggable` IDL attribute is set to the value `true`, the `draggable` content attribute must be set to the literal value `true`.

## Too slow? Try reading the multipage copy of

**8.7.8 The spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

All HTML elements separated by spaces must be accepted as a single ordered set of unique space-separated tokens.

**copy** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Indicates that dropping an accepted item on the element will result in the dragged data being copied to the new location.

**move**

Indicates that dropping an accepted item on the element will result in the dragged data being moved to the new location.

**link**

Indicates that dropping an accepted item on the element will result in a link to the original data.

**Any keyword with eight characters or more, beginning with the an ASCII case-insensitive match for the string "string":**

Indicates that items with the drag data item kind *Plain Unicode string* and the drag data item type string set to a value that matches the remainder of the keyword are accepted.

**Any keyword with six characters or more, beginning with an ASCII case-insensitive match for the string "file":**

Indicates that items with the drag data item kind *File* and the drag data item type string set to a value that matches the remainder of the keyword are accepted.

The `dropzone` content attribute's values must not have more than one of the three feedback values (`copy`, `move`, and `link`) specified. If none are specified, the `copy` value is implied.

An element with a `dropzone` attribute should also have a `title` attribute that names the element for the purpose of non-visual interactions.

A `dropzone` attribute **matches a drag data store** if the `dropzone` processing steps result in a match.

A `dropzone` attribute **specifies an operation** if the `dropzone` processing steps result in a specified operation. The specified operation is as given by those steps.

The **dropzone processing steps** are as follows. They either result in a match or not, and separate from this result either in a specified operation or not, as defined below.

1. Let `value` be the value of the `dropzone` attribute.
2. Let `keywords` be the result of splitting `value` on spaces.
3. Let `matched` be false.
4. Let `operation` be unspecified.
5. For each value in `keywords`, if any, in the order that they were found in `value`, run the following steps.
  1. Let `keyword` be the keyword.
  2. If `keyword` is one of "copy", "move", or "link", then: run the following substeps:
    1. If `operation` is still unspecified, then let `operation` be the string given by `keyword`.
    2. Skip to the step labeled `end of keyword` below.
  3. If `keyword` does not contain a U+003A COLON character (:), or if the first such character in `keyword` is either the first

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

5. Jump to the appropriate step from the list below, based on the value of *kind code*:↳ If *kind code* is the string "string"Let *kind* be Plain Unicode string.↳ If *kind code* is the string "file"Let *kind* be File.

↳ Otherwise



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↓ character (:) in the string,

d given in *kind* and whose

1, if there are no more.

6. The *args* ...

The algorithm results in a specified operation if *operation* is not unspecified. The specified operation, if one is specified, is the one given by *operation*.

The **dropzone** IDL attribute must reflect the content attribute of the same name.

In this example, a `div` element is made into a drop target for image files using the `dropzone` attribute. Images dropped into the target are then displayed.

```
<div dropzone="copy file:image/png file:image/gif file:image/jpeg" ondrop="receive(event, this)">
 <p>Drop an image here to have it displayed.</p>
</div>
<script>
 function receive(event, element) {
 var data = event.dataTransfer.items;
 for (var i = 0; i < data.length; i += 1) {
 if ((data[i].kind == 'file') && (data[i].type.match('^image/'))) {
 var img = new Image();
 img.src = window.createObjectURL(data[i].getAsFile());
 element.appendChild(img);
 }
 }
 }
</script>
```

### 8.7.9 Security risks in the drag-and-drop model

User agents must not make the data added to the `DataTransfer` object during the `dragstart` event available to scripts until the `drop` event, because otherwise, if a user were to drag sensitive information from one document to a second document, crossing a hostile third document in the process, the hostile document could intercept the data.

For the same reason, user agents must consider a drop to be successful only if the user specifically ended the drag operation — if any scripts end the drag operation, it must be considered unsuccessful (canceled) and the `drop` event must not be fired.

User agents should take care to not start drag-and-drop operations in response to script actions. For example, in a mouse-and-window environment, if a script moves a window while the user has his mouse button depressed, the UA would not consider that to start a drag. This is important because otherwise UAs could cause data to be dragged from sensitive sources and dropped into hostile documents without the user's consent.

User agents should filter potentially active (scripted) content (e.g. HTML) when it is dragged and when it is dropped, using a whitelist of known-safe features. Similarly, relative URLs should be turned into absolute URLs to avoid references changing in unexpected ways. This specification does not specify how this is performed.

|| Consider a hostile page providing some content and getting the user to select and drag and drop (or indeed, copy and paste)

[ ]

[Go](#) **NOV** **JAN** **FEB**  
◀ **03** ▶  
**2012** **2013** **2014**

[About this capture](#)

**3,445 captures**  
5 Jun 2004 - 22 Aug 2020



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

### 9.1.1 Scope

*This section is non-normative.*

This specification defines an API for running scripts in the background independently of any user interface scripts.

This allows for long-running scripts that are not interrupted by scripts that respond to clicks or other user interactions, and allows long tasks to be executed without yielding to keep the page responsive.

Workers (as these background scripts are called herein) are relatively heavy-weight, and are not intended to be used in large numbers. For example, it would be inappropriate to launch one worker for each pixel of a four megapixel image. The examples below show some appropriate

**Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>**

stance memory cost.

### 9.1.2 Examples

*This section is non-normative.*

There are a variety of examples illustrating various uses of this use.

#### 9.1.2.1 A background number-crunching worker

*This section is non-normative.*

The simplest use of workers is for performing a computationally expensive task without interrupting the user interface.

In this example, the main document spawns a worker to (naively) compute prime numbers, and progressively displays the most recently found prime number.

The main page is as follows:

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>Worker example: One-core computation</title>
 </head>
 <body>
 <p>The highest prime number discovered so far is: <output id="result"></output></p>
 <script>
 var worker = new Worker('worker.js');
 worker.onmessage = function (event) {
 document.getElementById('result').textContent = event.data;
 };
 </script>
 </body>
</html>
```

The `Worker()` constructor call creates a worker and returns a `Worker` object representing that worker, which is used to communicate with the worker. That object's `onmessage` event handler allows the code to receive messages from the worker.

The worker itself is as follows:

```
var n = 1;
search: while (true) {
 n += 1;
 for (var i = 2; i <= Math.sqrt(n); i += 1)
 if (n % i == 0)
 continue search;
 // found a prime!
 postMessage(n);
}
```

The bulk of this code is simply an unoptimized search for a prime number. The `postMessage()` method is used to send a message back to the page when a prime is found.

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*This section is non-normative.*

In this example, the main document uses two workers, one for fetching stock updates at regular intervals, and one for performing search queries that the user requests.

The main page is as follows:

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>
 <script>
 Too slow? Try reading the multipage copy of
 the spec instead: https://web.archive.org
 /web/20130103030411/http:
 //whatwg.org/html
 searcher.postMessage(query);
 </script>
 </head>
 <body onload="search('')">
 <p><output id="symbol"></output> <output id="value"></output></p>
 <script>
 ticker.onmessage = function (event) {
 var data = event.data.split(' ');
 document.getElementById('symbol').textContent = data[0];
 document.getElementById('value').textContent = data[1];
 };
 ticker.postMessage(symbol);
 </script>
 <p><label>Search: <input type="text" autofocus oninput="search(this.value)"></label></p>
 <ul id="results">
 <script>
 searcher.onmessage = function (event) {
 var data = event.data.split(' ');
 var results = document.getElementById('results');
 while (results.hasChildNodes()) // clear previous results
 results.removeChild(results.firstChild);
 for (var i = 0; i < data.length; i += 1) {
 // add a list item with a button for each result
 var li = document.createElement('li');
 var button = document.createElement('button');
 button.value = data[i];
 button.type = 'button';
 button.onclick = function () { select(this.value); };
 button.textContent = data[i];
 li.appendChild(button);
 results.appendChild(li);
 }
 };
 </script>
 <p>(The data in this example is not real. Try searching for "Google" or "Apple".)</p>
 </body>
 </html>
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

 xhr.open('GET', url, false);
 xhr.send();
 return xhr.responseText;
 } catch (e) {
 return ''; // turn all errors into empty results
 }
}

```

The stock updater worker is as follows:

```

import var ti Too slow? Try reading the multipage copy of
var sy the spec instead: https://web.archive.org
functi /web/20130103030411/http:
post time:
}
onmess /whatwg.org/html
if (t.
 clearTimeout(timer);
symbol = event.data;
update();
};

```



The search query worker is as follows:

```

importScripts('io.js');
onmessage = function (event) {
 postMessage(get('search.cgi?' + event.data));
};

```

[View this example online.](#)

### 9.1.2.3 Shared workers introduction

*This section is non-normative.*

This section introduces shared workers using a Hello World example. Shared workers use slightly different APIs, since each worker can have multiple connections.

This first example shows how you connect to a worker and how a worker can send a message back to the page when it connects to it. Received messages are displayed in a log.

Here is the HTML page:

```

<!DOCTYPE HTML>
<title>Shared workers: demo 1</title>
<pre id="log">Log:</pre>
<script>
 var worker = new SharedWorker('test.js');
 var log = document.getElementById('log');
 worker.port.onmessage = function(e) { // note: not worker.onmessage!
 log.textContent += '\n' + e.data;
 }
</script>

```

Here is the JavaScript worker:

```

onconnect = function(e) {
 var port = e.ports[0];
 port.postMessage('Hello World!');
}

```

[View this example online.](#)

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
<!DOCTYPE HTML>
<title>Shared workers: demo 2</title>
<pre id="log">Log:</pre>
<script>
 var worker = new SharedWorker('test.js');
 var log = document.getElementById('log');
 worker.port.addEventListener('message', function(e) {
 log.textContent += '\n' + e.data;
 }, false);
 worker.port.start();
</script>
```

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/shared-workers.html>**

Here is the outer HTML page:

```
onconnect = function(e) {
 var log = document.getElementById('log');
 log.textContent += '\n' + e.data;
 log.scrollTop = log.scrollHeight;
}
port.onmessage = function(e) {
 port.postMessage('pong'); // not e.ports[0].postMessage!
 // e.target.postMessage('pong'); would work also
}
```

[View this example online.](#)

Finally, the example is extended to show how two pages can connect to the same worker; in this case, the second page is merely in an iframe on the first page, but the same principle would apply to an entirely separate page in a separate top-level browsing context.

Here is the outer HTML page:

```
<!DOCTYPE HTML>
<title>Shared workers: demo 3</title>
<pre id="log">Log:</pre>
<script>
 var worker = new SharedWorker('test.js');
 var log = document.getElementById('log');
 worker.port.addEventListener('message', function(e) {
 log.textContent += '\n' + e.data;
 }, false);
 worker.port.start();
 worker.port.postMessage('ping');
</script>
<iframe src="inner.html"></iframe>
```

Here is the inner HTML page:

```
<!DOCTYPE HTML>
<title>Shared workers: demo 3 inner frame</title>
<pre id=log>Inner log:</pre>
<script>
 var worker = new SharedWorker('test.js');
 var log = document.getElementById('log');
 worker.port.onmessage = function(e) {
 log.textContent += '\n' + e.data;
 }
</script>
```

Here is the JavaScript worker:

```
var count = 0;
onconnect = function(e) {
 count += 1;
 var port = e.ports[0];
 port.postMessage(count);
}
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[View this example online.](#)

#### 9.1.2.4 Shared state using a shared worker

*This section is non-normative.*

In this example, multiple windows (viewers) can be opened that are all viewing the same map. All the windows share the same map information, with a single worker coordinating all the viewers. Each viewer can move around independently, but if they set any data on the map, all the

Too slow? Try reading the multipage copy of

The main pa

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
<!DOCTYPE html>
<html>
<head>
<title>Shared State Using a Shared Worker</title>
<script>
function openViewer() {
 window.open('viewer.html');
}
</script>
</head>
<body>
<p><button type="button" onclick="openViewer()">Open a new viewer</button></p>
<p>Each viewer opens in a new window. You can have as many viewers as you like, they all view the same data.</p>
</body>
</html>
```

The viewer is more involved:

```
<!DOCTYPE HTML>
<html>
<head>
<title>Workers example: Multiviewer viewer</title>
<script>
var worker = new SharedWorker('worker.js', 'core');

// CONFIGURATION
function configure(event) {
 if (event.data.substr(0, 4) != 'cfg ') return;
 var name = event.data.substr(4).split(' ', 1);
 // update display to mention our name is name
 document.getElementsByTagName('h1')[0].textContent += ' ' + name;
 // no longer need this listener
 worker.port.removeEventListener('message', configure, false);
}
worker.port.addEventListener('message', configure, false);

// MAP
function paintMap(event) {
 if (event.data.substr(0, 4) != 'map ') return;
 var data = event.data.substr(4).split(',');
 // display tiles data[0] .. data[8]
 var canvas = document.getElementById('map');
 var context = canvas.getContext('2d');
 for (var y = 0; y < 3; y += 1) {
 for (var x = 0; x < 3; x += 1) {
 var tile = data[y * 3 + x];
 if (tile == '0')
 context.fillStyle = 'green';
 else
```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
worker.port.addEventListener('message', paintMap, false);

// PUBLIC CHAT
function updatePublicChat(event) {
 if (event.data.substr(0, 4) != 'txt ') return;
 var name = event.data.substr(4).split(' ', 1);
 var message = event.data.substr(4 + length(name) + 1);
 // display "<name> message" in public chat
 var dialog = document.getElementById('public');
 v
 c Too slow? Try reading the multipage copy of
 c the spec instead: https://web.archive.org
 c /web/20130103030411/http:
}
wor //whatwg.org/html

// PRIVATE CHAT
function startPrivateChat(event) {
 if (event.data.substr(0, 4) != 'msg ') return;
 var name = event.data.substr(4).split(' ', 1);
 var port = event.ports[0];
 // display a private chat UI
 var ul = document.getElementById('private');
 var li = document.createElement('li');
 var h3 = document.createElement('h3');
 h3.textContent = 'Private chat with ' + name;
 li.appendChild(h3);
 var dialog = document.createElement('dialog');
 var addMessage = function(name, message) {
 var dt = document.createElement('dt');
 dt.textContent = name;
 dialog.appendChild(dt);
 var dd = document.createElement('dd');
 dd.textContent = message;
 dialog.appendChild(dd);
 };
 port.onmessage = function (event) {
 addMessage(name, event.data);
 };
 li.appendChild(dialog);
 var form = document.createElement('form');
 var p = document.createElement('p');
 var input = document.createElement('input');
 input.size = 50;
 p.appendChild(input);
 p.appendChild(document.createTextNode(' '));
 var button = document.createElement('button');
 button.textContent = 'Post';
 p.appendChild(button);
 form.onsubmit = function () {
 port.postMessage(input.value);
 addMessage('me', input.value);
 input.value = '';
 return false;
 };
 form.appendChild(p);
 li.appendChild(form);
}
worker.port.addEventListener('message', startPrivateChat, false);

worker.port.start();
```

Go

**NOV JAN FEB**



2012 2013 2014



3,445 captures

5 Jun 2004 - 22 Aug 2020

```
 value = ''; return
```

There are several key things worth noting about the way the viewer is written

**Multiple listeners.** Instead of a single message processing function, the code here attaches multiple event listeners, each one performing a quick check to see if it is relevant for the message. In this example it doesn't make much difference, but if multiple authors wanted to collaborate using a single port to communicate with a worker, it would allow for independent code instead of changes having to all be made to a single event handling function.

Registering event listeners in this way also allows you to unregister specific listeners when you are done with them, as is done with the `configure()` method in this example.

Finally, the worker:

```
var nextName = 0;
function getNextName() {
 // this could use more friendly names
 // but for now just return a number
 return nextName++;
}

var map = [
 [0, 0, 0, 0, 0, 0, 0],
 [1, 1, 0, 1, 0, 1, 1],
 [0, 1, 0, 1, 0, 0, 0],
 [0, 1, 0, 1, 0, 1, 1],
 [0, 0, 0, 1, 0, 0, 0],
 [1, 0, 0, 1, 1, 1, 1],
 [1, 1, 0, 1, 1, 0, 1],
];

function wrapX(x) {
 if (x < 0) return wrapX(x + map[0].length);
 if (x >= map[0].length) return wrapX(x - map[0].length);
 return x;
}

function wrapY(y) {
 if (y < 0) return wrapY(y + map.length);
 if (y >= map[0].length) return wrapY(y - map.length);
 return y;
}
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
 data += ',';
 data += map[y][x];
 }
}
callback('map ' + data);
}

var viewers = {};
onconnect = function (event) {
even
ever
view
ever
ever
senc
};

//whatwg.org/html
function
switch (event.data.substr(0, 4)) {
case 'mov':
 var direction = event.data.substr(4);
 var dx = 0;
 var dy = 0;
 switch (direction) {
 case 'up': dy = -1; break;
 case 'down': dy = 1; break;
 case 'left': dx = -1; break;
 case 'right': dx = 1; break;
 }
 event.target._data.x = wrapX(event.target._data.x + dx);
 event.target._data.y = wrapY(event.target._data.y + dy);
 sendMapData(event.target.postMessage);
 break;
case 'set':
 var value = event.data.substr(4);
 map[event.target._data.y][event.target._data.x] = value;
 for (var viewer in viewers)
 sendMapData(viewers[viewer].port.postMessage);
 break;
case 'txt':
 var name = event.target._name;
 var message = event.data.substr(4);
 for (var viewer in viewers)
 viewers[viewer].port.postMessage('txt ' + name + ' ' + message);
 break;
case 'msg':
 var party1 = event._data;
 var party2 = viewers[event.data.substr(4).split(' ', 1)];
 if (party2) {
 var channel = new MessageChannel();
 party1.port.postMessage('msg ' + party2.name, [channel.port1]);
 party2.port.postMessage('msg ' + party1.name, [channel.port2]);
 }
 break;
}
}
```



**Connecting to multiple pages.** The script uses the `onconnect` event listener to listen for multiple connections.

**Direct channels.** When the worker receives a "msg" message from one viewer naming another viewer, it sets up a direct connection between the two, so that the two viewers can communicate directly without the worker having to proxy all the messages.

[View this example online.](#)

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

multiple workers. In this example, a computationally expensive task that is to be performed for every number from 1 to 10,000,000 is farmed out to ten subworkers.

The main page is as follows, it just reports the result:

```
<!DOCTYPE HTML>
<html>
 <head>
 <title>Worker example: Multicore computation</title>
 </head>
 <body>
 <p>F
 <scr
 var wor
 c /web/20130103030411/http:
 };
 </sc //whatwg.org/html
 </body>
</html>
```



The worker itself is as follows:

```
// settings
var num_workers = 10;
var items_per_worker = 1000000;

// start the workers
var result = 0;
var pending_workers = num_workers;
for (var i = 0; i < num_workers; i += 1) {
 var worker = new Worker('core.js');
 worker.postMessage(i * items_per_worker);
 worker.postMessage((i+1) * items_per_worker);
 worker.onmessage = storeResult;
}

// handle the results
function storeResult(event) {
 result += 1*event.data;
 pending_workers -= 1;
 if (pending_workers <= 0)
 postMessage(result); // finished!
}
```

It consists of a loop to start the subworkers, and then a handler that waits for all the subworkers to respond.

The subworkers are implemented as follows:

```
var start;
onmessage = getStart;
function getStart(event) {
 start = 1*event.data;
 onmessage = getEnd;
}

var end;
function getEnd(event) {
 end = 1*event.data;
 onmessage = null;
 work();
}

function work() {
 var result = 0;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
 close();
}
```

They receive two numbers in two events, perform the computation for the range of numbers thus specified, and then report the result back to the parent.

[View this example online.](#)



### 9.1.3 Tutorial

#### 9.1.3.1 Creating a worker

This section is non-normative. [Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Creating a worker is as simple as creating a new `Worker` object; a URL is passed to the constructor:

```
var wc = new Worker('/whatwg.org/html');
```

to that file as its only

#### 9.1.3.2 Communicating with a dedicated worker

This section is non-normative.

Dedicated workers use `MessagePort` objects behind the scenes, and thus support all the same features, such as sending structured data, transferring binary data, and transferring other ports.

To receive messages from a dedicated worker, use the `onmessage` event handler IDL attribute on the `Worker` object:

```
worker.onmessage = function (event) { ... };
```

You can also use the `addEventListener()` method.

**Note:** The implicit `MessagePort` used by dedicated workers has its port message queue implicitly enabled when it is created, so there is no equivalent to the `MessagePort` interface's `start()` method on the `Worker` interface.

To send data to a worker, use the `postMessage()` method. Structured data can be sent over this communication channel. To send `ArrayBuffer` objects efficiently (by transferring them rather than cloning them), list them in an array in the second argument.

```
worker.postMessage({
 operation: 'find-edges',
 input: buffer, // an ArrayBuffer object
 threshold: 0.6,
}, [buffer]);
```

To receive a message inside the worker, the `onmessage` event handler IDL attribute is used.

```
onmessage = function (event) { ... };
```

You can again also use the `addEventListener()` method.

In either case, the data is provided in the event object's `data` attribute.

To send messages back, you again use `postMessage()`. It supports the structured data in the same manner.

```
postMessage(event.data.input, [event.data.input]); // transfer the buffer back
```

#### 9.1.3.3 Shared workers

This section is non-normative.

Shared workers are identified in one of two ways: either by the URL of the script used to create it, or by explicit name. When created by name, the URL used by the first page to create the worker with that name is the URL of the script that will be used for that worker. This allows multiple applications on a domain to all use a single shared worker to provide a common service, without the applications having to keep track of a common URL for the script used to provide the service.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
var worker = new SharedWorker('service.js');
```

Communicating with shared workers is done with explicit `MessagePort` objects. The object returned by the `SharedWorker()` constructor holds a reference to the port on its `port` attribute.

```
worker.port.onmessage = function (event) { ... };
worker.port.postMessage('some message');
worker.port.postMessage({ foo: 'structured', bar: ['data', 'also', 'possible']});
```

Inside the sh<sup>r</sup>  
event object

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
/whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

the new client is given by the

};

e

## 9.2 Infrastructure

There are two kinds of workers; dedicated workers, and shared workers. Dedicated workers, once created, are linked to their creator; but message ports can be used to communicate from a dedicated worker to multiple other browsing contexts or workers. Shared workers, on the other hand, are named, and once created any script running in the same origin can obtain a reference to that worker and communicate with it.

### 9.2.1 The global scope

The global scope is the "inside" of a worker.

#### 9.2.1.1 The `WorkerGlobalScope` common interface

```
interface WorkerGlobalScope : EventTarget {
 readonly attribute WorkerGlobalScope self;
 readonly attribute WorkerLocation location;

 void close();
 attribute EventHandler onerror;
 attribute EventHandler onoffline;
 attribute EventHandler ononline;
 };
 WorkerGlobalScope implements WorkerUtils;
```

The `self` attribute must return the `WorkerGlobalScope` object itself.

The `location` attribute must return the `WorkerLocation` object created for the `WorkerGlobalScope` object when the worker was created. It represents the absolute URL of the script that was used to initialize the worker, after any redirects.

When a script invokes the `close()` method on a `WorkerGlobalScope` object, the user agent must run the following steps (atomically):

1. Discard any tasks that have been added to the event loop's task queues.
2. Set the worker's `WorkerGlobalScope` object's closing flag to true. (This prevents any further tasks from being queued.)

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by objects implementing the `WorkerGlobalScope` interface:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Each `WorkerGlobalScope` object has a **worker origin** that is set when the object is created.

**Note:** For `data : URLs`, this is the origin of the entry script that called the constructor. For other URLs, this is the origin of the value of the absolute URL given in the worker's attribute.

#### 9.2.1.2 Dedicated workers and the `DedicatedWorkerGlobalScope` interface

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The DedicatedWorkerGlobalScope objects act as if they had an implicit MessagePort associated with them. This port is part of a channel that is set up when the worker is created, but it is not exposed. This object must never be garbage collected before the DedicatedWorkerGlobalScope object.

All messages received by that port must immediately be retargeted at the DedicatedWorkerGlobalScope object.

The `postMessage()` method on DedicatedWorkerGlobalScope objects must act as if, when invoked, it immediately invoked the `postMessage()` method of the same name on the port, with the same arguments, and returned the same return value.

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by objects implementing the DedicatedWorkerGlobalScope interface:

Event handler	Event handler event type
<code>onmessage</code>	<code>message</code>

For the purposes of the application cache networking model, a dedicated worker is an extension of the cache host from which it was created.

#### 9.2.1.3 Shared workers and the `SharedWorkerGlobalScope` interface

```
interface SharedWorkerGlobalScope : WorkerGlobalScope {
 readonly attribute DOMString name;
 readonly attribute ApplicationCache applicationCache;
 attribute EventHandler onconnect;
};
```

The SharedWorkerGlobalScope interface must only be exposed if the JavaScript global environment is a shared worker environment.

Shared workers receive message ports through `connect` events on their global object for each connection.

The `name` attribute must return the value it was assigned when the SharedWorkerGlobalScope object was created by the "run a worker" algorithm. Its value represents the name that can be used to obtain a reference to the worker using the `SharedWorker` constructor.

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by objects implementing the SharedWorkerGlobalScope interface:

Event handler	Event handler event type
<code>onconnect</code>	<code>connect</code>

For the purposes of the application cache networking model, a shared worker is its own cache host. The run a worker algorithm takes care of associating the worker with an application cache.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Each `WorkerGlobalScope` object has an event loop distinct from those defined for units of related similar-origin browsing contexts. This event loop has no associated browsing context, and its task queues only have events, callbacks, and networking activity as tasks. The processing model of these event loops is defined below in the run a worker algorithm.

Each `WorkerGlobalScope` object also has a **closing** flag, which must initially be false, but which can get set to true by the algorithms in the processing model section below.

Once the `WorkerGlobalScope`'s closing flag is set to true, the event loop's task queues must discard any further tasks that would be added to them, and any timers that were scheduled to fire in the future must be canceled. Once the closing flag is true, timers stop f

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Workers cor /[whatwg.org/html](http://whatwg.org/html)

MessagePort objects.

Each Worker are entangled.

MessagePort objects that list includes the implicit

MessagePort in the case of dedicated workers.

Each `WorkerGlobalScope` also has a list of **the worker's workers**. Initially this list is empty; it is populated when the worker creates or obtains further workers.

Finally, each `WorkerGlobalScope` also has a list of **the worker's Documents**. Initially this list is empty; it is populated when the worker is created.

Whenever a `Document d` is **added to the worker's Documents**, the user agent must, for each worker `q` in the list of the worker's workers whose list of the worker's `Documents` does not contain `d`, add `d` to `q`'s `WorkerGlobalScope` owner's list of the worker's `Documents`.

Whenever a `Document` object is discarded, it must be removed from the list of the worker's `Documents` of each worker whose list contains that `Document`.

Given a script's global object `o` when creating or obtaining a worker, the **list of relevant Document objects to add** depends on the type of `o`. If `o` is a `WorkerGlobalScope` object (i.e. if we are creating a nested worker), then the relevant `Documents` are the `Documents` that are in `o`'s own list of the worker's `Documents`. Otherwise, `o` is a `Window` object, and the **relevant Document** is just the `Document` that is the active document of the `Window` object `o`.

A worker is said to be a **permissible worker** if its list of the worker's `Documents` is not empty.

A worker is said to be a **protected worker** if it is a permissible worker and either it has outstanding timers, database transactions, or network connections, or its list of the worker's ports is not empty, or its `WorkerGlobalScope` is actually a `SharedWorkerGlobalScope` object (i.e. the worker is a shared worker).

A worker is said to be an **active needed worker** if any of the `Document` objects in the worker's `Documents` are fully active.

A worker is said to be a **suspendable worker** if it is not an active needed worker but it is a permissible worker.

## 9.2.4 Processing model

When a user agent is to **run a worker** for a script with URL `url`, a browsing context `owner browsing context`, a `Document owner document`, an origin `owner origin`, and with global scope `worker global scope`, it must run the following steps:

1. Create a separate parallel execution environment (i.e. a separate thread or process or equivalent construct), and run the rest of these steps asynchronously in that context.
2. If `worker global scope` is actually a `SharedWorkerGlobalScope` object (i.e. the worker is a shared worker), and there are any relevant application caches that are identified by a manifest URL with the same origin as `url` and that have `url` as one of their entries, *not* excluding entries marked as foreign, then associate the `worker global scope` with the most appropriate application cache of those that match.
3. Attempt to fetch the resource identified by `url`, from the `owner origin`, using `owner document` as the referrer source, with the `synchronous flag` set and the `force same-origin flag` set.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Let language be JavaScript.

**Note:** As with `script` elements, the MIME type of the script is ignored. Unlike with `script` elements, there is no way to override the type. It's always assumed to be JavaScript.

4. In the newly created execution environment, create a JavaScript global environment whose `global` object is `worker global scope`. If `worker global scope` is a `DedicatedWorkerGlobalScope` object, then this is a dedicated worker environment. Otherwise, `worker global scope` is a `SharedWorkerGlobalScope` object, and this is a shared worker environment. (In either case, by definition it is a worker environment.)

## 5. A note: Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

Part of the specification is available online at <https://web.archive.org/web/20130103030411/http://whatwg.org/html>.

Set the script's browsing context to *owner browsing context*.

Set the script's document to *owner document*.

Set the script's referrer source to *url*.

Set the script's URL character encoding to UTF-8. (This is just used for encoding non-ASCII characters in the query component of URLs.)

Set the script's base URL to *url*.

6. **Closing orphan workers:** Start monitoring the worker such that no sooner than it stops being either a protected worker or a suspendable worker, and no later than it stops being a permissible worker, `worker global scope`'s closing flag is set to true.

7. **Suspending workers:** Start monitoring the worker, such that whenever `worker global scope`'s closing flag is false and the worker is a suspendable worker, the user agent suspends execution of script in that worker until such time as either the closing flag switches to true or the worker stops being a suspendable worker.

8. Jump to the script's *initial code entry-point*, and let that run until it either returns, fails to catch an exception, or gets prematurely aborted by the "kill a worker" or "terminate a worker" algorithms defined below.

9. If `worker global scope` is actually a `DedicatedWorkerGlobalScope` object (i.e. the worker is a dedicated worker), then enable the port message queue of the worker's implicit port.

10. **Event loop:** Wait until either there is a task in one of the event loop's task queues or `worker global scope`'s closing flag is set to true.

11. Run the oldest task on one of the event loop's task queues, if any. The user agent may pick any task queue.

**Note:** The handling of events or the execution of callbacks might get prematurely aborted by the "kill a worker" or "terminate a worker" algorithms defined below.

12. If the storage mutex is now owned by the worker's event loop, release it so that it is once again free.

13. Remove the task just run in the earlier step, if any, from its task queue.

14. If there are any more events in the event loop's task queues or if `worker global scope`'s closing flag is set to false, then jump back to the step above labeled *event loop*.

15. Empty the `worker global scope`'s list of active timers.

16. Disentangle all the ports in the list of the worker's ports.

When a user agent is to **kill a worker** it must run the following steps in parallel with the worker's main loop (the "run a worker" processing model defined above):

1. Set the worker's `WorkerGlobalScope` object's closing flag to true.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

User agents may invoke the "kill a worker" processing model on a worker at any time, e.g. in response to user requests, in response to CPU quota management, or when a worker stops being an active needed worker if the worker continues executing even after its closing flag was set to true.

When a user agent is to **terminate a worker** it must run the following steps in parallel with the worker's main loop (the "run a worker" processing model defined above):

1. Set the worker's `WorkerGlobalScope` object's closing flag to true.
2. If the task source is a **script** (i.e. the worker is a **scripted** worker):
 

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**
3. **Abc**
4. If the task source is a **script** (i.e. the worker is a **scripted** worker):
 

The task source is a **script** (i.e. the worker is a **scripted** worker).

### 9.2.5 Runtime script errors

Whenever an uncaught runtime script error occurs in one of the worker's scripts, if the error did not occur while handling a previous script error, the user agent must report the error for that script, with the position (line number and column number) where the error occurred, using the `WorkerGlobalScope` object's `onerror` attribute.

For shared workers, if the error is still *not handled* afterwards, or if the error occurred while handling a previous script error, the error may be reported to the user.

For dedicated workers, if the error is still *not handled* afterwards, or if the error occurred while handling a previous script error, the user agent must queue a task to fire a trusted event that uses the `ErrorEvent` interface, with the name `error`, that doesn't bubble and is cancelable, with its `message`, `filename`, `lineno`, and `column` attributes initialized appropriately, at the `Worker` object associated with the worker. If the event is not canceled, the user agent must act as if the uncaught runtime script error had occurred in the global scope that the `Worker` object is in, thus repeating the entire runtime script error reporting process one level up.

If the implicit port connecting the worker to its `Worker` object has been disentangled (i.e. if the parent worker has been terminated), then the user agent must act as if the `Worker` object had no `error` event handler and as if that worker's `onerror` attribute was null, but must otherwise act as described above.

**Note:** Thus, error reports propagate up to the chain of dedicated workers up to the original document, even if some of the workers along this chain have been terminated and garbage collected.

The task source for the task mentioned above is the DOM manipulation task source.

```
[Constructor(DOMString type, optional ErrorEventInit eventInitDict)]
interface ErrorEvent : Event {
 readonly attribute DOMString message;
 readonly attribute DOMString filename;
 readonly attribute unsigned long lineno;
 readonly attribute unsigned long column;
};

dictionary ErrorEventInit : EventInit {
 DOMString message;
 DOMString filename;
 unsigned long lineno;
 unsigned long column;
};
```

The `message` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to the empty string. It represents the error message.

The `filename` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `column` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to zero. It represents the column number where the error occurred in the script.

## 9.2.6 Creating workers

### 9.2.6.1 The `AbstractWorker` abstract interface

```
[NoInterfaceObject]
interface
```

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

The following objects implement the `EventTarget` interface:  
 Event handle  
 onerror error

ported, as IDL attributes, by

### 9.2.6.2 Dedicated workers and the `Worker` interface

```
[Constructor(DOMString scriptURL)]
interface Worker : EventTarget {
 void terminate();

 void postMessage(any message, optional sequence<Transferable> transfer);
 attribute EventHandler onmessage;
};

Worker implements AbstractWorker;
```

The `terminate()` method, when invoked, must cause the "terminate a worker" algorithm to be run on the worker with the object is associated.

Worker objects act as if they had an implicit `MessagePort` associated with them. This port is part of a channel that is set up when the worker is created, but it is not exposed. This object must never be garbage collected before the `Worker` object.

All messages received by that port must immediately be retargeted at the `Worker` object.

The `postMessage()` method on `Worker` objects must act as if, when invoked, it immediately invoked the method of the same name on the port, with the same arguments, and returned the same return value.

The `postMessage()` method's first argument can be structured data:

```
worker.postMessage({opcode: 'activate', device: 1938, parameters: [23, 102]});
```

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by objects implementing the `Worker` interface:

Event handler	Event handler event type
onmessage	message

When the `Worker(scriptURL)` constructor is invoked, the user agent must run the following steps:

1. The user agent may throw a `SecurityError` exception and abort these steps if the request violates a policy decision (e.g. if the user agent is configured to not allow the page to start dedicated workers).
2. Resolve the `scriptURL` argument relative to the entry script's base URL, when the method is invoked.
3. If this fails, throw a `SyntaxError` exception.
4. If the `<scheme>` component of the resulting absolute URL is not "data", and the origin of the resulting absolute URL is not the same as the origin of the entry script, then throw a `SecurityError` exception and abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

5. Create a new `DedicatedWorkerGlobalScope` object whose worker origin is the origin of the entry script. Let `worker global scope` be this new object.
6. Create a new `Worker` object, associated with `worker global scope`. Let `worker` be this new object.
7. Create a new `MessagePort` object owned by the global object of the script that invoked the constructor. Let this be the `outside port`.
8. Associate the `outside port` with `worker`.
9. Create a new `Document` object with the same `origin` as the global object of the script that invoked the constructor. Let this be the `worker document`.
10. Associate the `worker document` with `worker`.
11. Enter the `worker` object's execution context.
12. Return the absolute URL of the script that invoked the constructor.
13. Enable the `Document` objects in `docs` to add given the global object of the script that invoked the constructor.
14. Let `docs` be the list of relevant `Document` objects to add given the global object of the script that invoked the constructor.
15. Add to `worker global scope`'s list of the worker's `Documents` the `Document` objects in `docs`.
16. If the global object of the script that invoked the constructor is a `WorkerGlobalScope` object (i.e. we are creating a nested worker), add `worker global scope` to the list of the worker's workers of the `WorkerGlobalScope` object that is the global object of the script that invoked the constructor.
17. Run a worker for the resulting absolute URL, with the script's browsing context of the script that invoked the method as the `owner browsing context`, with the script's document of the script that invoked the method as the `owner document`, with the origin of the entry script as the `owner origin`, and with `worker global scope` as the global scope.

This constructor must be visible when the script's global object is either a `Window` object or an object implementing the `WorkerUtils` interface.

#### 9.2.6.3 Shared workers and the `SharedWorker` interface

```
[Constructor(DOMString scriptURL, optional DOMString name)]
interface SharedWorker : EventTarget {
 readonly attribute MessagePort port;
}
SharedWorker implements AbstractWorker;
```

The `port` attribute must return the value it was assigned by the object's constructor. It represents the `MessagePort` for communicating with the shared worker.

When the `SharedWorker(scriptURL, name)` constructor is invoked, the user agent must run the following steps:

1. The user agent may throw a `SecurityError` exception and abort these steps if the request violates a policy decision (e.g. if the user agent is configured to not allow the page to start shared workers).
2. Resolve the `scriptURL` argument.
3. If this fails, throw a `SyntaxError` exception.
4. Otherwise, let `scriptURL` be the resulting absolute URL.
5. Let `name` be the value of the second argument, or the empty string if the second argument was omitted.
6. If the `<scheme>` component of `scriptURL` is not "data", and the origin of `scriptURL` is not the same as the origin of the entry script, then throw a `SecurityError` exception and abort these steps.

**Note:** Thus, scripts must either be external files with the same scheme, host, and port as the original page, or data: URLs. For example, you can't load a script from a javascript: URL, and an https: page couldn't start workers using scripts with http: URLs.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Let this SharedWorker object be *worker*.

2. Create a new MessagePort object owned by the global object of the script that invoked the method. Let this be the *outside port*.
3. Assign *outside port* to the *port* attribute of *worker*.
4. Let *worker global scope* be null.
5. If *name* is not the empty string and there exists a SharedWorkerGlobalScope object whose closing flag is false, in as *scriptURL*, then let

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

scope to null.

object whose closing flag is  
is an absolute URL that is  
scope object.  
munication between the  
, then set *worker global*

**Note:** For example, a user agent could have a development mode that isolates a particular top-level browsing context from all other pages, and scripts in that development mode could be blocked from connecting to shared workers running in the normal browser mode.

7. If *worker global scope* is not null, then run these steps:

1. If *worker global scope's* location attribute represents an absolute URL that is not exactly equal to *scriptURL*, then throw a `URLMismatchError` exception and abort all these steps.
2. Associate *worker* with *worker global scope*.
3. Create a new MessagePort object owned by *worker global scope*. Let this be the *inside port*.
4. Entangle *outside port* and *inside port*.
5. Return *worker* and perform the next step asynchronously.
6. Create a trusted event that uses the `MessageEvent` interface, with the name `connect`, which does not bubble, is not cancelable, has no default action, has a `data` attribute whose value is initialized to the empty string, has a `ports` attribute whose value is initialized to a read only array containing only the newly created port, and has a `source` attribute whose value is initialized to the newly created port, and queue a task to dispatch the event at *worker global scope*.
7. Add to *worker global scope's* list of the worker's `Documents` the `Document` objects in *docs*.
8. If the global object of the script that invoked the constructor is a `WorkerGlobalScope` object, add *worker global scope* to the list of the worker's workers of the `WorkerGlobalScope` object that is the global object of the script that invoked the constructor.
9. Abort all these steps.
8. Create a new `SharedWorkerGlobalScope` object whose worker origin is the origin of the entry script. Let *worker global scope* be this new object.
9. Associate *worker* with *worker global scope*.
10. Set the `name` attribute of *worker global scope* to *name*.
11. Create a new MessagePort object owned by *worker global scope*. Let *inside port* be this new object.
12. Entangle *outside port* and *inside port*.
9. Return *worker* and perform the remaining steps asynchronously.
10. Create a trusted event that uses the `MessageEvent` interface, with the name `connect`, which does not bubble, is not cancelable, has no default action, has a `data` attribute whose value is initialized to the empty string, has a `ports` attribute

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

12. If the global object of the script that invoked the constructor is a `WorkerGlobalScope` object, add `worker global scope` to the list of the worker's workers of the `WorkerGlobalScope` object that is the global object of the script that invoked the constructor.

13. Run a worker for `scriptURL`, with the script's browsing context of the script that invoked the method as the `owner browsing context`, with the script's document of the script that invoked the method as the `owner document`, with the origin of the entry script as the `owner origin`, and with `worker global scope` as the global scope.

This constructor must be visible when the script's global object is either a `Window` object or an object implementing the `WorkerUtils` interface.

The task [sol](#) **Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

### 9.3 APIs /[web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html):

```
[NoI
inte.
void importScripts(DOMString... urls);
readonly attribute WorkerNavigator navigator;
};
WorkerUtils implements WindowTimers;
WorkerUtils implements WindowBase64;
```

The DOM APIs — specifically, all the interfaces defined in the DOM specification other than exceptions — must not be exposed if the JavaScript global environment is a worker environment. [DOMCORE]

#### 9.3.1 Importing scripts and libraries

When a script invokes the `importScripts(urls)` method on a `WorkerGlobalScope` object, the user agent must run the following steps:

1. If there are no arguments, return without doing anything. Abort these steps.
2. Resolve each argument.
3. If any fail, throw a `SyntaxError` exception.
4. Attempt to fetch each resource identified by the resulting absolute URLs, from the entry script's origin, using the entry script's referrer source, and with the `synchronous` flag set.
5. For each argument in turn, in the order given, starting with the first one, run these substeps:

1. Wait for the fetching attempt for the corresponding resource to complete.

If the fetching attempt failed, throw a `NetworkError` exception and abort all these steps.

If the attempt succeeds, then let `source` be the script resource decoded as UTF-8, with error handling.

Let `language` be JavaScript.

**Note: As with the worker's script, the script here is always assumed to be JavaScript, regardless of the MIME type.**

2. Create a script, using `source` as the script source, the URL from which `source` was obtained, and `language` as the scripting language, using the same global object, browsing context, document, referrer source, URL character encoding, and base URL as the script that was created by the worker's run a worker algorithm.

If the script came from a resource whose URL does not have the same origin as the worker origin, then pass the `muted errors` flag to the create a script from a node algorithm.

Let the newly created script run until it either returns, fails to parse, fails to catch an exception, or gets prematurely aborted by the "kill a worker" or "terminate a worker" algorithms defined above.

If it failed to parse, then throw an ECMAScript `SyntaxError` exception and abort all these steps. [ECMA262]

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

### 9.3.2 The WorkerNavigator object

The `navigator` attribute of the `WorkerUtils` interface must return an instance of the `WorkerNavigator` interface, which represents the identity and state of the user agent (the client).

```
interface WorkerNavigator {};
WorkerNavigator implements NavigatorID;
WorkerNavigator implements NavigatorLanguage;
WorkerNavigator
```

**Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>**

### 9.3.3 Interface /web/20130103030411/http:

Nothing must be exposed to the following:

- `XMLHttpRequest` and all interface objects and constructors defined by the `XMLHttpRequest` specifications, except that the document response entity body must always be null. The `XMLHttpRequest` base URL is the script's base URL; the `XMLHttpRequest` origin is the script's origin, and the `XMLHttpRequest` referrer source is the script's referrer source. [XHR]
- The interface objects and constructors defined by this specification, except where is further restricted by explicit requirements in this specification.
- The interface objects of any objects made accessible through interfaces implemented by any objects that are themselves exposed (i.e. this requirement is transitive).

**Note: These requirements do not override the requirements defined by the Web IDL specification, in particular concerning the visibility of interfaces annotated with the `[NoInterfaceObject]` extended attribute.**

### 9.3.4 Worker locations

```
interface WorkerLocation {
 // URL decomposition IDL attributes
 stringifier readonly attribute DOMString href;
 readonly attribute DOMString protocol;
 readonly attribute DOMString host;
 readonly attribute DOMString hostname;
 readonly attribute DOMString port;
 readonly attribute DOMString pathname;
 readonly attribute DOMString search;
 readonly attribute DOMString hash;
}
```

A `WorkerLocation` object represents an absolute URL set at its creation.

The `href` attribute must return the absolute URL that the object represents.

The `WorkerLocation` interface also has the complement of URL decomposition IDL attributes, `protocol`, `host`, `port`, `hostname`, `pathname`, `search`, and `hash`. These must follow the rules given for URL decomposition IDL attributes, with the input being the absolute URL that the object represents (same as the `href` attribute), and the common setter action being a no-op, since the attributes are defined to be readonly.

The `WorkerLocation` interface must only be exposed if the JavaScript global environment is a worker environment.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Messages in server-sent events, Web sockets, cross-document messaging, and channel messaging use the `MessageEvent` interface.

The following interface is defined for this event:

```
[Constructor(DOMString type, optional MessageEventInit eventInitDict)]
interface MessageEvent : Event {
 readonly attribute any data;
 readonly attribute DOMString origin;
 readonly attribute DOMString lastEventId;
 readonly attribute WindowProxy source;
 ...
};

Too slow? Try reading the multipage copy of
the spec instead: https://web.archive.org
an /web/20130103030411/http:
DO //whatwg.org/html
Wi Mes.
};
```

*This box is non-normative. Implementation requirements are given below this box.*

**`event.data`**

Returns the data of the message.

**`event.origin`**

Returns the origin of the message, for server-sent events and cross-document messaging.

**`event.lastEventId`**

Returns the last event ID string, for server-sent events.

**`event.source`**

Returns the `WindowProxy` of the source window, for cross-document messaging, and the `MessagePort` being attached, in the `connect` event fired at `SharedWorkerGlobalScope` objects.

**`event.ports`**

Returns the `MessagePort` array sent with the message, for cross-document messaging and channel messaging.

The `data` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the message being sent.

The `origin` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to the empty string. It represents, in server-sent events and cross-document messaging, the origin of the document that sent the message (typically the scheme, hostname, and port of the document, but not its path or fragment identifier).

The `lastEventId` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to the empty string. It represents, in server-sent events, the last event ID string of the event source.

The `source` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents, in cross-document messaging, the `WindowProxy` of the browsing context of the `Window` object from which the message came; and in the `connect` events used by shared workers, the newly connecting `MessagePort`.

The `ports` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents, in cross-document messaging and channel messaging the `MessagePort` array being sent, if any.

## 10.2 Server-sent events

### 10.2.1 Introduction

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Using this API consists of creating an `EventSource` object and registering an event listener.

```
var source = new EventSource('updates.cgi');
source.onmessage = function (event) {
 alert(event.data);
};
```

On the server-side, the script ("`updates.cgi`" in this case) sends messages in the following form, with the `text/event-stream` MIME type:

```
data: '
 Too slow? Try reading the multipage copy of
data: the spec instead: https://web.archive.org
data: /web/20130103030411/http:
data: //whatwg.org/html
```

Authors can

d" and "remove":

```
event: add
data: 73857293
```

```
event: remove
data: 2153
```

```
event: add
data: 113411
```

The script to handle such a stream would look like this (where `addHandler` and `removeHandler` are functions that take one argument, the event):

```
var source = new EventSource('updates.cgi');
source.addEventListener('add', addHandler, false);
source.addEventListener('remove', removeHandler, false);
```

The default event type is "message".

Event stream requests can be redirected using HTTP 301 and 307 redirects as with normal HTTP requests. Clients will reconnect if the connection is closed; a client can be told to stop reconnecting using the HTTP 204 No Content response code.

Using this API rather than emulating it using XMLHttpRequest or an `iframe` allows the user agent to make better use of network resources in cases where the user agent implementor and the network operator are able to coordinate in advance. Amongst other benefits, this can result in significant savings in battery life on portable devices. This is discussed further in the section below on connectionless push.

### 10.2.2 The `EventSource` interface

```
[Constructor(DOMString url, optional EventSourceInit eventSourceInitDict)]
interface EventSource : EventTarget {
 readonly attribute DOMString url;
 readonly attribute boolean withCredentials;

 // ready state
 const unsigned short CONNECTING = 0;
 const unsigned short OPEN = 1;
 const unsigned short CLOSED = 2;
 readonly attribute unsigned short readyState;

 // networking
 attribute EventHandler onopen;
 attribute EventHandler onmessage;
 attribute EventHandler onerror;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

};

The `EventSource()` constructor takes one or two arguments. The first specifies the URL to which to connect. The second specifies the settings, if any, in the form of an `EventSourceInit` dictionary. When the `EventSource()` constructor is invoked, the UA must run these steps:

1. Resolve the URL specified in the first argument, relative to the entry script's base URL.
2. If the previous step failed, then throw a `SyntaxError` exception.

3. Create **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**  
 4. Let `url`, then set `CORS mode` to  
 5. If the `useCORS` flag is true, then set `CORS mode` to  
 6. Return `new EventSource(url)`.

7. Do a potentially CORS-enabled fetch of the resulting absolute URL using the entry script's referrer source, with the `mode` being `CORS mode`, and the `origin` being the entry script's origin, and process the resource obtained in this fashion, if any, as described below.

**Note:** The definition of the fetching algorithm (which is used by CORS) is such that if the browser is already fetching the resource identified by the given absolute URL, that connection can be reused, instead of a new connection being established. All messages received up to this point are dispatched immediately, in this case.

This constructor must be visible when the script's global object is either a `Window` object or an object implementing the `WorkerUtils` interface.

The `url` attribute must return the absolute URL that resulted from resolving the value that was passed to the constructor.

The `withCredentials` attribute must return the value to which it was last initialized. When the object is created, it must be initialized to `false`.

The `readyState` attribute represents the state of the connection. It can have the following values:

#### **CONNECTING (numeric value 0)**

The connection has not yet been established, or it was closed and the user agent is reconnecting.

#### **OPEN (numeric value 1)**

The user agent has an open connection and is dispatching events as it receives them.

#### **CLOSED (numeric value 2)**

The connection is not open, and the user agent is not trying to reconnect. Either there was a fatal error or the `close()` method was invoked.

When the object is created its `readyState` must be set to `CONNECTING` (0). The rules given below for handling the connection define when the value changes.

The `close()` method must abort any instances of the fetch algorithm started for this `EventSource` object, and must set the `readyState` attribute to `CLOSED`.

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by all objects implementing the `EventSource` interface:

Event handler	Event handler event type
<code>onopen</code>	<code>open</code>
<code>onmessage</code>	<code>message</code>
<code>onerror</code>	<code>error</code>

In addition to the above, each `EventSource` object has the following associated with it:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

These values are not currently exposed on the interface.

### 10.2.3 Processing model

The resource indicated in the argument to the `EventSource` constructor is fetched when the constructor is run.

For HTTP connections, the `Accept` header may be included; if included, it must contain only formats of event framing that are supported by the user agent (one of which must be `text/event-stream`, as described below)

If the event s  
request, whc

**Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

User agents  
(This header  
agents shou

'st be included with the  
equests of event sources.  
(request mechanism.) User

As data is re

HTTP 200 OK responses with a Content-Type header specifying the type `text/event-stream`, ignoring any MIME type parameters, must be processed line by line as described below.

When a successful response with a supported MIME type is received, such that the user agent begins parsing the contents of the stream, the user agent must announce the connection.

The task that the networking task source places on the task queue once the fetching algorithm for such a resource (with the correct MIME type) has completed must cause the user agent to asynchronously reestablish the connection. This applies whether the connection is closed gracefully or unexpectedly. It doesn't apply for the error conditions listed below except where explicitly specified.

HTTP 200 OK responses that have a Content-Type specifying an unsupported type, or that have no Content-Type at all, must cause the user agent to fail the connection.

HTTP 305 Use Proxy, 401 Unauthorized, and 407 Proxy Authentication Required should be treated transparently as for any other subresource.

HTTP 301 Moved Permanently, 302 Found, 303 See Other, and 307 Temporary Redirect responses are handled by the fetching and CORS algorithms. In the case of 301 redirects, the user agent must also remember the new URL so that subsequent requests for this resource for this `EventSource` object start with the URL given for the last 301 seen for requests for this object.

Network errors that prevent the connection from being established in the first place (e.g. DNS errors), must cause the user agent to asynchronously reestablish the connection.

Any other HTTP response code not listed here must cause the user agent to fail the connection.

For non-HTTP protocols, UAs should act in equivalent ways.

When a user agent is to **announce the connection**, the user agent must queue a task which, if the `readyState` attribute is set to a value other than `CLOSED`, sets the `readyState` attribute to `OPEN` and fires a simple event named `open` at the `EventSource` object.

When a user agent is to **reestablish the connection**, the user agent must run the following steps. These steps are run asynchronously, not as part of a task. (The tasks that it queues, of course, are run like normal tasks and not asynchronously.)

1. Queue a task to run the following steps:
  1. If the `readyState` attribute is set to `CLOSED`, abort the task.
  2. Set the `readyState` attribute to `CONNECTING`.
  3. Fire a simple event named `error` at the `EventSource` object.
2. Wait a delay equal to the reconnection time of the event source.
3. Optionally, wait some more. In particular, if the previous attempt failed, then user agents might introduce an exponential backoff delay to avoid overloading a potentially already overloaded server. Alternatively, if the operating system has reported that there is no network connectivity, user agents might wait for the operating system to announce that the network connection has returned before retrying.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2. Perform a potentially CORS-enabled fetch of the absolute URL of the event source resource, using the same *referrer source*, and with the same *mode* and *origin*, as those used in the original request triggered by the `EventSource()` constructor, and process the resource obtained in this fashion, if any, as described earlier in this section.

When a user agent is to **fail the connection**, the user agent must queue a task which, if the `readyState` attribute is set to a value other than `CLOSED`, sets the `readyState` attribute to `CLOSED` and fires a simple event named `error` at the `EventSource` object. **Once the user agent has failed the connection, it does not attempt to reconnect!**

The task sou'

**Too slow? Try reading the multipage copy of**

#### 10.2.4 Pa the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

This event s /web/20130103030411/http:

The event st //whatwg.org/html [ABNF]

;t for which is Unicode.

```

stream = [bom] *event
event = *(comment / field) end-of-line
comment = colon *any-char end-of-line
field = 1*name-char [colon [space] *any-char] end-of-line
end-of-line = (cr lf / cr / lf)

; characters
lf = %x000A ; U+000A LINE FEED (LF)
cr = %x000D ; U+000D CARRIAGE RETURN (CR)
space = %x0020 ; U+0020 SPACE
colon = %x003A ; U+003A COLON (:)
bom = %xFEFF ; U+FEFF BYTE ORDER MARK
name-char = %x0000-0009 / %x000B-000C / %x000E-0039 / %x003B-10FFFF
 ; a Unicode character other than U+000A LINE FEED (LF), U+000D CARRIAGE RETURN
 ; (CR), or U+003A COLON (:)
any-char = %x0000-0009 / %x000B-000C / %x000E-10FFFF
 ; a Unicode character other than U+000A LINE FEED (LF) or U+000D CARRIAGE RETURN
 ; (CR)

```

Event streams in this format must always be encoded as UTF-8. [RFC3629]

Lines must be separated by either a U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pair, a single U+000A LINE FEED (LF) character, or a single U+000D CARRIAGE RETURN (CR) character.

Since connections established to remote servers for such resources are expected to be long-lived, UAs should ensure that appropriate buffering is used. In particular, while line buffering with lines are defined to end with a single U+000A LINE FEED (LF) character is safe, block buffering or line buffering with different expected line endings can cause delays in event dispatch.

#### 10.2.5 Interpreting an event stream

Streams must be decoded as UTF-8, with error handling.

One leading U+FEFF BYTE ORDER MARK character must be ignored if any are present.

The stream must then be parsed by reading everything line by line, with a U+000D CARRIAGE RETURN U+000A LINE FEED (CRLF) character pair, a single U+000A LINE FEED (LF) character not preceded by a U+000D CARRIAGE RETURN (CR) character, and a single U+000D CARRIAGE RETURN (CR) character not followed by a U+000A LINE FEED (LF) character being the ways in which a line can end.

When a stream is parsed, a *data* buffer, an *event type* buffer, and a *last event ID* buffer must be associated with it. They must be initialized to the empty string.

Lines must be processed, in the order they are received, as follows:

##### ↳ If the line is empty (a blank line)

Dispatch the event, as defined below.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Collect the characters on the line before the first U+003A COLON character (:), and let *value* be that string. If *value* starts with a U+0020 SPACE character, remove it from *value*.

Process the field using the steps described below, using *field* as the field name and *value* as the field value.

↪ Otherwise, the string is not empty but does not contain a U+003A COLON character (:)

Process the field using the steps described below, using the whole line as the field name, and the empty string as the field value.

Once the empty line, t

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

n event, before the final

The steps to names must

the following list. Field

↪ If the fi //whatwg.org/html

↪ If the field name is "data"

Append the field value to the *data* buffer, then append a single U+000A LINE FEED (LF) character to the *data* buffer.

↪ If the field name is "id"

Set the *last event ID* buffer to the field value.

↪ If the field name is "retry"

If the field value consists of only ASCII digits, then interpret the field value as an integer in base ten, and set the event stream's reconnection time to that integer. Otherwise, ignore the field.

↪ Otherwise

The field is ignored.

When the user agent is required to **dispatch the event**, then the user agent must act as follows:

1. Set the last event ID string of the event source to value of the *last event ID* buffer. The buffer does not get reset, so the last event ID string of the event source remains set to this value until the next time it is set by the server.
2. If the *data* buffer is an empty string, set the *data* buffer and the *event type* buffer to the empty string and abort these steps.
3. If the *data* buffer's last character is a U+000A LINE FEED (LF) character, then remove the last character from the *data* buffer.
4. Create an event that uses the `MessageEvent` interface, with the event type `message`, which does not bubble, is not cancelable, and has no default action. The `data` attribute must be initialized to the value of the *data* buffer, the `origin` attribute must be initialized to the Unicode serialization of the origin of the event stream's final URL (i.e. the URL after redirects), and the `lastEventId` attribute must be initialized to the last event ID string of the event source. This event is not trusted.
5. If the *event type* buffer has a value other than the empty string, change the type of the newly created event to equal the value of the *event type* buffer.
6. Set the *data* buffer and the *event type* buffer to the empty string.
7. Queue a task which, if the `readyState` attribute is set to a value other than `CLOSED`, dispatches the newly created event at the `EventSource` object.

**Note: If an event doesn't have an "id" field, but an earlier event did set the event source's last event ID string, then the event's `lastEventId` field will be set to the value of whatever the last seen "id" field was.**

The following event stream, once followed by a blank line:

```
data: YHOO
data: +2
data: 10
```

...would cause an event message with the interface `MessageEvent` to be dispatched on the `EventSource` object. The event's

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

```
stocks.onmessage = function (event) {
 var data = event.data.split('\n');
 updateStocks(data[0], data[1], data[2]);
};

...where updateStocks() is a function defined as:

function updateStocks(symbol, delta, value) { ... }
```

...or some such.



The following stream fires two events:

```
last event
head value
recoil character
last event
```

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The second block has two "event", and will then set the be sent a Last-Event-ID 'id' field, this time with no now be sent in the event of a 'with a single leading space' trigger the dispatch of the

```
: test stream

data: first event
id: 1

data:second event
id

data: third event
```

The following stream fires two events:

```
data

data
data

data:
```

The first block fires events with the data set to the empty string, as would the last block if it was followed by a blank line. The middle block fires an event with the data set to a single newline character. The last block is discarded because it is not followed by a blank line.

The following stream fires two identical events:

```
data:test

data: test
```

This is because the space after the colon is ignored if present.

### 10.2.6 Notes

Legacy proxy servers are known to, in certain cases, drop HTTP connections after a short timeout. To protect against such proxy servers, authors can include a comment line (one starting with a ':' character) every 15 seconds or so.

Authors wishing to relate event source connections to each other or to specific documents previously served might find that relying on IP addresses doesn't work, as individual clients can have multiple IP addresses (due to having multiple proxy servers) and individual IP addresses can have multiple clients (due to sharing a proxy server). It is better to include a unique identifier in the document when it is served and then pass that identifier as part of the URL when the connection is established.

Authors are also cautioned that HTTP chunking can have unexpected negative effects on the reliability of this protocol. Where possible, chunking should be disabled for serving event streams unless the rate of messages is high enough for this not to matter.

Clients that support HTTP's per-server connection limitation might run into trouble when opening multiple pages from a site if each page has an `EventSource` to the same domain. Authors can avoid this using the relatively complex mechanism of using unique

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

### 10.2.7 Connectionless push and other features

User agents running in controlled environments, e.g. browsers on mobile handsets tied to specific carriers, may offload the management of the connection to a proxy on the network. In such a situation, the user agent for the purposes of conformance is considered to include both the handset software and the network proxy.

For example, a browser on a mobile device, after having established a connection, might detect that it is on a supporting network and request that a proxy server on the network take over the management of the connection. The timeline for such a situation might be as follows:



Author in the EventSource

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

involved in keeping the TCP

5. ... , ..., proxy", maintain the connection instead.
6. The "push proxy" service contacts the remote HTTP server and requests the resource specified by the author in the EventSource constructor (possibly including a `Last-Event-ID` HTTP header, etc).
7. The browser allows the mobile device to go to sleep.
8. The server sends another message.
9. The "push proxy" service uses a technology such as OMA push to convey the event to the mobile device, which wakes only enough to process the event and then returns to sleep.

This can reduce the total data usage, and can therefore result in considerable power savings.

As well as implementing the existing API and `text/event-stream` wire format as defined by this specification and in more distributed ways as described above, formats of event framing defined by other applicable specifications may be supported. This specification does not define how they are to be parsed or processed.

### 10.2.8 Garbage collection

While an `EventSource` object's `readyState` is `CONNECTING`, and the object has one or more event listeners registered for `open`, `message` or `error` events, there must be a strong reference from the `Window` or `WorkerUtils` object that the `EventSource` object's constructor was invoked from to the `EventSource` object itself.

While an `EventSource` object's `readyState` is `OPEN`, and the object has one or more event listeners registered for `message` or `error` events, there must be a strong reference from the `Window` or `WorkerUtils` object that the `EventSource` object's constructor was invoked from to the `EventSource` object itself.

While there is a task queued by an `EventSource` object on the remote event task source, there must be a strong reference from the `Window` or `WorkerUtils` object that the `EventSource` object's constructor was invoked from to that `EventSource` object.

If a user agent is to **forcibly close** an `EventSource` object (this happens when a `Document` object goes away permanently), the user agent must abort any instances of the fetch algorithm started for this `EventSource` object, and must set the `readyState` attribute to `CLOSED`.

If an `EventSource` object is garbage collected while its connection is still open, the user agent must abort any instance of the fetch algorithm opened by this `EventSource`.

**Note:** It's possible for one active network connection to be shared by multiple `EventSource` objects and their fetch algorithms, which is why the above is phrased in terms of aborting the fetch algorithm and not the actual underlying download.

### 10.2.9 IANA considerations

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

text

**Subtype name:**

event-stream

**Required parameters:**

No parameters

**Optional parameters:**

charset



parameter serves no

**Too slow? Try reading the multipage copy of****the spec instead: <https://web.archive.org>****/web/20130103030411/http:****An leak** //whatwg.org/html

can result in information

Event streams can overwhelm a user agent; a user agent is expected to apply suitable restrictions to avoid depleting local resources because of an overabundance of information from an event stream.

Servers can be overwhelmed if a situation develops in which the server is causing clients to reconnect rapidly. Servers should use a 5xx status code to indicate capacity problems, as this will prevent conforming clients from reconnecting automatically.

**Interoperability considerations:**

Rules for processing both conforming and non-conforming content are defined in this specification.

**Published specification:**

This document is the relevant specification.

**Applications that use this media type:**

Web browsers and tools using Web services.

**Additional information:****Magic number(s):**

No sequence of bytes can uniquely identify an event stream.

**File extension(s):**

No specific file extensions are recommended for this type.

**Macintosh file type code(s):**

No specific Macintosh file type codes are recommended for this type.

**Person & email address to contact for further information:**

Ian Hickson <ian@hixie.ch>

**Intended usage:**

Common

**Restrictions on usage:**

This format is only expected to be used by dynamic open-ended streams served using HTTP or a similar protocol. Finite resources are not expected to be labeled with this type.

**Author:**

Ian Hickson <ian@hixie.ch>

**Change controller:**

W3C

Fragment identifiers have no meaning with text/event-stream resources.

**10.2.9.2 Last-Event-ID**

This section describes a header field for registration in the Permanent Message Header Field Registry. [RFC3864]

**Header field name:**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Status:

standard

**Author/Change controller:**

W3C

**Specification document(s):**

This document is the relevant specification.

**Related information:**

Nonr



Too slow? Try reading the multipage copy of  
**10.3 We** the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

To enable Web applications to maintain bidirectional communications with server-side processes, this specification introduces the `WebSocket` interface.

**Note:** This interface does not allow for raw access to the underlying network. For example, this interface could not be used to implement an IRC client without proxying messages through a custom server.

### 10.3.2 The `WebSocket` interface

```
enum BinaryType { "blob", "arraybuffer" };
[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
 readonly attribute DOMString url;

 // ready state
 const unsigned short CONNECTING = 0;
 const unsigned short OPEN = 1;
 const unsigned short CLOSING = 2;
 const unsigned short CLOSED = 3;
 readonly attribute unsigned short readyState;
 readonly attribute unsigned long bufferedAmount;

 // networking
 attribute EventHandler onopen;
 attribute EventHandler onerror;
 attribute EventHandler onclose;
 readonly attribute DOMString extensions;
 readonly attribute DOMString protocol;
 void close([Clamp] optional unsigned short code, optional DOMString reason);

 // messaging
 attribute EventHandler onmessage;
 attribute BinaryType binaryType;
 void send(DOMString data);
 void send(Blob data);
 void send(ArrayBuffer data);
 void send(ArrayBufferView data);
};
```

The `WebSocket(url, protocols)` constructor takes one or two arguments. The first argument, `url`, specifies the URL to which to connect. The second, `protocols`, if present, is either a string or an array of strings. If it is a string, it is equivalent to an array consisting of just that string; if it is omitted, it is equivalent to the empty array. Each string in the array is a subprotocol name. The connection will only be established if the server reports that it has selected one of these subprotocols. The subprotocol names must all be strings that match the requirements for elements that comprise the value of `Sec-WebSocket-Protocol` header fields as defined by the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

throw a `SyntaxError` exception and abort these steps. [WSP]

2. If `secure` is false but the origin of the entry script has a scheme component that is itself a secure protocol, e.g. HTTPS, then throw a `SecurityError` exception.
3. If `port` is a port to which the user agent is configured to block access, then throw a `SecurityError` exception. (User agents typically block access to well-known ports like SMTP.)

Access to ports 80 and 443 should not be blocked, including the unlikely cases when `secure` is false but `port` is 443 or `secure` is true but `port` is 80

4. If `p1` **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**
5. If `ar` the `Synt` string for elements that comprise specification, then throw a
6. Let `c`.
7. Return a new `WebSocket` object, and continue these steps in the background (without blocking scripts).
8. Establish a `WebSocket` connection given the set (`host`, `port`, `resource name`, `secure`), along with the `protocols` list, an empty list for the extensions, and `origin`. The `headers` to send appropriate cookies must be a `Cookie` header whose value is the `cookie-string` computed from the user's cookie store and the URL `url`; for these purposes this is *not* a "non-HTTP" API. [WSP] [COOKIES]

When the user agent validates the server's response during the "establish a `WebSocket` connection" algorithm, if the status code received from the server is not 101 (e.g. it is a redirect), the user agent must fail the `WebSocket` connection.

**⚠️Warning! Following HTTP procedures here could introduce serious security problems in a Web browser context. For example, consider a host with a `WebSocket` server at one path and an open HTTP redirector at another. Suddenly, any script that can be given a particular `WebSocket` URL can be tricked into communicating to (and potentially sharing secrets with) any host on the Internet, even if the script checks that the URL has the right hostname.**

**Note: If the establish a `WebSocket` connection algorithm fails, it triggers the fail the `WebSocket` connection algorithm, which then invokes the close the `WebSocket` connection algorithm, which then establishes that the `WebSocket` connection is closed, which fires the close event as described below.**

This constructor must be visible when the script's global object is either a `Window` object or an object implementing the `WorkerUtils` interface.

The `url` attribute must return the result of resolving the URL that was passed to the constructor. (It doesn't matter what it is resolved relative to, since we already know it is an absolute URL.)

The `readyState` attribute represents the state of the connection. It can have the following values:

#### **CONNECTING (numeric value 0)**

The connection has not yet been established.

#### **OPEN (numeric value 1)**

The `WebSocket` connection is established and communication is possible.

#### **CLOSING (numeric value 2)**

The connection is going through the closing handshake, or the `close()` method has been invoked.

#### **CLOSED (numeric value 3)**

The connection has been closed or could not be opened.

When the object is created its `readyState` must be set to CONNECTING (0).

The `extensions` attribute must initially return the empty string. After the `WebSocket` connection is established, its value might change, as defined below.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

as defined below.

**Note:** The `protocol` attribute returns the subprotocol selected by the server, if any. It can be used in conjunction with the array form of the constructor's second argument to perform subprotocol negotiation.

The `close()` method must run the following steps:

1. If the method's first argument is present but is not an integer equal to 1000 or in the range 3000 to 4999, throw an `InvalidAccessError` exception and abort these steps.



2. If the **Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

actors.

, steps. [RFC3629]

3. Run the first matching steps from the following list:

↳ If the `readyState` attribute is in the CLOSING (2) or CLOSED (3) state

Do nothing.

**Note:** The connection is already closing or is already closed. If it has not already, a `close` event will eventually fire as described below.

↳ If the WebSocket connection is not yet established [WSP]

*Fail the WebSocket connection and set the `readyState` attribute's value to CLOSING (2). [WSP]*

**Note:** The fail the WebSocket connection algorithm invokes the close the WebSocket connection algorithm, which then establishes that the WebSocket connection is closed, which fires the `close` event as described below.

↳ If the WebSocket closing handshake has not yet been started [WSP]

*Start the WebSocket closing handshake and set the `readyState` attribute's value to CLOSING (2). [WSP]*

If the first argument is present, then the status code to use in the WebSocket Close message must be the integer given by the first argument. [WSP]

If the second argument is also present, then `reason` must be provided in the Close message after the status code. [RFC3629] [WSP]

**Note:** The start the WebSocket closing handshake algorithm eventually invokes the close the WebSocket connection algorithm, which then establishes that the WebSocket connection is closed, which fires the `close` event as described below.

↳ Otherwise

Set the `readyState` attribute's value to CLOSING (2).

**Note:** The WebSocket closing handshake is started, and will eventually invoke the close the WebSocket connection algorithm, which will establish that the WebSocket connection is closed, and thus the `close` event will fire, as described below.

The `bufferedAmount` attribute must return the number of bytes of application data (UTF-8 text and binary data) that have been queued using `send()` but that, as of the last time the event loop started executing a task, had not yet been transmitted to the network. (This thus includes any text sent during the execution of the current task, regardless of whether the user agent is able to transmit text asynchronously with script execution.) This does not include framing overhead incurred by the protocol, or buffering done by the operating system or network hardware. If the connection is closed, this attribute's value will only increase with each call to the `send()` method (the number does not reset to zero once the connection closes).

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
setInterval(function() {
 if (socket.bufferedAmount == 0)
 socket.send(getUpdateData());
}, 50);
};
```

The `bufferedAmount` attribute can also be used to saturate the network without sending the data at a higher rate than the network can handle, though this requires more careful monitoring of the value of the attribute over time.



When a Web  
value it was

## Too slow? Try reading the multipage copy of the spec instead: [https://web.archive.org /web/20130103030411/http: spool/i//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

*Note: To "blob" User agents must send a WebSocket Message comprised of `data` using a text frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must close the WebSocket connection with prejudice. Any invocation of this method with a string argument that does not throw an exception must increase the `bufferedAmount` attribute by the number of bytes needed to express the argument as UTF-8. [UNICODE] [RFC3629] [WSP]*

getting, it must return the last

*the attribute to arrayBuffer form. It to "blob", it is safe to memory. Naturally, user in memory or not, e.g.*

The `send(data)` method transmits data using the connection. If the `readyState` attribute is `CONNECTING`, it must throw an `InvalidStateError` exception. Otherwise, the user agent must run the appropriate set of steps from the following list:

### If the argument is a string

If `data` be the result of converting the `data` argument to a sequence of Unicode characters. If the `WebSocket connection is established` and the `WebSocket closing handshake has not yet started`, then the user agent must `send a WebSocket Message` comprised of `data` using a text frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must `close the WebSocket connection` with prejudice. Any invocation of this method with a string argument that does not throw an exception must increase the `bufferedAmount` attribute by the number of bytes needed to express the argument as UTF-8. [UNICODE] [RFC3629] [WSP]

### If the argument is a Blob object

If the `WebSocket connection is established`, and the `WebSocket closing handshake has not yet started`, then the user agent must `send a WebSocket Message` comprised of `data` using a binary frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must `close the WebSocket connection` with prejudice. The data to be sent is the raw data represented by the `Blob` object. Any invocation of this method with a `Blob` argument that does not throw an exception must increase the `bufferedAmount` attribute by the size of the `Blob` object's raw data, in bytes. [WSP] [FILEAPI]

### If the argument is an ArrayBuffer object

If the `WebSocket connection is established`, and the `WebSocket closing handshake has not yet started`, then the user agent must `send a WebSocket Message` comprised of `data` using a binary frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must `close the WebSocket connection` with prejudice. The data to be sent is the data stored in the buffer described by the `ArrayBuffer` object. Any invocation of this method with an `ArrayBuffer` argument that does not throw an exception must increase the `bufferedAmount` attribute by the length of the `ArrayBuffer` in bytes. [WSP] [TYPEDARRAY]

### If the argument is an ArrayBufferView object

If the `WebSocket connection is established`, and the `WebSocket closing handshake has not yet started`, then the user agent must `send a WebSocket Message` comprised of `data` using a binary frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must `close the WebSocket connection` with prejudice. The data to be sent is the data stored in the section of the buffer described by the `ArrayBuffer` object that the `ArrayBufferView` object references. Any invocation of this method with an `ArrayBufferView` argument that does not throw an exception must increase the `bufferedAmount` attribute by the length of the `ArrayBufferView` in bytes. [WSP] [TYPEDARRAY]

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by all objects implementing the `WebSocket` interface:

Event handler	Event handler event type
---------------	--------------------------

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

onclose close

### 10.3.3 Feedback from the protocol

When the WebSocket connection is established, the user agent must queue a task to run these steps:

1. Change the readyState attribute's value to OPEN (1).

2. Change the

3. Change the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html
4. Act for t
5. Fire

When a WebS

[WSP]

1. If the readyState attribute's value is not OPEN (1), then abort these steps.

2. Let event be a newly created trusted event that uses the MessageEvent interface, with the event type message, which does not bubble, is not cancelable, and has no default action.

3. Initialize event's origin attribute to the Unicode serialization of the origin of the URL that was passed to the WebSocket object's constructor.

4. If type indicates that the data is Text, then initialize event's data attribute to data.

If type indicates that the data is Binary, and binaryType is set to "blob", then initialize event's data attribute to a new Blob object that represents data as its raw data. [FILEAPI]

If type indicates that the data is Binary, and binaryType is set to "arraybuffer", then initialize event's data attribute to a new read-only ArrayBuffer object whose contents are data. [TYPEDARRAY]

5. Dispatch event at the WebSocket object.

**Note:** User agents are encouraged to check if they can perform the above steps efficiently before they run the task, picking tasks from other task queues while they prepare the buffers if not. For example, if the binaryType attribute was set to "blob" when the data arrived, and the user agent spooled all the data to disk, but just before running the above task for this particular message the script switched binaryType to "arraybuffer", the user agent would want to page the data back to RAM before running this task so as to avoid stalling the main thread while it created the ArrayBuffer object.

Here is an example of how to define a handler for the message event in the case of text frames:

```
mysocket.onmessage = function (event) {
 if (event.data == 'on') {
 turnLampOn();
 } else if (event.data == 'off') {
 turnLampOff();
 }
};
```

The protocol here is a trivial one, with the server just sending "on" or "off" messages.

When the WebSocket closing handshake is started, the user agent must queue a task to change the readyState attribute's value to CLOSING (2). (If the close() method was called, the readyState attribute's value will already be set to CLOSING (2) when this task runs.) [WSP]

When the WebSocket connection is closed, possibly cleanly, the user agent must queue a task to run the following substeps:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

3. Create a trusted event that uses the `CloseEvent` interface, with the event type `close`, which does not bubble, is not cancelable, has no default action, whose `wasClean` attribute is initialized to true if the connection closed *cleanly* and false otherwise, whose `code` attribute is initialized to the *WebSocket connection close code*, and whose `reason` attribute is initialized to the *WebSocket connection close reason* decoded as UTF-8, with error handling, and dispatch the event at the `WebSocket` object. [WSP]

**User agents must not convey any failure information to scripts in a way that would allow a script to distinguish the following situations:**

- A server that closed the connection because the user agent specified a subprotocol that the server did not offer.
- A slow server that caused the client to drop the connection (e.g. the server specified a subprotocol that the client did not offer).
- A server that closed the connection after successfully completing the opening handshake.
- A WebSocket server that abruptly closed the connection instead of sending a close frame with code 1006.
- A user agent that received a close frame with code 1006 from a server that did not support the subprotocol specified by the user agent.

In all of these cases, the the *WebSocket connection close code* would be 1006, as required by the *WebSocket Protocol specification*. [WSP]

Allowing a script to distinguish these cases would allow a script to probe the user's local network in preparation for an attack.

**Note:** In particular, this means the code 1015 is not used by the user agent (unless the server erroneously uses it in its close frame, of course).

The task source for all tasks queued in this section is the **WebSocket task source**.

#### 10.3.4 Ping and Pong frames

The *WebSocket protocol specification* defines Ping and Pong frames that can be used for keep-alive, heart-beats, network status probing, latency instrumentation, and so forth. These are not currently exposed in the API.

User agents may send ping and unsolicited pong frames as desired, for example in an attempt to maintain local network NAT mappings, to detect failed connections, or to display latency metrics to the user. User agents must not use pings or unsolicited pongs to aid the server; it is assumed that servers will solicit pongs whenever appropriate for the server's needs.

#### 10.3.5 Parsing WebSocket URLs

The steps to **parse a WebSocket URL's components** from a string `url` are as follows. These steps return either a `host`, a `port`, a `resource name`, and a `secure` flag, or they fail.

1. If the `url` string is not an absolute URL, then fail this algorithm.
2. Resolve the `url` string, with the URL character encoding set to UTF-8. [RFC3629]

**Note:** It doesn't matter what it is resolved relative to, since we already know it is an absolute URL at this point.

3. If `url` does not have a `<scheme>` component whose value, when converted to ASCII lowercase, is either "ws" or "wss", then fail this algorithm.
4. If `url` has a `<fragment>` component, then fail this algorithm.
5. If the `<scheme>` component of `url` is "ws", set `secure` to false; otherwise, the `<scheme>` component is "wss", set `secure` to true.
6. Let `host` be the value of the `<host>` component of `url`, converted to ASCII lowercase.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

10. If *resource name* is the empty string, set it to a single character U+002F SOLIDUS (/).
11. If *url* has a <query> component, then append a single U+003F QUESTION MARK character (?) to *resource name*, followed by the value of the <query> component.
12. Return *host*, *port*, *resource name*, and *secure*.

### 10.3.6 Event definitions

[Con  
inte  
re  
re  
re  
};  
dict.  
    boolean wasClean;  
    unsigned short code;  
    DOMString reason;  
};

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

The `wasClean` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to false. It represents whether the connection closed cleanly or not.

The `code` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to zero. It represents the WebSocket connection close code provided by the server.

The `reason` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to empty string. It represents the WebSocket connection close reason provided by the server.

### 10.3.7 Garbage collection

A WebSocket object whose `readyState` attribute's value was set to `CONNECTING` (0) as of the last time the event loop started executing a task must not be garbage collected if there are any event listeners registered for `open` events, `message` events, `error` events, or `close` events.

A WebSocket object whose `readyState` attribute's value was set to `OPEN` (1) as of the last time the event loop started executing a task must not be garbage collected if there are any event listeners registered for `message` events, `error`, or `close` events.

A WebSocket object whose `readyState` attribute's value was set to `CLOSING` (2) as of the last time the event loop started executing a task must not be garbage collected if there are any event listeners registered for `error` or `close` events.

A WebSocket object with an established connection that has data queued to be transmitted to the network must not be garbage collected. [WSP]

If a WebSocket object is garbage collected while its connection is still open, the user agent must start the WebSocket closing handshake, with no status code for the Close message. [WSP]

If a user agent is to make disappear a WebSocket object (this happens when a Document object goes away), the user agent must follow the first appropriate set of steps from the following list:

↳ If the WebSocket connection is not yet established [WSP]

    Fail the WebSocket connection. [WSP]

↳ If the WebSocket closing handshake has not yet been started [WSP]

    Start the WebSocket closing handshake, with the status code to use in the WebSocket Close message being 1001. [WSP]

↳ Otherwise

    Do nothing.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

While this is an important security feature, it prevents pages from different domains from communicating even when those pages are not hostile. This section introduces a messaging system that allows documents to communicate with each other regardless of their source domain, in a way designed to not enable cross-site scripting attacks.

The task source for the tasks in cross-document messaging is the **posted message task source**.

### 10.4.1 Introduction

This section is [part of the WHATWG Living Standard](#).



For `postMessage()`, the `document` object, marked as originating from the [spec instead](https://web.archive.org/web/20130103030411/http://whatwg.org/html): <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

To receive messages, the script in document `B` might look like:

```
window.addEventListener('message', receiver, false);
function receiver(e) {
 if (e.origin == 'http://example.com') {
 if (e.data == 'Hello world') {
 e.source.postMessage('Hello', e.origin);
 } else {
 alert(e.data);
 }
 }
}
```

`document` A calls `postMessage()` on the `document` object, marked as originating from the [spec instead](https://web.archive.org/web/20130103030411/http://whatwg.org/html).

(similar mechanisms). For example, the script in document `B` might look like:

This script first checks the domain is the expected domain, and then looks at the message, which it either displays to the user, or responds to by sending a message back to the document which sent the message in the first place.

### 10.4.2 Security

#### 10.4.2.1 Authors

**⚠ Warning!** *Use of this API requires extra care to protect users from hostile entities abusing a site for their own purposes.*

Authors should check the `origin` attribute to ensure that messages are only accepted from domains that they expect to receive messages from. Otherwise, bugs in the author's message handling code could be exploited by hostile sites.

Furthermore, even after checking the `origin` attribute, authors should also check that the data in question is of the expected format. Otherwise, if the source of the event has been attacked using a cross-site scripting flaw, further unchecked processing of information sent using the `postMessage()` method could result in the attack being propagated into the receiver.

Authors should not use the wildcard keyword (\*) in the `targetOrigin` argument in messages that contain any confidential information, as otherwise there is no way to guarantee that the message is only delivered to the recipient to which it was intended.

Authors who accept messages from any origin are encouraged to consider the risks of a denial-of-service attack. An attacker could send a high volume of messages; if the receiving page performs expensive computation or causes network traffic to be sent for each such message, the attacker's message could be multiplied into a denial-of-service attack. Authors are encouraged to employ rate limiting (only accepting a certain number of messages per minute) to make such attacks impractical.

#### 10.4.2.2 User agents

The integrity of this API is based on the inability for scripts of one origin to post arbitrary events (using `dispatchEvent()` or `postMessage()`) to objects in other origins (those that are not the same).

**Note:** *Implementors are urged to take extra care in the implementation of this feature. It allows authors to transmit*

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

of-service attacks.

#### 10.4.3 Posting messages

This box is non-normative. Implementation requirements are given below this box.

```
window.postMessage(message, targetOrigin [, transfer])
```

Posts a message to the given window. Messages can be structured objects. e.g. nested objects and arrays, can contain file Blob, FileList, and on the sending side. information leakage. To message to same-origin e cloned.

Jav Ar: Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org  
 Ob If t ser tarç Thr /web/20130103030411/http://whatwg.org/html

**Note:** When posting a message to a Window of a browsing context that has just been navigated to a new Document is likely to result in the message not receiving its intended recipient: the scripts in the target browsing context have to have had time to set up listeners for the messages. Thus, for instance, in situations where a message is to be sent to the Window of newly created child iframe, authors are advised to have the child Document post a message to their parent announcing their readiness to receive messages, and for the parent to wait for this message before beginning posting messages.

When a script invokes the `postMessage (message, targetOrigin, transfer)` method (with two or three arguments) on a Window object, the user agent must follow these steps:

- If the value of the `targetOrigin` argument is neither a single U+002A ASTERISK character (\*), a single U+002F SOLIDUS character (/), nor an absolute URL, then throw a `SyntaxError` exception and abort the overall set of steps.
- Let `new ports` be an empty array.
- Let `transfer map` be an empty association list of Transferable objects to placeholder objects.
- If the method was invoked with a third argument `transfer`, run these substeps:
  - If any object is listed in `transfer` more than once, or any of the Transferable objects listed in `transfer` are marked as neutered, then throw a `DataCloneError` exception and abort these steps.
  - For each object `x` in `transfer` in turn, add a mapping from `x` to a new unique placeholder object created for `x` to `transfer map`, and if `x` is a `MessagePort` object, also append the placeholder object to the `new ports` array.
- Let `message clone` be the result of obtaining a structured clone of the `message` argument, with `transfer map` as the `transfer map`. If this throws an exception, then throw that exception and abort these steps.
- If the method was invoked with a third argument `transfer`, run these substeps:
  - Let `new owner` be the `Window` object on which the method was invoked.
  - For each object `x` in `transfer` in turn, obtain a new object `y` by transferring the object `x` to `new owner`, and replace the placeholder object that was created for the object `x` by the new object `y` wherever the placeholder exists (i.e. in `message clone` and in `new ports`).
- Make `new ports` into a read only array.
- Return from the `postMessage ()` method, but asynchronously continue running these steps.
- If the `targetOrigin` argument is a single literal U+002F SOLIDUS character (/), and the `Document` of the `Window` object on which the method was invoked does not have the same origin as the entry script's document, then abort these steps silently.
- Otherwise, if the `targetOrigin` argument is an absolute URL, and the `Document` of the `Window` object on which the method was invoked does not have the same origin as `targetOrigin`, then abort these steps silently.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

must be initialized to the script's global object's `WindowProxy` object, and the `ports` attribute must be initialized to the new `ports` array.

11. Queue a task to dispatch the event created in the previous step at the `Window` object on which the method was invoked. The task source for this task is the posted message task source.

## 10.5 Channel messaging



### 10.5.1 Int

*This section*

To enable in messaging.

Communication are delivered

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

; authors can use channel

Messages sent in one port

To create a connection (two "entangled" ports), the `MessageChannel()` constructor is called:

```
var channel = new MessageChannel();
```

One of the ports is kept as the local port, and the other port is sent to the remote code, e.g. using `postMessage()`:

```
otherWindow.postMessage('hello', 'http://example.com', [channel.port2]);
```

To send messages, the `postMessage()` method on the port is used:

```
channel.port1.postMessage('hello');
```

To receive messages, one listens to `message` events:

```
channel.port1.onmessage = handleMessage;
function handleMessage(event) {
 // message is in event.data
 // ...
}
```

Data sent on a port can be structured data; for example here an array of strings is passed:

```
port1.postMessage(['hello', 'world'], 'http://example.com');
```

### 10.5.1.1 Examples

*This section is non-normative.*

In this example, two JavaScript libraries are connected to each other using `MessagePorts`. This allows the libraries to later be hosted in different frames, or in `Worker` objects, without any change to the APIs.

```
<script src="contacts.js"></script> <!-- exposes a contacts object -->
<script src="compose-mail.js"></script> <!-- exposes a composer object -->
<script>
 var channel = new MessageChannel();
 composer.addContactsProvider(channel.port1);
 contacts.registerConsumer(channel.port2);
</script>
```

Here's what the "`addContactsProvider()`" function's implementation could look like:

```
function addContactsProvider(port) {
 port.onmessage = function (event) {
 switch (event.data.messageType) {
 'search-result': handleSearchResult(event.data.results); break;
 'search-done': handleSearchDone(); break;
 'search-error': handleSearchError(event.data.message); break;
 }
 }
}
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Alternatively, it could be implemented as follows:

```
function addContactsProvider(port) {
 port.addEventListener('message', function (event) {
 if (event.data.messageType == 'search-result')
 handleSearchResult(event.data.results);
 });
 port.addEventListener('message', function (event) {
 if (event.data.messageType == 'search-done')
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The key difference is that when using `addEventListener()`, the `start()` method must also be invoked. When using `onmessage`, the call to `start()` is implied.

The `start()` method, whether called explicitly or implicitly (by setting `onmessage`), starts the flow of messages: messages posted on message ports are initially paused, so that they don't get dropped on the floor before the script has had a chance to set up its handlers.

#### 10.5.1.2 Ports as the basis of an object-capability model on the Web

*This section is non-normative.*

Ports can be viewed as a way to expose limited capabilities (in the object-capability model sense) to other actors in the system. This can either be a weak capability system, where the ports are merely used as a convenient model within a particular origin, or as a strong capability model, where they are provided by one origin *provider* as the only mechanism by which another origin *consumer* can effect change in or obtain information from *provider*.

For example, consider a situation in which a social Web site embeds in one `iframe` the user's e-mail contacts provider (an address book site, from a second origin), and in a second `iframe` a game (from a third origin). The outer social site and the game in the second `iframe` cannot access anything inside the first `iframe`; together they can only:

- Navigate the `iframe` to a new URL, such as the same URL but with a different fragment identifier, causing the `Window` in the `iframe` to receive a `hashchange` event.
- Resize the `iframe`, causing the `Window` in the `iframe` to receive a `resize` event.
- Send a `message` event to the `Window` in the `iframe` using the `window.postMessage()` API.

The contacts provider can use these methods, most particularly the third one, to provide an API that can be accessed by other origins to manipulate the user's address book. For example, it could respond to a message "`add-contact Guillaume Tell <tell@pomme.example.net>`" by adding the given person and e-mail address to the user's address book.

To avoid any site on the Web being able to manipulate the user's contacts, the contacts provider might only allow certain trusted sites, such as the social site, to do this.

Now suppose the game wanted to add a contact to the user's address book, and that the social site was willing to allow it to do so on its behalf, essentially "sharing" the trust that the contacts provider had with the social site. There are several ways it could do this; most simply, it could just proxy messages between the game site and the contacts site. However, this solution has a number of difficulties: it requires the social site to either completely trust the game site not to abuse the privilege, or it requires that the social site verify each request to make sure it's not a request that it doesn't want to allow (such as adding multiple contacts, reading the contacts, or deleting them); it also requires some additional complexity if there's ever the possibility of multiple games simultaneously trying to interact with the contacts provider.

Using message channels and `MessagePort` objects, however, all of these problems can go away. When the game tells the social site that it wants to add a contact, the social site can ask the contacts provider not for it to add a contact, but for the *capability* to add a single contact. The contacts provider then creates a pair of `MessagePort` objects, and sends one of them back to the social site, who forwards it on to the game. The game and the contacts provider then have a direct connection, and the contacts provider knows to only honor a single "add contact" request, nothing else. In other words, the game has been granted the capability to add a single contact.

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

have simply used XMLHttpRequest objects in the service's iframe, an evolution of the service might instead want to use a shared worker with a single WebSocket connection.

If the initial design used MessagePort objects to grant capabilities, or even just to allow multiple simultaneous independent sessions, the service implementation can switch from the XMLHttpRequests-in-each-iframe model to the shared-WebSocket model without changing the API at all: the ports on the service provider side can all be forwarded to the shared worker without it affecting the users of the API in the slightest.

### 10.5.2 MessageChannel

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Links to the multipage copy of the spec are given below this box.

`channel = new MessageChannel()`

Returns a new MessageChannel object with two new MessagePort objects.

`channel.port1`

Returns the first MessagePort object.

`channel.port2`

Returns the second MessagePort object.

When the `MessageChannel()` constructor is called, it must run the following algorithm:

1. Create a new MessagePort object owned by the script's global object, and let `port1` be that object.
2. Create a new MessagePort object owned by the script's global object, and let `port2` be that object.
3. Entangle the `port1` and `port2` objects.
4. Instantiate a new MessageChannel object, and let `channel` be that object.
5. Let the `port1` attribute of the `channel` object be `port1`.
6. Let the `port2` attribute of the `channel` object be `port2`.
7. Return `channel`.

This constructor must be visible when the script's global object is either a Window object or an object implementing the WorkerUtils interface.

The `port1` and `port2` attributes must return the values they were assigned when the `MessageChannel` object was created.

### 10.5.3 Message ports

Each channel has two message ports. Data sent through one port is received by the other port, and vice versa.

```
interface MessagePort : EventTarget {
 void postMessage(any message, optional sequence<Transferable> transfer);
 void start();
 void close();

 // event handlers
 attribute EventHandler onmessage;
};
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Posts a message through the channel. Objects listed in *transfer* are transferred, not just cloned, meaning that they are no longer usable on the sending side.

Throws a `DataCloneError` if *transfer* array contains duplicate objects or the source or target ports, or if *message* could not be cloned.

`port.start()`

Begins dispatching messages received on the port.



`port.close()`

**Dis** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Each MessagePort object also has a task queue, and is initially disabled. Once enabled, it can be removed altogether, without affecting the queue or removed.

When the user agent is to **create a new MessagePort object** owned by a script's global object *object owner*, it must instantiate a new `MessagePort` object, and let its owner be *owner*.

When the user agent is to **entangle** two `MessagePort` objects, it must run the following steps:

1. If one of the ports is already entangled, then disentangle it and the port that it was entangled with.

**Note:** If those two previously entangled ports were the two ports of a `MessageChannel` object, then that `MessageChannel` object no longer represents an actual channel: the two ports in that object are no longer entangled.

2. Associate the two ports to be entangled, so that they form the two parts of a new channel. (There is no `MessageChannel` object that represents this channel.)

When the user agent is to **clone a port** *original port*, with the clone being owned by *owner*, it must run the following steps, which return a new `MessagePort` object. These steps must be run atomically.

1. Create a new `MessagePort` object owned by *owner*, and let *new port* be that object.
2. Move all the events in the port message queue of *original port* to the port message queue of *new port*, if any, leaving the *new port*'s port message queue in its initial disabled state.
3. If the *original port* is entangled with another port, then run these substeps:
  1. Let the *remote port* be the port with which the *original port* is entangled.
  2. Entangle the *remote port* and *new port* objects. The *original port* object will be disentangled by this process.
4. Return *new port*. It is the clone.

To transfer a `MessagePort` object *old* to a new owner *owner*, a user agent must clone the *old* object with the clone being owned by *owner*, thus obtaining *new*, must neuter the *old* port, and must finally return *new*.

The `postMessage()` method, when called on a port *source port*, must cause the user agent to run the following steps:

1. Let *target port* be the port with which *source port* is entangled, if any.
2. Let *new ports* be an empty array.
3. Let *transfer map* be an empty association list of `Transferable` objects to placeholder objects.
4. If the method was invoked with a second argument *transfer*, run these substeps:
  1. If any object is listed in *transfer* more than once, or any of the `Transferable` objects listed in *transfer* are marked as neutered, then throw a `DataCloneError` exception and abort these steps.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

5. Let `message clone` be the result of obtaining a structured clone of the `message` argument, with `transfer map` as the `transfer map`. If this throws an exception, then throw that exception and abort these steps.

6. If the method was invoked with a second argument `transfer`, run these substeps:

1. Let `new owner` be the owner of `target port`, if there is a `target port`, or else some arbitrary owner. (This `new owner` is used when transferring objects below. If there is no `target port`, the Transferable objects given in the second argument, if any, are still transferred, but since they are then discarded, it doesn't matter where they are transferred)

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

7. Make `/web/20130103030411/http:`

8. If the `//whatwg.org/html`

9. Create a new `MessagePort` object. It bubble, is not cancelable, and has no default action.

10. Let the `data` attribute of the event be initialized to the value of `message clone`.

11. Let the `ports` attribute of the event be initialized to the `new ports` array.

12. Add the event to the port message queue of `target port`.

The `start()` method must enable its port's port message queue, if it is not already enabled.

When a port's port message queue is enabled, the event loop must use it as one of its task sources.

**Note:** If the Document of the port's event listeners' global object is not fully active, then the messages are lost.

The `close()` method, when called on a port `local port` that is entangled with another port, must cause the user agents to disentangle the two ports. If the method is called on a port that is not entangled, then the method must do nothing.

The following are the event handlers (and their corresponding event handler event types) that must be supported, as IDL attributes, by all objects implementing the `MessagePort` interface:

Event handler	Event handler event type
<code>onmessage</code>	<code>message</code>

The first time a `MessagePort` object's `onmessage` IDL attribute is set, the port's port message queue must be enabled, as if the `start()` method had been called.

### 10.5.4 Broadcasting to many ports

Broadcasting to many ports is in principle relatively simple: keep an array of `MessagePort` objects to send messages to, and iterate through the array to send a message. However, this has one rather unfortunate effect: it prevents the ports from being garbage collected, even if the other side has gone away.

To avoid this problem, the `PortCollection` object can be used. It acts as an opaque array of `MessagePort` objects, thus allowing the objects to be garbage collected when they stop being relevant, while still allowing scripts to iterate over the `MessagePort` objects.

```
[Constructor] interface PortCollection {
 void add(MessagePort port);
 void remove(MessagePort port);
 void clear();
 void iterate(PortCollectionCallback callback);
};
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Returns a new empty `PortCollection` object.

#### `portCollection . add(port)`

Adds `port` to the collection, if it isn't already present.

#### `portCollection . remove(port)`

Removes `port` from the collection, if it is present.

#### `portColl`

Re: Too slow? Try reading the multipage copy of  
`portCo`. the spec instead: [https://web.archive.org  
Cal /web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

A `PortColl`

orts is garbage collected, it must be silently removed from that list of ports. Objects in a list of ports are ordered chronologically by the time at which they were most recently added; the least-recently added `MessagePort` object is the first in the list, and the most-recently added `MessagePort` is the last in the list.

The `PortCollection()` constructor must return a new `PortCollection` object (with an empty list of ports).

The `add()` method must add the `MessagePort` given by the argument to the `PortCollection` object's list of ports, unless the `MessagePort` is already in the list of ports, in which case the method does nothing. (Calling this method with a port already in the list does not move the port to the end of the list.)

The `remove()` method must remove the `MessagePort` given by the argument from the `PortCollection` object's list of ports, unless the `MessagePort` is not in the list of ports, in which case the method does nothing.

The `clear()` method must remove all `MessagePort` objects from the `PortCollection` object's list of ports, returning it to the initial empty state. If the list of ports is already empty, the method does nothing.

The `iterate()` method must invoke its `PortCollectionCallback` argument once for each `MessagePort` object in the object's list of ports, in the order defined above, with each invocation being passed the corresponding `MessagePort` object as the callback's sole argument.

### 10.5.5 Ports and garbage collection

When a `MessagePort` object `o` is entangled, user agents must either act as if `o`'s entangled `MessagePort` object has a strong reference to `o`, or as if `o`'s owner has a strong reference to `o`.

*Thus, a message port can be received, given an event listener, and then forgotten, and so long as that event listener could receive a message, the channel will be maintained.*

*Of course, if this was to occur on both sides of the channel, then both ports could be garbage collected, since they would not be reachable from live code, despite having a strong reference to each other.*

Furthermore, a `MessagePort` object must not be garbage collected while there exists a message in a task queue that is to be dispatched on that `MessagePort` object, or while the `MessagePort` object's port message queue is open and there exists a message event in that queue.

There are no strong references from a `PortCollection` object to its `MessagePort` objects. (That is in fact the whole point of `PortCollection` objects: they allow for `MessagePort` objects to be referenced without preventing them from being garbage collected.)

*Note: Authors are strongly encouraged to explicitly close `MessagePort` objects to disentangle them, so that their resources can be recollected. Creating many `MessagePort` objects and discarding them without closing them can lead to high transient memory usage since garbage collection is not necessarily performed promptly.*

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

This section is non-normative.

This specification introduces two related mechanisms, similar to HTTP session cookies, for storing name-value pairs on the client side. [COOKIES]

The first is designed for scenarios where the user is carrying out a single transaction, but could be carrying out multiple transactions in different windows at the same time.

Cookies don't really handle this case well. For example, a user could be buying plane tickets in two different windows, using the same site. If the site used cookies to keep track of which ticket the user was buying, then as the user ~~walked~~ from page to page in both windows, the user would end up buying two tickets for the same flight.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/>**

To address this problem, session storage, and it will be accessible across multiple windows.

For example, /web/20130103030411/http://whatwg.org/html:

```
//whatwg.org/html
```

```
I want insurance on this trip.
</label>
```

A later page could then check, from script, whether the user had checked the checkbox or not:

```
if (sessionStorage.insurance) { ... }
```

If the user had multiple windows opened on the site, each one would have its own individual copy of the session storage object.

The second storage mechanism is designed for storage that spans multiple windows, and lasts beyond the current session. In particular, Web applications may wish to store megabytes of user data, such as entire user-authored documents or a user's mailbox, on the client side for performance reasons.

Again, cookies do not handle this case well, because they are transmitted with every request.

The `localStorage` IDL attribute is used to access a page's local storage area.

The site at example.com can display a count of how many times the user has loaded its page by putting the following at the bottom of its page:

```
<p>
 You have viewed this page
 an untold number of
 time(s).
</p>
<script>
 if (!localStorage.pageLoadCount)
 localStorage.pageLoadCount = 0;
 localStorage.pageLoadCount = parseInt(localStorage.pageLoadCount) + 1;
 document.getElementById('count').textContent = localStorage.pageLoadCount;
</script>
```

Each site has its own separate storage area.

## 11.2 The API

### 11.2.1 The Storage interface

```
interface Storage {
 readonly attribute unsigned long length;
 DOMString? key(unsigned long index);
 getter DOMString? getItem(DOMString key);
 setter creator void setItem(DOMString key, DOMString value);
 deleter void removeItem(DOMString key);
 void clear();
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Each Storage object is associated with a list of key/value pairs when it is created, as defined in the sections on the `sessionStorage` and `localStorage` attributes. Multiple separate objects implementing the `Storage` interface can all be associated with the same list of key/value pairs simultaneously.

The `length` attribute must return the number of key/value pairs currently present in the list associated with the object.

The `key(n)` method must return the name of the  $n$ th key in the list. The order of keys is user-agent defined, but must be consistent within an object so long as the number of keys doesn't change. (Thus, adding or removing a key may change the order of the keys, but merely changing the value of an existing key must not.) If  $n$  is greater than or equal to the number of key/value pairs in the object, then this method returns `null`.

The support object. **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org** in the list associated with the object.

The `getItem(key)` method must return the value of the key in the list associated with the object. If the key does not exist in the list, then this method returns `null`.

The `setItem(key, value)` method must add a new key/value pair to the list associated with the object. If the key already exists in the list, then its value is updated to `value`. If its previous value is equal to `value`, then the method must do nothing.

If it does not, then a new key/value pair must be added to the list, with the given `key` and with its value set to `value`.

If the given `key` does exist in the list, and its value is not equal to `value`, then it must have its value updated to `value`. If its previous value is equal to `value`, then the method must do nothing.

If it couldn't set the new value, the method must throw an `QuotaExceededError` exception. (Setting could fail if, e.g., the user has disabled storage for the site, or if the quota has been exceeded.)

The `removeItem(key)` method must cause the key/value pair with the given `key` to be removed from the list associated with the object, if it exists. If no item with that key exists, the method must do nothing.

The `setItem()` and `removeItem()` methods must be atomic with respect to failure. In the case of failure, the method does nothing. That is, changes to the data storage area must either be successful, or the data storage area must not be changed at all.

The `clear()` method must atomically cause the list associated with the object to be emptied of all key/value pairs, if there are any. If there are none, then the method must do nothing.

**Note:** When the `setItem()`, `removeItem()`, and `clear()` methods are invoked, events are fired on the `Window` objects of other Documents that can access the newly stored or removed data, as defined in the sections on the `sessionStorage` and `localStorage` attributes.

**Note:** This specification does not require that the above methods wait until the data has been physically written to disk. Only consistency in what different scripts accessing the same underlying list of key/value pairs see is required.

### 11.2.2 The `sessionStorage` attribute

```
[NoInterfaceObject]
interface WindowSessionStorage {
 readonly attribute Storage sessionStorage;
};
Window implements WindowSessionStorage;
```

The `sessionStorage` attribute represents the set of storage areas specific to the current top-level browsing context.

Each top-level browsing context has a unique set of session storage areas, one for each origin.

User agents should not expire data from a browsing context's session storage areas, but may do so when the user requests that such data be deleted, or when the UA detects that it has limited storage space, or for security reasons. User agents should always avoid deleting data while a script that could access that data is running. When a top-level browsing context is destroyed (and therefore permanently inaccessible to the user) the data stored in its session storage areas can be discarded with it, as the API described in this specification provides no way for that data to ever be subsequently retrieved.

**Note:** The lifetime of a browsing context can be unrelated to the lifetime of the actual user agent process itself, as the

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

session storage area. If it does not, a new storage area for that document's origin must be created, and then that is the document's assigned session storage area. A Document's assigned storage area does not change during the lifetime of a Document.

**Note:** In the case of an `iframe` being moved to another Document, the nested browsing context is destroyed and a new one created.

The `sessionStorage` attribute must return a `Storage` object associated with the Document's assigned session storage area, if any, or null if there isn't one. Each Document object must have a separate object for its Window's `sessionStorage` attribute.

When a new +  
same sessic  
any way.

When a new  
existing brov  
then the ses  
that point on

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

When the sel  
storage area, if the methods did not throw an exception or "do nothing" as defined above, then for every Document object whose Window object's `sessionStorage` attribute's `Storage` object is associated with the same storage area, other than x, send a storage notification.

ing context must start with the te, not affecting each other in

r following a link in an ew start for session storage, t when it is created. From er in any way.

associated with a session

### 11.2.3 The `localStorage` attribute

```
[NoInterfaceObject]
interface WindowLocalStorage {
 readonly attribute Storage localStorage;
};

Window implements WindowLocalStorage;
```

The `localStorage` object provides a `Storage` object for an origin.

User agents must have a set of local storage areas, one for each origin.



User agents should expire data from the local storage areas only for security reasons or when requested to do so by the user. User agents should always avoid deleting data while a script that could access that data is running.

When the `localStorage` attribute is accessed, the user agent must run the following steps, which are known as the `Storage object initialization steps`:

1. The user agent may throw a `SecurityError` exception instead of returning a `Storage` object if the request violates a policy decision (e.g. if the user agent is configured to not allow the page to persist data).
2. If the Document's origin is not a scheme/host/port tuple, then throw a `SecurityError` exception and abort these steps.
3. Check to see if the user agent has allocated a local storage area for the origin of the Document of the Window object on which the attribute was accessed. If it has not, create a new storage area for that origin.
4. Return the `Storage` object associated with that origin's local storage area. Each Document object must have a separate object for its Window's `localStorage` attribute.

When the `setItem()`, `removeItem()`, and `clear()` methods are called on a `Storage` object x that is associated with a local storage area, if the methods did not throw an exception or "do nothing" as defined above, then for every Document object whose Window object's `localStorage` attribute's `Storage` object is associated with the same storage area, other than x, send a storage notification.

Whenever the properties of a `localStorage` attribute's `Storage` object are to be examined, returned, set, or deleted, whether as part of a direct property access, when checking for the presence of a property, during property enumeration, when determining the number of properties present, or as part of the execution of any of the methods or attributes defined on the `Storage` interface, the user agent must first obtain the storage mutex.

#### 11.2.3.1 Security

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*Note. This means Storage objects are deleted when the document.domain attribute is used.*

#### 11.2.4 The storage event

The **storage** event is fired on a Document's Window object when a storage area changes, as described in the previous two sections (for session storage, for local storage).

When a user agent is to **send a storage notification** for a Document, the user agent must queue a task to fire a trusted event with the name `storage`, at the Document object's Window object.

#### Too slow? Try reading the multipage copy of

*Note: Since the spec instead: https://web.archive.org*

*/web/20130103030411/http:*

The task so if the event is initialized to the name or the key in question, its `oldValue` attribute initialized to the old value of the key in question, or null if the key is newly added, and its `newValue` attribute initialized to the new value of the key in question, or null if the key was removed.

Otherwise, if the event is being fired due to an invocation of the `clear()` method, the event must have its `key`, `oldValue`, and `newValue` attributes initialized to null.

In addition, the event must have its `url` attribute initialized to the address of the document whose `Storage` object was affected; and its `storageArea` attribute initialized to the `Storage` object from the `Window` object of the target `Document` that represents the same kind of `Storage` area as was affected (i.e. session or local).

##### 11.2.4.1 Event definition

```
[Constructor(DOMString type, optional StorageEventInit eventInitDict)]
interface StorageEvent : Event {
 readonly attribute DOMString? key;
 readonly attribute DOMString? oldValue;
 readonly attribute DOMString? newValue;
 readonly attribute DOMString url;
 readonly attribute Storage? storageArea;
};

dictionary StorageEventInit : EventInit {
 DOMString? key;
 DOMString? oldValue;
 DOMString? newValue;
 DOMString url;
 Storage? storageArea;
};
```

The `key` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the key being changed.

The `oldValue` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the old value of the key being changed.

The `newValue` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the new value of the key being changed.

The `url` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to the empty string. It represents the address of the document whose key changed.

The `storageArea` attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to null. It represents the `Storage` object that was affected.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Thus, the `length` attribute of a `Storage` object, and the value of the various properties of that object, cannot change while a script is executing, other than in a way that is predictable by the script itself.

## 11.3 Disk space

User agents should limit the total amount of space allowed for storage areas.

User agents should guard against sites storing data under their origin's other affiliated sites, e.g. ~~going~~ up to the limit in `a1.example.c`

User agents  
many user-c  
User agents  
A mostly art  
suggestion i

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

This enables sites to store

I will be used to update this

## 11.4 Privacy

### 11.4.1 User tracking

A third-party advertiser (or any entity capable of getting content distributed to multiple sites) could use a unique identifier stored in its local storage area to track a user across multiple sessions, building a profile of the user's interests to allow for highly targeted advertising. In conjunction with a site that is aware of the user's real identity (for example an e-commerce site that requires authenticated credentials), this could allow oppressive groups to target individuals with greater accuracy than in a world with purely anonymous Web usage.

There are a number of techniques that can be used to mitigate the risk of user tracking:

#### Blocking third-party storage

User agents may restrict access to the `localStorage` objects to scripts originating at the domain of the top-level document of the browsing context, for instance denying access to the API for pages from other domains running in `iframes`.

#### Expiring stored data

User agents may, possibly in a manner configured by the user, automatically delete stored data after a period of time.

For example, a user agent could be configured to treat third-party local storage areas as session-only storage, deleting the data once the user had closed all the browsing contexts that could access it.

This can restrict the ability of a site to track a user, as the site would then only be able to track the user across multiple sessions when he authenticates with the site itself (e.g. by making a purchase or logging in to a service).

However, this also reduces the usefulness of the API as a long-term storage mechanism. It can also put the user's data at risk, if the user does not fully understand the implications of data expiration.

#### Treating persistent storage as cookies

If users attempt to protect their privacy by clearing cookies without also clearing data stored in the local storage area, sites can defeat those attempts by using the two features as redundant backup for each other. User agents should present the interfaces for clearing these in a way that helps users to understand this possibility and enables them to delete data in all persistent storage features simultaneously. [COOKIES]

#### Site-specific white-listing of access to local storage areas

User agents may allow sites to access session storage areas in an unrestricted manner, but require the user to authorize access to local storage areas.

#### Origin-tracking of stored data

User agents may record the origins of sites that contained content from third-party origins that caused data to be stored.

If this information is then used to present the view of data currently in persistent storage, it would allow the user to make informed decisions about which parts of the persistent storage to prune. Combined with a blacklist ("delete this data and prevent this domain from ever storing data again"), the user can restrict the use of persistent storage to sites that he trusts.

#### Shared blacklists

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

can continue to track the user during a session, and can then pass all this information to the third party along with any identifying information (names, credit card numbers, addresses) obtained by the site. If a third party cooperates with multiple sites to obtain such information, a profile can still be created.

However, user tracking is to some extent possible even with no cooperation from the user agent whatsoever, for instance by using session identifiers in URLs, a technique already commonly used for innocuous purposes but easily repurposed for user tracking (even retroactively). This information can then be shared with other sites, using visitors' IP addresses and other user-specific data (e.g. user-agent headers and configuration settings) to combine separate sessions into coherent user profiles.



#### 11.4.2 Se

User agents records, or c  
**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

To this end,

[/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

endar appointments, health

storage.

#### 11.5 Sec.

##### 11.5.1 DNS spoofing attacks

Because of the potential for DNS spoofing attacks, one cannot guarantee that a host claiming to be in a certain domain really is from that domain. To mitigate this, pages can use TLS. Pages using TLS can be sure that only the user, software working on behalf of the user, and other pages using TLS that have certificates identifying them as being from the same domain, can access their storage areas.

##### 11.5.2 Cross-directory attacks

Different authors sharing one host name, for example users hosting content on `geocities.com`, all share one local storage object. There is no feature to restrict the access by pathname. Authors on shared hosts are therefore urged to avoid using these features, as it would be trivial for other authors to read the data and overwrite it.

**Note:** Even if a path-restriction feature was made available, the usual DOM scripting security model would make it trivial to bypass this protection and access the data from any path.

##### 11.5.3 Implementation risks

The two primary risks when implementing these persistent storage features are letting hostile sites read information from other domains, and letting hostile sites write information that is then read from other domains.

Letting third-party sites read data that is not supposed to be read from their domain causes *information leakage*. For example, a user's shopping wishlist on one domain could be used by another domain for targeted advertising; or a user's work-in-progress confidential documents stored by a word-processing site could be examined by the site of a competing company.

Letting third-party sites write data to the persistent storage of other domains can result in *information spoofing*, which is equally dangerous. For example, a hostile site could add items to a user's wishlist; or a hostile site could set a user's session identifier to a known ID that the hostile site can then use to track the user's actions on the victim site.

Thus, strictly following the origin model described in this specification is important for user security.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## 12.1 Writing HTML documents

This section only applies to documents, authoring tools, and markup generators. In particular, it does not apply to conformance checkers; conformance checkers must use the requirements given in the next section ("parsing HTML documents").

Documents must consist of the following parts, in the given order:

1. Optir
2. Any **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**
3. A D **/web/20130103030411/http:**
4. Any **//whatwg.org/html**
5. The **Any number of comments and space characters.**

The various types of content mentioned above are described in the next few sections.

In addition, there are some restrictions on how character encoding declarations are to be serialized, as discussed in the section on that topic.

**Space characters before the root `html` element, and space characters at the start of the `html` element and before the `head` element, will be dropped when the document is parsed; space characters after the root `html` element will be parsed as if they were at the end of the `body` element. Thus, space characters around the root element do not round-trip.**

**It is suggested that newlines be inserted after the DOCTYPE, after any comments that are before the root element, after the `html` element's start tag (if it is not omitted), and after any comments that are inside the `html` element but before the `head` element.**

Many strings in the HTML syntax (e.g. the names of elements and their attributes) are case-insensitive, but only for uppercase ASCII letters and lowercase ASCII letters. For convenience, in this section this is just referred to as "case-insensitive".

### 12.1.1 The DOCTYPE

A DOCTYPE is a required preamble.

**Note: DOCTYPES are required for legacy reasons. When omitted, browsers tend to use a different rendering mode that is incompatible with some specifications. Including the DOCTYPE in a document ensures that the browser makes a best-effort attempt at following the relevant specifications.**

A DOCTYPE must consist of the following components, in this order:

1. A string that is an ASCII case-insensitive match for the string "<!DOCTYPE".
2. One or more space characters.
3. A string that is an ASCII case-insensitive match for the string "html".
4. Optionally, a DOCTYPE legacy string or an obsolete permitted DOCTYPE string (defined below).
5. Zero or more space characters.
6. A U+003E GREATER-THAN SIGN character (>).

**Note: In other words, <!DOCTYPE html>, case-insensitively.**

For the purposes of HTML generators that cannot output HTML markup with the short DOCTYPE "<!DOCTYPE html>", a **DOCTYPE legacy string** may be inserted into the DOCTYPE (in the position defined above). This string must consist of:

1. One or more space characters.
2. A string that is an ASCII case-insensitive match for the string "SYSTEM".
3. One or more space characters.
4. A U+0022 QUOTATION MARK or U+0027 APOSTROPHE character (the quote mark).

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

'about:legacy-compat'&gt;, case-insensitively except for the part in single or double quotes.

The DOCTYPE legacy string should not be used unless the document is generated from a system that cannot output the shorter string.

To help authors transition from HTML4 and XHTML1, an **obsolete permitted DOCTYPE string** can be inserted into the DOCTYPE (in the position defined above). This string must consist of:

1. One or more space characters.
2. A string that is an ASCII case-insensitive match for the string "PUBLIC".

3. One

4. A U

5. The

6. A m

labe

7. If th

8. If th

cha

9. If th

row

10. If the

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

APOSTROPHE character (i.e. the same character as in the earlier step labeled *inner quote mark*).ongs is the selected row.  
acter as in the earlier step

U+0027 APOSTROPHE

nd column of the selected

, MARK or U+0027

Allowed values for public and system identifiers in an obsolete permitted DOCTYPE string.

Public identifier	System identifier
-//W3C//DTD HTML 4.0//EN	
-//W3C//DTD HTML 4.0//EN	http://www.w3.org/TR/REC-html40/strict.dtd
-//W3C//DTD HTML 4.01//EN	
-//W3C//DTD HTML 4.01//EN	http://www.w3.org/TR/html4/strict.dtd
-//W3C//DTD XHTML 1.0 Strict//EN	http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
-//W3C//DTD XHTML 1.1//EN	http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd

A DOCTYPE containing an obsolete permitted DOCTYPE string is an **obsolete permitted DOCTYPE**. Authors should not use obsolete permitted DOCTYPES, as they are unnecessarily long.

## 12.1.2 Elements

There are five different kinds of **elements**: void elements, raw text elements, RCDATA elements, foreign elements, and normal elements.

### Void elements

area, base, br, col, embed, hr, img, input, keygen, link, menuitem, meta, param, source, track, wbr

### Raw text elements

script, style

### RCDATA elements

textarea, title

### Foreign elements

Elements from the MathML namespace and the SVG namespace.

### Normal elements

All other allowed HTML elements are normal elements.

**Tags** are used to delimit the start and end of elements in the markup. Raw text, RCDATA, and normal elements have a start tag to indicate where they begin, and an end tag to indicate where they end. The start and end tags of certain normal elements can be omitted, as described later. Those that cannot be omitted must not be omitted. Void elements only have a start tag; end tags must not be specified for void elements. Foreign elements must either have a start tag and an end tag, or a start tag that is marked as self-closing, in which case they must not have an end tag.

The contents of the element must be placed between just after the start tag (which might be implied, in certain cases) and just before the end tag (which again, might be implied in certain cases). The exact allowed contents of each individual element depend on the content model of that element, as described earlier in this specification. Elements must not contain content that their content model disallows. In addition to the restrictions placed on the contents by those content models, however, the five types of elements have additional *syntactic* requirements.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

further restrictions described below.

Foreign elements whose start tag is marked as self-closing can't have any contents (since, again, as there's no end tag, no content can be put between the start tag and the end tag). Foreign elements whose start tag is *not* marked as self-closing can have text, character references, CDATA sections, other elements, and comments, but the text must not contain the character U+003C LESS-THAN SIGN (<) or an ambiguous ampersand.

**The HTML syntax does not support namespace declarations, even in foreign elements.**

For instance, consider the following HTML fragment:

```
<p> Too slow? Try reading the multipage copy of
<!--
 · the spec instead: https://web.archive.org
 /web/20130103030411/http:
 ·
 < //whatwg.org/html
</p>
```

T"/&gt;

**The innermost element, `cdr:license`, is actually in the SVG namespace, as the "`xmlns:cdr`" attribute has no effect (unlike in XML). In fact, as the comment in the fragment above says, the fragment is actually non-conforming. This is because the SVG specification does not define any elements called "`cdr:license`" in the SVG namespace.**

Normal elements can have text, character references, other elements, and comments, but the text must not contain the character U+003C LESS-THAN SIGN (<) or an ambiguous ampersand. Some normal elements also have yet more restrictions on what content they are allowed to hold, beyond the restrictions imposed by the content model and those described in this paragraph. Those restrictions are described below.

Tags contain a **tag name**, giving the element's name. HTML elements all have names that only use alphanumeric ASCII characters. In the HTML syntax, tag names, even those for foreign elements, may be written with any mix of lower- and uppercase letters that, when converted to all-lowercase, matches the element's tag name; tag names are case-insensitive.

### 12.1.2.1 Start tags

**Start tags** must have the following format:

1. The first character of a start tag must be a U+003C LESS-THAN SIGN character (<).
2. The next few characters of a start tag must be the element's tag name.
3. If there are to be any attributes in the next step, there must first be one or more space characters.
4. Then, the start tag may have a number of attributes, the syntax for which is described below. Attributes must be separated from each other by one or more space characters.
5. After the attributes, or after the tag name if there are no attributes, there may be one or more space characters. (Some attributes are required to be followed by a space. See the attributes section below.)
6. Then, if the element is one of the void elements, or if the element is a foreign element, then there may be a single U+002F SOLIDUS character (/). This character has no effect on void elements, but on foreign elements it marks the start tag as self-closing.
7. Finally, start tags must be closed by a U+003E GREATER-THAN SIGN character (>).

### 12.1.2.2 End tags

**End tags** must have the following format:

1. The first character of an end tag must be a U+003C LESS-THAN SIGN character (<).
2. The second character of an end tag must be a U+002F SOLIDUS character (/).
3. The next few characters of an end tag must be the element's tag name.
4. After the tag name, there may be one or more space characters.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Attributes for an element are expressed inside the element's start tag.

Attributes have a name and a value. **Attribute names** must consist of one or more characters other than the space characters, U+0000 NULL, U+0022 QUOTATION MARK ("), U+0027 APOSTROPHE ('), U+003E GREATER-THAN SIGN (>), U+002F SOLIDUS (/), and U+003D EQUALS SIGN (=) characters, the control characters, and any characters that are not defined by Unicode. In the HTML syntax, attribute names, even those for foreign elements, may be written with any mix of lower- and uppercase letters that are an ASCII case-insensitive match for the attribute's name.

**Attribute values** are a mixture of text and character references, except with the additional restriction that the text cannot contain an ambiguous ampersand.

Attributes ca **Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If an attribute using the empty attribute syntax is to be followed by another attribute, then there must be a space character separating the two.

#### Unquoted attribute value syntax

The attribute name, followed by zero or more space characters, followed by a single U+003D EQUALS SIGN character, followed by zero or more space characters, followed by the attribute value, which, in addition to the requirements given above for attribute values, must not contain any literal space characters, any U+0022 QUOTATION MARK characters ("), U+0027 APOSTROPHE characters ('), U+003D EQUALS SIGN characters (=), U+003C LESS-THAN SIGN characters (<), U+003E GREATER-THAN SIGN characters (>), or U+0060 GRAVE ACCENT characters (`), and must not be the empty string.

In the following example, the `value` attribute is given with the unquoted attribute value syntax:

```
<input value=yes>
```

If an attribute using the unquoted attribute syntax is to be followed by another attribute or by the optional U+002F SOLIDUS character (/) allowed in step 6 of the start tag syntax above, then there must be a space character separating the two.

#### Single-quoted attribute value syntax

The attribute name, followed by zero or more space characters, followed by a single U+003D EQUALS SIGN character, followed by zero or more space characters, followed by a single U+0027 APOSTROPHE character ('), followed by the attribute value, which, in addition to the requirements given above for attribute values, must not contain any literal U+0027 APOSTROPHE characters ('), and finally followed by a second single U+0027 APOSTROPHE character (').

In the following example, the `type` attribute is given with the single-quoted attribute value syntax:

```
<input type='checkbox'>
```

If an attribute using the single-quoted attribute syntax is to be followed by another attribute, then there must be a space character separating the two.

#### Double-quoted attribute value syntax

The attribute name, followed by zero or more space characters, followed by a single U+003D EQUALS SIGN character, followed by zero or more space characters, followed by a single U+0022 QUOTATION MARK character ("), followed by the attribute value, which, in addition to the requirements given above for attribute values, must not contain any literal U+0022 QUOTATION MARK characters ("), and finally followed by a second single U+0022 QUOTATION MARK character (").

In the following example, the `name` attribute is given with the double-quoted attribute value syntax:

```
<input name="be evil">
```

If an attribute using the double-quoted attribute syntax is to be followed by another attribute, then there must be a space character separating the two.

There must never be two or more attributes on the same start tag whose names are an ASCII case-insensitive match for each other.

When a foreign element has one of the namespaced attributes given by the local name and namespace of the first and second cells of a row from the following table, it must be written using the name given by the third cell from the same row.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

role	XLink namespace	xlink:role
show	XLink namespace	xlink:show
title	XLink namespace	xlink:title
type	XLink namespace	xlink:type
base	XML namespace	xml:base
lang	XML namespace	xml:lang
space	XML namespace	xml:space
xmlns	XMI NS namespace	xmle
xlink		



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

*Note: V  
and Ma  
HTML s*

*ficiations (e.g. the SVG  
rialised using the*

#### 12.1.2.4 Optional tags

Certain tags can be omitted.

*Note: Omitting an element's start tag in the situations described below does not mean the element is not present; it is implied, but it is still there. For example, an HTML document always has a root html element, even if the string <html> doesn't appear anywhere in the markup.*

An `html` element's start tag may be omitted if the first thing inside the `html` element is not a comment.

An `html` element's end tag may be omitted if the `html` element is not immediately followed by a comment.

A `head` element's start tag may be omitted if the element is empty, or if the first thing inside the `head` element is an element.

A `head` element's end tag may be omitted if the `head` element is not immediately followed by a space character or a comment.

A `body` element's start tag may be omitted if the element is empty, or if the first thing inside the `body` element is not a space character or a comment, except if the first thing inside the `body` element is a `script` or `style` element.

A `body` element's end tag may be omitted if the `body` element is not immediately followed by a comment.

An `li` element's end tag may be omitted if the `li` element is immediately followed by another `li` element or if there is no more content in the parent element.

A `dt` element's end tag may be omitted if the `dt` element is immediately followed by another `dt` element or a `dd` element.

A `dd` element's end tag may be omitted if the `dd` element is immediately followed by another `dd` element or a `dt` element, or if there is no more content in the parent element.

A `p` element's end tag may be omitted if the `p` element is immediately followed by an `address`, `article`, `aside`, `blockquote`, `dir`, `div`, `dl`, `fieldset`, `footer`, `form`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `header`, `hgroup`, `hr`, `menu`, `nav`, `ol`, `p`, `pre`, `section`, `table`, or `ul`, element, or if there is no more content in the parent element and the parent element is not an `a` element.

An `rt` element's end tag may be omitted if the `rt` element is immediately followed by an `rt` or `rp` element, or if there is no more content in the parent element.

An `rp` element's end tag may be omitted if the `rp` element is immediately followed by an `rt` or `rp` element, or if there is no more content in the parent element.

An `optgroup` element's end tag may be omitted if the `optgroup` element is immediately followed by another `optgroup` element, or if there is no more content in the parent element.

An `option` element's end tag may be omitted if the `option` element is immediately followed by another `option` element, or if it is immediately followed by an `optgroup` element, or if there is no more content in the parent element.

A `colgroup` element's start tag may be omitted if the first thing inside the `colgroup` element is a `col` element, and if the element is

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

A `thead` element's end tag may be omitted if the `thead` element is immediately followed by a `tbody` or `tfoot` element.

A `tbody` element's start tag may be omitted if the first thing inside the `tbody` element is a `tr` element, and if the element is not immediately preceded by a `tbody`, `thead`, or `tfoot` element whose end tag has been omitted. (It can't be omitted if the element is empty.)

A `tbody` element's end tag may be omitted if the `tbody` element is immediately followed by a `tbody` or `tfoot` element, or if there is no more content in the parent element.

A `tfoot` element's end tag may be omitted if the `tfoot` element is immediately followed by a `tbody` element, or if there is no more content in the parent element.

A `tr` element's end tag may be omitted if the `tr` element is immediately followed by a `tbody` element, or if there is no more content in the parent element.

A `td` element's end tag may be omitted if the `td` element is immediately followed by a `tbody` element, or if there is no more content in the parent element.

A `th` element's end tag may be omitted if the `th` element is immediately followed by a `tbody` element, or if there is no more content in the parent element.

However, a start tag must never be omitted if it has any attributes.

### 12.1.2.5 Restrictions on content models

For historical reasons, certain elements have extra restrictions beyond even the restrictions given by their content model.

A `table` element must not contain `tr` elements, even though these elements are technically allowed inside `table` elements according to the content models described in this specification. (If a `tr` element is put inside a `table` in the markup, it will in fact imply a `tbody` start tag before it.)

A single newline may be placed immediately after the start tag of `pre` and `textarea` elements. This does not affect the processing of the element. The otherwise optional newline *must* be included if the element's contents themselves start with a newline (because otherwise the leading newline in the contents would be treated like the optional newline, and ignored).

The following two `pre` blocks are equivalent:

```
<pre>Hello</pre>
<pre>
Hello</pre>
```

### 12.1.2.6 Restrictions on the contents of raw text and RCDATA elements

The text in raw text and RCDATA elements must not contain any occurrences of the string "</" (U+003C LESS-THAN SIGN, U+002F SOLIDUS) followed by characters that case-insensitively match the tag name of the element followed by one of U+0009 CHARACTER TABULATION (tab), U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), U+0020 SPACE, U+003E GREATER-THAN SIGN (>), or U+002F SOLIDUS (/).

## 12.1.3 Text

**Text** is allowed inside elements, attribute values, and comments. Extra constraints are placed on what is and what is not allowed in text based on where the text is to be put, as described in the other sections.

### 12.1.3.1 Newlines

**Newlines** in HTML may be represented either as U+000D CARRIAGE RETURN (CR) characters, U+000A LINE FEED (LF) characters, or pairs of U+000D CARRIAGE RETURN (CR), U+000A LINE FEED (LF) characters in that order.

Where character references are allowed, a character reference of a U+000A LINE FEED (LF) character (but not a U+000D CARRIAGE RETURN (CR) character) also represents a newline.

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Character references must start with a U+0026 AMPERSAND character (&). Following this, there are three possible kinds of character references:

### Named character references

The ampersand must be followed by one of the names given in the named character references section, using the same case. The name must be one that is terminated by a U+003B SEMICOLON character (;).

### Decimal numeric character reference

The ampersand must be followed by a U+0023 NUMBER SIGN character (#), followed by one or more ASCII digits, representing a base ten integer that corresponds to a Unicode code point that is allowed according to the definition below. The digit

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The numeric value must be followed by either a U+0078 semicolon, which must then be followed by a code point that is allowed according to the definition below. The character (;

other than U+0000, U+000D, permanently unassigned Unicode characters (noncharacters), and control characters other than space characters.

An **ambiguous ampersand** is a U+0026 AMPERSAND character (&) that is followed by one or more alphanumeric ASCII characters, followed by a U+003B SEMICOLON character (;), where these characters do not match any of the names given in the named character references section.

### 12.1.5 CDATA sections

**CDATA sections** must consist of the following components, in this order:

1. The string "<! [CDATA[".
2. Optionally, text, with the additional restriction that the text must not contain the string "]]>".
3. The string "]]>".

CDATA sections can only be used in foreign content (MathML or SVG). In this example, a CDATA section is used to escape the contents of an `ms` element:

```
<p>You can add a string to a number, but this stringifies the number:</p>
<math>
<ms><! [CDATA[x<y]]></ms>
<mo>+</mo>
<mn>3</mn>
<mo>=</mo>
<ms><! [CDATA[x<y3]]></ms>
</math>
```

### 12.1.6 Comments

**Comments** must start with the four character sequence U+003C LESS-THAN SIGN, U+0021 EXCLAMATION MARK, U+002D HYPHEN-MINUS, U+002D HYPHEN-MINUS (<!--). Following this sequence, the comment may have text, with the additional restriction that the text must not start with a single U+003E GREATER-THAN SIGN character (>), nor start with a U+002D HYPHEN-MINUS character (-) followed by a U+003E GREATER-THAN SIGN (>) character, nor contain two consecutive U+002D HYPHEN-MINUS characters (--), nor end with a U+002D HYPHEN-MINUS character (-). Finally, the comment must be ended by the three character sequence U+002D HYPHEN-MINUS, U+002D HYPHEN-MINUS, U+003E GREATER-THAN SIGN (-->).

## 12.2 Parsing HTML documents

This section only applies to user agents, data mining tools, and conformance checkers.

**Note:** The rules for parsing XML documents into DOM trees are covered by the next section, entitled "The XHTML syntax".

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

language with its own parsing rules.

*Some earlier versions of HTML (in particular from HTML2 to HTML4) were based on SGML and used SGML parsing rules. However, few (if any) web browsers ever implemented true SGML parsing for HTML documents; the only user agents to strictly handle HTML as an SGML application have historically been validators. The resulting confusion — with validators claiming documents to have one representation while widely deployed Web browsers interoperably implemented a different representation — has wasted decades of productivity. This version of HTML thus returns to a non-SGML basis.*

*Authors interested in using SGML tools in their authoring pipeline are encouraged to XML tools and the XML serialize*

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

This specific parsing algo described below wish to apply Conformance document at parse error condition if more than one parse error condition exists in the document. Conformance checkers are not required to recover from parse errors.

not. Certain points in the nts must either act as counter for which they do not

conditions exist in the s may report more than one

**Note:** Parse errors are only errors with the syntax of HTML. In addition to checking for parse errors, conformance checkers will also verify that the document obeys all the other conformance requirements described in this specification.

For the purposes of conformance checkers, if a resource is determined to be in the HTML syntax, then it is an HTML document.

### 12.2.1 Overview of the parsing model

The input to the HTML parsing process consists of a stream of Unicode code points, which is passed through a tokenization stage followed by a tree construction stage. The output is a `Document` object.

**Note:** Implementations that do not support scripting do not have to actually create a DOM `Document` object, but the DOM tree in such cases is still used as the model for the rest of the specification.

In the common case, the data handled by the tokenization stage comes from the network, but it can also come from script running in the user agent, e.g. using the `document.write()` API.

There is only one set of states for the tokenizer stage and the tree construction stage, but the tree construction stage is reentrant, meaning that while the tree construction stage is handling one token, the tokenizer might be resumed, causing further tokens to be emitted and processed before the first token's processing is complete.

In the following example, the tree construction stage will be called upon to handle a "p" start tag token while handling the "script" end tag token:

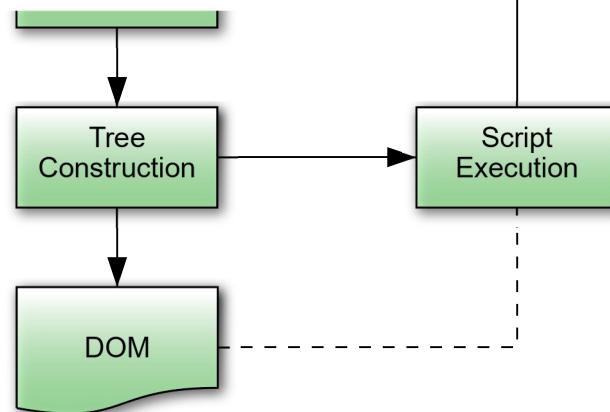
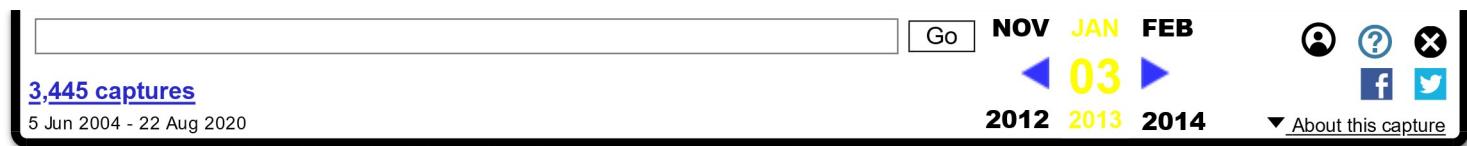
```
...
<script>
 document.write('<p>');
</script>
...
```

To handle these cases, parsers have a **script nesting level**, which must be initially set to zero, and a **parser pause flag**, which must be initially set to false.

### 12.2.2 The input byte stream

The stream of Unicode code points that comprises the input to the tokenization stage will be initially seen by the user agent as a stream of bytes (typically coming over the network or from the local file system). The bytes encode the actual characters according to a particular **character encoding**, which the user agent must use to decode the bytes into characters.

**Note:** For XML documents, the algorithm user agents must use to determine the character encoding is given by the



Any byte or sequence of bytes in the original byte stream that is misinterpreted for compatibility is a parse error.

#### 12.2.2.1 Determining the character encoding

In some cases, it might be impractical to unambiguously determine the encoding before parsing the document. Because of this, this specification provides for a two-pass mechanism with an optional pre-scan. Implementations are allowed, as described below, to apply a simplified parsing algorithm to whatever bytes they have available before beginning to parse the document. Then, the real parser is started, using a tentative encoding derived from this pre-parse and other out-of-band metadata. If, while the document is being loaded, the user agent discovers a character encoding declaration that conflicts with this information, then the parser can get reinvoked to perform a parse of the document with the real encoding.

User agents must use the following algorithm, called the **encoding sniffing algorithm**, to determine the character encoding to use when decoding a document in the first pass. This algorithm takes as input any out-of-band metadata available to the user agent (e.g. the Content-Type metadata of the document) and all the bytes available so far, and returns a character encoding and a **confidence**. The confidence is either *tentative*, *certain*, or *irrelevant*. The encoding used, and whether the confidence in that encoding is *tentative* or *certain*, is used during the parsing to determine whether to change the encoding. If no encoding is necessary, e.g. because the parser is operating on a Unicode stream and doesn't have to use a character encoding at all, then the confidence is *irrelevant*.

1. If the user has explicitly instructed the user agent to override the document's character encoding with a specific encoding, optionally return that encoding with the confidence *certain* and abort these steps.

**Note:** Typically, user agents remember such user requests across sessions, and in some cases apply them to documents in iframes as well.

2. The user agent may wait for more bytes of the resource to be available, either in this step or at any later step in this algorithm. For instance, a user agent might wait 500ms or 1024 bytes, whichever came first. In general preparing the source to find the encoding improves performance, as it reduces the need to throw away the data structures used when parsing upon finding the encoding information. However, if the user agent delays too long to obtain data to determine the encoding, then the cost of the delay could outweigh any performance improvements from the preparse.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

3. For each of the rows in the following table, starting with the first one and going down, if there are as many or more bytes available than the number of bytes in the first column, and the first bytes of the file match the bytes given in the first column, then return the encoding given in the cell in the second column of that row, with the confidence *certain*, and abort these steps:

Bytes in Hexadecimal	Encoding
FE FF	Big-endian UTF-16
FF FE	Little-endian UTF-16
EF BB BF	UTF-8



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

*Header is a willful  
with legacy content.*

4. If the algorithm aborts, with the confidence *certain*, and

5. Optionally prescan the byte stream to determine its encoding. The *end condition* is that the user agent decides that scanning further bytes would not be efficient. User agents are encouraged to only prescan the first 1024 bytes. User agents may decide that scanning *any* bytes is not efficient, in which case these substeps are entirely skipped.

The aforementioned algorithm either aborts unsuccessfully or returns a character encoding. If it returns a character encoding, then this algorithm must be aborted, returning the same encoding, with confidence *tentative*.

6. If the HTML parser for which this algorithm is being run is associated with a `Document` that is itself in a nested browsing context, run these substeps:

1. Let *new document* be the `Document` with which the HTML parser is associated.
2. Let *parent document* be the `Document` through which *new document* is nested (the active document of the parent browsing context of *new document*).
3. If *parent document*'s origin is not the same origin as *new document*'s origin, then abort these substeps.
4. If *parent document*'s character encoding is not an ASCII-compatible character encoding, then abort these substeps.
5. Return *parent document*'s character encoding, with the confidence *tentative*, and abort the encoding sniffing algorithm's steps.

7. Otherwise, if the user agent has information on the likely encoding for this page, e.g. based on the encoding of the page when it was last visited, then return that encoding, with the confidence *tentative*, and abort these steps.

8. The user agent may attempt to autodetect the character encoding from applying frequency analysis or other algorithms to the data stream. Such algorithms may use information about the resource other than the resource's contents, including the address of the resource. If autodetection succeeds in determining a character encoding, and that encoding is a supported encoding, then return that encoding, with the confidence *tentative*, and abort these steps. [UNIVCHARDET]

**Note:** The UTF-8 encoding has a highly detectable bit pattern. Documents that contain bytes with values greater than 0x7F which match the UTF-8 pattern are very likely to be UTF-8, while documents with byte sequences that do not match it are very likely not. User-agents are therefore encouraged to search for this common encoding. [PPUTF8] [UTF8DET]

9. Otherwise, return an implementation-defined or user-specified default character encoding, with the confidence *tentative*.

In controlled environments or in environments where the encoding of documents can be prescribed (for example, for user agents intended for dedicated use in new networks), the comprehensive `UTF-8` encoding is suggested.

In other environments, the default encoding is typically dependent on the user's locale (an approximation of the languages, and thus often encodings, of the pages that the user is likely to frequent). The following table gives suggested defaults based on the user's locale, for compatibility with legacy content. Locales are identified by BCP 47 language tags. [BCP47]

Locale language	Suggested default encoding
ar	UTF-8

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

cy	UTF-8
fa	UTF-8
he	windows-1255
hr	UTF-8
hu	ISO-8859-2
ja	Windows-31J
kk	UTF-8
ko	windows-949



ku

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

sk	windows-1255
sl	ISO-8859-2
sr	UTF-8
th	windows-874
tr	windows-1254
uk	windows-1251
vi	UTF-8
zh-CN	GB18030
zh-TW	Big5
All other locales	windows-1252

The document's character encoding must immediately be set to the value returned from this algorithm, at the same time as the user agent uses the returned value to select the decoder to use for the input byte stream.

When an algorithm requires a user agent to **prescan a byte stream to determine its encoding**, given some defined *end condition*, then it must run the following steps. These steps either abort unsuccessfully or return a character encoding.

1. Let *position* be a pointer to a byte in the input byte stream, initially pointing at the first byte. If at any point during these steps the user agent either runs out of bytes or reaches its *end condition*, then abort the prescan a byte stream to determine its encoding algorithm unsuccessfully.

2. *Loop*: If *position* points to:

↳ A sequence of bytes starting with: 0x3C 0x21 0x2D 0x2D (ASCII '<!--')

Advance the *position* pointer so that it points at the first 0x3E byte which is preceded by two 0x2D bytes (i.e. at the end of an ASCII '--> sequence) and comes after the 0x3C byte that was found. (The two 0x2D bytes can be the same as the those in the '<!--' sequence.)

↳ A sequence of bytes starting with: 0x3C, 0x4D or 0x6D, 0x45 or 0x65, 0x54 or 0x74, 0x41 or 0x61, and one of 0x09, 0x0A, 0x0C, 0x0D, 0x20, 0x2F (case-insensitive ASCII '<meta' followed by a space or slash)

1. Advance the *position* pointer so that it points at the next 0x09, 0x0A, 0x0C, 0x0D, 0x20, or 0x2F byte (the one in sequence of characters matched above).

2. Let *attribute list* be an empty list of strings.

3. Let *got pragma* be false.

4. Let *need pragma* be null.

5. Let *charset* be the null value (which, for the purposes of this algorithm, is distinct from an unrecognised encoding or the empty string).

6. *Attributes*: Get an attribute and its value. If no attribute was sniffed, then jump to the *processing* step below.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## ↳ If the attribute's name is "http-equiv"

If the attribute's value is "content-type", then set *got pragma* to true.

## ↳ If the attribute's name is "content"

Apply the algorithm for extracting a character encoding from a `meta` element, giving the attribute's value as the string to parse. If a character encoding is returned, and if *charset* is still set to null, let *charset* be the encoding returned, and set *need pragma* to true.

## ↳ If the attribute's name is "charset"

'ue, and set *need pragma* to

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

byte.

labeled *next byte*.14. If *charset* is not a supported character encoding, then jump to the step below labeled *next byte*.15. Abort the prescan a byte stream to determine its encoding algorithm, returning the encoding given by *charset*.

## ↳ A sequence of bytes starting with a 0x3C byte (ASCII &lt;), optionally a 0x2F byte (ASCII /), and finally a byte in the range 0x41-0x5A or 0x61-0x7A (an ASCII letter)

1. Advance the *position* pointer so that it points at the next 0x09 (ASCII TAB), 0x0A (ASCII LF), 0x0C (ASCII FF), 0x0D (ASCII CR), 0x20 (ASCII space), or 0x3E (ASCII >) byte.2. Repeatedly get an attribute until no further attributes can be found, then jump to the step below labeled *next byte*.

## ↳ A sequence of bytes starting with: 0x3C 0x21 (ASCII '&lt;!')

## ↳ A sequence of bytes starting with: 0x3C 0x2F (ASCII '&lt;/')

## ↳ A sequence of bytes starting with: 0x3C 0x3F (ASCII '&lt;?')

Advance the *position* pointer so that it points at the first 0x3E byte (ASCII >) that comes after the 0x3C byte that was found.

## ↳ Any other byte

Do nothing with that byte.

3. *Next byte*: Move *position* so it points at the next byte in the input byte stream, and return to the step above labeled *loop*.When the prescan a byte stream to determine its encoding algorithm says to **get an attribute**, it means doing this:1. If the byte at *position* is one of 0x09 (ASCII TAB), 0x0A (ASCII LF), 0x0C (ASCII FF), 0x0D (ASCII CR), 0x20 (ASCII space), or 0x2F (ASCII /) then advance *position* to the next byte and redo this step.2. If the byte at *position* is 0x3E (ASCII >), then abort the get an attribute algorithm. There isn't one.3. Otherwise, the byte at *position* is the start of the attribute name. Let *attribute name* and *attribute value* be the empty string.4. Process the byte at *position* as follows:↳ If it is 0x3D (ASCII =), and the *attribute name* is longer than the empty stringAdvance *position* to the next byte and jump to the step below labeled *value*.

## ↳ If it is 0x09 (ASCII TAB), 0x0A (ASCII LF), 0x0C (ASCII FF), 0x0D (ASCII CR), or 0x20 (ASCII space)

Jump to the step below labeled *spaces*.

## ↳ If it is 0x2F (ASCII /) or 0x3E (ASCII &gt;)

Abort the get an attribute algorithm. The attribute's name is the value of *attribute name*, its value is the empty string.

## ↳ If it is in the range 0x41 (ASCII A) to 0x5A (ASCII Z)

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

doesn't actually matter how bytes outside the ASCII range are handled here, since only ASCII characters can contribute to the detection of a character encoding.)

5. Advance *position* to the next byte and return to the previous step.
6. **Spaces:** If the byte at *position* is one of 0x09 (ASCII TAB), 0x0A (ASCII LF), 0x0C (ASCII FF), 0x0D (ASCII CR), or 0x20 (ASCII space) then advance *position* to the next byte, then, repeat this step.
7. If the byte at *position* is not 0x3D (ASCII =), abort the get an attribute algorithm. The attribute's name is the value of *attribute name*, its value is the empty string

## 8. Adv Too slow? Try reading the multipage copy of

9. *Value* the spec instead: <https://web.archive.org> (ASCII CR), or 0x20 (ASCII

- spa
10. Pro /web/20130103030411/http:  
↳ //whatwg.org/html

1. Let *b* be the value of the byte at *position*.

2. **Quote loop:** Advance *position* to the next byte.

3. If the value of the byte at *position* is the value of *b*, then advance *position* to the next byte and abort the "get an attribute" algorithm. The attribute's name is the value of *attribute name*, and its value is the value of *attribute value*.

4. Otherwise, if the value of the byte at *position* is in the range 0x41 (ASCII A) to 0x5A (ASCII Z), then append a Unicode character to *attribute value* whose code point is 0x20 more than the value of the byte at *position*.

5. Otherwise, append a Unicode character to *attribute value* whose code point is the same as the value of the byte at *position*.

6. Return to the step above labeled *quote loop*.

### ↳ If it is 0x3E (ASCII >)

Abort the get an attribute algorithm. The attribute's name is the value of *attribute name*, its value is the empty string.

### ↳ If it is in the range 0x41 (ASCII A) to 0x5A (ASCII Z)

Append the Unicode character with code point *b*+0x20 to *attribute value* (where *b* is the value of the byte at *position*). Advance *position* to the next byte.

### ↳ Anything else

Append the Unicode character with the same code point as the value of the byte at *position*) to *attribute value*. Advance *position* to the next byte.

11. Process the byte at *position* as follows:

### ↳ If it is 0x09 (ASCII TAB), 0x0A (ASCII LF), 0x0C (ASCII FF), 0x0D (ASCII CR), 0x20 (ASCII space), or 0x3E (ASCII >)

Abort the get an attribute algorithm. The attribute's name is the value of *attribute name* and its value is the value of *attribute value*.

### ↳ If it is in the range 0x41 (ASCII A) to 0x5A (ASCII Z)

Append the Unicode character with code point *b*+0x20 to *attribute value* (where *b* is the value of the byte at *position*).

### ↳ Anything else

Append the Unicode character with the same code point as the value of the byte at *position*) to *attribute value*.

12. Advance *position* to the next byte and return to the previous step.

For the sake of interoperability, user agents should not use a pre-scan algorithm that returns different results than the one described above. (But, if you do, please at least let us know, so that we can improve this algorithm and benefit everyone...)

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

User agents must support the preferred MIME name of every character encoding they support, and should support all the IANA-registered names and aliases of every character encoding they support. [IANACHARSET]

When comparing a string specifying a character encoding with the name or alias of a character encoding to determine if they are equal, user agents must remove any leading or trailing space characters in both names, and then perform the comparison in an ASCII case-insensitive manner.

When a user agent would otherwise use a character encoding given in the first column of the following table to either convert content to Unicode characters in the second column of the same row, it must instead use the character encoding given in the third column of the same row. This is to have been misinterpreted.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

Input encoding: /web/20130103030411/http://whatwg.org/html  
 EUC-KR  
 EUC-JP  
 GB2312

GB_2312-80	GBK	[RFC1345] [GBK]
ISO-2022-JP	CP50220	[RFC1468] [RFC2237] [RFC1554] [CP50220]
ISO-8859-1	windows-1252	[RFC1345] [WIN1252]
ISO-8859-9	windows-1254	[RFC1345] [WIN1254]
ISO-8859-11	windows-874	[ISO885911] [WIN874]
KS_C_5601-1987	windows-949	[RFC1345] [WIN949]
Shift_JIS	Windows-31J	[SHIFTJIS] [WIN31J]
TIS-620	windows-874	[TIS620] [WIN874]
US-ASCII	windows-1252	[RFC1345] [WIN1252]

**Note: The requirement to treat certain encodings as other encodings according to the table above is a willful violation of the W3C Character Model specification, motivated by a desire for compatibility with legacy content. [CHARMOD]**

When a user agent is to use the self-describing UTF-16 encoding but no BOM has been found, user agents must default to little-endian UTF-16.

**Note: The requirement to default UTF-16 to little-endian rather than big-endian is a willful violation of RFC 2781, motivated by a desire for compatibility with legacy content. [RFC2781]**

User agents must not support the CESU-8, UTF-7, BOCU-1 and SCSU encodings. [CESU8] [UTF7] [BOCU1] [SCSU]

Support for encodings based on EBCDIC is discouraged. This encoding is rarely used for publicly-facing Web content.

Support for UTF-32 is also discouraged. This encoding is rarely used, and frequently implemented incorrectly.

**Note: This specification does not make any attempt to support EBCDIC-based encodings and UTF-32 in its algorithms; support and use of these encodings can thus lead to unexpected behavior in implementations of this specification.**

### 12.2.2.3 Changing the encoding while parsing

When the parser requires the user agent to **change the encoding**, it must run the following steps. This might happen if the encoding sniffing algorithm described above failed to find a character encoding, or if it found a character encoding that was not the actual encoding of the file.

1. If the encoding that is already being used to interpret the input stream is a UTF-16 encoding, then set the confidence to *certain* and abort these steps. The new encoding is ignored; if it was anything but the same encoding, then it would be clearly incorrect.
2. If the new encoding is a UTF-16 encoding, change it to UTF-8.
3. If the new encoding is identical or equivalent to the encoding that is already being used to interpret the input stream, then set

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

encoding and the new encoding, and if the user agent supports changing the converter on the fly, then the user agent may change to the new converter for the encoding on the fly. Set the document's character encoding and the encoding used to convert the input stream to the new encoding, set the confidence to *certain*, and abort these steps.

5. Otherwise, navigate to the document again, with replacement enabled, and using the same source browsing context, but this time skip the encoding sniffing algorithm and instead just set the encoding to the new encoding and the confidence to *certain*. Whenever possible, this should be done without actually contacting the network layer (the bytes should be re-parsed from memory), even if, e.g., the document is marked as not being cacheable. If this is not possible and contacting the network layer would involve repeating a request that uses a method other than HTTP GET (or equivalent for non-HTTP URLs), then instead set the

'User agents may notify the

use **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org**

#### 12.2.2.4 P

The **input** step manipulates the **input** stream. One leading

various APIs that directly manipulate the **input** stream.

**Note:** The requirement to strip a U+FEFF BYTE ORDER MARK character regardless of whether that character was used to determine the byte order is a willful violation of Unicode, motivated by a desire to increase the resilience of user agents in the face of naïve transcoders.

Any occurrences of any characters in the ranges U+0001 to U+0008, U+000E to U+001F, U+007F to U+009F, U+FDD0 to U+FDEF, and characters U+000B, U+FFFF, U+1FFF, U+1FFF, U+2FFF, U+2FFF, U+3FFF, U+3FFF, U+4FFF, U+4FFF, U+5FFF, U+5FFF, U+6FFF, U+6FFF, U+7FFF, U+7FFF, U+8FFF, U+8FFF, U+9FFF, U+9FFF, U+AFFF, U+AFFF, U+BFFF, U+BFFF, U+CFFF, U+CFFF, U+DFFF, U+DFFF, U+EFFF, U+EFFF, U+FFFF, U+FFFF, U+10FFF, and U+10FFF are parse errors. These are all control characters or permanently undefined Unicode characters (noncharacters).

U+000D CARRIAGE RETURN (CR) characters and U+000A LINE FEED (LF) characters are treated specially. All CR characters must be converted to LF characters, and any LF characters that immediately follow a CR character must be ignored. Thus, newlines in HTML DOMs are represented by LF characters, and there are never any CR characters in the input to the tokenization stage.

The **next input character** is the first character in the input stream that has not yet been **consumed** or explicitly ignored by the requirements in this section. Initially, the **next input character** is the first character in the input. The **current input character** is the last character to have been **consumed**.

The **insertion point** is the position (just before a character or just before the end of the input stream) where content inserted using `document.write()` is actually inserted. The insertion point is relative to the position of the character immediately after it; it is not an absolute offset into the input stream. Initially, the insertion point is undefined.

The "EOF" character in the tables below is a conceptual character representing the end of the input stream. If the parser is a script-created parser, then the end of the input stream is reached when an **explicit "EOF" character** (inserted by the `document.close()` method) is consumed. Otherwise, the "EOF" character is not a real character in the stream, but rather the lack of any further characters.

**Note:** The handling of U+0000 NULL characters varies based on where the characters are found. In general, they are ignored except where doing so could plausibly introduce an attack vector. This handling is, by necessity, spread across both the tokenization stage and the tree construction stage.

### 12.2.3 Parse state

#### 12.2.3.1 The insertion mode

The **insertion mode** is a state variable that controls the primary operation of the tree construction stage.

Initially, the insertion mode is "initial". It can change to "before html", "before head", "in head", "in head noscript", "after head", "in body", "text", "in table", "in table text", "in caption", "in column group", "in table body", "in row", "in cell", "in select", "in select in table", "after body", "in frameset", "after frameset", "after after body", and "after after frameset" during the course of the parsing, as described in the tree construction stage. The insertion mode affects how tokens are processed and whether CDATA sections are supported.

Several of these modes, namely "in head", "in body", "in table", and "in select", are special, in that the other modes defer to them at various times. When the algorithm below says that the user agent is to do something "**using the rules for the m insertion mode**",

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

When the steps below require the UA to **reset the insertion mode appropriately**, it means the UA must follow these steps:

1. Let *last* be false.
2. Let *node* be the last node in the stack of open elements.
3. *Loop*: If *node* is the first node in the stack of open elements, then set *last* to true and set *node* to the *context* element. (fragment case)
4. If *nc* **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html** 'fragment case)
5. If *nc* these steps.
6. If *nc*
7. If *nc* id abort these steps.
8. If *nc* 3.
9. If *node* is a `colgroup` element, then switch the insertion mode to "in column group" and abort these steps. (fragment case)
10. If *node* is a `table` element, then switch the insertion mode to "in table" and abort these steps.
11. If *node* is a `head` element, then switch the insertion mode to "in body" ("in body"! *not* "in head"!) and abort these steps. (fragment case)
12. If *node* is a `body` element, then switch the insertion mode to "in body" and abort these steps.
13. If *node* is a `frameset` element, then switch the insertion mode to "in frameset" and abort these steps. (fragment case)
14. If *node* is an `html` element, then switch the insertion mode to "before head" Then, abort these steps. (fragment case)
15. If *last* is true, then switch the insertion mode to "in body" and abort these steps. (fragment case)
16. Let *node* now be the node before *node* in the stack of open elements.
17. Return to the step labeled *loop*.

### 12.2.3.2 The stack of open elements

Initially, the **stack of open elements** is empty. The stack grows downwards; the topmost node on the stack is the first one added to the stack, and the bottommost node of the stack is the most recently added node in the stack (notwithstanding when the stack is manipulated in a random access fashion as part of the handling for misnested tags).

The "before html" insertion mode creates the `html` root element node, which is then added to the stack.

In the fragment case, the stack of open elements is initialized to contain an `html` element that is created as part of that algorithm. (The fragment case skips the "before html" insertion mode.)

The `html` node, however it is created, is the topmost node of the stack. It only gets popped off the stack when the parser finishes.

The **current node** is the bottommost node in this stack.

Elements in the stack fall into the following categories:

#### Special

The following elements have varying levels of special parsing rules: HTML's `address`, `applet`, `area`, `article`, `aside`, `base`, `basefont`, `bgsound`, `blockquote`, `body`, `br`, `button`, `caption`, `center`, `col`, `colgroup`, `menuitem`, `dd`, `details`, `dir`, `div`, `dl`, `dt`, `embed`, `fieldset`, `figcaption`, `figure`, `footer`, `form`, `frame`, `frameset`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `head`, `header`, `hgroup`, `hr`, `html`, `iframe`, `img`, `input`, `isindex`, `li`, `link`, `listing`, `marquee`, `menu`, `meta`, `nav`, `noembed`, `noframes`, `noscript`, `object`, `ol`, `p`, `param`, `plaintext`, `pre`, `script`, `section`, `select`, `source`, `style`, `summary`, `table`, `tbody`, `td`, `textarea`, `tfoot`, `th`, `thead`, `title`, `tr`, `track`, `ul`, `wbr`, and `xmp`; MathML's `mi`, `mo`, `mn`, `ms`, `mtext`, and `annotation-xml`; and SVG's `foreignObject`, `desc`, and `title`.

#### Formatting

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The stack of open elements is said to **have an element in a specific scope** consisting of a list of element types *list* when the following algorithm terminates in a match state:

1. Initialize *node* to be the current node (the bottommost node of the stack).
2. If *node* is the target node, terminate in a match state.
3. Otherwise, if *node* is one of the element types in *list*, terminate in a failure state.
4. Otherwise, set *node* to its parent node. If *node* has no parent node, terminate in a failure state.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The stack of open elements is said to **have an element in list item scope** when it has an element in the specific scope consisting of the following element types:

- `app`
- `cap`
- `htr`
- `tab`
- `td` in the `HTML` namespace
- `th` in the `HTML` namespace
- `marquee` in the `HTML` namespace
- `object` in the `HTML` namespace
- `mi` in the `MathML` namespace
- `mo` in the `MathML` namespace
- `mn` in the `MathML` namespace
- `ms` in the `MathML` namespace
- `mtext` in the `MathML` namespace
- `annotation-xml` in the `MathML` namespace
- `foreignObject` in the `SVG` namespace
- `desc` in the `SVG` namespace
- `title` in the `SVG` namespace

The stack of open elements is said to **have an element in list item scope** when it has an element in the specific scope consisting of the following element types:

- All the element types listed above for the *has an element in scope* algorithm.
- `ol` in the `HTML` namespace
- `ul` in the `HTML` namespace

The stack of open elements is said to **have an element in button scope** when it has an element in the specific scope consisting of the following element types:

- All the element types listed above for the *has an element in scope* algorithm.
- `button` in the `HTML` namespace

The stack of open elements is said to **have an element in table scope** when it has an element in the specific scope consisting of the following element types:

- `html` in the `HTML` namespace
- `table` in the `HTML` namespace

The stack of open elements is said to **have an element in select scope** when it has an element in the specific scope consisting of all element types *except* the following:

- `optgroup` in the `HTML` namespace
- `option` in the `HTML` namespace

Nothing happens if at any time any of the elements in the stack of open elements are moved to a new location in, or removed from, the Document tree. In particular, the stack is not changed in this situation. This can cause, amongst other strange effects, content to be appended to nodes that are no longer in the DOM.

**Note:** In some cases (namely, when closing misnested formatting elements), the stack is manipulated in a random-access fashion.

### 12.2.3.3 The list of active formatting elements

Initially, the **list of active formatting elements** is empty. It is used to handle mis-nested formatting element tags.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*Note. The scope markers are unrelated to the concept of an element being in scope.*

In addition, each element in the list of active formatting elements is associated with the token for which it was created, so that further elements can be created for that token if necessary.

When the steps below require the UA to **push onto the list of active formatting elements** an element *element*, the UA must perform the following steps:

1. If there are already three elements in the list of active formatting elements after the last list marker, if any, or anywhere in the list if there are no list markers, that have the same tag name, namespace, and attributes as *element*, then remove the earliest such element. The compared as they were parsed attributes can be in the order of the attributes does not matter.
- Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**
2. Add *element* to the list of active formatting elements.

When the steps below require the UA to **reconstruct the active formatting elements**, the UA must perform the following steps:

1. If there are no entries in the list of active formatting elements, then there is nothing to reconstruct; stop this algorithm.
2. If the last (most recently added) entry in the list of active formatting elements is a marker, or if it is an element that is in the stack of open elements, then there is nothing to reconstruct; stop this algorithm.
3. Let *entry* be the last (most recently added) element in the list of active formatting elements.
4. If there are no entries before *entry* in the list of active formatting elements, then jump to step 8.
5. Let *entry* be the entry one earlier than *entry* in the list of active formatting elements.
6. If *entry* is neither a marker nor an element that is also in the stack of open elements, go to step 4.
7. Let *entry* be the element one later than *entry* in the list of active formatting elements.
8. Create an element for the token for which the element *entry* was created, to obtain *new element*.
9. Append *new element* to the current node and push it onto the stack of open elements so that it is the new current node.
10. Replace the entry for *entry* in the list with an entry for *new element*.
11. If the entry for *new element* in the list of active formatting elements is not the last entry in the list, return to step 7.

This has the effect of reopening all the formatting elements that were opened in the current body, cell, or caption (whichever is youngest) that haven't been explicitly closed.

**Note:** *The way this specification is written, the list of active formatting elements always consists of elements in chronological order with the least recently added element first and the most recently added element last (except for while steps 8 to 11 of the above algorithm are being executed, of course).*

When the steps below require the UA to **clear the list of active formatting elements up to the last marker**, the UA must perform the following steps:

1. Let *entry* be the last (most recently added) entry in the list of active formatting elements.
2. Remove *entry* from the list of active formatting elements.
3. If *entry* was a marker, then stop the algorithm at this point. The list has been cleared up to the last marker.
4. Go to step 1.

#### 12.2.3.4 The element pointers

Initially, the `head` element pointer and the `form` element pointer are both null.

Once a `head` element has been parsed (whether implicitly or explicitly) the `head` element pointer gets set to point to this node.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The **scripting flag** is set to "enabled" if scripting was enabled for the Document with which the parser is associated when the parser was created, and "disabled" otherwise.

**Note:** The scripting flag can be enabled even when the parser was originally created for the HTML fragment parsing algorithm, even though `script` elements don't execute in that case.

The **frameset-ok flag** is set to "ok" when the parser is created. It is set to "not ok" after certain tokens are seen.



## 12.2.4 To Too slow? Try reading the multipage copy of

Implemental  
Most states  
*reconsume*  
next charact  
some cases

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

must start in the data state.  
machine to a new state to  
same state to consume the  
switching to another state. In

The exact behavior depends on the state machine. Certain states also use a **temporary buffer** to track progress.

The output of the tokenization step is a series of zero or more of the following tokens: DOCTYPE, start tag, end tag, comment, character, end-of-file. DOCTYPE tokens have a name, a public identifier, a system identifier, and a **force-quirks flag**. When a DOCTYPE token is created, its name, public identifier, and system identifier must be marked as missing (which is a distinct state from the empty string), and the **force-quirks flag** must be set to *off* (its other state is *on*). Start and end tag tokens have a tag name, a **self-closing flag**, and a list of attributes, each of which has a name and a value. When a start or end tag token is created, its **self-closing flag** must be unset (its other state is that it be set), and its attributes list must be empty. Comment and character tokens have data.

When a token is emitted, it must immediately be handled by the tree construction stage. The tree construction stage can affect the state of the tokenization stage, and can insert additional characters into the stream. (For example, the `script` element can result in scripts executing and using the dynamic markup insertion APIs to insert characters into the stream being tokenized.)

When a start tag token is emitted with its **self-closing flag** set, if the flag is not **acknowledged** when it is processed by the tree construction stage, that is a parse error.

When an end tag token is emitted with attributes, that is a parse error.

When an end tag token is emitted with its **self-closing flag** set, that is a parse error.

An **appropriate end tag token** is an end tag token whose tag name matches the tag name of the last start tag to have been emitted from this tokenizer, if any. If no start tag has been emitted from this tokenizer, then no end tag token is appropriate.

Before each step of the tokenizer, the user agent must first check the parser pause flag. If it is true, then the tokenizer must abort the processing of any nested invocations of the tokenizer, yielding control back to the caller.

The tokenizer state machine consists of the states defined in the following subsections.

### 12.2.4.1 Data state

Consume the next input character:

↳ U+0026 AMPERSAND (&)

Switch to the character reference in data state.

↳ U+003C LESS-THAN SIGN (<)

Switch to the tag open state.

↳ U+0000 NULL

Parse error. Emit the current input character as a character token.

↳ EOF

Emit an end-of-file token.

↳ Anything else

Emit the current input character as a character token.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If nothing is returned, emit a U+0026 AMPERSAND character (&) token.

Otherwise, emit the character tokens that were returned.

#### 12.2.4.3 RCDATA state

Consume the next input character:

↳ U+0026 AMPERSAND (&)



↳ U+003C Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ U+0000 /

↳ EOF //whatwg.org/html

E.....

↳ Anything else

Emit the current input character as a character token.

#### 12.2.4.4 Character reference in RCDATA state

Switch to the RCDATA state.

Attempt to consume a character reference, with no additional allowed character.

If nothing is returned, emit a U+0026 AMPERSAND character (&) token.

Otherwise, emit the character tokens that were returned.

#### 12.2.4.5 RAWTEXT state

Consume the next input character:

↳ U+003C LESS-THAN SIGN (<)

Switch to the RAWTEXT less-than sign state.

↳ U+0000 NULL

Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↳ EOF

Emit an end-of-file token.

↳ Anything else

Emit the current input character as a character token.

#### 12.2.4.6 Script data state

Consume the next input character:

↳ U+003C LESS-THAN SIGN (<)

Switch to the script data less-than sign state.

↳ U+0000 NULL

Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↳ EOF

Emit an end-of-file token.

↳ Anything else

Emit the current input character as a character token.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↳ EOF

Emit an end-of-file token.

↳ Anything else

Emit the current input character as a character token.

#### 12.2.4.8 Tag open state



Consume the next input character:

↳ U+0021

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

↳ U+002F

/web/20130103030411/http:

↳ Uppercase ASCII letter

Create a new start tag token, set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point), then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ Lowercase ASCII letter

Create a new start tag token, set its tag name to the current input character, then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ U+003F QUESTION MARK (?)

Parse error. Switch to the bogus comment state.

↳ Anything else

Parse error. Switch to the data state. Emit a U+003C LESS-Than SIGN character token. Reconsume the current input character.

#### 12.2.4.9 End tag open state

Consume the next input character:

↳ Uppercase ASCII letter

Create a new end tag token, set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point), then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ Lowercase ASCII letter

Create a new end tag token, set its tag name to the current input character, then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ U+003E GREATER-Than SIGN (&gt;)

Parse error. Switch to the data state.

↳ EOF

Parse error. Switch to the data state. Emit a U+003C LESS-Than SIGN character token and a U+002F SOLIDUS character token. Reconsume the EOF character.

↳ Anything else

Parse error. Switch to the bogus comment state.

#### 12.2.4.10 Tag name state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

→ U+003E GREATER-THAN SIGN (&gt;)

Switch to the data state. Emit the current tag token.

↳ Uppercase ASCII letter

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name.

↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current tag token's tag name.



↳ EOF

Too slow? Try reading the multipage copy of

↳ Anythin

the spec instead: <https://web.archive.org>

/web/20130103030411/http:

#### 12.2.4.11

//whatwg.org/html

Consume the

↳ U+002F SOLIDUS (/)

Set the *temporary buffer* to the empty string. Switch to the RCDATA end tag open state.

↳ Anything else

Switch to the RCDATA state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

#### 12.2.4.12 RCDATA end tag open state

Consume the next input character:

↳ Uppercase ASCII letter

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the RCDATA end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ Lowercase ASCII letter

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the RCDATA end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ Anything else

Switch to the RCDATA state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

#### 12.2.4.13 RCDATA end tag name state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

↳ U+002F SOLIDUS (/)

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

↳ U+003E GREATER-THAN SIGN (&gt;)

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

↳ Uppercase ASCII letter

[Go](#)

NOV JAN FEB

03

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*temporary buffer.*

## ↳ Anything else

Switch to the RCDATA state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

**12.2.4.14 RAWTEXT less-than sign state**

Consume the next input character:

## ↳ U+002F

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

## ↳ Anythin

/web/20130103030411/http://whatwg.org/html

ne the current input

**12.2.4.15 RAWTEXT end tag open state**

Consume the next input character:

## ↳ Uppercase ASCII letter

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the RAWTEXT end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

## ↳ Lowercase ASCII letter

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the RAWTEXT end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

## ↳ Anything else

Switch to the RAWTEXT state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

**12.2.4.16 RAWTEXT end tag name state**

Consume the next input character:

## ↳ U+0009 CHARACTER TABULATION (tab)

## ↳ U+000A LINE FEED (LF)

## ↳ U+000C FORM FEED (FF)

## ↳ U+0020 SPACE

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

## ↳ U+002F SOLIDUS (/)

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

## ↳ U+003E GREATER-THAN SIGN (&gt;)

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

## ↳ Uppercase ASCII letter

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name. Append the current input character to the *temporary buffer*.

## ↳ Lowercase ASCII letter

Append the current input character to the current tag token's tag name. Append the current input character to the *temporary buffer*.

## ↳ Anything else

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

#### 12.2.4.17 Script data less-than sign state

Consume the next input character:

↳ **U+002F SOLIDUS (/)**

Set the *temporary buffer* to the empty string. Switch to the script data end tag open state.

↳ **U+0021 EXCLAMATION MARK (!)**

Switch to the script data escape start state. Emit a U+003C LESS-THAN SIGN character token and a U+0021 EXCLAMATION MARK character token.

↳ **Anything else?** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

#### 12.2.4.18

Consume the next input character:

↳ **Uppercase ASCII letter**

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the script data end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **Lowercase ASCII letter**

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the script data end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **Anything else**

Switch to the script data state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

#### 12.2.4.19 Script data end tag name state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**

↳ **U+000A LINE FEED (LF)**

↳ **U+000C FORM FEED (FF)**

↳ **U+0020 SPACE**

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

↳ **U+002F SOLIDUS (/)**

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

↳ **U+003E GREATER-THAN SIGN (>)**

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

↳ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↳ **Lowercase ASCII letter**

Append the current input character to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↳ **Anything else**

Switch to the script data state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Switch to the script data escape start dash state. Emit a U+002D HYPHEN-MINUS character token.

## ↳ Anything else

Switch to the script data state. Reconsume the current input character.

**12.2.4.21 Script data escape start dash state**

Consume the next input character:



## ↳ U+002D (-)

Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

## ↳ Anythin

/web/20130103030411/http://whatwg.org/html

**12.2.4.22**

Consume the next input character:

## ↳ U+002D HYPHEN-MINUS (-)

Switch to the script data escaped dash state. Emit a U+002D HYPHEN-MINUS character token.

## ↳ U+003C LESS-THAN SIGN (&lt;)

Switch to the script data escaped less-than sign state.

## ↳ U+0000 NULL

Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

## ↳ EOF

Switch to the data state. Parse error. Reconsume the EOF character.

## ↳ Anything else

Emit the current input character as a character token.

**12.2.4.23 Script data escaped dash state**

Consume the next input character:

## ↳ U+002D HYPHEN-MINUS (-)

Switch to the script data escaped dash dash state. Emit a U+002D HYPHEN-MINUS character token.

## ↳ U+003C LESS-THAN SIGN (&lt;)

Switch to the script data escaped less-than sign state.

## ↳ U+0000 NULL

Parse error. Switch to the script data escaped state. Emit a U+FFFD REPLACEMENT CHARACTER character token.

## ↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

## ↳ Anything else

Switch to the script data escaped state. Emit the current input character as a character token.

**12.2.4.24 Script data escaped dash dash state**

Consume the next input character:

## ↳ U+002D HYPHEN-MINUS (-)

Emit a U+002D HYPHEN-MINUS character token.

## ↳ U+003C LESS-THAN SIGN (&lt;)

Switch to the script data escaped less-than sign state.

## ↳ U+003E GREATER-THAN SIGN (&gt;)

Switch to the script data state. Emit a U+003E GREATER-THAN SIGN character token.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

↳ Anything else

Switch to the script data escaped state. Emit the current input character as a character token.

#### 12.2.4.25 Script data escaped less-than sign state

Consume the next input character:

↳ U+002F SOLIDUS (/)



state.

↳ Upperc **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

ut character (add 0x0020 to art state. Emit a U+003C

↳ Lowerc **//whatwg.org/html**

*rary buffer*. Switch to the

Script data double escape start state. Emit a U+003C LESS-THAN SIGN character token, and the current input character as a character token.

↳ Anything else

Switch to the script data escaped state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

#### 12.2.4.26 Script data escaped end tag open state

Consume the next input character:

↳ Uppercase ASCII letter

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the script data escaped end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ Lowercase ASCII letter

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the script data escaped end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ Anything else

Switch to the script data escaped state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

#### 12.2.4.27 Script data escaped end tag name state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

↳ U+002F SOLIDUS (/)

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

↳ U+003E GREATER-THAN SIGN (>)

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

↳ Uppercase ASCII letter

[Go](#)

NOV JAN FEB

03

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*temporary buffer.*

## ↳ Anything else

Switch to the script data escaped state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

**12.2.4.28 Script data double escape start state**

Consume the next input character:

- ↳ U+0009
- ↳ U+000A
- ↳ U+000C
- ↳ U+0020 /web/20130103030411/http://whatwg.org/html
- ↳ U+002F
- ↳ U+003E

If the *temporary buffer* is the string "script", then switch to the script data double escaped state. Otherwise, switch to the script data escaped state. Emit the current input character as a character token.

## ↳ Uppercase ASCII letter

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the *temporary buffer*. Emit the current input character as a character token.

## ↳ Lowercase ASCII letter

Append the current input character to the *temporary buffer*. Emit the current input character as a character token.

## ↳ Anything else

Switch to the script data escaped state. Reconsume the current input character.

**12.2.4.29 Script data double escaped state**

Consume the next input character:

- ↳ U+002D HYPHEN-MINUS (-)
- Switch to the script data double escaped dash state. Emit a U+002D HYPHEN-MINUS character token.
- ↳ U+003C LESS-THAN SIGN (<)
- Switch to the script data double escaped less-than sign state. Emit a U+003C LESS-THAN SIGN character token.
- ↳ U+0000 NULL
- Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.
- ↳ EOF
- Parse error. Switch to the data state. Reconsume the EOF character.
- ↳ Anything else
- Emit the current input character as a character token.

**12.2.4.30 Script data double escaped dash state**

Consume the next input character:

- ↳ U+002D HYPHEN-MINUS (-)
- Switch to the script data double escaped dash dash state. Emit a U+002D HYPHEN-MINUS character token.
- ↳ U+003C LESS-THAN SIGN (<)
- Switch to the script data double escaped less-than sign state. Emit a U+003C LESS-THAN SIGN character token.
- ↳ U+0000 NULL
- Parse error. Switch to the script data double escaped state. Emit a U+FFFD REPLACEMENT CHARACTER character token.
- ↳ EOF

[Go](#)

NOV JAN FEB

03

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

#### 12.2.4.31 Script data double escaped dash dash state

Consume the next input character:

↳ U+002D HYPHEN-MINUS (-)

Emit a U+002D HYPHEN-MINUS character token.

↳ U+003C LESS-THAN SIGN (<)

Switch to the script data double escaped less-than sign state. Emit a U+003C LESS-THAN SIGN character token.

↳ U+003E Too slow? Try reading the multipage copy of

↳ U+0000 the spec instead: https://web.archive.org

/web/20130103030411/http:

CHARACTER character

↳ EOF //whatwg.org/html

↳ Anything else

Switch to the script data double escaped state. Emit the current input character as a character token.

#### 12.2.4.32 Script data double escaped less-than sign state

Consume the next input character:

↳ U+002F SOLIDUS (/)

Set the *temporary buffer* to the empty string. Switch to the script data double escape end state. Emit a U+002F SOLIDUS character token.

↳ Anything else

Switch to the script data double escaped state. Reconsume the current input character.

#### 12.2.4.33 Script data double escape end state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

↳ U+002F SOLIDUS (/)

↳ U+003E GREATER-THAN SIGN (>)

If the *temporary buffer* is the string "script", then switch to the script data escaped state. Otherwise, switch to the script data double escaped state. Emit the current input character as a character token.

↳ Uppercase ASCII letter

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the *temporary buffer*. Emit the current input character as a character token.

↳ Lowercase ASCII letter

Append the current input character to the *temporary buffer*. Emit the current input character as a character token.

↳ Anything else

Switch to the script data double escaped state. Reconsume the current input character.

#### 12.2.4.34 Before attribute name state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Switch to the self-closing start tag state.

↳ U+003E GREATER-THAN SIGN (&gt;)

Switch to the data state. Emit the current tag token.

↳ Uppercase ASCII letter

Start a new attribute in the current tag token. Set that attribute's name to the lowercase version of the current input character (add 0x0020 to the character's code point), and its value to the empty string. Switch to the attribute name state.

↳ U+0000 NULL

FD REPLACEMENT

Too slow? Try reading the multipage copy of

↳ U+0022 the spec instead: https://web.archive.org

↳ U+0027

↳ U+003C /web/20130103030411/http:

↳ U+003E //whatwg.org/html

↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

↳ Anything else

Start a new attribute in the current tag token. Set that attribute's name to the current input character, and its value to the empty string. Switch to the attribute name state.

#### 12.2.4.35 Attribute name state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Switch to the after attribute name state.

↳ U+002F SOLIDUS (/)

Switch to the self-closing start tag state.

↳ U+003D EQUALS SIGN (=)

Switch to the before attribute value state.

↳ U+003E GREATER-THAN SIGN (&gt;)

Switch to the data state. Emit the current tag token.

↳ Uppercase ASCII letter

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current attribute's name.

↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's name.

↳ U+0022 QUOTATION MARK ("")

↳ U+0027 APOSTROPHE ('')

↳ U+003C LESS-THAN SIGN (&lt;)

Parse error. Treat it as per the "anything else" entry below.

↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

↳ Anything else

Append the current input character to the current attribute's name.

When the user agent leaves the attribute name state (and before emitting the tag token, if appropriate), the complete attribute's name must be compared to the other attributes on the same token; if there is already an attribute on the token with the exact same name,

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Ignore the character.

↳ U+002F SOLIDUS (/)

↳ U+003E Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org

↳ U+003E /web/20130103030411/http:

↳ Upperc //whatwg.org/html

    on of the current input  
character (add 0x0020 to the character's code point), and its value to the empty string. Switch to the attribute name state.

↳ U+0000 NULL

    Parse error. Start a new attribute in the current tag token. Set that attribute's name to a U+FFF4 REPLACEMENT  
CHARACTER character, and its value to the empty string. Switch to the attribute name state.

↳ U+0022 QUOTATION MARK ("")

↳ U+0027 APOSTROPHE ('')

↳ U+003C LESS-THAN SIGN (&lt;)

Parse error. Treat it as per the "anything else" entry below.

↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

↳ Anything else

    Start a new attribute in the current tag token. Set that attribute's name to the current input character, and its value to the  
empty string. Switch to the attribute name state.**12.2.4.37 Before attribute value state**

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Ignore the character.

↳ U+0022 QUOTATION MARK ("")

Switch to the attribute value (double-quoted) state.

↳ U+0026 AMPERSAND (&amp;)

Switch to the attribute value (unquoted) state. Reconsume the current input character.

↳ U+0027 APOSTROPHE ('')

Switch to the attribute value (single-quoted) state.

↳ U+0000 NULL

    Parse error. Append a U+FFF4 REPLACEMENT CHARACTER character to the current attribute's value. Switch to the  
attribute value (unquoted) state.

↳ U+003E GREATER-THAN SIGN (&gt;)

Parse error. Switch to the data state. Emit the current tag token.

↳ U+003C LESS-THAN SIGN (&lt;)

↳ U+003D EQUALS SIGN (=)

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

#### ↳ Anything else

Append the current input character to the current attribute's value. Switch to the attribute value (unquoted) state.

### 12.2.4.38 Attribute value (double-quoted) state

Consume the next input character:

#### ↳ U+0022 QUOTATION MARK ("")



↳ U+0026 Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org

· being U+0022 QUOTATION

↳ U+0000 /web/20130103030411/http:  
//whatwg.org/html

ute's value.

#### ↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

#### ↳ Anything else

Append the current input character to the current attribute's value.

### 12.2.4.39 Attribute value (single-quoted) state

Consume the next input character:

#### ↳ U+0027 APOSTROPHE ('')

Switch to the after attribute value (quoted) state.

#### ↳ U+0026 AMPERSAND (&)

Switch to the character reference in attribute value state, with the additional allowed character being U+0027 APOSTROPHE ('').

#### ↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's value.

#### ↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

#### ↳ Anything else

Append the current input character to the current attribute's value.

### 12.2.4.40 Attribute value (unquoted) state

Consume the next input character:

#### ↳ U+0009 CHARACTER TABULATION (tab)

#### ↳ U+000A LINE FEED (LF)

#### ↳ U+000C FORM FEED (FF)

#### ↳ U+0020 SPACE

Switch to the before attribute name state.

#### ↳ U+0026 AMPERSAND (&)

Switch to the character reference in attribute value state, with the additional allowed character being U+003E GREATER-THAN SIGN (>).

#### ↳ U+003E GREATER-THAN SIGN (>)

Switch to the data state. Emit the current tag token.

#### ↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's value.

#### ↳ U+0022 QUOTATION MARK ("")

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**[3,445 captures](#)**

5 Jun 2004 - 22 Aug 2020

Parse error. Treat it as per the "anything else" entry below.

↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

↳ Anything else

Append the current input character to the current attribute's value.

**12.2.4.41 Character reference in attribute value state**

Attempt to consume the character reference. If nothing is consumed, then if the character reference is valid, then switch to the attribute value state; otherwise, switch to the data state. Finally, switch to the data state.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**12.2.4.42 After attribute value (quoted) state**

Consume the next input character:

- ↳ U+0009 CHARACTER TABULATION (tab)
- ↳ U+000A LINE FEED (LF)
- ↳ U+000C FORM FEED (FF)
- ↳ U+0020 SPACE

Switch to the before attribute name state.

↳ U+002F SOLIDUS (/)

Switch to the self-closing start tag state.

↳ U+003E GREATER-THAN SIGN (&gt;)

Switch to the data state. Emit the current tag token.

↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

↳ Anything else

Parse error. Switch to the before attribute name state. Reconsume the character.

**12.2.4.43 Self-closing start tag state**

Consume the next input character:

↳ U+003E GREATER-THAN SIGN (&gt;)

Set the *self-closing flag* of the current tag token. Switch to the data state. Emit the current tag token.

↳ EOF

Parse error. Switch to the data state. Reconsume the EOF character.

↳ Anything else

Parse error. Switch to the before attribute name state. Reconsume the character.

**12.2.4.44 Bogus comment state**

Consume every character up to and including the first U+003E GREATER-THAN SIGN character (>) or the end of the file (EOF), whichever comes first. Emit a comment token whose data is the concatenation of all the characters starting from and including the character that caused the state machine to switch into the bogus comment state, up to and including the character immediately before the last consumed character (i.e. up to the character just before the U+003E or EOF character), but with any U+0000 NULL characters replaced by U+FFFD REPLACEMENT CHARACTER characters. (If the comment was started by the end of the file (EOF), the token is empty. Similarly, the token is empty if it was generated by the string "<!>".)

Switch to the data state.

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

About this capture

If the next two characters are both U+002D HYPHEN-MINUS characters (-), consume those two characters, create a comment token whose data is the empty string, and switch to the comment start state.

Otherwise, if the next seven characters are an ASCII case-insensitive match for the word "DOCTYPE", then consume those characters and switch to the DOCTYPE state.

Otherwise, if there is a current node and it is not an element in the HTML namespace and the next seven characters are a case-sensitive match for the string "[CDATA[" (the five uppercase letters "CDATA" with a U+005B LEFT SQUARE BRACKET character before and after), then consume those characters and switch to the CDATA section state.



' if any, is the first character

Otherwise, th'

that will be in **Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>**

Consume th /web/20130103030411/http:

↳ U+002F //whatwg.org/html

↳ U+0000 NULL

Parse error. Append a U+FFF4 REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↳ U+003E GREATER-THAN SIGN (>)

Parse error. Switch to the data state. Emit the comment token.

↳ EOF

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↳ Anything else

Append the current input character to the comment token's data. Switch to the comment state.

### 12.2.4.47 Comment start dash state

Consume the next input character:

↳ U+002D HYPHEN-MINUS (-)

Switch to the comment end state

↳ U+0000 NULL

Parse error. Append a U+002D HYPHEN-MINUS character (-) and a U+FFF4 REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↳ U+003E GREATER-THAN SIGN (>)

Parse error. Switch to the data state. Emit the comment token.

↳ EOF

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↳ Anything else

Append a U+002D HYPHEN-MINUS character (-) and the current input character to the comment token's data. Switch to the comment state.

### 12.2.4.48 Comment state

Consume the next input character:

↳ U+002D HYPHEN-MINUS (-)

Switch to the comment end dash state

↳ U+0000 NULL

Parse error. Append a U+FFF4 REPLACEMENT CHARACTER character to the comment token's data.

↳ EOF

[Go](#)

NOV JAN FEB

03

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

#### 12.2.4.49 Comment end dash state

Consume the next input character:

↳ **U+002D HYPHEN-MINUS (-)**

Switch to the comment end state

↳ **U+0000 NULL**

Parse error. Append a U+002D HYPHEN-MINUS character (-) and a U+FFFD REPLACEMENT CHARACTER character

↳ **EOF** Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ **Anything else**

Parse error. Append two U+002D HYPHEN-MINUS characters (-) and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

#### 12.2.4.50 Comment end state

Consume the next input character:

↳ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the comment token.

↳ **U+0000 NULL**

Parse error. Append two U+002D HYPHEN-MINUS characters (-) and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↳ **U+0021 EXCLAMATION MARK (!)**

Parse error. Switch to the comment end bang state.

↳ **U+002D HYPHEN-MINUS (-)**

Parse error. Append a U+002D HYPHEN-MINUS character (-) to the comment token's data.

↳ **EOF**

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Append two U+002D HYPHEN-MINUS characters (-) and the current input character to the comment token's data. Switch to the comment state.

#### 12.2.4.51 Comment end bang state

Consume the next input character:

↳ **U+002D HYPHEN-MINUS (-)**

Append two U+002D HYPHEN-MINUS characters (-) and a U+0021 EXCLAMATION MARK character (!) to the comment token's data. Switch to the comment end dash state.

↳ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the comment token.

↳ **U+0000 NULL**

Parse error. Append two U+002D HYPHEN-MINUS characters (-), a U+0021 EXCLAMATION MARK character (!), and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↳ **EOF**

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↳ **Anything else**

Append two U+002D HYPHEN-MINUS characters (-), a U+0021 EXCLAMATION MARK character (!), and the current input character to the comment token's data. Switch to the comment state.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

- ↳ U+000A LINE FEED (LF)
- ↳ U+000C FORM FEED (FF)
- ↳ U+0020 SPACE

Switch to the before DOCTYPE name state.

- ↳ EOF

Parse error. Switch to the data state. Create a new DOCTYPE token. Set its *force-quirks flag* to *on*. Emit the token. Reconsume the EOF character.

- ↳ Anything

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

### 12.2.4.53

Consume the next input character:

- ↳ U+0009 //whatwg.org/html
- ↳ U+000A
- ↳ U+000C FORM FEED (FF)
- ↳ U+0020 SPACE

Ignore the character.

- ↳ Uppercase ASCII letter

Create a new DOCTYPE token. Set the token's name to the lowercase version of the current input character (add 0x0020 to the character's code point). Switch to the DOCTYPE name state.

- ↳ U+0000 NULL

Parse error. Create a new DOCTYPE token. Set the token's name to a U+FFFD REPLACEMENT CHARACTER character. Switch to the DOCTYPE name state.

- ↳ U+003E GREATER-THAN SIGN (>)

Parse error. Create a new DOCTYPE token. Set its *force-quirks flag* to *on*. Switch to the data state. Emit the token.

- ↳ EOF

Parse error. Switch to the data state. Create a new DOCTYPE token. Set its *force-quirks flag* to *on*. Emit the token. Reconsume the EOF character.

- ↳ Anything else

Create a new DOCTYPE token. Set the token's name to the current input character. Switch to the DOCTYPE name state.

### 12.2.4.54 DOCTYPE name state

Consume the next input character:

- ↳ U+0009 CHARACTER TABULATION (tab)
- ↳ U+000A LINE FEED (LF)
- ↳ U+000C FORM FEED (FF)
- ↳ U+0020 SPACE

Switch to the after DOCTYPE name state.

- ↳ U+003E GREATER-THAN SIGN (>)

Switch to the data state. Emit the current DOCTYPE token.

- ↳ Uppercase ASCII letter

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current DOCTYPE token's name.

- ↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's name.

- ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Ignore the character.



↳ U+003E `

Too slow? Try reading the multipage copy of

↳ EOF the spec instead: https://web.archive.org

hat DOCTYPE token.

/web/20130103030411/http:

↳ Anythin

//whatwg.org/html

atch for the word "PUBLIC",

tis

Otherwise, if the six characters starting from the current input character are an ASCII case-insensitive match for the word "SYSTEM", then consume those characters and switch to the after DOCTYPE system keyword state.

Otherwise, this is a parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

#### 12.2.4.56 After DOCTYPE public keyword state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Switch to the before DOCTYPE public identifier state.

↳ U+0022 QUOTATION MARK (" )

Parse error. Set the DOCTYPE token's public identifier to the empty string (not missing), then switch to the DOCTYPE public identifier (double-quoted) state.

↳ U+0027 APOSTROPHE (' )

Parse error. Set the DOCTYPE token's public identifier to the empty string (not missing), then switch to the DOCTYPE public identifier (single-quoted) state.

↳ U+003E GREATER-THAN SIGN (&gt; )

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ Anything else

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

#### 12.2.4.57 Before DOCTYPE public identifier state

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Ignore the character.

↳ U+0022 QUOTATION MARK (" )

[Go](#)

NOV JAN FEB

03

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

(single-quoted) state.

## ↳ U+003E GREATER-THAN SIGN (&gt;)

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

## ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

## ↳ Anything else



Ξ state.

Too slow? Try reading the multipage copy of

**12.2.4.58 the spec instead: https://web.archive.org**

Consume the next input character:

## ↳ U+0022

//whatwg.org/html

## ↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's public identifier.

## ↳ U+003E GREATER-THAN SIGN (&gt;)

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

## ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

## ↳ Anything else

Append the current input character to the current DOCTYPE token's public identifier.

**12.2.4.59 DOCTYPE public identifier (single-quoted) state**

Consume the next input character:

## ↳ U+0027 APOSTROPHE (')

Switch to the after DOCTYPE public identifier state.

## ↳ U+0000 NULL

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's public identifier.

## ↳ U+003E GREATER-THAN SIGN (&gt;)

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

## ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

## ↳ Anything else

Append the current input character to the current DOCTYPE token's public identifier.

**12.2.4.60 After DOCTYPE public identifier state**

Consume the next input character:

## ↳ U+0009 CHARACTER TABULATION (tab)

## ↳ U+000A LINE FEED (LF)

## ↳ U+000C FORM FEED (FF)

## ↳ U+0020 SPACE

Switch to the between DOCTYPE public and system identifiers state.

## ↳ U+003E GREATER-THAN SIGN (&gt;)

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

#### ↳ U+0027 APOSTROPHE (')

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

#### ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

#### ↳ Anything else



Ξ state.

Too slow? Try reading the multipage copy of

#### 12.2.4.61 the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Consume th

/web/20130103030411/http:

#### ↳ U+0009

#### ↳ U+000A

//whatwg.org/html

#### ↳ U+000C ↳..... ↳...

#### ↳ U+0020 SPACE

Ignore the character.

#### ↳ U+003E GREATER-THAN SIGN (>)

Switch to the data state. Emit the current DOCTYPE token.

#### ↳ U+0022 QUOTATION MARK ("")

Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (double-quoted) state.

#### ↳ U+0027 APOSTROPHE ('')

Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

#### ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

#### ↳ Anything else

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

#### 12.2.4.62 After DOCTYPE system keyword state

Consume the next input character:

#### ↳ U+0009 CHARACTER TABULATION (tab)

#### ↳ U+000A LINE FEED (LF)

#### ↳ U+000C FORM FEED (FF)

#### ↳ U+0020 SPACE

Switch to the before DOCTYPE system identifier state.

#### ↳ U+0022 QUOTATION MARK ("")

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (double-quoted) state.

#### ↳ U+0027 APOSTROPHE ('')

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

#### ↳ U+003E GREATER-THAN SIGN (>)

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

#### ↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Ignore the character.



↳ U+0022 `

Too slow? Try reading the multipage copy of

the DOCTYPE system

↳ U+0027 the spec instead: https://web.archive.org

/web/20130103030411/http:

the DOCTYPE system

↳ U+003E //whatwg.org/html

`

...that DOCTYPE token.

↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token.

Reconsume the EOF character.

↳ Anything else

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

#### 12.2.4.64 DOCTYPE system identifier (double-quoted) state

Consume the next input character:

↳ U+0022 QUOTATION MARK (" )

Switch to the after DOCTYPE system identifier state.

↳ U+0000 NULL

Parse error. Append a U+FFF4 REPLACEMENT CHARACTER character to the current DOCTYPE token's system identifier.

↳ U+003E GREATER-THAN SIGN (&gt;)

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ Anything else

Append the current input character to the current DOCTYPE token's system identifier.

#### 12.2.4.65 DOCTYPE system identifier (single-quoted) state

Consume the next input character:

↳ U+0027 APOSTROPHE (' )

Switch to the after DOCTYPE system identifier state.

↳ U+0000 NULL

Parse error. Append a U+FFF4 REPLACEMENT CHARACTER character to the current DOCTYPE token's system identifier.

↳ U+003E GREATER-THAN SIGN (&gt; )

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ EOF

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Consume the next input character:

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

Ignore the character.



↳ U+003E `

Too slow? Try reading the multipage copy of

↳ EOF

the spec instead: https://web.archive.org  
/web/20130103030411/http:

hat DOCTYPE token.

↳ Anythin

//whatwg.org/html

(orce-quirks flag to on.)

#### 12.2.4.67 Bogus DOCTYPE state

Consume the next input character:

↳ U+003E GREATER-THAN SIGN (&gt;)

Switch to the data state. Emit the DOCTYPE token.

↳ EOF

Switch to the data state. Emit the DOCTYPE token. Reconsume the EOF character.

↳ Anything else

Ignore the character.

#### 12.2.4.68 CDATA section state

Switch to the data state.

Consume every character up to the next occurrence of the three character sequence U+005D RIGHT SQUARE BRACKET U+005D RIGHT SQUARE BRACKET U+003E GREATER-THAN SIGN (]]>), or the end of the file (EOF), whichever comes first. Emit a series of character tokens consisting of all the characters consumed except the matching three character sequence at the end (if one was found before the end of the file).

If the end of the file was reached, reconsume the EOF character.

#### 12.2.4.69 Tokenizing character references

This section defines how to **consume a character reference**. This definition is used when parsing character references in text and in attributes.

The behavior depends on the identity of the next character (the one immediately after the U+0026 AMPERSAND character):

↳ U+0009 CHARACTER TABULATION (tab)

↳ U+000A LINE FEED (LF)

↳ U+000C FORM FEED (FF)

↳ U+0020 SPACE

↳ U+003C LESS-Than SIGN

↳ U+0026 AMPERSAND

↳ EOF

↳ The additional allowed character, if there is one

Not a character reference. No characters are consumed, and nothing is returned. (This is not an error, either.)

↳ U+0023 NUMBER SIGN (#)

Consume the U+0023 NUMBER SIGN.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Follow the steps below, but using ASCII hex digits.

When it comes to interpreting the number, interpret it as a hexadecimal number.

### ↳ Anything else

Follow the steps below, but using ASCII digits.

When it comes to interpreting the number, interpret it as a decimal number.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If this number is one of the numbers in the first column of the following table, then this is a parse error. Find the row with that number in the first column, and return a character token for the Unicode character given in the second column of that row.

Number	Unicode character
0x00	U+FFFD REPLACEMENT CHARACTER
0x0D	U+000D CARRIAGE RETURN (CR)
0x80	U+20AC EURO SIGN (€)
0x81	U+0081 <control>
0x82	U+201A SINGLE LOW-9 QUOTATION MARK („)
0x83	U+0192 LATIN SMALL LETTER F WITH HOOK (ƒ)
0x84	U+201E DOUBLE LOW-9 QUOTATION MARK („)
0x85	U+2026 HORIZONTAL ELLIPSIS (...)
0x86	U+2020 DAGGER (†)
0x87	U+2021 DOUBLE DAGGER (‡)
0x88	U+02C6 MODIFIER LETTER CIRCUMFLEX ACCENT (^)
0x89	U+2030 PER MILLE SIGN (‰)
0x8A	U+0160 LATIN CAPITAL LETTER S WITH CARON (Š)
0x8B	U+2039 SINGLE LEFT-POINTING ANGLE QUOTATION MARK (⟨)
0x8C	U+0152 LATIN CAPITAL LIGATURE OE (Œ)
0x8D	U+008D <control>
0x8E	U+017D LATIN CAPITAL LETTER Z WITH CARON (Ž)
0x8F	U+008F <control>
0x90	U+0090 <control>
0x91	U+2018 LEFT SINGLE QUOTATION MARK (‘)
0x92	U+2019 RIGHT SINGLE QUOTATION MARK (’)
0x93	U+201C LEFT DOUBLE QUOTATION MARK (“)
0x94	U+201D RIGHT DOUBLE QUOTATION MARK (”)
0x95	U+2022 BULLET (•)
0x96	U+2013 EN DASH (–)
0x97	U+2014 EM DASH (—)
0x98	U+02DC SMALL TILDE (˜)
0x99	U+2122 TRADE MARK SIGN (™)
0x9A	U+0161 LATIN SMALL LETTER S WITH CARON (ſ)
0x9B	U+203A SINGLE RIGHT-POINTING ANGLE QUOTATION MARK (⟩)
0x9C	U+0153 LATIN SMALL LIGATURE OE (œ)
0x9D	U+009D <control>
0x9E	U+017E LATIN SMALL LETTER Z WITH CARON (ž)
0x9F	U+0178 LATIN CAPITAL LETTER Y WITH DIAERESIS (Ŷ)

Otherwise, if the number is in the range 0xD800 to 0xDFFF or is greater than 0x10FFFF, then this is a parse error. Return

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

0x6FFE, 0x6FFF, 0x7FFF, 0x7FFF, 0x8FFF, 0x8FFF, 0x9FFF, 0x9FFF, 0xAFFF, 0xAFFF, 0xBFFF, 0xBFFF, 0xCFFF, 0xCFFF, 0xDFFF, 0xEFFF, 0xFFFF, 0xFFFF, 0x10FFF, or 0x10FFF, then this is a parse error.

#### ↳ Anything else

Consume the maximum number of characters possible, with the consumed characters matching one of the identifiers in the first column of the named character references table (in a case-sensitive manner).

If no match can be made, then no characters are consumed, and nothing is returned. In this case, if the characters after '#' are ASCII characters followed by

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

atched is not a U+003B  
ter (=) or an alphanumeric  
-0026 AMPERSAND

MICOLON character (;),

Return one or two character tokens for the character(s) corresponding to the character reference name (as given by the second column of the named character references table).

If the markup contains (not in an attribute) the string I'm &notit; I tell you, the character reference is parsed as "not", as in, I'm -it; I tell you (and this is a parse error). But if the markup was I'm &notin; I tell you, the character reference would be parsed as "notin;", resulting in I'm # I tell you (and no parse error).

### 12.2.5 Tree construction

The input to the tree construction stage is a sequence of tokens from the tokenization stage. The tree construction stage is associated with a DOM `Document` object when a parser is created. The "output" of this stage consists of dynamically modifying or extending that document's DOM tree.

This specification does not define when an interactive user agent has to render the `Document` so that it is available to the user, or when it has to begin accepting user input.

As each token is emitted from the tokenizer, the user agent must follow the appropriate steps from the following list:

- ↳ If there is no current node
- ↳ If the current node is an element in the HTML namespace
- ↳ If the current node is a MathML text integration point and the token is a start tag whose tag name is neither "mglyph" nor "malignmark"
- ↳ If the current node is a MathML text integration point and the token is a character token
- ↳ If the current node is an `annotation-xml` element in the MathML namespace and the token is a start tag whose tag name is "svg"
- ↳ If the current node is an HTML integration point and the token is a start tag
- ↳ If the current node is an HTML integration point and the token is a character token
- ↳ If the token is an end-of-file token

Process the token according to the rules given in the section corresponding to the current insertion mode in HTML content.

- ↳ Otherwise

Process the token according to the rules given in the section for parsing tokens in foreign content.

When the specification says that a user agent is to **act as if a token had been seen**, it means that the user agent is to follow the appropriate steps from the above list.

The current node is a **MathML text integration point** if it is one of the following elements:

- An `mi` element in the MathML namespace
- An `mo` element in the MathML namespace
- An `mn` element in the MathML namespace

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

- An annotation-xml element in the MathML namespace whose start tag token had an attribute with the name "encoding" whose value was an ASCII case-insensitive match for the string "text/html"
- An annotation-xml element in the MathML namespace whose start tag token had an attribute with the name "encoding" whose value was an ASCII case-insensitive match for the string "application/xhtml+xml"
- A foreignObject element in the SVG namespace
- A desc element in the SVG namespace
- A title element in the SVG namespace

When the steps below require the UA to **insert a character** into a node, if that node has a child immediately before where the character is to be inserted, and that child is a `Text` node, then the character must be appended to that `Text` node; otherwise, a new `Text` node v

Here  
agen  
the spec instead: <https://web.archive.org>  
— /web/20130103030411/http://whatwg.org/html

result in, assuming a user

### Text nodes

aining "AB".

```
document.body.removeChild(script);
</script>B
```

Three `Text` nodes; "A" before the script, the script's contents, and "BC" after the script (the parser appends to the `Text` node created by the script).

```
A<script>
var text = document.createTextNode('B');
document.body.appendChild(text);
</script>C
```

Two adjacent `Text` nodes in the document, containing "A" and "BC".

```
A<script>
var text = document.getElementsByTagName('script')[0].firstChild;
text.data = 'B';
document.body.appendChild(text);
</script>C
```

One `Text` node before the table, containing "ABCD". (This is caused by foster parenting.)

```
A<table>B<tr>C</tr>D</table>
```

One `Text` node before the table, containing "A B C" (A-space-B-space-C). (This is caused by foster parenting.)

```
A<table><tr> B</tr> C</table>
```

One `Text` node before the table, containing "A BC" (A-space-B-C), and one `Text` node inside the table (as a child of a `tbody`) with a single space character. (Space characters separated from non-space characters by non-character tokens are not affected by foster parenting, even if those other tokens then get ignored.)

```
A<table><tr> B</tr> C</table>
```

DOM mutation events must not fire for changes caused by the UA parsing the document. This includes the parsing of any content inserted using `document.write()` and `document.writeln()` calls. [DOMEVENTS]

However, mutation observers *do* fire, as required by the DOM specification.

**Note:** Not all of the tag names mentioned below are conformant tag names in this specification; many are included to handle legacy content. They still form part of the algorithm that implementations are required to implement to claim conformance.

**Note:** The algorithm described below places no limit on the depth of the DOM tree generated, or on the length of tag names, attribute names, attribute values, `Text` nodes, etc. While implementors are encouraged to avoid arbitrary limits, it is recognized that practical concerns will likely force user agents to impose nesting depth constraints.

### 12.2.5.1 Creating and inserting elements

When the steps below require the UA to **create an element for a token** in a particular namespace, the UA must create a node implementing the interface appropriate for the element type corresponding to the tag name of the token in the given namespace (as given in the specification that defines that element, e.g. for an `a` element in the HTML namespace, this specification defines it to be the `HTMLAnchorElement` interface), with the tag name being the name of that element, with the node being in the given namespace, and

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

▼ About this capture

When a resettable element is created in this manner, its reset algorithm must be invoked once the attributes are set. (This initializes the element's value and checkedness based on the element's attributes.)

When the steps below require the UA to **insert an HTML element** for a token, the UA must first create an element for the token in the HTML namespace, and then append this node to the current node, and push it onto the stack of open elements so that it is the new current node.

The steps below may also require that the UA insert an HTML element in a particular place. In ~~which~~ case the UA must follow the same steps except

Too slow? Try reading the multipage copy of

If an element has been removed from the spec instead: <https://web.archive.org>

and the new element pair [/web/20130103030411/http](http://web/20130103030411/)

When the st //whatwg.org/html

When the student  
given names:

given namespace. The parser creates new elements so that it is the new current node. If the newly created element has an `xmlns` attribute in the XMLNS namespace whose value is not exactly the same as the element's namespace, that is a parse error. Similarly, if the newly created element has an `xmlns:xlink` attribute in the XMLNS namespace whose value is not the XLink Namespace, that is a parse error.

When the steps below require the user agent to **adjust MathML attributes** for a token, then, if the token has an attribute named `definitionurl`, change its name to `definitionURI` (note the case difference).

When the steps below require the user agent to **adjust SVG attributes** for a token, then, for each attribute on the token whose attribute name is one of the ones in the first column of the following table, change the attribute's name to the name given in the corresponding cell in the second column. (This fixes the case of SVG attributes that are not all lowercase.)

Attribute name on token	Attribute name on element
attributename	attributeName
attributetype	attributeType
basefrequency	baseFrequency
baseprofile	baseProfile
calcmode	calcMode
clippathunits	clipPathUnits
contentscripttype	contentScriptType
contentstyletype	contentStyleType
diffuseconstant	diffuseConstant
edgemode	edgeMode
externalresourcesrequired	externalResourcesRequire
filtereres	filterRes
filterunits	filterUnits
glyphref	glyphRef
gradienttransform	gradientTransform
gradientunits	gradientUnits
kernelmatrix	kernelMatrix
kernelunitlength	kernelUnitLength
keypoints	keyPoints
keysplines	keySplines
keytimes	keyTimes
lengthadjust	lengthAdjust
limitingconeangle	limitingConeAngle
markerheight	markerHeight
markerunits	markerUnits
markerwidth	markerWidth
maskcontentunits	maskContentUnits
maskunits	maskUnits

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

patterntransform	patternTransform
patternunits	patternUnits
pointsatx	pointsAtX
pointsaty	pointsAtY
pointsatz	pointsAtZ
preservealpha	preserveAlpha
preserveaspectratio	preserveAspectRatio
primitiveunits	primitiveUnits
refx	
refy	
repeatcount	
repeatdur	
requiredext	
requiredfeat	
specularcolor	
specularexp0..	
spreadmethod	spreadMethod
startoffset	startOffset
stddeviation	stdDeviation
stitchtiles	stitchTiles
surfacescale	surfaceScale
systemlanguage	systemLanguage
tablevalues	tableValues
targetx	targetX
targety	targetY
textlength	textLength
viewbox	viewBox
viewtarget	viewTarget
xchannelselector	xChannelSelector
ychannelselector	yChannelSelector
zoomandpan	zoomAndPan

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



When the steps below require the user agent to **adjust foreign attributes** for a token, then, if any of the attributes on the token match the strings given in the first column of the following table, let the attribute be a namespaced attribute, with the prefix being the string given in the corresponding cell in the second column, the local name being the string given in the corresponding cell in the third column, and the namespace being the namespace given in the corresponding cell in the fourth column. (This fixes the use of namespaced attributes, in particular lang attributes in the XML namespace.)

Attribute name	Prefix	Local name	Namespace
xlink:actuate	xlink	actuate	XLink namespace
xlink:arcrole	xlink	arcrole	XLink namespace
xlink:href	xlink	href	XLink namespace
xlink:role	xlink	role	XLink namespace
xlink:show	xlink	show	XLink namespace
xlink:title	xlink	title	XLink namespace
xlink:type	xlink	type	XLink namespace
xml:base	xml	base	XML namespace
xml:lang	xml	lang	XML namespace
xml:space	xml	space	XML namespace
xmlns	(none)	xmlns	XMLNS namespace
xmlns:xlink	xlink	xlink	XMLNS namespace

The **generic raw text element parsing algorithm** and the **generic RCDATA element parsing algorithm** consist of the following steps. These algorithms are always invoked in response to a start tag token.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

3. Let the original insertion mode be the current insertion mode.
4. Then, switch the insertion mode to "text".

### 12.2.5.2 Closing elements that have implied end tags

When the steps below require the UA to **generate implied end tags**, then, while the current node is a `dd` element, a `dt` element, an `li` element, an `option` element, an `optgroup` element, a `p` element, an `rp` element, or an `rt` element, the UA must pop the current node off the stack of open elements and the UA must perform the steps above.

If a step requires the UA to generate implied end tags, then the UA must perform the steps above.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

### 12.2.5.3 Foster parenting

Foster parent <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

When a node `node` is to be **foster parented**, the node `node` must be inserted into the **foster parent element**.

**Note:** This typically happens as part of the insert an HTML element algorithm above: when that algorithm would normally insert the newly created node into the current node, it gets inserted into the foster parent element instead.

The **foster parent element** is the parent element of the last `table` element in the stack of open elements, if there is a `table` element and it has such a parent element.

**Note:** It might have no parent or some other kind parent if a script manipulated the DOM after the element was inserted by the parser.

If there is no `table` element in the stack of open elements (fragment case), then the **foster parent element** is the first element in the stack of open elements (the `html` element). Otherwise, if there is a `table` element in the stack of open elements, but the last `table` element in the stack of open elements has no parent, or its parent node is not an element, then the **foster parent element** is the element before the last `table` element in the stack of open elements.

If the **foster parent element** is the parent element of the last `table` element in the stack of open elements, then `node` must be inserted into the **foster parent element**, immediately *before* the last `table` element in the stack of open elements; otherwise, `node` must be appended to the **foster parent element**.

### 12.2.5.4 The rules for parsing tokens in HTML content

#### 12.2.5.4.1 The "initial" insertion mode

When the user agent is to apply the rules for the "initial" insertion mode, the user agent must handle the token as follows:

↪ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

Ignore the token.

↪ A comment token

Append a `Comment` node to the `Document` object with the `data` attribute set to the data given in the comment token.

↪ A DOCTYPE token

If the DOCTYPE token's name is not a case-sensitive match for the string "`html`", or the token's public identifier is not missing, or the token's system identifier is neither missing nor a case-sensitive match for the string "`about:legacy-compat`", and none of the sets of conditions in the following list are matched, then there is a parse error.

- The DOCTYPE token's name is a case-sensitive match for the string "`html`", the token's public identifier is the case-sensitive string "`-//W3C//DTD HTML 4.0//EN`", and the token's system identifier is either missing or the case-sensitive string "`http://www.w3.org/TR/REC-html40/strict.dtd`".
- The DOCTYPE token's name is a case-sensitive match for the string "`html`", the token's public identifier is the case-sensitive string "`-//W3C//DTD HTML 4.01//EN`", and the token's system identifier is either missing or the case-sensitive string "`http://www.w3.org/TR/REC-html40/strict.dtd`".

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- The DOCTYPE token's name is a case-sensitive match for the string "html", the token's public identifier is the case-sensitive string "-//W3C//DTD XHTML 1.1//EN", and the token's system identifier is the case-sensitive string "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd".

Conformance checkers may, based on the values (including presence or lack thereof) of the DOCTYPE token's name, public identifier, or system identifier, switch to a conformance checking mode for another language (e.g. based on the DOCTYPE token a conformance checker could recognize that the document is an HTML4-era document, and defer to an HTML4 conformance checker.)



## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ne given in the DOCTYPE  
lentifier given in the  
ute set to the system  
; and the other attributes  
documentType node with the  
: object.  
Document to quirks mode:

- The *force-quirks flag* is set to *on*.
- The name is set to anything other than "html" (compared case-sensitively).
- The public identifier starts with: "+//Silmaril//DTD html Pro v0r11 19970101//"
- The public identifier starts with: "-//AdvaSoft Ltd//DTD HTML 3.0 asWedit + extensions//"
- The public identifier starts with: "-//AS//DTD HTML 3.0 asWedit + extensions//"
- The public identifier starts with: "-//IETF//DTD HTML 2.0 Level 1//"
- The public identifier starts with: "-//IETF//DTD HTML 2.0 Level 2//"
- The public identifier starts with: "-//IETF//DTD HTML 2.0 Strict Level 1//"
- The public identifier starts with: "-//IETF//DTD HTML 2.0 Strict Level 2//"
- The public identifier starts with: "-//IETF//DTD HTML 2.0 Strict//"
- The public identifier starts with: "-//IETF//DTD HTML 2.0//"
- The public identifier starts with: "-//IETF//DTD HTML 2.1E//"
- The public identifier starts with: "-//IETF//DTD HTML 3.0//"
- The public identifier starts with: "-//IETF//DTD HTML 3.2 Final//"
- The public identifier starts with: "-//IETF//DTD HTML 3.2//"
- The public identifier starts with: "-//IETF//DTD HTML 3//"
- The public identifier starts with: "-//IETF//DTD HTML Level 0//"
- The public identifier starts with: "-//IETF//DTD HTML Level 1//"
- The public identifier starts with: "-//IETF//DTD HTML Level 2//"
- The public identifier starts with: "-//IETF//DTD HTML Level 3//"
- The public identifier starts with: "-//IETF//DTD HTML Strict Level 0//"
- The public identifier starts with: "-//IETF//DTD HTML Strict Level 1//"
- The public identifier starts with: "-//IETF//DTD HTML Strict Level 2//"
- The public identifier starts with: "-//IETF//DTD HTML Strict Level 3//"
- The public identifier starts with: "-//IETF//DTD HTML Strict//"
- The public identifier starts with: "-//IETF//DTD HTML//"
- The public identifier starts with: "-//Metrius//DTD Metrius Presentational//"
- The public identifier starts with: "-//Microsoft//DTD Internet Explorer 2.0 HTML Strict//"
- The public identifier starts with: "-//Microsoft//DTD Internet Explorer 2.0 HTML//"
- The public identifier starts with: "-//Microsoft//DTD Internet Explorer 2.0 Tables//"
- The public identifier starts with: "-//Microsoft//DTD Internet Explorer 3.0 HTML Strict//"
- The public identifier starts with: "-//Microsoft//DTD Internet Explorer 3.0 HTML//"
- The public identifier starts with: "-//Microsoft//DTD Internet Explorer 3.0 Tables//"
- The public identifier starts with: "-//Netscape Comm. Corp//DTD HTML//"
- The public identifier starts with: "-//Netscape Comm. Corp//DTD Strict HTML//"
- The public identifier starts with: "-//O'Reilly and Associates//DTD HTML 2.0//"
- The public identifier starts with: "-//O'Reilly and Associates//DTD HTML Extended 1.0//"
- The public identifier starts with: "-//O'Reilly and Associates//DTD HTML Extended Relaxed 1.0//"
- The public identifier starts with: "-//SoftQuad Software//DTD HotMetal PRO 6.0::19990601::extensions to HTML 4.0//"
- The public identifier starts with: "-//SoftQuad//DTD HotMetal PRO 4.0::19971010::extensions to HTML 4.0//"
- The public identifier starts with: "-//Spyglass//DTD HTML 2.0 Extended//"
- The public identifier starts with: "-//SQ//DTD HTML 2.0 HotMetal + extensions//"
- The public identifier starts with: "-//Sun Microsystems Corp//DTD HotJava HTML//"
- The public identifier starts with: "-//Sun Microsystems Corp//DTD HotJava Strict HTML//"
- The public identifier starts with: "-//W3C//DTD HTML 3 1995-03-24//"
- The public identifier starts with: "-//W3C//DTD HTML 3.2 Draft//"
- The public identifier starts with: "-//W3C//DTD HTML 3.2 Final//"
- The public identifier starts with: "-//W3C//DTD HTML 3.2//"

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- The public identifier starts with: "-//W3C//DTD W3 HTML//"
- The public identifier starts with: "-//W3O//DTD W3 HTML 3.0//"
- The public identifier is set to: "-//W3O//DTD W3 HTML Strict 3.0//EN//"
- The public identifier starts with: "-//WebTechs//DTD Mozilla HTML 2.0//"
- The public identifier starts with: "-//WebTechs//DTD Mozilla Mozilla HTML//"
- The public identifier is set to: "-//W3C/DTD HTML 4.0 Transitional//EN"
- The public identifier is set to: "HTML"
- The system identifier is set to: "http://www.ibm.com/data/dtd/v11/ibmxhtml1-transitional.dtd"
- The system identifier is missing and the public identifier starts with: "-//W3C//DTD HTML 4.01 Frameset//"
- The system identifier is missing and the public identifier starts with: "-//W3C//DTD HTML 4.01

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the Document to limited-

ID HTML 4.01

ID HTML 4.01

The system identifier and public identifier strings must be compared to the values given in the lists above in an ASCII case-insensitive manner. A system identifier whose value is the empty string is not considered missing for the purposes of the conditions above.

Then, switch the insertion mode to "before html".

#### ↳ Anything else

If the document is *not* an `iframe srcdoc` document, then this is a parse error; set the Document to quirks mode.

In any case, switch the insertion mode to "before html", then reprocess the current token.

#### 12.2.5.4.2 The "before html" insertion mode

When the user agent is to apply the rules for the "before html" insertion mode, the user agent must handle the token as follows:

##### ↳ A DOCTYPE token

Parse error. Ignore the token.

##### ↳ A comment token

Append a `Comment` node to the `Document` object with the `data` attribute set to the data given in the comment token.

##### ↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

Ignore the token.

##### ↳ A start tag whose tag name is "html"

Create an element for the token in the HTML namespace. Append it to the `Document` object. Put this element in the stack of open elements.

If the `Document` is being loaded as part of navigation of a browsing context, then: if the newly created element has a `manifest` attribute whose value is not the empty string, then resolve the value of that attribute to an absolute URL, relative to the newly created element, and if that is successful, run the application cache selection algorithm with the resulting absolute URL with any `<fragment>` component removed; otherwise, if there is no such attribute, or its value is the empty string, or resolving its value fails, run the application cache selection algorithm with no manifest. The algorithm must be passed the `Document` object.

Switch the insertion mode to "before head".

##### ↳ An end tag whose tag name is one of: "head", "body", "html", "br"

Act as described in the "anything else" entry below.

##### ↳ Any other end tag

Parse error. Ignore the token.

##### ↳ Anything else

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

Switch the insertion mode to "before head", then reprocess the current token.

The root element can end up being removed from the `Document` object, e.g. by scripts; nothing in particular happens in such cases, content continues being appended to the nodes as described in the next section.

#### 12.2.5.4.3 The "before head" insertion mode

When the user agent is to apply the rules for the "before head" insertion mode, the user agent must handle the token as follows:

- ↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE
  - Insert the character into the current node.
- ↳ A comment token
  - Append a `Comment` node to the current node with the `data` attribute set to the data given in the comment token.
- ↳ A DOCTYPE token
  - Parse error. Ignore the token.
- ↳ A start tag whose tag name is "html"
  - Process the token using the rules for the "in body" insertion mode.
- ↳ A start tag whose tag name is "head"
  - Insert an HTML element for the token.
  - Set the `head` element pointer to the newly created `head` element.
  - Switch the insertion mode to "in head".
- ↳ An end tag whose tag name is one of: "head", "body", "html", "br"
  - Act as if a start tag token with the tag name "head" and no attributes had been seen, then reprocess the current token.
- ↳ Any other end tag
  - Parse error. Ignore the token.
- ↳ Anything else
  - Act as if a start tag token with the tag name "head" and no attributes had been seen, then reprocess the current token.

#### 12.2.5.4.4 The "in head" insertion mode

When the user agent is to apply the rules for the "in head" insertion mode, the user agent must handle the token as follows:

- ↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE
  - Insert the character into the current node.
- ↳ A comment token
  - Append a `Comment` node to the current node with the `data` attribute set to the data given in the comment token.
- ↳ A start tag whose tag name is "html"
  - Process the token using the rules for the "in body" insertion mode.
- ↳ A start tag whose tag name is one of: "base", "basefont", "bgsound", "command", "link"
  - Insert an HTML element for the token. Immediately pop the current node off the stack of open elements.
  - Acknowledge the token's *self-closing flag*, if it is set.
- ↳ A start tag whose tag name is "meta"
  - Insert an HTML element for the token. Immediately pop the current node off the stack of open elements.
  - Acknowledge the token's *self-closing flag*, if it is set.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

"Content-Type", and the element has a `content` attribute, and applying the algorithm for extracting a character encoding from a `meta` element to that attribute's value returns a supported ASCII-compatible character encoding or a UTF-16 encoding, and the confidence is currently *tentative*, then change the encoding to the extracted encoding.

↳ A start tag whose tag name is "title"

Follow the generic RCDATA element parsing algorithm.

↳ A start tag whose tag name is "noscript", if the scripting flag is enabled

↳ A start tag whose tag name is one of: "noframes", "style"



↳ A start Too slow? Try reading the multipage copy of  
start the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. Create an element for the token in the HTML namespace.
2. Mark the element as being "parser-inserted" and unset the element's "force-async" flag.

**Note:** This ensures that, if the script is external, any `document.write()` calls in the script will execute in-line, instead of blowing the document away, as would happen in most other cases. It also prevents the script from executing until the end tag is seen.

3. If the parser was originally created for the HTML fragment parsing algorithm, then mark the `script` element as "already started". (fragment case)
4. Append the new element to the current node and push it onto the stack of open elements.
5. Switch the tokenizer to the script data state.
6. Let the original insertion mode be the current insertion mode.
7. Switch the insertion mode to "text".

↳ An end tag whose tag name is "head"

Pop the current node (which will be the `head` element) off the stack of open elements.

Switch the insertion mode to "after head".

↳ An end tag whose tag name is one of: "body", "html", "br"

Act as described in the "anything else" entry below.

↳ A start tag whose tag name is "head"

↳ Any other end tag

Parse error. Ignore the token.

↳ Anything else

Act as if an end tag token with the tag name "head" had been seen, and reprocess the current token.

#### 12.2.5.4.5 The "in head noscript" insertion mode

When the user agent is to apply the rules for the "in head noscript" insertion mode, the user agent must handle the token as follows:

↳ A DOCTYPE token

Parse error. Ignore the token.

↳ A start tag whose tag name is "html"

Process the token using the rules for the "in body" insertion mode.

↳ An end tag whose tag name is "noscript"

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

↪ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

↪ A comment token

↪ A start tag whose tag name is one of: "basefont", "bgsound", "link", "meta", "noframes", "style"

Process the token using the rules for the "in head" insertion mode.

↪ An end tag whose tag name is "br"

Act as described in the "anything else" entry below.



↪ A start †

↪ Any o† Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

↪ Anythir /web/20130103030411/http://whatwg.org/html

e current token.

## 12.2.5.4.6 Ti.

When the user agent is to apply the rules for the "after head" insertion mode, the user agent must handle the token as follows:

↪ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

Insert the character into the current node.

↪ A comment token

Append a Comment node to the current node with the data attribute set to the data given in the comment token.

↪ A DOCTYPE token

Parse error. Ignore the token.

↪ A start tag whose tag name is "html"

Process the token using the rules for the "in body" insertion mode.

↪ A start tag whose tag name is "body"

Insert an HTML element for the token.

Set the frameset-ok flag to "not ok".

Switch the insertion mode to "in body".

↪ A start tag whose tag name is "frameset"

Insert an HTML element for the token.

Switch the insertion mode to "in frameset".

↪ A start tag token whose tag name is one of: "base", "basefont", "bgsound", "link", "meta", "noframes", "script", "style", "title"

Parse error.

Push the node pointed to by the head element pointer onto the stack of open elements.

Process the token using the rules for the "in head" insertion mode.

Remove the node pointed to by the head element pointer from the stack of open elements.

**Note: The head element pointer cannot be null at this point.**

↪ An end tag whose tag name is one of: "body", "html", "br"

Act as described in the "anything else" entry below.

↪ A start tag whose tag name is "head"

↪ Any other end tag

Parse error. Ignore the token.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

#### 12.2.5.4.7 The "in body" insertion mode

When the user agent is to apply the rules for the "in body" insertion mode, the user agent must handle the token as follows:

↳ A character token that is U+0000 NULL

Parse error. Ignore the token.

↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

F



Too slow? Try reading the multipage copy of

↳ Any other the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ A comment token

Append a Comment node to the current node with the data attribute set to the data given in the comment token.

↳ A DOCTYPE token

Parse error. Ignore the token.

↳ A start tag whose tag name is "html"

Parse error. For each attribute on the token, check to see if the attribute is already present on the top element of the stack of open elements. If it is not, add the attribute and its corresponding value to that element.

↳ A start tag token whose tag name is one of: "base", "basefont", "bgsound", "command", "link", "meta", "noframes", "script", "style", "title"

Process the token using the rules for the "in head" insertion mode.

↳ A start tag whose tag name is "body"

Parse error.

If the second element on the stack of open elements is not a body element, or, if the stack of open elements has only one node on it, then ignore the token. (fragment case)

Otherwise, set the frameset-ok flag to "not ok"; then, for each attribute on the token, check to see if the attribute is already present on the body element (the second element) on the stack of open elements, and if it is not, add the attribute and its corresponding value to that element.

↳ A start tag whose tag name is "frameset"

Parse error.

If the second element on the stack of open elements is not a body element, or, if the stack of open elements has only one node on it, then ignore the token. (fragment case)

If the frameset-ok flag is set to "not ok", ignore the token.

Otherwise, run the following steps:

1. Remove the second element on the stack of open elements from its parent node, if it has one.
2. Pop all the nodes from the bottom of the stack of open elements, from the current node up to, but not including, the root html element.
3. Insert an HTML element for the token.
4. Switch the insertion mode to "in frameset".

↳ An end-of-file token

If there is a node in the stack of open elements that is not either a dd element, a dt element, an li element, a p element, a tbody element, a td element, a tfoot element, a th element, a thead element, a tr element, the body element, or



Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

1. Set the frameset-ok flag to "not ok".
2. Initialize `node` to be the current node (the bottommost node of the stack).
3. *Loop:* If `node` is a `dd` or `dt` element, then act as if an end tag with the same tag name as `node` had been seen, then jump to the last step.
4. If `node` is in the special category, but is not an `address`, `div`, or `p` element, then jump to the last step.
5. Otherwise, set `node` to the previous entry in the stack of open elements and return to the step labeled *loop*.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

ag with the tag name "p" had

↳ A start tag [//whatwg.org/html](http://whatwg.org/html)

seen.

the tag name "p" had been

Insert an HTML element for the token.

Switch the tokenizer to the PLAINTEXT state.

**Note:** Once a start tag with the tag name "plaintext" has been seen, that will be the last token ever seen other than character tokens (and the end-of-file token), because there is no way to switch out of the PLAINTEXT state.

↳ A start tag whose tag name is "button"

If the stack of open elements has a `button` element in scope, then this is a parse error; act as if an end tag with the tag name "button" had been seen, then reprocess the token.

Otherwise:

Reconstruct the active formatting elements, if any.

Insert an HTML element for the token.

Set the frameset-ok flag to "not ok".

↳ An end tag whose tag name is one of: "address", "article", "aside", "blockquote", "button", "center", "details", "dialog", "dir", "div", "dl", "fieldset", "figcaption", "figure", "footer", "header", "hgroup", "listing", "menu", "nav", "ol", "pre", "section", "summary", "ul"

If the stack of open elements does not have an element in scope with the same tag name as that of the token, then this is a parse error; ignore the token.

Otherwise, run these steps:

1. Generate implied end tags.
2. If the current node is not an element with the same tag name as that of the token, then this is a parse error.
3. Pop elements from the stack of open elements until an element with the same tag name as the token has been popped from the stack.

↳ An end tag whose tag name is "form"

Let `node` be the element that the `form` element pointer is set to.

Set the `form` element pointer to null.

If `node` is null or the stack of open elements does not have `node` in scope, then this is a parse error; ignore the token.

Otherwise, run these steps:

1. Generate implied end tags.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

If the stack of open elements does not have an element in button scope with the same tag name as that of the token, then this is a parse error; act as if a start tag with the tag name "p" had been seen, then reprocess the current token.

Otherwise, run these steps:

1. Generate implied end tags, except for elements with the same tag name as the token.
2. If the current node is not an element with the same tag name as that of the token, then this is a parse error.
3. Pop elements from the stack of open elements until an element with the same tag name as the token has been

↳ **An end** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ame as that of the token,

i.

2. If the current node is not an element with the same tag name as that of the token, then this is a parse error.
3. Pop elements from the stack of open elements until an element with the same tag name as the token has been popped from the stack.

↳ **An end tag whose tag name is one of: "dd", "dt"**

If the stack of open elements does not have an element in scope with the same tag name as that of the token, then this is a parse error; ignore the token.

Otherwise, run these steps:

1. Generate implied end tags, except for elements with the same tag name as the token.
2. If the current node is not an element with the same tag name as that of the token, then this is a parse error.
3. Pop elements from the stack of open elements until an element with the same tag name as the token has been popped from the stack.

↳ **An end tag whose tag name is one of: "h1", "h2", "h3", "h4", "h5", "h6"**

If the stack of open elements does not have an element in scope whose tag name is one of "h1", "h2", "h3", "h4", "h5", or "h6", then this is a parse error; ignore the token.

Otherwise, run these steps:

1. Generate implied end tags.
2. If the current node is not an element with the same tag name as that of the token, then this is a parse error.
3. Pop elements from the stack of open elements until an element whose tag name is one of "h1", "h2", "h3", "h4", "h5", or "h6" has been popped from the stack.

↳ **An end tag whose tag name is "sarcasm"**

Take a deep breath, then act as described in the "any other end tag" entry below.

↳ **A start tag whose tag name is "a"**

If the list of active formatting elements contains an element whose tag name is "a" between the end of the list and the last marker on the list (or the start of the list if there is no marker on the list), then this is a parse error; act as if an end tag with the tag name "a" had been seen, then remove that element from the list of active formatting elements and the stack of open elements if the end tag didn't already remove it (it might not have if the element is not in table scope).

In the non-conforming stream `<a href="a"><a><table><a href="b">b</table>x`, the first `a` element would be closed upon seeing the second one, and the "x" character would be inside a link to "b", not to "a". This is despite the fact that the outer `a` element is not in table scope (meaning that a regular `</a>` end tag at the start of the table wouldn't close the outer `a` element). The result is that the two `a` elements are indirectly nested inside each other — non-conforming markup will often result in non-conforming DOMs when parsed.

Reconstruct the active formatting elements, if any.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Insert an HTML element for the token. Push onto the list of active formatting elements that element.

**↶ A start tag whose tag name is "nobr"**

Reconstruct the active formatting elements, if any.

If the stack of open elements has a `nobr` element in scope, then this is a parse error; act as if an end tag with the tag name "nobr" had been seen, then once again reconstruct the active formatting elements, if any.

Insert an HTML element for the token. Push onto the list of active formatting elements that element.

**↶ An end 'u'**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

"all", "strike", "strong", "tt",

:ps.

4. Let the *formatting element* be the last element in the list of active formatting elements that:

- is between the end of the list and the last scope marker in the list, if any, or the start of the list otherwise, and
- has the same tag name as the token.

If there is no such node, then abort these steps and instead act as described in the "any other end tag" entry below.

Otherwise, if there is such a node, but that node is not in the stack of open elements, then this is a parse error; remove the element from the list, and abort these steps.

Otherwise, if there is such a node, and that node is also in the stack of open elements, but the element is not in scope, then this is a parse error; ignore the token, and abort these steps.

Otherwise, there is a *formatting element* and that element is in the stack and is in scope. If the element is not the current node, this is a parse error. In any case, proceed with the algorithm as written in the following steps.

5. Let the *furthest block* be the topmost node in the stack of open elements that is lower in the stack than the *formatting element*, and is an element in the special category. There might not be one.
6. If there is no *furthest block*, then the UA must first pop all the nodes from the bottom of the stack of open elements, from the current node up to and including the *formatting element*, then remove the *formatting element* from the list of active formatting elements, and finally abort these steps.
7. Let the *common ancestor* be the element immediately above the *formatting element* in the stack of open elements.
8. Let a bookmark note the position of the *formatting element* in the list of active formatting elements relative to the elements on either side of it in the list.
9. Let *node* and *last node* be the *furthest block*. Follow these steps:
  1. Let *inner loop counter* be zero.
  2. *Inner loop*: If *inner loop counter* is greater than or equal to three, then go to the next step in the overall algorithm.
  3. Increment *inner loop counter* by one.
  4. Let *node* be the element immediately above *node* in the stack of open elements, or if *node* is no longer in the stack of open elements (e.g. because it got removed by the next step), the element that was immediately above *node* in the stack of open elements before *node* was removed.
  5. If *node* is not in the list of active formatting elements, then remove *node* from the stack of open elements and then go back to the step labeled *inner loop*.
  6. Otherwise, if *node* is the *formatting element*, then go to the next step in the overall algorithm.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

new *node* in the list of active formatting elements.

9. Insert *last node* into *node*, first removing it from its previous parent node if any.

10. Let *last node* be *node*.

11. Return to the step labeled *inner loop*.

10. If the *common ancestor* node is a `table`, `tbody`, `tfoot`, `thead`, or `tr` element, then, foster parent whatever *last node* ended up being in the previous step, first removing it from its previous parent node if any.

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

14. Remove the *formatting element* from the list of active formatting elements, and insert the new element into the list of active formatting elements at the position of the aforementioned bookmark.

15. Remove the *formatting element* from the stack of open elements, and insert the new element into the stack of open elements immediately below the position of the *furthest block* in that stack.

16. Jump back to the step labeled *outer loop*.

**Note: Because of the way this algorithm causes elements to change parents, it has been dubbed the "adoption agency algorithm" (in contrast with other possible algorithms for dealing with misnested content, which included the "incest algorithm", the "secret affair algorithm", and the "Heisenberg algorithm").**

#### ↳ A start tag token whose tag name is one of: "applet", "marquee", "object"

Reconstruct the active formatting elements, if any.

Insert an HTML element for the token.

Insert a marker at the end of the list of active formatting elements.

Set the frameset-ok flag to "not ok".

#### ↳ An end tag token whose tag name is one of: "applet", "marquee", "object"

If the stack of open elements does not have an element in scope with the same tag name as that of the token, then this is a parse error; ignore the token.

Otherwise, run these steps:

1. Generate implied end tags.
2. If the current node is not an element with the same tag name as that of the token, then this is a parse error.
3. Pop elements from the stack of open elements until an element with the same tag name as the token has been popped from the stack.
4. Clear the list of active formatting elements up to the last marker.

#### ↳ A start tag whose tag name is "table"

If the `Document` is *not* set to quirks mode, and the stack of open elements has a `p` element in button scope, then act as if an end tag with the tag name "p" had been seen.

Insert an HTML element for the token.

Set the frameset-ok flag to "not ok".

Switch the insertion mode to "in table".

#### ↳ A start tag whose tag name is one of: "area", "br", "embed", "img", "keygen", "wbr"

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Set the frameset-ok flag to "not ok".

↶ A start tag whose tag name is "input"

Reconstruct the active formatting elements, if any.

Insert an HTML element for the token. Immediately pop the current node off the stack of open elements.

Acknowledge the token's *self-closing flag*, if it is set.

If the `value` attribute of the resulting `input` element is not an ASCII case-  
elements.

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↶ A start tag whose tag name is "p"

If the `value` attribute of the resulting `p` element is not an ASCII case-  
seen.

Insert an HTML element for the token. Immediately pop the current node off the stack of open elements.

Acknowledge the token's *self-closing flag*, if it is set.

Set the frameset-ok flag to "not ok".

↶ A start tag whose tag name is "image"

Parse error. Change the token's tag name to "img" and reprocess it. (Don't ask.)

↶ A start tag whose tag name is "isindex"

Parse error.

If the `form` element pointer is not null, then ignore the token.

Otherwise:

Acknowledge the token's *self-closing flag*, if it is set.

Act as if a start tag token with the tag name "form" had been seen.

If the token has an attribute called "action", set the `action` attribute on the resulting `form` element to the value of the "action" attribute of the token.

Act as if a start tag token with the tag name "hr" had been seen.

Act as if a start tag token with the tag name "label" had been seen.

Act as if a stream of character tokens had been seen (see below for what they should say).

Act as if a start tag token with the tag name "input" had been seen, with all the attributes from the "isindex" token except "name", "action", and "prompt". Set the `name` attribute of the resulting `input` element to the value "isindex".

Act as if a stream of character tokens had been seen (see below for what they should say).

Act as if an end tag token with the tag name "label" had been seen.

Act as if a start tag token with the tag name "hr" had been seen.

Act as if an end tag token with the tag name "form" had been seen.

If the token has an attribute with the name "prompt", then the first stream of characters must be the same string as given in that attribute, and the second stream of characters must be empty. Otherwise, the two streams of character tokens together should, together with the `input` element, express the equivalent of "This is a searchable index. Enter search keywords: (input field)" in the user's preferred language.

↶ A start tag whose tag name is "textarea"

Run these steps:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

3. Switch the tokenizer to the RCDATA state.
4. Let the original insertion mode be the current insertion mode.
5. Set the frameset-ok flag to "not ok".
6. Switch the insertion mode to "text".

↳ A start tag whose tag name is "xmp"

If the stack of open elements has a `p` element in button scope, then act as if an end tag with the tag name "p" had been seen.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ A start tag whose tag name is "script"

Follow the generic raw text element parsing algorithm.

↳ A start tag whose tag name is "noembed"

↳ A start tag whose tag name is "noscript", if the scripting flag is enabled

Follow the generic raw text element parsing algorithm.

↳ A start tag whose tag name is "select"

Reconstruct the active formatting elements, if any.

Insert an HTML element for the token.

Set the frameset-ok flag to "not ok".

If the insertion mode is one of "in table", "in caption", "in table body", "in row", or "in cell", then switch the insertion mode to "in select in table". Otherwise, switch the insertion mode to "in select".

↳ A start tag whose tag name is one of: "optgroup", "option"

If the current node is an `option` element, then act as if an end tag with the tag name "option" had been seen.

Reconstruct the active formatting elements, if any.

Insert an HTML element for the token.

↳ A start tag whose tag name is one of: "rp", "rt"

If the stack of open elements has a `ruby` element in scope, then generate implied end tags. If the current node is not then a `ruby` element, this is a parse error.

Insert an HTML element for the token.

↳ An end tag whose tag name is "br"

Parse error. Act as if a start tag token with the tag name "br" had been seen. Ignore the end tag token.

↳ A start tag whose tag name is "math"

Reconstruct the active formatting elements, if any.

Adjust MathML attributes for the token. (This fixes the case of MathML attributes that are not all lowercase.)

Adjust foreign attributes for the token. (This fixes the use of namespaced attributes, in particular XLink.)

Insert a foreign element for the token, in the MathML namespace.

If the token has its *self-closing flag* set, pop the current node off the stack of open elements and acknowledge the token's *self-closing flag*.

↳ A start tag whose tag name is "svg"

Reconstruct the active formatting elements, if any.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

If the token has its *self-closing flag* set, pop the current node off the stack of open elements and acknowledge the token's *self-closing flag*.

↳ A start tag whose tag name is one of: "caption", "col", "colgroup", "frame", "head", "tbody", "td", "tfoot", "th", "thead", "tr"

Parse error. Ignore the token.

↳ Any other start tag

Reconstruct the active formatting elements, if any.



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ Any off /web/20130103030411/http://whatwg.org/html

2. *Loop*: If *node* has the same tag name as the token, then:

1. Generate implied end tags, except for elements with the same tag name as the token.

2. If the tag name of the end tag token does not match the tag name of the current node, this is a parse error.

3. Pop all the nodes from the current node up to *node*, including *node*, then stop these steps.

3. Otherwise, if *node* is in the special category, then this is a parse error; ignore the token, and abort these steps.

4. Set *node* to the previous entry in the stack of open elements.

5. Return to the step labeled *loop*.

#### 12.2.5.4.8 The "text" insertion mode

When the user agent is to apply the rules for the "text" insertion mode, the user agent must handle the token as follows:

↳ A character token

Insert the token's character into the current node.

**Note: This can never be a U+0000 NULL character; the tokenizer converts those to U+FFFD REPLACEMENT CHARACTER characters.**

↳ An end-of-file token

Parse error.

If the current node is a `script` element, mark the `script` element as "already started".

Pop the current node off the stack of open elements.

Switch the insertion mode to the original insertion mode and reprocess the current token.

↳ An end tag whose tag name is "script"

Perform a microtask checkpoint.

Provide a stable state.

Let `script` be the current node (which will be a `script` element).

Pop the current node off the stack of open elements.

Switch the insertion mode to the original insertion mode.

Let the *old insertion point* have the same value as the current insertion point. Let the insertion point be just before the next input character.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Decrement the parser's script nesting level by one. If the parser's script nesting level is zero, then set the parser pause flag to false.

Let the insertion point have the value of the *old insertion point*. (In other words, restore the insertion point to its previous value. This value might be the "undefined" value.)

At this stage, if there is a pending parsing-blocking script, then:

↳ **If the script nesting level is not zero:**

Set the parser pause flag to true, and abort the processing of any nested invocations of the tokenizer, which returns to the "outer" tree

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

*ed reentrantly, say*

1. Let *the script* be the pending parsing-blocking script. There is no longer a pending parsing-blocking script.
2. Block the tokenizer for this instance of the HTML parser, such that the event loop will not run tasks that invoke the tokenizer.
3. If the parser's `Document` has a style sheet that is blocking scripts or *the script*'s "ready to be parser-executed" flag is not set: spin the event loop until the parser's `Document` has no style sheet that is blocking scripts and *the script*'s "ready to be parser-executed" flag is set.
4. Unblock the tokenizer for this instance of the HTML parser, such that tasks that invoke the tokenizer can again be run.
5. Let the insertion point be just before the next input character.
6. Increment the parser's script nesting level by one (it should be zero before this step, so this sets it to one).
7. Execute *the script*.
8. Decrement the parser's script nesting level by one. If the parser's script nesting level is zero (which it always should be at this point), then set the parser pause flag to false.
9. Let the insertion point be undefined again.
10. If there is once again a pending parsing-blocking script, then repeat these steps from step 1.

↳ **Any other end tag**

Pop the current node off the stack of open elements.

Switch the insertion mode to the original insertion mode.

#### 12.2.5.4.9 The "in table" insertion mode

When the user agent is to apply the rules for the "in table" insertion mode, the user agent must handle the token as follows:

↳ **A character token, if the current node is `table`, `tbody`, `tfoot`, `thead`, or `tr` element**

Let the *pending table character tokens* be an empty list of tokens.

Let the original insertion mode be the current insertion mode.

Switch the insertion mode to "in table text" and reprocess the token.

↳ **A comment token**

Append a `Comment` node to the current node with the `data` attribute set to the data given in the comment token.

↳ **A DOCTYPE token**

Parse error. Ignore the token.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Insert an HTML element for the token, then switch the insertion mode to "in caption".

↶ A start tag whose tag name is "colgroup"

Clear the stack back to a table context. (See below.)

Insert an HTML element for the token, then switch the insertion mode to "in column group".

↶ A start tag whose tag name is "col"

Act as if a start tag token with the tag name "colgroup" had been seen, then reprocess the current token.

↶ A start '

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>

↶ A start /[web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

it token.

↶ A start .

Parse error. Act as if an end tag token with the tag name "table" had been seen, then, if that token wasn't ignored, reprocess the current token.

**Note: The fake end tag token here can only be ignored in the fragment case.**

↶ An end tag whose tag name is "table"

If the stack of open elements does not have an element in table scope with the same tag name as the token, this is a parse error. Ignore the token. (fragment case)

Otherwise:

Pop elements from this stack until a `table` element has been popped from the stack.

Reset the insertion mode appropriately.

↶ An end tag whose tag name is one of: "body", "caption", "col", "colgroup", "html", "tbody", "td", "tfoot", "th", "thead", "tr"

Parse error. Ignore the token.

↶ A start tag whose tag name is one of: "style", "script"

Process the token using the rules for the "in head" insertion mode.

↶ A start tag whose tag name is "input"

If the token does not have an attribute with the name "type", or if it does, but that attribute's value is not an ASCII case-insensitive match for the string "hidden", then: act as described in the "anything else" entry below.

Otherwise:

Parse error.

Insert an HTML element for the token.

Pop that `input` element off the stack of open elements.

Acknowledge the token's *self-closing flag*, if it is set.

↶ A start tag whose tag name is "form"

Parse error.

If the `form` element pointer is not null, ignore the token.

Otherwise:

Insert an HTML element for the token, and set the `form` element pointer to point to the element created.

Pop that `form` element off the stack of open elements.

↶ An end-of-file token

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Stop parsing.

#### ↳ Anything else

Parse error. Process the token using the rules for the "in body" insertion mode, except that whenever a node would be inserted into the current node when the current node is a `table`, `tbody`, `tfoot`, `thead`, or `tr` element, then it must instead be foster parented.

When the steps above require the UA to **clear the stack back to a table context**, it means that the UA must, while the current node is not a `table` element or an `html` element, pop elements from the stack of open elements.

**Note:** T

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org>**

**12.2.5.4.10**

When the user agent handles the token as follows:

#### ↳ A character token

##### ↳ Any other character token

Append the character token to the *pending table character tokens* list.

#### ↳ Anything else

If any of the tokens in the *pending table character tokens* list are character tokens that are not space characters, then reprocess the character tokens in the *pending table character tokens* list using the rules given in the "anything else" entry in the "in table" insertion mode.

Otherwise, insert the characters given by the *pending table character tokens* list into the current node.

Switch the insertion mode to the original insertion mode and reprocess the token.

**12.2.5.4.11 The "in caption" insertion mode**

When the user agent is to apply the rules for the "in caption" insertion mode, the user agent must handle the token as follows:

#### ↳ An end tag whose tag name is "caption"

If the stack of open elements does not have an element in table scope with the same tag name as the token, this is a parse error. Ignore the token. (fragment case)

Otherwise:

Generate implied end tags.

Now, if the current node is not a `caption` element, then this is a parse error.

Pop elements from this stack until a `caption` element has been popped from the stack.

Clear the list of active formatting elements up to the last marker.

Switch the insertion mode to "in table".

#### ↳ A start tag whose tag name is one of: "caption", "col", "colgroup", "tbody", "td", "tfoot", "th", "thead", "tr"

#### ↳ An end tag whose tag name is "table"

Parse error. Act as if an end tag with the tag name "caption" had been seen, then, if that token wasn't ignored, reprocess the current token.

**Note:** The fake end tag token here can only be ignored in the fragment case.

#### ↳ An end tag whose tag name is one of: "body", "col", "colgroup", "html", "tbody", "td", "tfoot", "th", "thead", "tr"

Parse error. Ignore the token.

#### ↳ Anything else

Process the token using the rules for the "in body" insertion mode.

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

U+000D CARRIAGE RETURN (CR), OR U+0020 SPACE

Insert the character into the current node.

## ↳ A comment token

Append a Comment node to the current node with the data attribute set to the data given in the comment token.

## ↳ A DOCTYPE token

Parse error. Ignore the token.

## ↳ A start tag whose tag name is "html"



Too slow? Try reading the multipage copy of

## ↳ A start the spec instead: https://web.archive.org

elements.

/web/20130103030411/http:

## ↳ An end //whatwg.org/html

(fragment case)

Otherwise, pop the current node (which will be a colgroup element) from the stack of open elements. Switch the insertion mode to "in table".

## ↳ An end tag whose tag name is "col"

Parse error. Ignore the token.

## ↳ An end-of-file token

If the current node is the root html element, then stop parsing. (fragment case)

Otherwise, act as described in the "anything else" entry below.

## ↳ Anything else

Act as if an end tag with the tag name "colgroup" had been seen, and then, if that token wasn't ignored, reprocess the current token.

**Note: The fake end tag token here can only be ignored in the fragment case.****12.2.5.4.13 The "in table body" insertion mode**

When the user agent is to apply the rules for the "in table body" insertion mode, the user agent must handle the token as follows:

## ↳ A start tag whose tag name is "tr"

Clear the stack back to a table body context. (See below.)

Insert an HTML element for the token, then switch the insertion mode to "in row".

## ↳ A start tag whose tag name is one of: "th", "td"

Parse error. Act as if a start tag with the tag name "tr" had been seen, then reprocess the current token.

## ↳ An end tag whose tag name is one of: "tbody", "tfoot", "thead"

If the stack of open elements does not have an element in table scope with the same tag name as the token, this is a parse error. Ignore the token.

Otherwise:

Clear the stack back to a table body context. (See below.)

Pop the current node from the stack of open elements. Switch the insertion mode to "in table".

## ↳ A start tag whose tag name is one of: "caption", "col", "colgroup", "tbody", "tfoot", "thead"

## ↳ An end tag whose tag name is "table"

If the stack of open elements does not have a tbody, thead, or tfoot element in table scope, this is a parse error. Ignore the token. (fragment case)

Otherwise:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- ↳ An end tag whose tag name is one of: "body", "caption", "col", "colgroup", "html", "td", "th", "tr"
  - Parse error. Ignore the token.

#### ↳ Anything else

Process the token using the rules for the "in table" insertion mode.

When the steps above require the UA to **clear the stack back to a table body context**, it means that the UA must, while the current node is not a `tbody`, `tfoot`, `thead`, or `html` element, pop elements from the stack of open elements.

**Note: Th**

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

When the user agent sees a start tag, it processes the token as follows:

- ↳ A start tag whose tag name is "tbody"

When the user agent sees a start tag, it processes the token as follows:

Insert an HTML element for the token, then switch the insertion mode to "in cell".

Insert a marker at the end of the list of active formatting elements.

#### ↳ An end tag whose tag name is "tr"

If the stack of open elements does not have an element in table scope with the same tag name as the token, this is a parse error. Ignore the token. (fragment case)

Otherwise:

Clear the stack back to a table row context. (See below.)

Pop the current node (which will be a `tr` element) from the stack of open elements. Switch the insertion mode to "in table body".

#### ↳ A start tag whose tag name is one of: "caption", "col", "colgroup", "tbody", "tfoot", "thead", "tr"

#### ↳ An end tag whose tag name is "table"

Act as if an end tag with the tag name "tr" had been seen, then, if that token wasn't ignored, reprocess the current token.

**Note: The fake end tag token here can only be ignored in the fragment case.**

#### ↳ An end tag whose tag name is one of: "tbody", "tfoot", "thead"

If the stack of open elements does not have an element in table scope with the same tag name as the token, this is a parse error. Ignore the token.

Otherwise, act as if an end tag with the tag name "tr" had been seen, then reprocess the current token.

#### ↳ An end tag whose tag name is one of: "body", "caption", "col", "colgroup", "html", "td", "th"

Parse error. Ignore the token.

#### ↳ Anything else

Process the token using the rules for the "in table" insertion mode.

When the steps above require the UA to **clear the stack back to a table row context**, it means that the UA must, while the current node is not a `tr` element or an `html` element, pop elements from the stack of open elements.

**Note: The current node being an html element after this process is a fragment case.**

#### 12.2.5.4.15 The "in cell" insertion mode

When the user agent is to apply the rules for the "in cell" insertion mode, the user agent must handle the token as follows:

#### ↳ An end tag whose tag name is one of: "td", "th"

If the stack of open elements does not have an element in table scope with the same tag name as that of the token, then this is a parse error and the token must be ignored.



Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

First, if the current node is an `option` element, and the node immediately before it in the stack of open elements is an `optgroup` element, then act as if an end tag with the tag name "option" had been seen.

If the current node is an `optgroup` element, then pop that node from the stack of open elements. Otherwise, this is a parse error; ignore the token.

#### ↳ An end tag whose tag name is "option"

If the current node is an `option` element, then pop that node from the stack of open elements. Otherwise, this is a parse error; ignore the token.



#### ↳ An end Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ne as the token, this is a

in the stack.

#### ↳ A start tag whose tag name is "select"

Parse error. Act as if the token had been an end tag with the tag name "select" instead.

#### ↳ A start tag whose tag name is one of: "input", "keygen", "textarea"

Parse error.

If the stack of open elements does not have a `select` element in select scope, ignore the token. (fragment case)

Otherwise, act as if an end tag with the tag name "select" had been seen, and reprocess the token.

#### ↳ A start tag token whose tag name is "script"

Process the token using the rules for the "in head" insertion mode.

#### ↳ An end-of-file token

If the current node is not the root `html` element, then this is a parse error.

**Note:** The current node can only be the root `html` element in the fragment case.

Stop parsing.

#### ↳ Anything else

Parse error. Ignore the token.

### 12.2.5.4.17 The "in select in table" insertion mode

When the user agent is to apply the rules for the "in select in table" insertion mode, the user agent must handle the token as follows:

#### ↳ A start tag whose tag name is one of: "caption", "table", "tbody", "tfoot", "thead", "tr", "td", "th"

Parse error. Act as if an end tag with the tag name "select" had been seen, and reprocess the token.

#### ↳ An end tag whose tag name is one of: "caption", "table", "tbody", "tfoot", "thead", "tr", "td", "th"

Parse error.

If the stack of open elements has an element in table scope with the same tag name as that of the token, then act as if an end tag with the tag name "select" had been seen, and reprocess the token. Otherwise, ignore the token.

#### ↳ Anything else

Process the token using the rules for the "in select" insertion mode.

### 12.2.5.4.18 The "after body" insertion mode

When the user agent is to apply the rules for the "after body" insertion mode, the user agent must handle the token as follows:

#### ↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF),

[Go](#)

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Appending a Comment node to the first element in the stack of open elements (the `html` element), with the `data` attribute set to the data given in the comment token.

## ↳ A DOCTYPE token

Parse error. Ignore the token.

## ↳ A start tag whose tag name is "html"

Process the token using the rules for the "in body" insertion mode.

## ↳ An end tag whose tag name is "html"



Parse error; ignore the token.

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

## ↳ An end /web/20130103030411/http:

## ↳ Anything //whatwg.org/html

Parse error. Switch the insertion mode to "in body" and reprocess the token.

**12.2.5.4.19 The "in frameset" insertion mode**

When the user agent is to apply the rules for the "in frameset" insertion mode, the user agent must handle the token as follows:

## ↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

Insert the character into the current node.

## ↳ A comment token

Append a Comment node to the current node with the `data` attribute set to the data given in the comment token.

## ↳ A DOCTYPE token

Parse error. Ignore the token.

## ↳ A start tag whose tag name is "html"

Process the token using the rules for the "in body" insertion mode.

## ↳ A start tag whose tag name is "frameset"

Insert an HTML element for the token.

## ↳ An end tag whose tag name is "frameset"

If the current node is the root `html` element, then this is a parse error; ignore the token. (fragment case)

Otherwise, pop the current node from the stack of open elements.

If the parser was *not* originally created as part of the HTML fragment parsing algorithm (fragment case), and the current node is no longer a `frameset` element, then switch the insertion mode to "after frameset".

## ↳ A start tag whose tag name is "frame"

Insert an HTML element for the token. Immediately pop the current node off the stack of open elements.

Acknowledge the token's *self-closing flag*, if it is set.

## ↳ A start tag whose tag name is "noframes"

Process the token using the rules for the "in head" insertion mode.

## ↳ An end-of-file token

If the current node is not the root `html` element, then this is a parse error.

**Note: The current node can only be the root `html` element in the fragment case.**

Stop parsing.

## ↳ Anything else

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the user agent is to apply the rules for the "after frameset" insertion mode, the user agent must handle the token as follows:

- ↪ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

Insert the character into the current node.

- ↪ A comment token

Append a Comment node to the current node with the data attribute set to the data given in the comment token.

- ↪ A DOCTYPE token



Too slow? Try reading the multipage copy of

- ↪ A start tag whose tag name is "html"
- ↪ the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

- ↪ An end tag whose tag name is "html"

Process the token using the rules for the "in head" insertion mode.

- ↪ An end-of-file token

Stop parsing.

- ↪ Anything else

Parse error. Ignore the token.

#### 12.2.5.4.21 The "after after body" insertion mode

When the user agent is to apply the rules for the "after after body" insertion mode, the user agent must handle the token as follows:

- ↪ A comment token

Append a Comment node to the Document object with the data attribute set to the data given in the comment token.

- ↪ A DOCTYPE token

- ↪ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

- ↪ A start tag whose tag name is "html"

Process the token using the rules for the "in body" insertion mode.

- ↪ An end-of-file token

Stop parsing.

- ↪ Anything else

Parse error. Switch the insertion mode to "in body" and reprocess the token.

#### 12.2.5.4.22 The "after after frameset" insertion mode

When the user agent is to apply the rules for the "after after frameset" insertion mode, the user agent must handle the token as follows:

- ↪ A comment token

Append a Comment node to the Document object with the data attribute set to the data given in the comment token.

- ↪ A DOCTYPE token

- ↪ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR), or U+0020 SPACE

- ↪ A start tag whose tag name is "html"

Process the token using the rules for the "in body" insertion mode.

- ↪ An end-of-file token

Stop parsing.

- ↪ A start tag whose tag name is "noframes"

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

### 12.2.5.5 The rules for parsing tokens in foreign content

When the user agent is to apply the rules for parsing tokens in foreign content, the user agent must handle the token as follows:

↳ A character token that is U+0000 NULL

Parse error. Insert a U+FFFD REPLACEMENT CHARACTER character into the current node.

↳ A character token that is one of U+0009 CHARACTER TABULATION, U+000A LINE FEED (LF), U+000C FORM FEED (FF), U+000D CARRIAGE RETURN (CR) or U+0020 SPACE



Too slow? Try reading the multipage copy of

↳ Any other

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

↳ A comment

//[whatwg.org/html](http://whatwg.org/html)

.he comment token.

↳ A DOCTYPE token

Parse error. Ignore the token.

↳ A start tag whose tag name is one of: "b", "big", "blockquote", "body", "br", "center", "code", "dd", "div", "dl", "dt", "em", "embed", "h1", "h2", "h3", "h4", "h5", "h6", "head", "hr", "i", "img", "li", "listing", "menu", "meta", "nobr", "ol", "p", "pre", "ruby", "s", "small", "span", "strong", "strike", "sub", "sup", "table", "tt", "u", "ul", "var"

↳ A start tag whose tag name is "font", if the token has any attributes named "color", "face", or "size"

Parse error.

Pop an element from the stack of open elements, and then keep popping more elements from the stack of open elements until the current node is a MathML text integration point, an HTML integration point, or an element in the HTML namespace.

Then, reprocess the token.

↳ Any other start tag

If the current node is an element in the MathML namespace, adjust MathML attributes for the token. (This fixes the case of MathML attributes that are not all lowercase.)

If the current node is an element in the SVG namespace, and the token's tag name is one of the ones in the first column of the following table, change the tag name to the name given in the corresponding cell in the second column. (This fixes the case of SVG elements that are not all lowercase.)

Tag name	Element name
altglyph	altGlyph
altglyphdef	altGlyphDef
altglyphitem	altGlyphItem
animatecolor	animateColor
animatemotion	animateMotion
animatetransform	animateTransform
clippath	clipPath
feblend	feBlend
fecolormatrix	feColorMatrix
fecomponenttransfer	feComponentTransfer
fecomposite	feComposite
feconvolvematrix	feConvolveMatrix
fediffuselighting	feDiffuseLighting
fedisplacementmap	feDisplacementMap
fedistantlight	feDistantLight
feflood	feFlood
fefunca	feFuncA

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

3,445 captures

5 Jun 2004 - 22 Aug 2020

fegaussianblur	feGaussianBlur
feimage	feImage
femerge	feMerge
femergenode	feMergeNode
femorphology	feMorphology
feoffset	feOffset
fepointlight	fePointLight
fespecularlighting	feSpecularLighting



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

If the current node is an element in the SVG namespace, adjust SVG attributes for the token. (This fixes the case of SVG attributes that are not all lowercase.)

Adjust foreign attributes for the token. (This fixes the use of namespaced attributes, in particular XLink in SVG.)

Insert a foreign element for the token, in the same namespace as the current node.

If the token has its *self-closing flag* set, then run the appropriate steps from the following list:

↪ If the token's tag name is "script"

Acknowledge the token's *self-closing flag*, and then act as if an end tag with the tag name "script" had been seen.

↪ Otherwise

Pop the current node off the stack of open elements and acknowledge the token's *self-closing flag*.

↪ An end tag whose tag name is "script", if the current node is a script element in the SVG namespace

Pop the current node off the stack of open elements.

Let the *old insertion point* have the same value as the current insertion point. Let the insertion point be just before the next input character.

Increment the parser's script nesting level by one. Set the parser pause flag to true.

Process the `script` element according to the SVG rules, if the user agent supports SVG. [SVG]

**Note: Even if this causes new characters to be inserted into the tokenizer, the parser will not be executed reentrantly, since the parser pause flag is true.**

Decrement the parser's script nesting level by one. If the parser's script nesting level is zero, then set the parser pause flag to false.

Let the insertion point have the value of the *old insertion point*. (In other words, restore the insertion point to its previous value. This value might be the "undefined" value.)

↪ Any other end tag

Run these steps:

1. Initialize `node` to be the current node (the bottommost node of the stack).
2. If `node` is not an element with the same tag name as the token, then this is a parse error.
3. Loop: If `node`'s tag name, converted to ASCII lowercase, is the same as the tag name of the token, pop elements from the stack of open elements until `node` has been popped from the stack, and then abort these steps.
4. Set `node` to the previous entry in the stack of open elements.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## 12.2.6 The end

Once the user agent **stops parsing** the document, the user agent must run the following steps:

1. Set the current document readiness to "interactive" and the insertion point to undefined.

2. Pop *all* the nodes off the stack of open elements.

3. If the list of scripts that will execute when the document has finished parsing is not empty, then repeat these substeps:

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

again from substep 1.

4. Queue a task to fire a simple event that bubbles named `DOMContentLoaded` at the Document.

5. Spin the event loop until the set of scripts that will execute as soon as possible and the list of scripts that will execute in order as soon as possible are empty.

6. Spin the event loop until there is nothing that **delays the load event** in the Document.

7. Queue a task to run the following substeps:

1. Set the current document readiness to "complete".

2. If the Document is in a browsing context, fire a simple event named `load` at the Document's Window object, but with its target set to the Document object (and the `currentTarget` set to the Window object).

8. If the Document is in a browsing context, then queue a task to run the following substeps:

1. If the Document's page showing flag is true, then abort this task (i.e. don't fire the event below).

2. Set the Document's page showing flag to true.

3. Fire a trusted event with the name `pageshow` at the Window object of the Document, but with its target set to the Document object (and the `currentTarget` set to the Window object), using the `PageTransitionEvent` interface, with the `persisted` attribute initialized to false. This event must not bubble, must not be cancelable, and has no default action.

9. If the Document has any pending application cache download process tasks, then queue each such task in the order they were added to the list of pending application cache download process tasks, and then empty the list of pending application cache download process tasks. The task source for these tasks is the networking task source.

10. If the Document's print when loaded flag is set, then run the printing steps.

11. The Document is now **ready for post-load tasks**.

12. Queue a task to mark the Document as **completely loaded**.

When the user agent is to **abort a parser**, it must run the following steps:

1. Throw away any pending content in the input stream, and discard any future content that would have been added to it.

2. Set the current document readiness to "interactive".

3. Pop *all* the nodes off the stack of open elements.

4. Set the current document readiness to "complete".

Except where otherwise specified, the task source for the tasks mentioned in this section is the DOM manipulation task source.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

that isn't included in the DOM itself. This section specifies some rules for handling these issues.

If the XML API being used doesn't support DOCTYPEs, the tool may drop DOCTYPEs altogether.

If the XML API doesn't support attributes in no namespace that are named "`xmllns`", attributes whose names start with "`xmlns:`", or attributes in the XMLNS namespace, then the tool may drop such attributes.

The tool may annotate the output with any namespace declarations required for proper operation.

If the XML API being used restricts the allowable characters in the local names of elements and attributes, then the tool may map all characters to either a legal HTML element name or a well-formed XML element name.

## Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

/web/20130103030411/http://whatwg.org/html

As all attributes will have a prefix `xlink` and a local name `href`. However, used on an `HTML` element, it becomes an attribute with no prefix and the local name `"xlink:href"`, which is not a valid NCName, and thus might not be accepted by an XML API. It could thus get converted, becoming `"xlinkU00003Ahref"`.

**Note:** *The resulting names from this conversion conveniently can't clash with any attribute generated by the HTML parser, since those are all either lowercase or those listed in the adjust foreign attributes algorithm's table.*

If the XML API restricts comments from having two consecutive U+002D HYPHEN-MINUS characters (--), the tool may insert a single U+0020 SPACE character between any such offending characters.

If the XML API restricts comments from ending in a U+002D HYPHEN-MINUS character (-), the tool may insert a single U+0020 SPACE character at the end of such comments.

If the XML API restricts allowed characters in character data, attribute values, or comments, the tool may replace any U+000C FORM FEED (FF) character with a U+0020 SPACE character, and any other literal non-XML character with a U+FFFD REPLACEMENT CHARACTER.

If the tool has no way to convey out-of-band information, then the tool may drop the following information:

- Whether the document is set to *no-quirks mode*, *limited-quirks mode*, or *quirks mode*
- The association between form controls and forms that aren't their nearest `form` element ancestor (use of the `form` element pointer in the parser)

**Note:** *The mutations allowed by this section apply after the HTML parser's rules have been applied. For example, a `<a::>` start tag will be closed by a `</a::>` end tag, and never by a `</aU00003AU00003A>` end tag, even if the user agent is using the rules above to then generate an actual element in the DOM with the name `aU00003AU00003A` for that start tag.*

## 12.2.8 An introduction to error handling and strange cases in the parser

*This section is non-normative.*

This section examines some erroneous markup and discusses how the HTML parser handles these cases.

### 12.2.8.1 Misnested tags: `<b><i></b></i>`

*This section is non-normative.*

The most-often discussed example of erroneous markup is as follows:

```
<p>12<i>34</i>5</p>
```

The parsing of this markup is straightforward up to the "3". At this point, the DOM looks like this:

Go

NOV JAN FEB

03

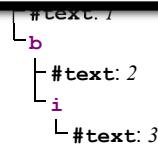
2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

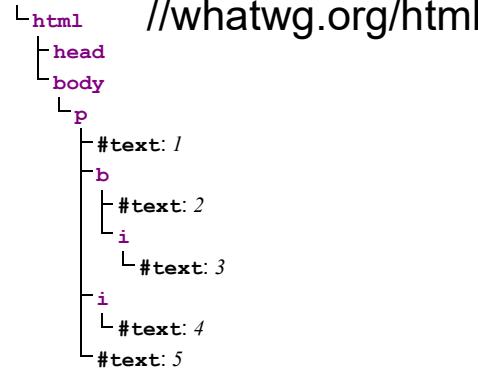


Here, the stack of open elements has five elements on it: `html`, `body`, `p`, `b`, and `i`. The list of active formatting elements just has two: `b` and `i`. The insertion mode is "in body".

Upon receiving "`</b>`" the parsing here is straightforward: the stack of open elements ends up with just three elements: `html`, `body` and `i` at this point.

The next token is an `i` element. The DOM location of the `i` element is

`/web/20130103030411/http://whatwg.org/html`



is just the `i` element. A new "5" Text node is inserted,

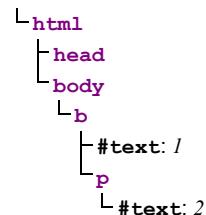
#### 12.2.8.2 Misnested tags: `<b><p></b></p>`

*This section is non-normative.*

A case similar to the previous one is the following:

```
1<p>23</p>
```

Up to the "2" the parsing here is straightforward:



The interesting part is when the end tag token with the tag name "b" is parsed.

Before that token is seen, the stack of open elements has four elements on it: `html`, `body`, `b`, and `p`. The list of active formatting elements just has the one: `b`. The insertion mode is "in body".

Upon receiving the end tag token with the tag name "b", the "adoption agency algorithm" is invoked, as in the previous example. However, in this case, there is a *furthest block*, namely the `p` element. Thus, this time the adoption agency algorithm isn't skipped over.

The *common ancestor* is the `body` element. A conceptual "bookmark" marks the position of the `b` in the list of active formatting elements, but since that list has only one element in it, the bookmark won't have much effect.

As the algorithm progresses, *node* ends up set to the formatting element (`b`), and *last node* ends up set to the *furthest block* (`p`).

The *last node* gets appended (moved) to the *common ancestor*, so that the DOM looks like:



Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

 p
 #text: 2

```

A new **b** element is created, and the children of the **p** element are moved to it:

```

 L html
 L head
 L body
 L b
 L #text: .
 L b
 L #text: Too slow? Try reading the multipage copy of
 L #text: the spec instead: https://web.archive.org
Finally, the r /web/20130103030411/http:
 L html
 L head
 L body
 L b
 L #text: 1
 L p
 L b
 L #text: 2

```

The **b** element is removed from the list of active formatting elements and the stack of open elements, so that when the "3" is parsed, it is appended to the **p** element:

```

 L html
 L head
 L body
 L b
 L #text: 1
 L p
 L b
 L #text: 2
 L #text: 3

```

### 12.2.8.3 Unexpected markup in tables

*This section is non-normative.*

Error handling in tables is, for historical reasons, especially strange. For example, consider the following markup:

```
<table><tr><td>aaa</td></tr>bbb</table>ccc
```

The highlighted **b** element start tag is not allowed directly inside a table like that, and the parser handles this case by placing the element *before* the table. (This is called *foster parenting*.) This can be seen by examining the DOM tree as it stands just after the **table** element's start tag has been seen:

```

 L html
 L head
 L body
 L table

```

...and then immediately after the **b** element start tag has been seen:

```

 L html
 L head
 L body
 L b
 L table

```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

the DOM looks as follows:

```

└ html
 └ head
 └ body
 └ b
 └ table
 └ tbody
 └ tr

```



Here, the stack has the `b` element. The `td` element switches to `table`:

```

└ html
 └ head
 └ body
 └ b
 └ table
 └ tbody
 └ tr
 └ td

```

The marker means that when the "aaa" character tokens are seen, no `b` element is created to hold the resulting `Text` node:

```

└ html
 └ head
 └ body
 └ b
 └ table
 └ tbody
 └ tr
 └ td
 └ #text: aaa

```

The end tags are handled in a straight-forward manner; after handling them, the stack of open elements has on it the elements `html`, `body`, `table`, and `tbody`; the list of active formatting elements still has the `b` element in it (the marker having been removed by the "td" end tag token); and the insertion mode is "in table body".

Thus it is that the "bbb" character tokens are found. These trigger the "in table text" insertion mode to be used (with the original insertion mode set to "in table body"). The character tokens are collected, and when the next token (the `table` element end tag) is seen, they are processed as a group. Since they are not all spaces, they are handled as per the "anything else" rules in the "in table" insertion mode, which defer to the "in body" insertion mode but with foster parenting.

When the active formatting elements are reconstructed, a `b` element is created and foster parented, and then the "bbb" `Text` node is appended to it:

```

└ html
 └ head
 └ body
 └ b
 └ b
 └ #text: bbb
 └ table
 └ tbody
 └ tr
 └ td
 └ #text: aaa

```

The stack of open elements has on it the elements `html`, `body`, `table`, `tbody`, and the new `b` (again, note that this doesn't match the

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

Finally, the `table` is closed by a "table" end tag. This pops all the nodes from the stack of open elements up to and including the `table` element, but it doesn't affect the list of active formatting elements, so the "ccc" character tokens after the `table` result in yet another `b` element being created, this time after the `table`:

```

L html
 |- head
 |- body
 |- b
 |- b
 L #text: Too slow? Try reading the multipage copy of
 L table
 L tbody
 L tr
 L td
 L #text: the spec instead: https://web.archive.org
 L #text: /web/20130103030411/http:
 L #text: //whatwg.org/html

```

**12.2.8.4 Scripts that modify the page as it is being parsed**

*This section is non-normative.*

Consider the following markup, which for this example we will assume is the document with URL `http://example.com/inner`, being rendered as the content of an `iframe` in another document with the URL `http://example.com/outer`:

```

<div id=a>
 <script>
 var div = document.getElementById('a');
 parent.document.body.appendChild(div);
 </script>
 <script>
 alert(document.URL);
 </script>
</div>
<script>
 alert(document.URL);
</script>

```

Up to the first "script" end tag, before the script is parsed, the result is relatively straightforward:

```

L html
 |- head
 |- body
 |- div id="a"
 L #text:
 L script
 L #text: var div = document.getElementById('a'); parent.document.body.appendChild(div);

```

After the script is parsed, though, the `div` element and its child `script` element are gone:

```

L html
 |- head
 |- body

```

They are, at this point, in the `Document` of the aforementioned outer browsing context. However, the stack of open elements *still contains the `div` element*.

Thus, when the second `script` element is parsed, it is inserted *into the outer Document object*.

Those parsed into different `Documents` than the one the parser was created for do not execute, so the first alert does not show.

Once the `div` element's end tag is parsed, the `div` element is popped off the stack, and so the next `script` element is in the inner `Document`:

[3,445 captures](#)  
5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB

◀ 03 ▶

2012 2013 2014

f
t

#text: alert(document.URL);

This script does execute, resulting in an alert that says "http://example.com/inner".

### 12.2.8.5 The execution of scripts that are moving across multiple documents

*This section is non-normative.*

Elaborating on the example in the previous section, consider the case where the second `script` element is an external script (i.e. one with a `src` attribute). In this case, the browser will download the external script and execute it, even if the external script is not even being downloaded. Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

*Note: It*

while the external script is not execute.

### 12.2.8.6 Unclosed formatting elements

*This section is non-normative.*

The following markup shows how nested formatting elements (such as `b`) get collected and continue to be applied even as the elements they are contained in are closed, but that excessive duplicates are thrown away.

```
<!DOCTYPE html>
<p><b class=x><b class=x><b class=x><b class=x>X
<p>X
<p><b class=x>X
<p>X
```

The resulting DOM tree is as follows:

```

└─DOCTYPE: html
 └─html
 └─head
 └─body
 └─p
 └─b class="x"
 └─b class="x"
 └─b
 └─b class="x"
 └─b class="x"
 └─b
 └─#text: X
 └─p
 └─b class="x"
 └─b
 └─b class="x"
 └─b class="x"
 └─b
 └─#text: X
 └─p
 └─b class="x"
 └─b
 └─b class="x"
 └─b class="x"
 └─b
 └─b
 └─#text: X

```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

element's "X".

Also note how this means that in the final paragraph only six `b` end tags are needed to completely clear the list of formatting elements, even though nine `b` start tags have been seen up to this point.

## 12.3 Serializing HTML fragments

The following steps form the steps from the [HTML Fragment serialization algorithm](#). The algorithm takes as input a DOM Element, Document, or DocumentFragment.

### Too slow? Try reading the multipage copy of

**Note:** Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

1. Let `/web/20130103030411/http://whatwg.org/html`
2. For

2. Append the appropriate string from the following list to `s`:

↳ If **current node** is an **Element**

If `current node` is an element in the HTML namespace, the MathML namespace, or the SVG namespace, then let `tagname` be `current node`'s local name. Otherwise, let `tagname` be `current node`'s qualified name.

Append a U+003C LESS-Than SIGN character (<), followed by `tagname`.

**Note:** For HTML elements created by the HTML parser or `Document.createElement()`, `tagname` will be lowercase.

For each attribute that the element has, append a U+0020 SPACE character, the attribute's serialized name as described below, a U+003D EQUALS SIGN character (=), a U+0022 QUOTATION MARK character ("'), the attribute's value, escaped as described below in **attribute mode**, and a second U+0022 QUOTATION MARK character ("').

An **attribute's serialized name** for the purposes of the previous paragraph must be determined as follows:

↳ If the attribute has no namespace

The attribute's serialized name is the attribute's local name.

**Note:** For attributes on HTML elements set by the HTML parser or by `Element.setAttribute()`, the local name will be lowercase.

↳ If the attribute is in the XML namespace

The attribute's serialized name is the string "`xml:`" followed by the attribute's local name.

↳ If the attribute is in the XMLNS namespace and the attribute's local name is `xmlns`

The attribute's serialized name is the string "`xmlns`".

↳ If the attribute is in the XMLNS namespace and the attribute's local name is not `xmlns`

The attribute's serialized name is the string "`xmlns:`" followed by the attribute's local name.

↳ If the attribute is in the XLink namespace

The attribute's serialized name is the string "`xlink:`" followed by the attribute's local name.

↳ If the attribute is in some other namespace

The attribute's serialized name is the attribute's qualified name.

While the exact order of attributes is UA-defined, and may depend on factors such as the order that the attributes were given in the original markup, the sort order must be stable, such that consecutive invocations of this algorithm serialize an element's attributes in the same order.

Append a U+003E GREATER-Than SIGN character (>).

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

is a Text node whose character data has as its first character a U+000A LINE FEED (LF) character, then append a U+000A LINE FEED (LF) character.

Append the value of running the HTML fragment serialization algorithm on the *current node* element (thus recursing into this algorithm for that element), followed by a U+003C LESS-Than SIGN character (<), a U+002F SOLIDUS character (/), *tagname* again, and finally a U+003E GREATER-Than SIGN character (>).

↳ If *current node* is a Text node



noframes, or plaintext  
is enabled for the node, then

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

as described below.

MATION MARK, U+002D  
*ent node*'s data IDL attribute,

followed by the literal string --> (U+002D HYPHEN-MINUS, U+002D HYPHEN-MINUS, U+003E  
GREATER-Than SIGN).

↳ If *current node* is a ProcessingInstruction

Append the literal string <? (U+003C LESS-Than SIGN, U+003F QUESTION MARK), followed by the  
value of *current node*'s target IDL attribute, followed by a single U+0020 SPACE character, followed by  
the value of *current node*'s data IDL attribute, followed by a single U+003E GREATER-Than SIGN  
character (>).

↳ If *current node* is a DocumentType

Append the literal string <!DOCTYPE (U+003C LESS-Than SIGN, U+0021 EXCLAMATION MARK,  
U+0044 LATIN CAPITAL LETTER D, U+004F LATIN CAPITAL LETTER O, U+0043 LATIN CAPITAL  
LETTER C, U+0054 LATIN CAPITAL LETTER T, U+0059 LATIN CAPITAL LETTER Y, U+0050 LATIN  
CAPITAL LETTER P, U+0045 LATIN CAPITAL LETTER E), followed by a space (U+0020 SPACE),  
followed by the value of *current node*'s name IDL attribute, followed by the literal string > (U+003E  
GREATER-Than SIGN).

3. The result of the algorithm is the string s.

**⚠Warning!** It is possible that the output of this algorithm, if parsed with an HTML parser, will not return the original tree structure.

For instance, if a textarea element to which a Comment node has been appended is serialized and the output is then reparsed, the comment will end up being displayed in the text field. Similarly, if, as a result of DOM manipulation, an element contains a comment that contains the literal string "-->", then when the result of serializing the element is parsed, the comment will be truncated at that point and the rest of the comment will be interpreted as markup. More examples would be making a script element contain a Text node with the text string "</script>", or having a p element that contains a ul element (as the ul element's start tag would imply the end tag for the p).

This can enable cross-site scripting attacks. An example of this would be a page that lets the user enter some font family names that are then inserted into a CSS style block via the DOM and which then uses the innerHTML IDL attribute to get the HTML serialization of that style element: if the user enters "</style><script>attack</script>" as a font family name, innerHTML will return markup that, if parsed in a different context, would contain a script node, even though no script node existed in the original DOM.

**Escaping a string** (for the purposes of the algorithm above) consists of running the following steps:

1. Replace any occurrence of the "&" character by the string "&";
2. Replace any occurrences of the U+00A0 NO-BREAK SPACE character by the string " ".
3. If the algorithm was invoked in the *attribute mode*, replace any occurrences of the "" character by the string """.
4. If the algorithm was *not* invoked in the *attribute mode*, replace any occurrences of the "<" character by the string "<", and any occurrences of the ">" character by the string ">".

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

View

**Note:** Parts marked fragment case in algorithms in the parser section are parts that only occur if the parser was created for the purposes of this algorithm (and with a context element). The algorithms have been annotated with such markings for informational purposes only; such markings have no normative weight. If it is possible for a condition described as a fragment case to occur even when the parser wasn't created for the purposes of handling this algorithm, then that is an error in the specification.

1. Create a new Document node, and mark it as being an HTML document.



2. If the

mode

Doc

3. Cre

4. If th

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

the Document be in quirks mode, then let the

↳ If it is a title or textarea element

Switch the tokenizer to the RCDATA state.

↳ If it is a style, xmp, iframe, noembed, ornoframes element

Switch the tokenizer to the RAWTEXT state.

↳ If it is a script element

Switch the tokenizer to the script data state.

↳ If it is a noscript element

If the scripting flag is enabled, switch the tokenizer to the RAWTEXT state. Otherwise, leave the tokenizer in the data state.

↳ If it is a plaintext element

Switch the tokenizer to the PLAINTEXT state.

↳ Otherwise

Leave the tokenizer in the data state.

**Note:** For performance reasons, an implementation that does not report errors and that uses the actual state machine described in this specification directly could use the PLAINTEXT state instead of the RAWTEXT and script data states where those are mentioned in the list above. Except for rules regarding parse errors, they are equivalent, since there is no appropriate end tag token in the fragment case, yet they involve far fewer state transitions.

2. Let root be a new html element with no attributes.

3. Append the element root to the Document node created above.

4. Set up the parser's stack of open elements so that it contains just the single element root.

5. Reset the parser's insertion mode appropriately.

**Note:** The parser will reference the context element as part of that algorithm.

6. Set the parser's form element pointer to the nearest node to the context element that is a form element (going straight up the ancestor chain, and including the element itself, if it is a form element), or, if there is no such form element, to null.

5. Place into the input stream for the HTML parser just created the input. The encoding confidence is irrelevant.

6. Start the parser and let it run until it has consumed all the characters just inserted into the input stream.

7. If there is a context element, return the child nodes of root, in tree order.

Otherwise, return the children of the Document object, in tree order.

[ ]

[Go](#) **NOV** **JAN** **FEB**  
◀ **03** ▶  
**2012** **2013** **2014**

[About this capture](#)

**3,445 captures**  
5 Jun 2004 - 22 Aug 2020



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

ac;	U+0223E	~
acd;	U+0223F	~
acE;	U+0223E U+00333	~
Acirc;	U+000C2	Ã
Acirc;	U+000C2	Ã
acirc;	U+000E2	â
acirc;	U+000E2	â
acute;	U+000B4	'
acute	U+000B4	'
Acy;	U+00410	A
acy;	U+00430	a
AElig;	U+000C6	Æ
AElig	U+000C6	Æ
aelig;		
aelig		
af;		
Afr;		
afr;		
Agrave;		
Agrave		
agrave;		
grave		
grave		
alefsym;		
aleph;		
Alpha;		
alpha;		
Amacr;	U+00100	Å
amacr;	U+00101	å
amalq;	U+0243F	ܲ
AMP;	U+00026	&
AMP	U+00026	&
amp;	U+00026	&
amp	U+00026	&
And;	U+02A53	݂
and;	U+02227	݂
andand;	U+02A55	݂
andd;	U+0245C	݂
andalope;	U+02A58	݂
andv;	U+02A5A	݂
ang;	U+02220	݂
ange;	U+02944	݂
angle;	U+02220	݂
angmsd;	U+02221	݂
angmsdaa;	U+029A8	݂
angmsdab;	U+029A9	݂
angmsdac;	U+029AA	݂
angmsdad;	U+029AB	݂
angmsdaei;	U+029AC	݂
angmsdaf;	U+029AD	݂
angmsdag;	U+029AE	݂
angmsdah;	U+029AF	݂
angrt;	U+0221F	݂
angrtvb;	U+0228E	݂
angrtvbd;	U+029D0	݂
angsph;	U+02222	݂
angst;	U+000C5	A
angzarr;	U+0237C	݂
Aogen;	U+00104	A
aogen;	U+00105	â
Aopf;	U+1D538	A
aopf;	U+1D552	â
ap;	U+02248	=
apacir;	U+0246F	â
apE;	U+02470	=
ape;	U+0224A	=
apid;	U+0224B	=
apos;	U+00027	'
ApplyFunction;	U+02061	
approx;	U+02248	=
approxeq;	U+0224A	=
Aring;	U+000C5	Å
Aring	U+000C5	Å
azing;	U+000E5	â
aring;	U+000E5	â
aring;	U+000E5	â
Ascr;	U+1D49C	Ӓ
ascr;	U+1D4B6	ӓ
Assign;	U+02254	=
ast;	U+0002A	*
asympt;	U+02248	=
asympeq;	U+0224D	x
Atilde;	U+000C3	Ã
Atilde	U+000C3	Ã
atilde;	U+000E3	â
atilde	U+000E3	â
Auml;	U+000C4	Ã
Auml	U+000C4	Ã
auml;	U+000E4	â
auml	U+000E4	â
awconint;	U+02233	݂
awint;	U+02A11	݂
backcong;	U+0224C	=

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

[ ]

[Go](#) **NOV** **JAN** **FEB**  
◀ **03** ▶  
**2012** **2013** **2014**

[About this capture](#)

**3,445 captures**  
5 Jun 2004 - 22 Aug 2020



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## 13.1 Writing XHTML documents

The syntax for using HTML with XML, whether in XHTML documents or embedded in other XML documents, is defined in the XML and Namespaces in XML specifications. [XML] [XMLNS]

This specification does not define any syntax-level requirements beyond those defined for XML proper.

XML docume

define a pub

**Too slow? Try reading the multipage copy of the spec instead:** <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Note: A

referen

docum

This specification does not

xternal DTD subset

rs in XHTML

t; and &amp;apos;).

## 13.2 Parsing XHTML documents

This section describes the relationship between XML and the DOM, with a particular emphasis on how this interacts with HTML.

An **XML parser**, for the purposes of this specification, is a construct that follows the rules given in the XML specification to map a string of bytes or characters into a `Document` object.

An XML parser is either associated with a `Document` object when it is created, or creates one implicitly.

This `Document` must then be populated with DOM nodes that represent the tree structure of the input passed to the parser, as defined by the XML specification, the Namespaces in XML specification, and the DOM specification. DOM mutation events must not fire for the operations that the XML parser performs on the `Document`'s tree, but the user agent must act as if elements and attributes were individually appended and set respectively so as to trigger rules in this specification regarding what happens when an element is inserted into a document or has its attributes set, and the DOM specification's requirements regarding mutation observers mean that mutation observers are fired (unlike mutation events). [XML] [XMLNS] [DOM] [DOMEVENTS]

Between the time an element's start tag is parsed and the time either the element's end tag is parsed or the parser detects a well-formedness error, the user agent must act as if the element was in a stack of open elements.

**Note:** This is used by the `object` element to avoid instantiating plugins before the `param` element children have been parsed.

This specification provides the following additional information that user agents should use when retrieving an external entity: the public identifiers given in the following list all correspond to the URL given by this link. (This URL is a DTD containing the entity reference declarations for the names listed in the named character references section.)

- -//W3C//DTD XHTML 1.0 Transitional//EN
- -//W3C//DTD XHTML 1.1//EN
- -//W3C//DTD XHTML 1.0 Strict//EN
- -//W3C//DTD XHTML 1.0 Frameset//EN
- -//W3C//DTD XHTML Basic 1.0//EN
- -//W3C//DTD XHTML 1.1 plus MathML 2.0//EN
- -//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG 1.1//EN
- -//W3C//DTD MathML 2.0//EN
- -//WAPFORUM//DTD XHTML Mobile 1.0//EN

Furthermore, user agents should attempt to retrieve the above external entity's content when one of the above public identifiers is used, and should not attempt to retrieve any other external entity's content.

**Note:** This is not strictly a violation of the XML specification, but it does contradict the spirit of the XML specification's requirements. This is motivated by a desire for user agents to all handle entities in an interoperable fashion without requiring any network access for handling external subsets. [XML]

When an XML parser creates a `script` element, it must be marked as being "parser-inserted" and its "force-async" flag must be unset. If the parser was originally created for the XML fragment parsing algorithm, then the element must be marked as "already started" also. When the element's end tag is parsed, the user agent must perform a microtask checkpoint, provide a stable state, and

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

2. Spin the event loop until the parser's Document has no style sheet that is blocking scripts and the pending parsing-blocking script's "ready to be parser-executed" flag is set.
3. Unblock this instance of the XML parser, such that tasks that invoke it can again be run.
4. Execute the pending parsing-blocking script.
5. There is no longer a pending parsing-blocking script.

**Note:** Since the `document.write()` API is not available for XML documents, much complexity in the HTML parser is removed.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Certain algorithms would have to be modified to work with the XML parser.

es, the XML parser must act

When an XML document can also be read by an XML parser. An XML parser

is an XML document.

For the purpose of this section, an XML document is considered XML namespace-well-formed if a document consisting of an element with no namespace declarations whose contents are the internal general parsed entity would itself be XML namespace-well-formed.

### 13.3 Serializing XHTML fragments

The XML fragment serialization algorithm for a Document or Element node either returns a fragment of XML that represents that node or throws an exception.

For Documents, the algorithm must return a string in the form of a document entity, if none of the error cases below apply.

For Elements, the algorithm must return a string in the form of an internal general parsed entity, if none of the error cases below apply.

In both cases, the string returned must be XML namespace-well-formed and must be an isomorphic serialization of all of that node's child nodes, in tree order. User agents may adjust prefixes and namespace declarations in the serialization (and indeed might be forced to do so in some cases to obtain namespace-well-formed XML). User agents may use a combination of regular text and character references to represent Text nodes in the DOM.

For Elements, if any of the elements in the serialization are in no namespace, the default namespace in scope for those elements must be explicitly declared as the empty string. (This doesn't apply in the Document case.) [XML] [XMLNS]

For the purposes of this section, an internal general parsed entity is considered XML namespace-well-formed if a document consisting of an element with no namespace declarations whose contents are the internal general parsed entity would itself be XML namespace-well-formed.

If any of the following error cases are found in the DOM subtree being serialized, then the algorithm must throw an `InvalidStateError` exception instead of returning a string:

- A Document node with no child element nodes.
- A DocumentType node that has an external subset public identifier that contains characters that are not matched by the XML PubidChar production. [XML]
- A DocumentType node that has an external subset system identifier that contains both a U+0022 QUOTATION MARK ("") and a U+0027 APOSTROPHE ('') or that contains characters that are not matched by the XML Char production. [XML]
- A node with a local name containing a U+003A COLON (:).
- A node with a local name that does not match the XML Name production. [XML]
- An Attr node with no namespace whose local name is the lowercase string "xmlns". [XMLNS]
- An Element node with two or more attributes with the same local name and namespace.
- An Attr node, Text node, Comment node, or ProcessingInstruction node whose data contains characters that are not matched by the XML Char production. [XML]
- A Comment node whose data contains two adjacent U+002D HYPHEN-MINUS characters (-) or ends with such a character.

3,445 captures

5 Jun 2004 - 22 Aug 2020

Go

**NOV JAN FEB**

03

2012 2013 2014



▼ About this capture

**Note:** These are the only ways to make a DOM unserializable. The DOM enforces all the other XML constraints; for example, trying to append two elements to a Document node will throw a HierarchyRequestError exception.

## 13.4 Parsing XHTML fragments

The **XML fragment parsing algorithm** either returns a `Document` or throws a `SyntaxError` exception. Given a string `input` and an optional contr

1. Create a new file named `index.html` in the `src` folder.
  2. If the file is too slow, try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The default namespace is the namespace for which the DOM `isDefaultNamespace()` method on the element would return true.

**Note:** If there is a context element, no DOCTYPE is passed to the parser, and therefore no external subset is referenced, and therefore no entities will be recognized.

3. Feed the parser just created the string *input*.
  4. If there is a *context* element, feed the parser just created the string corresponding to the end tag of that element.
  5. If there is an XML well-formedness or XML namespace well-formedness error, then throw a `SyntaxError` exception and abort these steps.
  6. If there is a *context* element, and the root element of the resulting `Document` has any sibling nodes, then throw a `SyntaxError` exception and abort these steps.
  7. If there is a *context* element, then return the child nodes of the root element of the resulting `Document`, in tree order.  
Otherwise, return the children of the `Document` object, in tree order.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

the documents' authors. So as to avoid confusion regarding the normativity of this section, RFC2119 terms have not been used.

Instead, the term "expected" is used to indicate behavior that will lead to this experience. For the purposes of conformance for user agents designated as supporting the suggested default rendering, the term "expected" in this section has the same conformance implications as the RFC2119-defined term "must".

### 14.1 Introduction

In general, user agents are expected to support CSS, and many of the suggestions in this section are expressed in CSS terms. User agents that do not honor author-level CSS style sheets are nonetheless expected to act as if they applied the CSS rules given in this section.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The suggestion applies to multiple sections of this specification, and it is intended to adapt the suggestions in this section to the needs of the document object model.

An element is **being rendered** if it is in a `Document`, either its parent node is itself being rendered or it is the `Document` node, and it is not explicitly excluded from the rendering using either:

- the CSS 'display' property's 'none' value, or
- the 'visibility' property's 'collapse' value unless it is being treated as equivalent to the 'hidden' value, or
- some equivalent in other styling languages.

**Note:** Just being off-screen does not mean the element is not being rendered. The presence of the `hidden` attribute normally means the element is not being rendered, though this might be overridden by the style sheets.

User agents that do not honor author-level CSS style sheets are nonetheless expected to act as if they applied the CSS rules given in these sections in a manner consistent with this specification and the relevant CSS and Unicode specifications. [CSS] [UNICODE] [BIDI]

**Note:** This is especially important for issues relating to the 'display', 'unicode-bidi', and 'direction' properties.

### 14.2 The CSS user agent style sheet and presentational hints

The CSS rules given in these subsections are, except where otherwise specified, expected to be used as part of the user-agent level style sheet defaults for all documents that contain HTML elements.

Some rules are intended for the author-level zero-specificity presentational hints part of the CSS cascade; these are explicitly called out as **presentational hints**.

Some of the rules regarding left and right margins are given here as appropriate for elements whose 'direction' property is 'ltr', and are expected to be flipped around on elements whose 'direction' property is 'rtl'. These are marked "**LTR-specific**".

**Note:** These markings only affect the handling of attribute values, not attribute names or element names.

When the text below says that an attribute `attribute` on an element `element` **maps to the pixel length property** (or properties) `properties`, it means that if `element` has an attribute `attribute` set, and parsing that attribute's value using the rules for parsing non-negative integers doesn't generate an error, then the user agent is expected to use the parsed value as a pixel length for a presentational hint for `properties`.

When the text below says that an attribute `attribute` on an element `element` **maps to the dimension property** (or properties) `properties`, it means that if `element` has an attribute `attribute` set, and parsing that attribute's value using the rules for parsing dimension values doesn't generate an error, then the user agent is expected to use the parsed dimension as the value for a presentational hint for `properties`, with the value given as a pixel length if the dimension was an integer, and with the value given as a percentage if the dimension was a percentage.

When a user agent is to **align descendants** of a node, the user agent is expected to align only those descendants that have both their 'margin-left' and 'margin-right' properties computing to a value other than 'auto', that are over-constrained and that have one of those two margins with a used value forced to a greater value, and that do not themselves have an applicable `align` attribute. When multiple

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

## 14.5 Non-replaced elements

### 14.3.1 Hidden elements

```
@namespace url (http://www.w3.org/1999/xhtml) ;

[hidden], area, base, basefont, command, datalist, head,
input[type=hidden i], link, menu[type=popup i], meta, noembed, noframes,
param, rp, script, source, style, track, title {
 d:
}
```

**Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>**

The user agent is expected to compute to 'none',  
irrespective of the value of the 'display' property.

### 14.3.2 The page

```
@namespace url (http://www.w3.org/1999/xhtml) ;

html, body { display: block; }
```

For each property in the table below, given a `body` element, the first attribute that exists maps to the pixel length property on the `body` element. If none of the attributes for a property are found, or if the value of the attribute that was found cannot be parsed successfully, then, if the `body` element's `Document`'s browsing context does not have its seamless browsing context flag set, a default value of 8px is expected to be used for that property instead.

Property	Source
'margin-top'	<code>body</code> element's <code>marginheight</code> attribute
	The <code>body</code> element's container frame element's <code>marginheight</code> attribute
	<code>body</code> element's <code>topmargin</code> attribute
'margin-right'	<code>body</code> element's <code>marginwidth</code> attribute
	The <code>body</code> element's container frame element's <code>marginwidth</code> attribute
	<code>body</code> element's <code>rightmargin</code> attribute
'margin-bottom'	<code>body</code> element's <code>marginheight</code> attribute
	The <code>body</code> element's container frame element's <code>marginheight</code> attribute
	<code>body</code> element's <code>bottommargin</code> attribute
'margin-left'	<code>body</code> element's <code>marginwidth</code> attribute
	The <code>body</code> element's container frame element's <code>marginwidth</code> attribute
	<code>body</code> element's <code>leftmargin</code> attribute

If the `body` element's `Document`'s browsing context is a nested browsing context, and the browsing context container of that nested browsing context is a `frame` or `iframe` element, then the `container frame element` of the `body` element is that `frame` or `iframe` element. Otherwise, there is no container frame element.

**⚠ Warning! The above requirements imply that a page can change the margins of another page (including one from another origin) using, for example, an `iframe`. This is potentially a security risk, as it might in some cases allow an attack to contrive a situation in which a page is rendered not as the author intended, possibly for the purposes of phishing or otherwise misleading the user.**

If the `Document` has a root element, and the `Document`'s browsing context is a nested browsing context, and the browsing context container of that nested browsing context is a `frame` or `iframe` element, and that element has a `scrolling` attribute, then the user agent is expected to compare the value of the attribute in an ASCII case-insensitive manner to the values in the first column of the following table, and if one of them matches, then the user agent is expected to treat that attribute as a presentational hint for the aforementioned root element's `'overflow'` property, setting it to the value given in the corresponding cell on the same row in the second column:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

off	'hidden'
noscroll	'hidden'
no	'hidden'
auto	'auto'

When a Document is in quirks mode, vertical margins on HTML elements at the top or bottom of body elements are expected to be collapsed to zero.



When a body element, and its 'background' value, and if its 'background' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's

be resolved relative to the element's

When a body element has a 'background' attribute, and if its 'background' value, and if its 'background' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's

s for parsing a legacy color value, and if its 'background' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's

When a body element has a 'color' attribute, and if its 'color' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's 'color' property to the resulting color.

a legacy color value, and if its 'color' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's 'color' property to the resulting color.

When a body element has a 'link' attribute, and if its 'link' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the 'color' property of any element in the Document matching the ':link' pseudo-class to the resulting color.

When a body element has a 'vlink' attribute, and if its 'vlink' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the 'color' property of any element in the Document matching the ':visited' pseudo-class to the resulting color.

When a body element has a 'alink' attribute, and if its 'alink' attribute does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the 'color' property of any element in the Document matching the ':active' pseudo-class and either the ':link' pseudo-class or the ':visited' pseudo-class to the resulting color.

### 14.3.3 Flow content

```
@namespace url("http://www.w3.org/1999/xhtml");

address, blockquote, center, div, figure, figcaption, footer, form,
header, hr, legend, listing, p, plaintext, pre, summary, xmp {
 display: block;
}

blockquote, figure, listing, p, plaintext, pre, xmp {
 margin-top: 1em; margin-bottom: 1em;
}

blockquote, figure { margin-left: 40px; margin-right: 40px; }

address { font-style: italic; }
listing, plaintext, pre, xmp {
 font-family: monospace; white-space: pre;
}

dialog:not([open]) { display: none; }
dialog {
 position: absolute;
 left: 0; right: 0;
 margin: auto;
 border: solid;
 padding: 1em;
}
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The following rules are also expected to apply, as presentational hints:

```
@namespace url(http://www.w3.org/1999/xhtml);
pre[wrap] { white-space: pre-wrap; }
```

In quirks mode, the following rules are also expected to apply:

`@n@`  
`fo` Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org>

The `center` /web/20130103030411/http:  
the string "c  
'center' in a //whatwg.org/html

insensitive match for either  
eir 'text-align' property set to

The `div` element, when it has an `align` attribute whose value is an ASCII case-insensitive match for the string "left", is expected to left-align text within itself, as if it had its 'text-align' property set to 'left' in a presentational hint, and to align descendants to the left.

The `div` element, when it has an `align` attribute whose value is an ASCII case-insensitive match for the string "right", is expected to right-align text within itself, as if it had its 'text-align' property set to 'right' in a presentational hint, and to align descendants to the right.

The `div` element, when it has an `align` attribute whose value is an ASCII case-insensitive match for the string "justify", is expected to full-justify text within itself, as if it had its 'text-align' property set to 'justify' in a presentational hint, and to align descendants to the left.

#### 14.3.4 Phrasing content

```
@namespace url(http://www.w3.org/1999/xhtml);

cite, dfn, em, i, var { font-style: italic; }
b, strong { font-weight: bolder; }
code, kbd, samp, tt { font-family: monospace; }
big { font-size: larger; }
small { font-size: smaller; }

sub { vertical-align: sub; }
sup { vertical-align: super; }
sub, sup { line-height: normal; font-size: smaller; }

ruby { display: ruby; }
rt { display: ruby-text; }

:link { color: #0000EE; }
:visited { color: #551A8B; }
:link, :visited { text-decoration: underline; }
a:link[rel~=help], a:visited[rel~=help],
area:link[rel~=help], area:visited[rel~=help] { cursor: help; }

:focus { outline: auto; }

mark { background: yellow; color: black; /* this color is just a suggestion and can be
changed based on implementation feedback */

abbr[title], acronym[title] { text-decoration: dotted underline; }
ins, u { text-decoration: underline; }
del, s, strike { text-decoration: line-through; }
blink { text-decoration: blink; }

q::before { content: open-quote; }
```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

nobr wbr { white-space: normal; }

The following rules are also expected to apply, as presentational hints:

```
@namespace url("http://www.w3.org/1999/xhtml");
br[clear=left i] { clear: left; }
br[clear=right i] { clear: right; }
br[c]
```



**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

For the pur  
in anonymou

; are expected to be wrapped

When a part  
as to minimi

both sides of the base text so

**Note: Ruby rendering is expected to be wrapped around the text of rt elements in the absence of rp elements.**

ressed in terms of CSS elements, which is how

User agents that do not support correct ruby rendering are expected to render parentheses around the text of `rt` elements in the absence of `rp` elements.

User agents are expected to support the 'clear' property on inline elements (in order to render `br` elements with `clear` attributes) in the manner described in the non-normative note to this effect in CSS2.1.

The initial value for the 'color' property is expected to be black. The initial value for the 'background-color' property is expected to be 'transparent'. The canvas' background is expected to be white.

When a `font` element has a `color` attribute, its value is expected to be parsed using the rules for parsing a legacy color value, and if that does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's 'color' property to the resulting color.

When a `font` element has a `face` attribute, the user agent is expected to treat the attribute as a presentational hint setting the element's 'font-family' property to the attribute's value.

When a `font` element has a `size` attribute, the user agent is expected to use the following steps, known as the **rules for parsing a legacy font size**, to treat the attribute as a presentational hint setting the element's 'font-size' property:

1. Let `input` be the attribute's value.
2. Let `position` be a pointer into `input`, initially pointing at the start of the string.
3. Skip whitespace.
4. If `position` is past the end of `input`, there is no presentational hint. Abort these steps.
5. If the character at `position` is a U+002B PLUS SIGN character (+), then let `mode` be `relative-plus`, and advance `position` to the next character. Otherwise, if the character at `position` is a U+002D HYPHEN-MINUS character (-), then let `mode` be `relative-minus`, and advance `position` to the next character. Otherwise, let `mode` be `absolute`.
6. Collect a sequence of characters that are ASCII digits, and let the resulting sequence be `digits`.
7. If `digits` is the empty string, there is no presentational hint. Abort these steps.
8. Interpret `digits` as a base-ten integer. Let `value` be the resulting number.
9. If `mode` is `relative-plus`, then increment `value` by 3. If `mode` is `relative-minus`, then let `value` be the result of subtracting `value` from 3.
10. If `value` is greater than 7, let it be 7.
11. If `value` is less than 1, let it be 1.
12. Set 'font-size' to the keyword corresponding to the value of `value` according to the following table:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

4	large	
5	x-large	
6	xx-large	
7	xxx-large	see below

The 'xxx-large' value is a non-CSS value used here to indicate a font size 50% larger than 'xx-large'.

#### 14.3.5 Bidirectional text

**@n** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```

:d
:d
:ad
:he
:as

colgroup, col, thead, tbody, tr, td, th, ol, ul, li {
 unicode-bidi: isolate;
}

:matches([dir=ltr i], [dir=rtl i], [dir=auto i]):not(address):not(blockquote)
:matches([dir=ltr i], [dir=rtl i]):not(div):not(figure):not(figcaption):not(footer):not(form)
:matches([dir=ltr i], [dir=rtl i]):not(header):not(hr):not(legend):not(listing):not(p):not(plaintext):not(pre)
:matches([dir=ltr i], [dir=rtl i]):not(summary):not(xmp):not(article):not(aside):not(h1):not(h2):not(h3):not(h4)
:matches([dir=ltr i], [dir=rtl i]):not(h5):not(h6):not(hgroup):not(nav):not(section):not(table):not(caption)
:matches([dir=ltr i], [dir=rtl i]):not(colgroup):not(col):not(thead):not(tbody):not(tfoot):not(tr):not(td)
:matches([dir=ltr i], [dir=rtl i]):not(th):not(dir):not(dd):not(dl):not(dt):not(menu):not(ol):not(ul):not(li) {
 unicode-bidi: embed;
}

bdi, bdi:matches([dir=ltr i], [dir=rtl i]),
output, output:matches([dir=ltr i], [dir=rtl i]),
[dir=auto i] {
 unicode-bidi: isolate;
}

bdo, bdo:matches([dir=ltr i], [dir=rtl i]) { unicode-bidi: bidi-override; }
bdo[dir=auto i] { unicode-bidi: isolate-override; }

textarea[dir=auto i], pre[dir=auto i] { unicode-bidi: plaintext; }

```

#### 14.3.6 Quotes

This block is automatically generated from the Unicode Common Locale Data Repository. [CLDR]

User agents are expected to use either the block below (which will be regularly updated) or to automatically generate their own copy directly from the source material. The language codes are derived from the CLDR file names. The quotes are derived from the delimiter blocks, with fallback handled as specified in the CLDR documentation.

```

@namespace url("http://www.w3.org/1999/xhtml");

:root { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(af), :not(:lang(af)) > :lang(af) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(agq), :not(:lang(agq)) > :lang(agq) { quotes: '\201e' '\201d'
'\201a' '\2019' } /* " , ' */
:root:lang(ak), :not(:lang(ak)) > :lang(ak) { quotes: '\201c' '\201d'

```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

:root:lang(asa), :not(:lang(asa)) > :lang(asa) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(az-Cyrl), :not(:lang(az-Cyrl)) > :lang(az-Cyrl) { quotes: '\00ab' '\00bb'
'\2039' '\203a' } /* <> < > */
:root:lang(bas), :not(:lang(bas)) > :lang(bas) { quotes: '\00ab' '\00bb'
'\201e' '\201c' } /* <> " ' */
:root:lang(bem), :not(:lang(bem)) > :lang(bem) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(bet), :not(:lang(bet)) > :lang(bet) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(bz), :not(:lang(bz)) > :lang(bz) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(brx), :not(:lang(brx)) > :lang(brx) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(bs-Cyrl), :not(:lang(bs-Cyrl)) > :lang(bs-Cyrl) { quotes: '\201e' '\201c'
'\201a' '\2018' } /* " , ' */
:root:lang(ca), :not(:lang(ca)) > :lang(ca) { quotes: '\201c' '\201d'
'\00ab' '\00bb' } /* " <> */
:root:lang(cgg), :not(:lang(cgg)) > :lang(cgg) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(chr), :not(:lang(chr)) > :lang(chr) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(cs), :not(:lang(cs)) > :lang(cs) { quotes: '\201e' '\201c'
'\201a' '\2018' } /* " , ' */
:root:lang(da), :not(:lang(da)) > :lang(da) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(dav), :not(:lang(dav)) > :lang(dav) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(de), :not(:lang(de)) > :lang(de) { quotes: '\201e' '\201c'
'\201a' '\2018' } /* " , ' */
:root:lang(de-CH), :not(:lang(de-CH)) > :lang(de-CH) { quotes: '\00ab' '\00bb'
'\2039' '\203a' } /* <> < > */
:root:lang(dje), :not(:lang(dje)) > :lang(dje) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(dua), :not(:lang(dua)) > :lang(dua) { quotes: '\00ab' '\00bb'
'\2018' '\2019' } /* <> ' ' */
:root:lang(dy), :not(:lang(dy)) > :lang(dy) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " */
:root:lang(dz), :not(:lang(dz)) > :lang(dz) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(ebu), :not(:lang(ebu)) > :lang(ebu) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(ee), :not(:lang(ee)) > :lang(ee) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(el), :not(:lang(el)) > :lang(el) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " */
:root:lang(en), :not(:lang(en)) > :lang(en) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " ' ' */
:root:lang(es), :not(:lang(es)) > :lang(es) { quotes: '\201c' '\201d'
'\00ab' '\00bb' } /* " <> */
:root:lang(et), :not(:lang(et)) > :lang(et) { quotes: '\201e' '\201c'
'\201a' '\2018' } /* " , ' */
:root:lang(eu), :not(:lang(eu)) > :lang(eu) { quotes: '\201c' '\201d'
'\00ab' '\00bb' } /* " <> */
:root:lang(ewo), :not(:lang(ewo)) > :lang(ewo) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " */

```

**Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

Go

NOV JAN FEB

03



2012 2013 2014

About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```

'\"2019' '\"2019' } /* " " ' ' */
:root:lang(fr), :not(:lang(fr)) > :lang(fr) { quotes: '\00ab' '\00bb'
'\00ab' '\00bb' } /* <> <> */
:root:lang(fr-CA), :not(:lang(fr-CA)) > :lang(fr-CA) { quotes: '\00ab' '\00bb'
'\2039' '\"203a' } /* <> < > */
:root:lang(fr-CH), :not(:lang(fr-CH)) > :lang(fr-CH) { quotes: '\00ab' '\00bb'
'\2039' '\"203a' } /* <> < > */
:root:lang(gsw), :not(:lang(gsw)) > :lang(gsw) { quotes: '\00ab' '\00bb'
'\2039' '\"203a' } /* <> < > */

:r
'\" Too slow? Try reading the multipage copy of
:r
'\" the spec instead: https://web.archive.org
:r
'\" /web/20130103030411/http:
:r
'\" //whatwg.org/html
:rc
'\"2018' '\"2019' } /* " " ' ' */
:root:lang(hr), :not(:lang(hr)) > :lang(hr) { quotes: '\201e' '\201c'
'\201a' '\"2018' } /* " " , ' */
:root:lang(hu), :not(:lang(hu)) > :lang(hu) { quotes: '\201e' '\201d'
'\00bb' '\00ab' } /* " " <> */
:root:lang(id), :not(:lang(id)) > :lang(id) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(ig), :not(:lang(ig)) > :lang(ig) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(it), :not(:lang(it)) > :lang(it) { quotes: '\00ab' '\00bb'
'\201c' '\"201d' } /* <> " " */
:root:lang(ja), :not(:lang(ja)) > :lang(ja) { quotes: '\300c' '\300d'
'\300e' '\"300f' } /* [] 『 』 */
:root:lang(jgo), :not(:lang(jgo)) > :lang(jgo) { quotes: '\00ab' '\00bb'
'\2039' '\"203a' } /* <> < > */
:root:lang(jmc), :not(:lang(jmc)) > :lang(jmc) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(kab), :not(:lang(kab)) > :lang(kab) { quotes: '\00ab' '\00bb'
'\201c' '\"201d' } /* <> " " */
:root:lang(kam), :not(:lang(kam)) > :lang(kam) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(kde), :not(:lang(kde)) > :lang(kde) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(kea), :not(:lang(kea)) > :lang(kea) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(khq), :not(:lang(khq)) > :lang(khq) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(ki), :not(:lang(ki)) > :lang(ki) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(kkj), :not(:lang(kkj)) > :lang(kkj) { quotes: '\00ab' '\00bb'
'\2039' '\"203a' } /* <> < > */
:root:lang(kln), :not(:lang(kln)) > :lang(kln) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(km), :not(:lang(km)) > :lang(km) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(kn), :not(:lang(kn)) > :lang(kn) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(ko), :not(:lang(ko)) > :lang(ko) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(ksb), :not(:lang(ksb)) > :lang(ksb) { quotes: '\201c' '\201d'
'\2018' '\"2019' } /* " " ' ' */
:root:lang(ksf), :not(:lang(ksf)) > :lang(ksf) { quotes: '\00ab' '\00bb'
'\2018' '\"2019' } /* <> ' ' */
:root:lang(lag), :not(:lang(lag)) > :lang(lag) { quotes: '\201d' '\201d'

```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

```

:root:lang(lo), :not(:lang(lo)) > :lang(lo) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(lt), :not(:lang(lt)) > :lang(lt) { quotes: '\201e' '\201c'
'\201e' '\201c' } /* " " " */
:root:lang(lu), :not(:lang(lu)) > :lang(lu) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(luo), :not(:lang(luo)) > :lang(luo) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(luv), :not(:lang(luv)) > :lang(luv) { quotes: '\201e' '\201c'
'\2018' '\2019' } /* " " ' */
:root:lang(lv), :not(:lang(lv)) > :lang(lv) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(mg), :not(:lang(mg)) > :lang(mg) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " " */
:root:lang(mgo), :not(:lang(mgo)) > :lang(mgo) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(mk), :not(:lang(mk)) > :lang(mk) { quotes: '\201e' '\201c'
'\201a' '\2018' } /* " , ' */
:root:lang(ml), :not(:lang(ml)) > :lang(ml) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(mr), :not(:lang(mr)) > :lang(mr) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(ms), :not(:lang(ms)) > :lang(ms) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(mua), :not(:lang(mua)) > :lang(mua) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " */
:root:lang(my), :not(:lang(my)) > :lang(my) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(naq), :not(:lang(naq)) > :lang(naq) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(nb), :not(:lang(nb)) > :lang(nb) { quotes: '\00ab' '\00bb'
'\2018' '\2019' } /* <> ' ' */
:root:lang(nd), :not(:lang(nd)) > :lang(nd) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(nl), :not(:lang(nl)) > :lang(nl) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(nmg), :not(:lang(nmg)) > :lang(nmg) { quotes: '\201e' '\201d'
'\00ab' '\00bb' } /* " " <> */
:root:lang(nn), :not(:lang(nn)) > :lang(nn) { quotes: '\00ab' '\00bb'
'\2018' '\2019' } /* <> ' ' */
:root:lang(nnh), :not(:lang(nnh)) > :lang(nnh) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " */
:root:lang(nus), :not(:lang(nus)) > :lang(nus) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(nyn), :not(:lang(nyn)) > :lang(nyn) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(pl), :not(:lang(pl)) > :lang(pl) { quotes: '\201e' '\201d'
'\00ab' '\00bb' } /* " " <> */
:root:lang(pt), :not(:lang(pt)) > :lang(pt) { quotes: '\201c' '\201d'
'\2018' '\2019' } /* " " ' */
:root:lang(pt-PT), :not(:lang(pt-PT)) > :lang(pt-PT) { quotes: '\00ab' '\00bb'
'\201c' '\201d' } /* <> " " */
:root:lang(rn), :not(:lang(rn)) > :lang(rn) { quotes: '\201d' '\201d'
'\2019' '\2019' } /* " " ' */
:root:lang(ro), :not(:lang(ro)) > :lang(ro) { quotes: '\201e' '\201d'
'\00ab' '\00bb' } /* " " <> */

```

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ [About this capture](#)

```

'\\2018' '\\2019' } /* <> `` */
:root:lang(rwk), :not(:lang(rwk)) > :lang(rwk) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` `` */
:root:lang(saq), :not(:lang(saq)) > :lang(saq) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` `` */
:root:lang(sbp), :not(:lang(sbp)) > :lang(sbp) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` `` */
:root:lang(seh), :not(:lang(seh)) > :lang(seh) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` `` */

:rc
'\\ Too slow? Try reading the multipage copy of
:r
'\\ the spec instead: https://web.archive.org
:r
'\\ /web/20130103030411/http:
:r
'\\ //whatwg.org/html
:rc
'\\2018' '\\2019' } /* `` `` */
:root:lang(sk), :not(:lang(sk)) > :lang(sk) { quotes: '\\201e' '\\201c'
'\\201a' '\\2018' } /* `` , ` */
:root:lang(sl), :not(:lang(sl)) > :lang(sl) { quotes: '\\201e' '\\201c'
'\\201a' '\\2018' } /* `` , ` */
:root:lang(sn), :not(:lang(sn)) > :lang(sn) { quotes: '\\201d' '\\201d'
'\\2019' '\\2019' } /* `` , ` */
:root:lang(so), :not(:lang(so)) > :lang(so) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(sq), :not(:lang(sq)) > :lang(sq) { quotes: '\\201e' '\\201c'
'\\201a' '\\2018' } /* `` , ` */
:root:lang(sr), :not(:lang(sr)) > :lang(sr) { quotes: '\\201e' '\\201c'
'\\201a' '\\2018' } /* `` , ` */
:root:lang(sr-Latn), :not(:lang(sr-Latn)) > :lang(sr-Latn) { quotes: '\\201e' '\\201c'
'\\201a' '\\2018' } /* `` , ` */
:root:lang(sv), :not(:lang(sv)) > :lang(sv) { quotes: '\\201d' '\\201d'
'\\2019' '\\2019' } /* `` , ` */
:root:lang(sw), :not(:lang(sw)) > :lang(sw) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(swc), :not(:lang(swc)) > :lang(swc) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(ta), :not(:lang(ta)) > :lang(ta) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(te), :not(:lang(te)) > :lang(te) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(teo), :not(:lang(teo)) > :lang(teo) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(th), :not(:lang(th)) > :lang(th) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(ti-ER), :not(:lang(ti-ER)) > :lang(ti-ER) { quotes: '\\2018' '\\2019'
'\\201c' '\\201d' } /* ' ' " */
:root:lang(to), :not(:lang(to)) > :lang(to) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(tr), :not(:lang(tr)) > :lang(tr) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(twq), :not(:lang(twq)) > :lang(twq) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(tzm), :not(:lang(tzm)) > :lang(tzm) { quotes: '\\201c' '\\201d'
'\\2018' '\\2019' } /* `` , ` */
:root:lang(uk), :not(:lang(uk)) > :lang(uk) { quotes: '\\00ab' '\\00bb'
'\\201e' '\\201c' } /* <> `` */
:root:lang(ur), :not(:lang(ur)) > :lang(ur) { quotes: '\\201d' '\\201c'
'\\2019' '\\2018' } /* `` , ` */
:root:lang(vai), :not(:lang(vai)) > :lang(vai) { quotes: '\\201c' '\\201d'

```

Go

**NOV JAN FEB**



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

▼ About this capture

### **14.3.7 Sections and headings**

```
@namespace url(http://www.w3.org/1999/xhtml);

article, aside, h1, h2, h3, h4, h5, h6, hgroup, nav, section {
 display: block;
}

h1 { margin-top: 0.67em; margin-bottom: 0.67em; font-size: 2.00em; font-weight: bold; }
h2 { margin-top: 0.83em; margin-bottom: 0.83em; font-size: 1.50em; font-weight: bold; }
h3 { margin-top: 1.00em; margin-bottom: 1.00em; font-size: 1.17em; font-weight: bold; }
h4 { margin-top: 1.33em; margin-bottom: 1.33em; font-size: 1.00em; font-weight: bold; }
h5 { margin-top: 1.67em; margin-bottom: 1.67em; font-size: 0.83em; font-weight: bold; }
h6 { margin-top: 2.33em; margin-bottom: 2.33em; font-size: 0.67em; font-weight: bold; }
```

The `article`, `aside`, `nav`, and `section` elements are expected to affect the margins and font size of `h1` elements, as well as `h2`–`h5` elements that follow `h1` elements in `hgroup` elements, based on the nesting depth. If `x` is a selector that matches elements that are either `article`, `aside`, `nav`, or `section` elements, then the following rules capture what is expected:

```
@namespace url (http://www.w3.org/1999/xhtml);

x h1 { margin-top: 0.83em; margin-bottom: 0.83em; font-size: 1.50em; }
x x h1 { margin-top: 1.00em; margin-bottom: 1.00em; font-size: 1.17em; }
x x x h1 { margin-top: 1.33em; margin-bottom: 1.33em; font-size: 1.00em; }
x x x x h1 { margin-top: 1.67em; margin-bottom: 1.67em; font-size: 0.83em; }
x x x x x h1 { margin-top: 2.33em; margin-bottom: 2.33em; font-size: 0.67em; }

x hgroup > h1 ~ h2 { margin-top: 1.00em; margin-bottom: 1.00em; font-size: 1.17em; }
x x hgroup > h1 ~ h2 { margin-top: 1.33em; margin-bottom: 1.33em; font-size: 1.00em; }
x x x hgroup > h1 ~ h2 { margin-top: 1.67em; margin-bottom: 1.67em; font-size: 0.83em; }
x x x x hgroup > h1 ~ h2 { margin-top: 2.33em; margin-bottom: 2.33em; font-size: 0.67em; }

x hgroup > h1 ~ h3 { margin-top: 1.33em; margin-bottom: 1.33em; font-size: 1.00em; }
x x hgroup > h1 ~ h3 { margin-top: 1.67em; margin-bottom: 1.67em; font-size: 0.83em; }
x x x hgroup > h1 ~ h3 { margin-top: 2.33em; margin-bottom: 2.33em; font-size: 0.67em; }

x hgroup > h1 ~ h4 { margin-top: 1.67em; margin-bottom: 1.67em; font-size: 0.83em; }
x x hgroup > h1 ~ h4 { margin-top: 2.33em; margin-bottom: 2.33em; font-size: 0.67em; }

x hgroup > h1 ~ h5 { margin-top: 2.33em; margin-bottom: 2.33em; font-size: 0.67em; }
```

### **14.3.8 Lists**

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
dir, dl, menu, ol, ul { margin-top: 1em; margin-bottom: 1em; }
```

```
dir dir, dir dl, dir menu, dir ol, dir ul,
dl dir, dl dl, dl menu, dl ol, dl ul,
menu dir, menu dl, menu menu, menu ol, menu ul,
ol dir, ol dl, ol menu, ol ol, ol ul,
ul dir, ul dl, ul menu, ul ol, ul ul {
 margin-top: 0; margin-bottom: 0;
}
```



**dd Too slow? Try reading the multipage copy of  
di the spec instead: https://web.archive.org  
el /web/20130103030411/http:  
ol //whatwg.org/html  
di.**

elements \*/  
-right' for rtl

```
dir dl, dir menu, dir ul,
menu dl, menu menu, menu ul,
ol dl, ol menu, ol ul,
ul dl, ul menu, ul ul {
 list-style-type: circle;
}
```

```
dir dir dl, dir dir menu, dir dir ul,
dir menu dl, dir menu menu, dir menu ul,
dir ol dl, dir ol menu, dir ol ul,
dir ul dl, dir ul menu, dir ul ul,
menu dir dl, menu dir menu, menu dir ul,
menu menu dl, menu menu menu, menu menu ul,
menu ol dl, menu ol menu, menu ol ul,
menu ul dl, menu ul menu, menu ul ul,
ol dir dl, ol dir menu, ol dir ul,
ol menu dl, ol menu menu, ol menu ul,
ol ol dl, ol ol menu, ol ol ul,
ol ul dl, ol ul menu, ol ul ul,
ul dir dl, ul dir menu, ul dir ul,
ul menu dl, ul menu menu, ul menu ul,
ul ol dl, ul ol menu, ul ol ul,
ul ul dl, ul ul menu, ul ul ul {
 list-style-type: square;
}
```

The following rules are also expected to apply, as presentational hints:

```
@namespace url (http://www.w3.org/1999/xhtml);

ol[type=1], li[type=1] { list-style-type: decimal; }
ol[type=a], li[type=a] { list-style-type: lower-alpha; }
ol[type=A], li[type=A] { list-style-type: upper-alpha; }
ol[type=i], li[type=i] { list-style-type: lower-roman; }
ol[type=I], li[type=I] { list-style-type: upper-roman; }
ul[type=disc i], li[type=disc i] { list-style-type: disc; }
ul[type=circle i], li[type=circle i] { list-style-type: circle; }
ul[type=square i], li[type=square i] { list-style-type: square; }
```

When rendering `li` elements, non-CSS user agents are expected to use the ordinal value of the `li` element to render the counter in the list item marker.

This specification does not yet define the CSS-specific rules for rendering `li` elements, because CSS doesn't yet provide

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

WHATWG

▼ About this capture

```
@namespace url("http://www.w3.org/1999/xhtml");\n\n table { display: table; }\n caption { display: table-caption; }\n colgroup, colgroup[hidden] { display: table-column-group; }\n col, col[hidden] { display: table-column; }\n thead, thead[hidden] { display: table-header-group; }\n tbody, tbody[hidden], tfoot, tfoot[hidden] { display: table-row-group; }\n tr { vertical-align: middle; }\n td, th { padding: 1px; }\n th { font-weight: bold; }\n\n thead, tbody, tfoot, table > tr { vertical-align: middle; }\n tr, td, th { vertical-align: inherit; }\n\n table, td, th { border-color: gray; }\n thead, tbody, tfoot, tr { border-color: inherit; }\n table[rules=none i], table[rules=groups i], table[rules=rows i],\n table[rules=cols i], table[rules=all i], table[frame=void i],\n table[frame=above i], table[frame=below i], table[frame=hsides i],\n table[frame=lhs i], table[frame=rhs i], table[frame=vsides i],\n table[frame=box i], table[frame=border i],\n table[rules=none i] > tr > td, table[rules=none i] > tr > th,\n table[rules=groups i] > tr > td, table[rules=groups i] > tr > th,\n table[rules=rows i] > tr > td, table[rules=rows i] > tr > th,\n table[rules=cols i] > tr > td, table[rules=cols i] > tr > th,\n table[rules=all i] > tr > td, table[rules=all i] > tr > th,\n table[rules=none i] > thead > tr > td, table[rules=none i] > thead > tr > th,\n table[rules=groups i] > thead > tr > td, table[rules=groups i] > thead > tr > th,\n table[rules=rows i] > thead > tr > td, table[rules=rows i] > thead > tr > th,\n table[rules=cols i] > thead > tr > td, table[rules=cols i] > thead > tr > th,\n table[rules=all i] > thead > tr > td, table[rules=all i] > thead > tr > th,\n table[rules=none i] > tbody > tr > td, table[rules=none i] > tbody > tr > th,\n table[rules=groups i] > tbody > tr > td, table[rules=groups i] > tbody > tr > th,\n table[rules=rows i] > tbody > tr > td, table[rules=rows i] > tbody > tr > th,\n table[rules=cols i] > tbody > tr > td, table[rules=cols i] > tbody > tr > th,\n table[rules=all i] > tbody > tr > td, table[rules=all i] > tbody > tr > th,\n table[rules=none i] > tfoot > tr > td, table[rules=none i] > tfoot > tr > th,\n table[rules=groups i] > tfoot > tr > td, table[rules=groups i] > tfoot > tr > th,\n table[rules=rows i] > tfoot > tr > td, table[rules=rows i] > tfoot > tr > th,\n table[rules=cols i] > tfoot > tr > td, table[rules=cols i] > tfoot > tr > th,\n table[rules=all i] > tfoot > tr > td, table[rules=all i] > tfoot > tr > th {\n border-color: black;\n }
```

The following rules are also expected to apply, as presentational hints:

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
table[align=center i] { margin-left: auto; margin-right: auto; }
thead[align=absmiddle i], tbody[align=absmiddle i], tfoot[align=absmiddle i],
tr[align=absmiddle i], td[align=absmiddle i], th[align=absmiddle i] {
 text-align: center;
}

caption[align=bottom i] { caption-side: bottom; }
p[align=left i], h1[align=left i], h2[align=left i], h3[align=left i],
h4[align=left i], h5[align=left i], h6[align=left i] {
 text-align: left;
}
p[align=right i], h1[align=right i], h2[align=right i], h3[align=right i],
h4[align=right i], h5[align=right i], h6[align=right i] {
 text-align: right;
}
p[align=justify i], h1[align=justify i], h2[align=justify i], h3[align=justify i],
h4[align=justify i], h5[align=justify i], h6[align=justify i] {
 text-align: justify;
}
thead[v-align=top i], tbody[v-align=top i], tfoot[v-align=top i],
tr[v-align=top i], td[v-align=top i], th[v-align=top i] {
 vertical-align: top;
}
thead[v-align=middle i], tbody[v-align=middle i], tfoot[v-align=middle i],
tr[v-align=middle i], td[v-align=middle i], th[v-align=middle i] {
 vertical-align: middle;
}
thead[v-align=bottom i], tbody[v-align=bottom i], tfoot[v-align=bottom i],
tr[v-align=bottom i], td[v-align=bottom i], th[v-align=bottom i] {
 vertical-align: bottom;
}
thead[v-align=baseline i], tbody[v-align=baseline i], tfoot[v-align=baseline i],
tr[v-align=baseline i], td[v-align=baseline i], th[v-align=baseline i] {
 vertical-align: baseline;
}

td[nowrap], th[nowrap] { white-space: nowrap; }

table[rules=none i], table[rules=groups i], table[rules=rows i],
table[rules=cols i], table[rules=all i] {
 border-style: hidden;
 border-collapse: collapse;
}
table[border] { border-style: outset; } /* only if border is not equivalent to zero */
table[frame=void i] { border-style: hidden; }
table[frame=above i] { border-style: outset hidden hidden hidden; }
table[frame=below i] { border-style: hidden hidden outset hidden; }
table[frame=hsides i] { border-style: outset hidden outset hidden; }
table[frame=lhs i] { border-style: hidden hidden hidden outset; }
table[frame=rhs i] { border-style: hidden outset hidden hidden; }
table[frame=vsides i] { border-style: hidden outset; }
table[frame=box i], table[frame=border i] { border-style: outset; }

table[border] > tr > td, table[border] > tr > th,
table[border] > thead > tr > td, table[border] > thead > tr > th,
table[border] > tbody > tr > td, table[border] > tbody > tr > th,
table[border] > tfoot > tr > td, table[border] > tfoot > tr > th {
 /* only if border is not equivalent to zero */
 border-width: 1px;
```

Go

**NOV JAN FEB**



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

2014

▼ About this capture

```
table[rules=none i] > tfoot > tr > td, table[rules=none i] > tfoot > tr > th,
table[rules=groups i] > tr > td, table[rules=groups i] > tr > th,
table[rules=groups i] > thead > tr > td, table[rules=groups i] > thead > tr > th,
table[rules=groups i] > tbody > tr > td, table[rules=groups i] > tbody > tr > th,
table[rules=groups i] > tfoot > tr > td, table[rules=groups i] > tfoot > tr > th,
table[rules=rows i] > tr > td, table[rules=rows i] > tr > th,
table[rules=rows i] > thead > tr > td, table[rules=rows i] > thead > tr > th,
table[rules=rows i] > tbody > tr > td, table[rules=rows i] > tbody > tr > th,
table[rules=rows i] > tfoot > tr > td. table[rules=rows i] > tfoot > tr > th {
}
:
Too slow? Try reading the multipage copy of
}
the spec instead: https://web.archive.org
ta > th,
ta > th,
ta > th {
/web/20130103030411/http:
//whatwg.org/html
k
}
table[rules=all i] > tr > td, table[rules=all i] > tr > th,
table[rules=all i] > thead > tr > td, table[rules=all i] > thead > tr > th,
table[rules=all i] > tbody > tr > td, table[rules=all i] > tbody > tr > th,
table[rules=all i] > tfoot > tr > td, table[rules=all i] > tfoot > tr > th {
border-width: 1px;
border-style: solid;
}

table[rules=groups i] > colgroup {
border-left-width: 1px;
border-left-style: solid;
border-right-width: 1px;
border-right-style: solid;
}
table[rules=groups i] > thead,
table[rules=groups i] > tbody,
table[rules=groups i] > tfoot {
border-top-width: 1px;
border-top-style: solid;
border-bottom-width: 1px;
border-bottom-style: solid;
}

table[rules=rows i] > tr, table[rules=rows i] > thead > tr,
table[rules=rows i] > tbody > tr, table[rules=rows i] > tfoot > tr {
border-top-width: 1px;
border-top-style: solid;
border-bottom-width: 1px;
border-bottom-style: solid;
}
```

In quirks mode, the following rules are also expected to apply:

```
@namespace url (http://www.w3.org/1999/xhtml)

table {
 font-weight: initial;
 font-style: initial;
 font-variant: initial;
 font-size: initial;
 line-height: initial;
 white-space: initial;
 text-align: initial;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

For the purposes of the CSS table model, the `colgroup` element, if it contains no `col` element, is expected to be treated as if it had as many such children as its `span` attribute specifies.

For the purposes of the CSS table model, the `colspan` and `rowspan` attributes on `td` and `th` elements are expected to provide the *special knowledge* regarding cells spanning rows and columns.

In HTML documents, the user agent is expected to force the 'display' property of `form` elements that are children of `table`, `thead`, `tbody`, `tfoot`, or `tr` elements to compute to 'none', irrespective of CSS rules.



The `table`: Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

ment.

ght', 'padding-bottom', and `table` element.the `table` element.on the `table` element.

The `table` element's `height` attribute maps to the dimension property 'height' on the `table` element.

The `table` element's `width` attribute maps to the dimension property 'width' on the `table` element.

The `col` element's `width` attribute maps to the dimension property 'width' on the `col` element.

The `tr` element's `height` attribute maps to the dimension property 'height' on the `tr` element.

The `td` and `th` elements' `height` attributes map to the dimension property 'height' on the `td` element.

The `td` and `th` elements' `width` attributes map to the dimension property 'width' on the `td` element.

When a Document is in quirks mode, vertical margins on HTML elements at the top or bottom of `td` or `th` elements are expected to be collapsed to zero.

The `caption` element unless specified otherwise below, and the `thead`, `tbody`, `tfoot`, `tr`, `td`, and `th` elements when they have an `align` attribute whose value is an ASCII case-insensitive match for either the string "center" or the string "middle", are expected to center text within themselves, as if they had their 'text-align' property set to 'center' in a presentational hint, and to align descendants to the center.

The `caption`, `thead`, `tbody`, `tfoot`, `tr`, `td`, and `th` elements, when they have an `align` attribute whose value is an ASCII case-insensitive match for the string "left", are expected to left-align text within themselves, as if they had their 'text-align' property set to 'left' in a presentational hint, and to align descendants to the left.

The `caption`, `thead`, `tbody`, `tfoot`, `tr`, `td`, and `th` elements, when they have an `align` attribute whose value is an ASCII case-insensitive match for the string "right", are expected to right-align text within themselves, as if they had their 'text-align' property set to 'right' in a presentational hint, and to align descendants to the right.

The `caption`, `thead`, `tbody`, `tfoot`, `tr`, `td`, and `th` elements, when they have an `align` attribute whose value is an ASCII case-insensitive match for the string "justify", are expected to full-justify text within themselves, as if they had their 'text-align' property set to 'justify' in a presentational hint, and to align descendants to the left.

User agents are expected to have a rule in their user agent stylesheet that matches `th` elements that have a parent node whose computed value for the 'text-align' property is its initial value, whose declaration block consists of just a single declaration that sets the 'text-align' property to the value 'center'.

When a `table`, `thead`, `tbody`, `tfoot`, `tr`, `td`, or `th` element has a `background` attribute set to a non-empty value, the new value is expected to be resolved relative to the element, and if this is successful, the user agent is expected to treat the attribute as a presentational hint setting the element's 'background-image' property to the resulting absolute URL.

When a `table`, `thead`, `tbody`, `tfoot`, `tr`, `td`, or `th` element has a `bgcolor` attribute set, the new value is expected to be parsed using the rules for parsing a legacy color value, and if that does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's 'background-color' property to the resulting color.

When a `table` element has a `bordercolor` attribute, its value is expected to be parsed using the rules for parsing a legacy color

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

'border-left-width' on the element. If the attribute is present but parsing the attribute's value using the rules for parsing non-negative integers generates an error, a default value of 1px is expected to be used for that property instead.

Rules marked "only if border is not equivalent to zero" in the CSS block above is expected to only be applied if the border attribute mentioned in the selectors for the rule is not only present but, when parsed using the rules for parsing non-negative integers, is also found to have a value other than zero or to generate an error.

In quirks mode, a td element or a th element that has a nowrap attribute but also has a width attribute whose value, when parsed using the rules for parsing dimension values, is found to be a length (not an error or a number followed as a percentage), is expected to have a pre

sets it to 'no' Too slow? Try reading the multipage copy of

User agents the spec instead: https://web.archive.org

cause the ta /web/20130103030411/http:

//whatwg.org/html

### 14.3.10 F

```
@namespace url(http://www.w3.org/1999/xhtml);

input, select, option, optgroup, button, textarea, keygen {
 text-indent: initial;
}

textarea { white-space: pre-wrap; }

input[type="radio"], input[type="checkbox"], input[type="reset"], input[type="button"],
input[type="submit"], select, button {
 box-sizing: border-box;
}
```

In quirks mode, the following rules are also expected to apply:

```
@namespace url(http://www.w3.org/1999/xhtml);

input { box-sizing: border-box; }
```

Each kind of form control is also given a specific default binding, as described in subsequent sections, which implements the look and feel of the control.

### 14.3.11 The hr element

```
@namespace url(http://www.w3.org/1999/xhtml);

hr { color: gray; border-style: inset; border-width: 1px; margin: 0.5em auto; }
```

The following rules are also expected to apply, as presentational hints:

```
@namespace url(http://www.w3.org/1999/xhtml);

hr[align=left] { margin-left: 0; margin-right: auto; }
hr[align=right] { margin-left: auto; margin-right: 0; }
hr[align=center] { margin-left: auto; margin-right: auto; }
hr[color], hr[noshade] { border-style: solid; }
```

If an hr element has either a color attribute or a noshade attribute, and furthermore also has a size attribute, and parsing that attribute's value using the rules for parsing non-negative integers doesn't generate an error, then the user agent is expected to use the parsed value divided by two as a pixel length for presentational hints for the properties 'border-top-width', 'border-right-width', 'border-bottom-width', and 'border-left-width' on the element.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

hints for the height property on the element

The `width` attribute on an `hr` element maps to the dimension property 'width' on the element.

When an `hr` element has a `color` attribute, its value is expected to be parsed using the rules for parsing a legacy color value, and if that does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's 'color' property to the resulting color.

### 14.3.12 The `fieldset` element



@n Too slow? Try reading the multipage copy of  
fi the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `fieldset` element is expected to establish a new block formatting context.

If the `fieldset` element has a child that matches the conditions in the list below, then the first such child is the `fieldset` element's **rendered legend**:

- The child is a `legend` element.
- The child is not out-of-flow (e.g. not absolutely positioned or floated).
- The child is generating a box (e.g. it is not 'display:none').

A `fieldset` element's rendered legend, if any, is expected to be rendered over the top border edge of the `fieldset` element as a 'block' box (overriding any explicit 'display' value). In the absence of an explicit width, the box should shrink-wrap. If the `legend` element in question has an `align` attribute, and its value is an ASCII case-insensitive match for one of the strings in the first column of the following table, then the `legend` is expected to be rendered horizontally aligned over the border edge in the position given in the corresponding cell on the same row in the second column. If the attribute is absent or has a value that doesn't match any of the cases in the table, then the position is expected to be on the right if the 'direction' property on this element has a computed value of 'rtl', and on the left otherwise.

Attribute value	Alignment position
left	On the left
right	On the right
center	In the middle

## 14.4 Replaced elements

### 14.4.1 Embedded content

The `embed`, `iframe`, and `video` elements are expected to be treated as replaced elements.

A `canvas` element that represents embedded content is expected to be treated as a replaced element; the contents of such elements are the element's bitmap, if any, or else a transparent black bitmap with the same intrinsic dimensions as the element. Other `canvas` elements are expected to be treated as ordinary elements in the rendering model.

An `object` element that represents an image, plugin, or nested browsing context is expected to be treated as a replaced element. Other `object` elements are expected to be treated as ordinary elements in the rendering model.

An `applet` element that represents a plugin is expected to be treated as a replaced element. Other `applet` elements are expected to be treated as ordinary elements in the rendering model.

The `audio` element, when it is exposing a user interface, is expected to be treated as a replaced element about one line high, as wide as is necessary to expose the user agent's user interface features. When an `audio` element is not exposing a user interface, the user agent is expected to force its 'display' property to compute to 'none', irrespective of CSS rules.

Whether a `video` element is exposing a user interface is not expected to affect the size of the rendering; controls are expected to be overlaid above the page content without causing any layout changes, and are expected to disappear when the user does not need

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Any subtitles or captions are expected to be overlayed directly on top of their `video` element, as defined by the relevant rendering rules; for WebVTT, those are the WebVTT cue text rendering rules. [WEBVTT]

When the user agent starts exposing a user interface for a `video` element, the user agent should run the rules for updating the text track rendering of each of the text tracks in the `video` element's list of text tracks that are showing (e.g., for text tracks based on WebVTT, the rules for updating the display of WebVTT text tracks). [WEBVTT]

**Note:** Resizing `video` and `canvas` elements does not interrupt video playback or clear `canvas`.

The following text is from the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

```
@n
if
if
vic
```

#### 14.4.2 Images

When an `img` element or an `input` element when its `type` attribute is in the Image Button state represents an image, it is expected to be treated as a replaced element.

When an `img` element or an `input` element when its `type` attribute is in the Image Button state does not represent an image, but the element already has intrinsic dimensions (e.g. from the dimension attributes or CSS rules), and either the user agent has reason to believe that the image will become *available* and be rendered in due course or the Document is in quirks mode, the element is expected to be treated as a replaced element whose content is the text that the element represents, if any, optionally alongside an icon indicating that the image is being obtained. For `input` elements, the text is expected to appear button-like to indicate that the element is a button.

When an `img` element represents some text and the user agent does not expect this to change, the element is expected to be treated as a non-replaced phrasing element whose content is the text, optionally with an icon indicating that an image is missing.

When an `img` element represents nothing and the user agent does not expect this to change, the element is expected to not be rendered at all.

When an `img` element might be a key part of the content, but neither the image nor any kind of alternative text is available, and the user agent does not expect this to change, the element is expected to be treated as a non-replaced phrasing element whose content is an icon indicating that an image is missing.

When an `input` element whose `type` attribute is in the Image Button state does not represent an image and the user agent does not expect this to change, the element is expected to be treated as a replaced element consisting of a button whose content is the element's alternative text. The intrinsic dimensions of the button are expected to be about one line in height and whatever width is necessary to render the text on one line.

The icons mentioned above are expected to be relatively small so as not to disrupt most text but be easily clickable. In a visual environment, for instance, icons could be 16 pixels by 16 pixels square, or 1em by 1em if the images are scalable. In an audio environment, the icon could be a short bleep. The icons are intended to indicate to the user that they can be used to get to whatever options the UA provides for images, and, where appropriate, are expected to provide access to the context menu that would have come up if the user interacted with the actual image.

All animated images with the same absolute URL and the same image data are expected to be rendered synchronized to the same timeline as a group, with the timeline starting at the time of the most recent addition to the group.

**Note:** In other words, the animation loop of an animated image is restarted each time another image with the same absolute URL and image data begins to animate, e.g. after being inserted into the document.

The following CSS rules are expected to apply when the Document is in quirks mode:

```
@namespace url (http://www.w3.org/1999/xhtml) ;
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

#### 14.4.3 Attributes for embedded content and images

The following CSS rules are expected to apply as presentational hints:

```
@namespace url("http://www.w3.org/1999/xhtml");

iframe[frameborder=0], iframe[frameborder=no i] { border: none; }

applet[align=top i], embed[align=top i], img[align=top i], input[type=image i][align=top i], object[align=top i] {
 vertical-align: top;
}

applet[align=baseline i], embed[align=baseline i], iframe[align=baseline i], img[align=baseline i], input[type=image i][align=baseline i], object[align=baseline i] {
 vertical-align: baseline;
}

applet[align=texttop i], embed[align=texttop i], iframe[align=texttop i], img[align=texttop i], input[type=image i][align=texttop i], object[align=texttop i] {
 vertical-align: text-top;
}

applet[align=absmiddle i], embed[align=absmiddle i], iframe[align=absmiddle i], img[align=absmiddle i], input[type=image i][align=absmiddle i], object[align=absmiddle i],
applet[align=abscenter i], embed[align=abscenter i], iframe[align=abscenter i], img[align=abscenter i], input[type=image i][align=abscenter i], object[align=abscenter i] {
 vertical-align: middle;
}

applet[align=bottom i], embed[align=bottom i], iframe[align=bottom i], img[align=bottom i], input[type=image i][align=bottom i],
object[align=bottom i] {
 vertical-align: bottom;
}
```

When an `applet`, `embed`, `iframe`, `img`, or `object` element, or an `input` element whose `type` attribute is in the Image Button state, has an `align` attribute whose value is an ASCII case-insensitive match for the string "center" or the string "middle", the user agent is expected to act as if the element's 'vertical-align' property was set to a value that aligns the vertical middle of the element with the parent element's baseline.

The `hspace` attribute of `applet`, `embed`, `iframe`, `img`, or `object` elements, and `input` elements with a `type` attribute in the Image Button state, maps to the dimension properties 'margin-left' and 'margin-right' on the element.

The `vspace` attribute of `applet`, `embed`, `iframe`, `img`, or `object` elements, and `input` elements with a `type` attribute in the Image Button state, maps to the dimension properties 'margin-top' and 'margin-bottom' on the element.

When an `img` element, `object` element, or `input` element with a `type` attribute in the Image Button state has a `border` attribute whose value, when parsed using the rules for parsing non-negative integers, is found to be a number greater than zero, the user agent is expected to use the parsed value for eight presentational hints: four setting the parsed value as a pixel length for the element's 'border-top-width', 'border-right-width', 'border-bottom-width', and 'border-left-width' properties, and four setting the element's 'border-top-style', 'border-right-style', 'border-bottom-style', and 'border-left-style' properties to the value 'solid'.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

#### 14.4.4 Image maps

Shapes on an image map are expected to act, for the purpose of the CSS cascade, as elements independent of the original `area` element that happen to match the same style rules but inherit from the `img` or `object` element.

For the purposes of the rendering, only the 'cursor' property is expected to have any effect on the shape.

Thus, for example, if an `area` element has a `style` attribute that sets the 'cursor' property to 'help', then when the user designates that shape, the cursor would change to a Help cursor.



Similar to the `area` element, the `img` element also has a `style` attribute that sets the 'cursor' property to 'help'. This is because the `area` element is a child of the `img` element, and the `area` element's styling is inherited by the `img` element.

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

ting the 'cursor' property of the image map, not from

#### 14.5 Bin

#### 14.5.1 Int

A number of elements have their rendering defined in terms of the 'binding' property. [BECSS]

The CSS snippets below set the 'binding' property to a user-agent-defined value, represented below by keywords like `button`. The rules then described for these bindings are only expected to apply if the element's 'binding' property has not been overridden (e.g. by the author) to have another value.

Exactly how the bindings are implemented is not specified by this specification. User agents are encouraged to make their bindings set the 'appearance' CSS property appropriately to achieve platform-native appearances for widgets, and are expected to implement any relevant animations, etc, that are appropriate for the platform. [CSSUI]

#### 14.5.2 The `button` element

```
@namespace url("http://www.w3.org/1999/xhtml");
button { binding: button; }
```

When the `button` binding applies to a `button` element, the element is expected to render as an 'inline-block' box rendered as a button whose contents are the contents of the element.

When the `button` element's `type` attribute is in the Menu state, the user agent is expected to indicate that activating the element will display a menu, e.g. by displaying a down-pointing triangle after the button's label.

#### 14.5.3 The `details` element

```
@namespace url("http://www.w3.org/1999/xhtml");
details { binding: details; }
```

When the `details` binding applies to a `details` element, the element is expected to render as a 'block' box with its 'padding-left' property set to '40px' for left-to-right elements (LTR-specific) and with its 'padding-right' property set to '40px' for right-to-left elements. The element's shadow tree is expected to take the element's first child `summary` element, if any, and place it in a first 'block' box container, and then take the element's remaining descendants, if any, and place them in a second 'block' box container.

The first container is expected to contain at least one line box, and that line box is expected to contain a disclosure widget (typically a triangle), horizontally positioned within the left padding of the `details` element. That widget is expected to allow the user to request that the details be shown or hidden.

The second container is expected to have its 'overflow' property set to 'hidden'. When the `details` element does not have an `open` attribute, this second container is expected to be removed from the rendering.

#### 14.5.4 The `input` element as a text entry widget

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

/\* later rules override this for other values of type="" \*/

When the *input-textfield* binding applies to an `input` element whose `type` attribute is in the Text, Search, Telephone, URL, or E-mail state, the element is expected to render as an 'inline-block' box rendered as a text field.

When the *input-password* binding applies, to an `input` element whose `type` attribute is in the Password state, the element is expected to render as an 'inline-block' box rendered as a text field whose contents are obscured.

If an `input` element whose `type` attribute is in one of the above states has a `size` attribute, and parsing that attribute's value using the rules for `pixels` results in a value between 1 and 1000 pixels, the user agent is expected to use the attribute as a presentation pixels algorithm.

## Too slow? Try reading the multipage copy of

the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The converter converts the average character width to convert, `avg` is

the average character width of that same font, also in pixels. (The element's 'letter-spacing' property does not affect the result.)

### 14.5.5 The `input` element as domain-specific widgets

```
@namespace url (http://www.w3.org/1999/xhtml) ;

input[type=datetime] { binding: input-datetime; }
input[type=date] { binding: input-date; }
input[type=month] { binding: input-month; }
input[type=week] { binding: input-week; }
input[type=time] { binding: input-time; }
input[type=datetime-local] { binding: input-datetime-local; }
input[type=number] { binding: input-number; }
```

When the *input-datetime* binding applies to an `input` element whose `type` attribute is in the Date and Time state, the element is expected to render as an 'inline-block' box depicting a Date and Time control.

When the *input-date* binding applies to an `input` element whose `type` attribute is in the Date state, the element is expected to render as an 'inline-block' box depicting a Date control.

When the *input-month* binding applies to an `input` element whose `type` attribute is in the Month state, the element is expected to render as an 'inline-block' box depicting a Month control.

When the *input-week* binding applies to an `input` element whose `type` attribute is in the Week state, the element is expected to render as an 'inline-block' box depicting a Week control.

When the *input-time* binding applies to an `input` element whose `type` attribute is in the Time state, the element is expected to render as an 'inline-block' box depicting a Time control.

When the *input-datetime-local* binding applies to an `input` element whose `type` attribute is in the Local Date and Time state, the element is expected to render as an 'inline-block' box depicting a Local Date and Time control.

When the *input-number* binding applies to an `input` element whose `type` attribute is in the Number state, the element is expected to render as an 'inline-block' box depicting a Number control.

These controls are all expected to be about one line high, and about as wide as necessary to show the widest possible value.

### 14.5.6 The `input` element as a range control

```
@namespace url (http://www.w3.org/1999/xhtml) ;

input[type=range] { binding: input-range; }
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

### 3,445 captures

5 Jun 2004 - 22 Aug 2020

it is expected to be a vertical slider, with the lowest value on the bottom.

Predefined suggested values (provided by the `list` attribute) are expected to be shown as tick marks on the slider, which the slider can snap to.

User agents are expected to use the used value of the 'direction' property on the element to determine the direction in which the slider operates. Typically, a left-to-right ('ltr') horizontal control would have the lowest value on the left and the highest value on the right, and vice versa.



### 14.5.7 Th Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org

`@n` `in` /web/20130103030411/http://whatwg.org/html

When the `in` attribute is present, the element is expected to render as an 'inline-block' box containing a color picker interface (e.g. a color wheel or color palette) from which the color can be changed.

Predefined suggested values (provided by the `list` attribute) are expected to be shown in the color picker interface, not on the color well itself.

### 14.5.8 The `input` element as a checkbox and radio button widgets

```
@namespace url("http://www.w3.org/1999/xhtml");
input[type=checkbox] i { binding: input-checkbox; }
input[type=radio] i { binding: input-radio; }
```

When the `input-checkbox` binding applies to an `input` element whose `type` attribute is in the Checkbox state, the element is expected to render as an 'inline-block' box containing a single checkbox control, with no label.

When the `input-radio` binding applies to an `input` element whose `type` attribute is in the Radio Button state, the element is expected to render as an 'inline-block' box containing a single radio button control, with no label.

### 14.5.9 The `input` element as a file upload control

```
@namespace url("http://www.w3.org/1999/xhtml");
input[type=file] i { binding: input-file; }
```

When the `input-file` binding applies to an `input` element whose `type` attribute is in the File Upload state, the element is expected to render as an 'inline-block' box containing a span of text giving the filename(s) of the selected files, if any, followed by a button that, when activated, provides the user with a file picker from which the selection can be changed.

### 14.5.10 The `input` element as a button

```
@namespace url("http://www.w3.org/1999/xhtml");
input[type=submit] i, input[type=reset] i, input[type=button] i {
 binding: input-button;
}
```

When the `input-button` binding applies to an `input` element whose `type` attribute is in the Submit Button, Reset Button, or Button state, the element is expected to render as an 'inline-block' box rendered as a button, about one line high, containing the contents of the element's `value` attribute, if any, or text derived from the element's `type` attribute in a user-agent-defined (and probably locale-specific) fashion, if not.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When the `marquee` binding applies to a `marquee` element, while the element is turned on, the element is expected to render in an animated fashion according to its attributes as follows:

#### If the element's `behavior` attribute is in the scroll state

Slide the contents of the element in the direction described by the `direction` attribute as defined below, such that it begins off the start side of the `marquee`, and ends flush with the inner end side.

Once you turn on the element, slide the contents of the element in the direction described by the `direction` attribute as defined below, such that their left edge are off the start side of the `marquee`, and ends flush with the inner end side.

For example, if the `direction` attribute is `left` (the default), then the contents would begin such that their left edge are off the start side of the `marquee`'s content area, and the contents would then slide up to the point where the `right` edge of the contents are flush with the inner end side of the `marquee`'s content area.

If the element's `behavior` attribute is in the alternate state

Slide the contents of the element in the direction described by the `direction` attribute as defined below, such that it begins off the start side of the `marquee`.

Once the animation has ended, the user agent is expected to increment the `marquee` current loop index. If the element is still turned on after this, then the user agent is expected to restart the animation.

#### If the element's `behavior` attribute is in the alternate state

When the `marquee` current loop index is even (or zero), slide the contents of the element in the direction described by the `direction` attribute as defined below, such that it begins flush with the start side of the `marquee`, and ends flush with the end side of the `marquee`.

When the `marquee` current loop index is odd, slide the contents of the element in the opposite direction than that described by the `direction` attribute as defined below, such that it begins flush with the end side of the `marquee`, and ends flush with the start side of the `marquee`.

For example, if the `direction` attribute is `left` (the default), then the contents would begin with their right edge flush with the right inner edge of the `marquee`'s content area, and the contents would then slide up to the point where the `left` edge of the contents are flush with the left inner edge of the `marquee`'s content area.

Once the animation has ended, the user agent is expected to increment the `marquee` current loop index. If the element is still turned on after this, then the user agent is expected to continue the animation.

The `direction` attribute has the meanings described in the following table:

direction attribute state	Direction of animation	Start edge	End edge	Opposite direction
left	← Right to left	Right	Left	→ Left to Right
right	→ Left to Right	Left	Right	← Right to left
up	↑ Up (Bottom to Top)	Bottom	Top	↓ Down (Top to Bottom)
down	↓ Down (Top to Bottom)	Top	Bottom	↑ Up (Bottom to Top)

In any case, the animation should proceed such that there is a delay given by the `marquee` scroll interval between each frame, and such that the content moves at most the distance given by the `marquee` scroll distance with each frame.

When a `marquee` element has a `bgcolor` attribute set, the value is expected to be parsed using the rules for parsing a legacy color value, and if that does not return an error, the user agent is expected to treat the attribute as a presentational hint setting the element's '`background-color`' property to the resulting color.

The `width` and `height` attributes on a `marquee` element map to the dimension properties '`width`' and '`height`' on the element respectively.

The intrinsic height of a `marquee` element with its `direction` attribute in the up or down states is 200 CSS pixels.

The `vspace` attribute of a `marquee` element maps to the dimension properties '`margin-top`' and '`margin-bottom`' on the element. The

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

#### 14.5.12 The meter element

```
@namespace url("http://www.w3.org/1999/xhtml");
meter { binding: meter; }
```

When the `meter` binding applies to a `meter` element, the element is expected to render as an 'inline-block' box with a 'height' of '1em' and a 'width' of '5em', a 'vertical-align' of '-0.2em', and with its contents depicting a gauge.

When the element is wider than it is tall, the element is expected to be depicted as a horizontal progress bar, with the start on the right and the end on the left if the 'direction' property on this element has a computed value of 'rtl', and with the start on the left and the end on the right otherwise.

User agents

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

with the minimum value on the left, and the maximum value on the right if the element is taller than it is wide.

Note: [F](#)

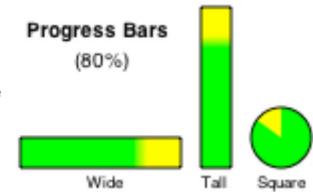
[? meter element.](#)

#### 14.5.13 The progress element

```
@namespace url("http://www.w3.org/1999/xhtml");
progress { binding: progress; }
```

When the `progress` binding applies to a `progress` element, the element is expected to render as an 'inline-block' box with a 'height' of '1em' and a 'width' of '10em', and a 'vertical-align' of '-0.2em'.

When the element is wider than it is tall, the element is expected to be depicted as a horizontal progress bar, with the start on the right and the end on the left if the 'direction' property on this element has a computed value of 'rtl', and with the start on the left and the end on the right otherwise. When the element is taller than it is wide, it is expected to be depicted as a vertical progress bar, with the lowest value on the bottom. When the element is square, it is expected to be depicted as a direction-independent progress widget (e.g. a circular progress ring).



User agents are expected to use a presentation consistent with platform conventions for progress bars. In particular, user agents are expected to use different presentations for determinate and indeterminate progress bars. User agents are also expected to vary the presentation based on the dimensions of the element.

For example, on some platforms for showing indeterminate progress there is an asynchronous progress indicator with square dimensions, which could be used when the element is square, and an indeterminate progress bar, which could be used when the element is wide.

**Note: Requirements for how to determine if the progress bar is determinate or indeterminate, and what progress a determinate progress bar is to show, are included in the definition of the progress element.**

#### 14.5.14 The select element

```
@namespace url("http://www.w3.org/1999/xhtml");
select { binding: select; }
```

When the `select` binding applies to a `select` element whose `multiple` attribute is present, the element is expected to render as a multi-select list box.

When the `select` binding applies to a `select` element whose `multiple` attribute is absent, and the element's display size is greater than 1, the element is expected to render as a single-select list box.

When the element renders as a list box, it is expected to render as an 'inline-block' box whose 'height' is the height necessary to contain as many rows for items as given by the element's display size, or four rows if the attribute is absent, and whose 'width' is the width of the `select`'s labels plus the width of a scrollbar.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

An `optgroup` element is expected to be rendered by displaying the element's `label` attribute.

An `option` element is expected to be rendered by displaying the element's label, indented under its `optgroup` element if it has one.

The **width of the select's labels** is the wider of the width necessary to render the widest `optgroup`, and the width necessary to render the widest `option` element in the element's list of options (including its indent, if any).

If a `select` element contains a placeholder label option, the user agent is expected to render that `option` in a manner that conveys that it is a label selected by the user agent. The user agent is also expected to prevent the placeholder label option from being explicitly displayed in a fashion that indicates that it is a label selected by the user agent.

## Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

User agents are being displayed in a fashion that is consistent whether the label

**14.5.15** [T](#) [@namespace url\(<http://www.w3.org/1999/xhtml>\) ;](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
textarea { binding: textarea; white-space: pre-wrap; }
```

When the `textarea` binding applies to a `textarea` element, the element is expected to render as an 'inline-block' box rendered as a multiline text field.

If the element has a `cols` attribute, and parsing that attribute's value using the rules for parsing non-negative integers doesn't generate an error, then the user agent is expected to use the attribute as a presentational hint for the 'width' property on the element, with the value being the `textarea` effective width (as defined below). Otherwise, the user agent is expected to act as if it had a user-agent-level style sheet rule setting the 'width' property on the element to the `textarea` effective width.

The **textarea effective width** of a `textarea` element is  $\text{size} \times \text{avg} + \text{sbw}$ , where `size` is the element's character width, `avg` is the average character width of the primary font of the element, in CSS pixels, and `sbw` is the width of a scroll bar, in CSS pixels. (The element's 'letter-spacing' property does not affect the result.)

If the element has a `rows` attribute, and parsing that attribute's value using the rules for parsing non-negative integers doesn't generate an error, then the user agent is expected to use the attribute as a presentational hint for the 'height' property on the element, with the value being the `textarea` effective height (as defined below). Otherwise, the user agent is expected to act as if it had a user-agent-level style sheet rule setting the 'height' property on the element to the `textarea` effective height.

The **textarea effective height** of a `textarea` element is the height in CSS pixels of the number of lines specified the element's character height, plus the height of a scrollbar in CSS pixels.

User agents are expected to apply the 'white-space' CSS property to `textarea` elements. For historical reasons, if the element has a `wrap` attribute whose value is an ASCII case-insensitive match for the string "off", then the user agent is expected to treat the attribute as a presentational hint setting the element's 'white-space' property to 'pre'.

## 14.5.16 The `keygen` element

```
@namespace url(http://www.w3.org/1999/xhtml) ;
```

```
keygen { binding: keygen; }
```

When the `keygen` binding applies to a `keygen` element, the element is expected to render as an 'inline-block' box containing a user interface to configure the key pair to be generated.

## 14.6 Frames and framesets

User agents are expected to render `frameset` elements as a box with the height and width of the viewport, with a surface rendered according to the following layout algorithm:

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Use the rules for parsing a list of dimensions to parse the value of the element's `rows` attribute, if there is one. Let `rows` be the result, or an empty list if there is no such attribute.

2. For any of the entries in `cols` or `rows` that have the number zero and the unit `relative`, change the entry's number to one.

3. If `cols` has no entries, then add a single entry consisting of the value 1 and the unit `relative` to `cols`.

If `rows` has no entries, then add a single entry consisting of the value 1 and the unit `relative` to `rows`.

4. Invok

widt

rest

Invoc

heig

rest

5. Spli

sizeo

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

`cols` as the input list, and the on. Let `sized cols` be the

`rows` as the input list, and the ion. Let `sized rows` be the

`n` is the number of entries in

Size the columns so that each column in the grid is as many CSS pixels wide as the corresponding entry in the `sized cols` list.

Size the rows so that each row in the grid is as many CSS pixels high as the corresponding entry in the `sized rows` list.

6. Let `children` be the list of `frame` and `frameset` elements that are children of the `frameset` element for which the algorithm was invoked.

7. For each row of the grid of rectangles created in the previous step, from top to bottom, run these substeps:

1. For each rectangle in the row, from left to right, run these substeps:

1. If there are any elements left in `children`, take the first element in the list, and assign it to the rectangle.

If this is a `frameset` element, then recurse the entire `frameset` layout algorithm for that `frameset` element, with the rectangle as the surface.

Otherwise, it is a `frame` element; create a nested browsing context sized to fit the rectangle.

2. If there are any elements left in `children`, remove the first element from `children`.

8. If the `frameset` element has a border, draw an outer set of borders around the rectangles, using the element's frame border color.

For each rectangle, if there is an element assigned to that rectangle, and that element has a border, draw an inner set of borders around that rectangle, using the element's frame border color.

For each (visible) border that does not abut a rectangle that is assigned a `frame` element with a `noresize` attribute (including rectangles in further nested `frameset` elements), the user agent is expected to allow the user to move the border, resizing the rectangles within, keeping the proportions of any nested `frameset` grids.

A `frameset` or `frame` element **has a border** if the following algorithm returns true:

1. If the element has a `frameborder` attribute whose value is not the empty string and whose first character is either a U+0031 DIGIT ONE (1) character, a U+0079 LATIN SMALL LETTER Y character (y), or a U+0059 LATIN CAPITAL LETTER Y character (Y), then return true.
2. Otherwise, if the element has a `frameborder` attribute, return false.
3. Otherwise, if the element has a parent element that is a `frameset` element, then return true if *that* element has a border, and false if it does not.
4. Otherwise, return true.

The **frame border color** of a `frameset` or `frame` element is the color obtained from the following algorithm:

1. If the element has a `bordercolor` attribute, and applying the rules for parsing a legacy color value to that attribute's value does not result in an error, then return the color so obtained.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The algorithm to convert a list of dimensions to a list of pixel values consists of the following steps:

1. Let *input list* be the list of numbers and units passed to the algorithm.

Let *output list* be a list of numbers the same length as *input list*, all zero.

Entries in *output list* correspond to the entries in *input list* that have the same position.

2. Let *input dimension* be the size passed to the algorithm.

3. Let *co*

Let **Too slow? Try reading the multipage copy of**

**the spec instead: https://web.archive.org**

**/web/20130103030411/http:**

**//whatwg.org/html**

Let *t*

Let *remaining space* be the value of *input dimension*.

4. If *total absolute* is greater than *remaining space*, then for each entry in *input list* whose unit is *absolute*, set the corresponding value in *output list* to the number of the entry in *input list* multiplied by *remaining space* and divided by *total absolute*. Then, set *remaining space* to zero.

Otherwise, for each entry in *input list* whose unit is *absolute*, set the corresponding value in *output list* to the number of the entry in *input list*. Then, decrement *remaining space* by *total absolute*.

5. If *total percentage* multiplied by the *input dimension* and divided by 100 is greater than *remaining space*, then for each entry in *input list* whose unit is *percentage*, set the corresponding value in *output list* to the number of the entry in *input list* multiplied by *remaining space* and divided by *total percentage*. Then, set *remaining space* to zero.

Otherwise, for each entry in *input list* whose unit is *percentage*, set the corresponding value in *output list* to the number of the entry in *input list* multiplied by the *input dimension* and divided by 100. Then, decrement *remaining space* by *total percentage* multiplied by the *input dimension* and divided by 100.

6. For each entry in *input list* whose unit is *relative*, set the corresponding value in *output list* to the number of the entry in *input list* multiplied by *remaining space* and divided by *total relative*.

7. Return *output list*.

User agents working with integer values for frame widths (as opposed to user agents that can lay frames out with subpixel accuracy) are expected to distribute the remainder first to the last entry whose unit is *relative*, then equally (not proportionally) to each entry whose unit is *percentage*, then equally (not proportionally) to each entry whose unit is *absolute*, and finally, failing all else, to the last entry.

## 14.7 Interactive media

### 14.7.1 Links, forms, and navigation

User agents are expected to allow the user to control aspects of hyperlink activation and form submission, such as which browsing context is to be used for the subsequent navigation.

User agents are expected to allow users to discover the destination of hyperlinks and of forms before triggering their navigation.

User agents are expected to inform the user of whether a hyperlink includes hyperlink auditing, and to let them know at a minimum which domains will be contacted as part of such auditing.

User agents may allow users to navigate browsing contexts to the resources indicated by the `cite` attributes on `q`, `blockquote`, `ins`, and `del` elements.

User agents may surface hyperlinks created by `link` elements in their user interface.

**Note:** While `link` elements that create hyperlinks will match the ':link' or ':visited' pseudo-classes, will react to clicks

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

### 14.7.2 The title attribute

User agents are expected to expose the advisory information of elements upon user request, and to make the user aware of the presence of such information.

On interactive graphical systems where the user can use a pointing device, this could take the form of a tooltip. When the user is unable to use a pointing device, then the user agent is expected to make the content available in some other fashion, e.g. by making the element focusable and always displaying the advisory information of the currently focused element, or by showing the advisory information of the elements under the user's finger on a touch device as the user pans around the screen.

U+000A LIN characters a occurring at the spec instead: <https://web.archive.org>

R TABULATION (tab)  
ab stop, with tab stops

For e with : docu //whatwg.org/html

make any focused element allow a user to tab around the

As an... to read the last tooltip for which a cue was played.

↳ tooltip, with an associated key

### 14.7.3 Editing hosts

The current text editing caret (i.e. the active range, if it is empty and in an editing host), if any, is expected to act like an inline replaced element with the vertical dimensions of the caret and with zero width for the purposes of the CSS rendering model.

**Note:** This means that even an empty block can have the caret inside it, and that when the caret is in such an element, it prevents margins from collapsing through the element.

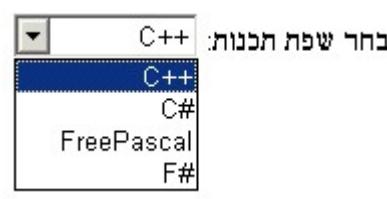
### 14.7.4 Text rendered in native user interfaces

User agents are expected to honor the Unicode semantics of text that is exposed in user interfaces, for example supporting the bidirectional algorithm in text shown in dialogs, title bars, pop-up menus, and tooltips. Text from elements (either attribute values or the contents of elements) is expected to be rendered in a manner that honors the directionality of the element from which the text was obtained.

Consider the following markup, which has Hebrew text asking for a programming language, the languages being text for which a left-to-right direction is important given the punctuation in some of their names:

```
<p dir="rtl" lang="he">
 <label>
 בחר שפת הכנות:
 <select>
 <option dir="ltr">C++</option>
 <option dir="ltr">C#</option>
 <option dir="ltr">FreePascal</option>
 <option dir="ltr">F#</option>
 </select>
 </label>
</p>
```

If the select element was rendered as a drop down box, a correct rendering would ensure that the punctuation was the same both in the drop down, and in the box showing the current selection.



Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

When necessary, authors can enforce a particular direction for a given paragraph by starting it with the Unicode U+200E LEFT-TO-RIGHT MARK or U+200F RIGHT-TO-LEFT MARK characters.

Thus, the following script:

```
alert('＼u05DC＼u05DE＼u05D3 HTML \u05D4＼u05D9＼u05D5＼u05DD!')
```

...would always result in a message reading "ה ים HTML ל" (not "ם ה HTML ים!"), regardless of the language of the user agent interface or the direction of the page or any of its elements.



For a

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

When

car..."

right-to-left, and so the output will be the following unintended mess:

"..and Wilma will get the car , פה אַ, Ok, Fred ! פה אַ"

.y \*/

ilma will get the  
one direction of the paragraph is

To force an alert that starts with user-provided text (or other text of unknown directionality) to render left-to-right, the string can be prefixed with a U+200E LEFT-TO-RIGHT MARK character:

```
var s;
if (s = prompt('What is your name?')) {
 alert('\u200E' + s + '! Ok, Fred, ' + s + ', and Wilma will get the car.');
}
```

## 14.8 Print media

User agents are expected to allow the user to request the opportunity to **obtain a physical form** (or a representation of a physical form) of a Document. For example, selecting the option to print a page or convert it to PDF format. [PDF]

When the user actually obtains a physical form (or a representation of a physical form) of a Document, the user agent is expected to create a new rendering of the Document for the print media.

## 14.9 Unstyled XML documents

HTML user agents may, in certain circumstances, find themselves rendering non-HTML documents that use vocabularies for which they lack any built-in knowledge. This section provides for a way for user agents to handle such documents in a somewhat useful manner.

While a Document is an unstyled document, the user agent is expected to render an unstyled document view.

A Document is an **unstyled document** while it matches the following conditions:

- The Document has no author style sheets (whether referenced by HTTP headers, processing instructions, elements like link, inline elements like style, or any other mechanism).
- None of the elements in the Document have any presentational hints.
- None of the elements in the Document have any CSS styling attributes.
- None of the elements in the Document are in any of the following namespaces: HTML namespace, SVG namespace, MathML namespace
- The Document has no focusable elements (e.g. from XLink).
- The Document has no hyperlinks (e.g. from XLink).
- There exists no script whose document is this Document.

3,445 captures

5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB  
◀ 03 ▶  
2012 2013 2014

About this capture

simply a message saying that the page is not a styled document.

*Note: If a Document stops being an unstyled document, then the conditions above stop applying, and thus a user agent following these requirements will switch to using the regular CSS rendering.*



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

**3,445 captures**

5 Jun 2004 - 22 Aug 2020

Features listed in this section will trigger warnings in conformance checkers.

Authors should not specify a `border` attribute on an `img` element. If the attribute is present, its value must be the string "0". CSS should be used instead.

Authors should not specify a `language` attribute on a `script` element. If the attribute is present, its value must be an ASCII case-insensitive match for the string "JavaScript" and either the `type` attribute must be omitted or its value must be an ASCII case-insensitive match for the string "text/javascript". The attribute should be entirely omitted instead (with the value "JavaScript", it has no effect), or replaced with use of the `type` attribute.

Authors should not specify a `name` attribute on an `a` element if its value is the empty string and must not be equal to the ID, if any, nor be equal to the value of any other attribute on the element. This attribute was intended as a way to...

**Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

Authors should not specify attributes on `input` elements to help legacy...

...be the empty string and must not be equal to the ID, if any, nor be equal to the value of any other attribute on the element. This attribute was intended as a way to...

...be `maxlength` and `size`. These attributes regardless is the `input` field with a useful width.

**Note:** In the HTML syntax, specifying a DOCTYPE that is an obsolete permitted DOCTYPE will also trigger a warning.

### 15.1.1 Warnings for obsolete but conforming features

To ease the transition from HTML4 Transitional documents to the language defined in *this* specification, and to discourage certain features that are only allowed in very few circumstances, conformance checkers are required to warn the user when the following features are used in a document. These are generally old obsolete features that have no effect, and are allowed only to distinguish between likely mistakes (regular conformance errors) and mere vestigial markup or unusual and discouraged practices (these warnings).

The following features must be categorized as described above:

- The presence of an obsolete permitted DOCTYPE in an HTML document.
- The presence of a `border` attribute on an `img` element if its value is the string "0".
- The presence of a `language` attribute on a `script` element if its value is an ASCII case-insensitive match for the string "JavaScript" and if there is no `type` attribute or there is and its value is an ASCII case-insensitive match for the string "text/javascript".
- The presence of a `name` attribute on an `a` element, if its value is not the empty string.
- The presence of a `maxlength` attribute on an `input` element whose `type` attribute is in the Number state.
- The presence of a `size` attribute on an `input` element whose `type` attribute is in the Number state.

Conformance checkers must distinguish between pages that have no conformance errors and have none of these obsolete features, and pages that have no conformance errors but do have some of these obsolete features.

|| For example, a validator could report some pages as "Valid HTML" and others as "Valid HTML with warnings".

## 15.2 Non-conforming features

Elements in the following list are entirely obsolete, and must not be used by authors:

### `applet`

Use `embed` or `object` instead.

### `acronym`

Use `abbr` instead.

### `bgsound`

Use `audio` instead.

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

### frameset

### noframes

Either use `iframe` and CSS instead, or use server-side includes to generate complete pages with the various invariant parts merged in.

### isindex

Use an explicit `form` and text field combination instead.



### listing

Use

**nextid** Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

### noembed

Use

Use

### plaintext

Use the "text/plain" MIME type instead.

### rb

Providing the ruby base directly inside the `ruby` element is sufficient; the `rb` element is unnecessary. Omit it altogether.

### strike

Use `del` instead if the element is marking an edit, otherwise use `s` instead.

### xmp

Use `pre` and `code` instead, and escape "<" and "&" characters as "&lt;" and "&amp;" respectively.

### basefont

### big

### blink

### center

### font

### marquee

### multicol

### nobr

### spacer

### tt

Use appropriate elements or CSS instead.

Where the `tt` element would have been used for marking up keyboard input, consider the `kbd` element; for variables, consider the `var` element; for computer code, consider the `code` element; and for computer output, consider the `samp` element.

Similarly, if the `big` element is being used to denote a heading, consider using the `h1` element; if it is being used for marking up important passages, consider the `strong` element; and if it is being used for highlighting text for reference purposes, consider the `mark` element.

See also the text-level semantics usage summary for more suggestions with examples.

The following attributes are obsolete (though the elements are still part of the language), and must not be used by authors:

### charset on a elements

### charset on link elements

Use an HTTP Content-Type header on the linked resource instead.

### coords on a elements

### shape on a elements

Use `area` instead of `a` for image maps.

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**name on a elements** (except as noted in the previous section)**name on embed elements****name on img elements****name on option elements**Use the `id` attribute instead.**rev on a elements****rev on link elements**Use '`'`

**urn on a el** Too slow? Try reading the multipage copy of  
**urn on lin** the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html/>

**Spec**

**accept on** Use `//whatwg.org/html`

**nohref on area elements**Omitting the `href` attribute is sufficient; the `nohref` attribute is unnecessary. Omit it altogether.**profile on head elements**When used for declaring which `meta` terms are used in the document, unnecessary; omit it altogether, and register the names.When used for triggering specific user agent behaviors: use a `link` element instead.**version on html elements**

Unnecessary. Omit it altogether.

**usemap on input elements**Use `img` instead of `input` for image maps.**longdesc on iframe elements****longdesc on img elements**Use a regular `a` element to link to the description, or (in the case of images) use an image map to provide a link from the image to the image's description.**lowsrc on img elements**Use a progressive JPEG image (given in the `src` attribute), instead of using two separate images.**target on link elements**

Unnecessary. Omit it altogether.

**scheme on meta elements**

Use only one scheme per field, or make the scheme declaration part of the value.

**archive on object elements****classid on object elements****code on object elements****codebase on object elements****codetype on object elements**Use the `data` and `type` attributes to invoke plugins. To set parameters with these names in particular, the `param` element can be used.**declare on object elements**Repeat the `object` element completely each time the resource is to be reused.**standby on object elements**

Optimize the linked resource so that it loads quickly or, at least, incrementally.

**type on param elements**

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

## 3,445 captures

5 Jun 2004 - 22 Aug 2020

Use the `type` attribute instead.**event on script elements****for on script elements**

Use DOM Events mechanisms to register event listeners. [DOM]

**datapagesize on table elements**

Unnecessary. Omit it altogether.

**summary on**

Use **Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html**

**axis on td**Use the `scope` attribute on the relevant `th`.ter that. The `title` attribute  
it's a heading, use `th` (which**scope on td elements**Use `th` elements for heading cells.**datasrc on a, applet, button, div, frame, iframe, img, input, label, legend, marquee, object, option, select, span, table, and textarea elements****datafld on a, applet, button, div, fieldset, frame, iframe, img, input, label, legend, marquee, object, param, select, span, and textarea elements****dataformatas on button, div, input, label, legend, marquee, object, option, select, span, and table elements**

Use script and a mechanism such as XMLHttpRequest to populate the page dynamically. [XHR]

**alink on body elements****bgcolor on body elements****link on body elements****marginbottom on body elements****marginheight on body elements****marginleft on body elements****marginright on body elements****margintop on body elements****marginwidth on body elements****text on body elements****vlink on body elements****clear on br elements****align on caption elements****align on col elements****char on col elements****charoff on col elements****valign on col elements****width on col elements****align on div elements****compact on dl elements****align on embed elements****hspace on embed elements****vspace on embed elements****align on hr elements****color on hr elements****noshade on hr elements**

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*allowtransparency* on **iframe** elements*frameborder* on **iframe** elements*hspace* on **iframe** elements*marginheight* on **iframe** elements*marginwidth* on **iframe** elements*scrolling* on **iframe** elements*vspace* on **iframe** elements*align* on **ir***hspace* on Too slow? Try reading the multipage copy of*vspace* on*align* on **i***border* on /web/20130103030411/http:*hspace* on*vspace* on*align* on **legenda** elements*type* on **li** elements*compact* on **menu** elements*align* on **object** elements*border* on **object** elements*hspace* on **object** elements*vspace* on **object** elements*compact* on **ol** elements*align* on **p** elements*width* on **pre** elements*align* on **table** elements*bgcolor* on **table** elements*border* on **table** elements*bordercolor* on **table** elements*cellpadding* on **table** elements*cellspacing* on **table** elements*frame* on **table** elements*rules* on **table** elements*width* on **table** elements*align* on **tbody**, **thead**, and **tfoot** elements*char* on **tbody**, **thead**, and **tfoot** elements*charoff* on **tbody**, **thead**, and **tfoot** elements*valign* on **tbody**, **thead**, and **tfoot** elements*align* on **td** and **th** elements*bgcolor* on **td** and **th** elements*char* on **td** and **th** elements*charoff* on **td** and **th** elements*height* on **td** and **th** elements*nowrap* on **td** and **th** elements*valign* on **td** and **th** elements*width* on **td** and **th** elements*align* on **tr** elements*bgcolor* on **tr** elements*char* on **tr** elements*charoff* on **tr** elements*valign* on **tr** elements

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

## 15.3 Requirements for implementations

### 15.3.1 The applet element

The `applet` element is a Java-specific variant of the `embed` element. The `applet` element is now obsolete so that all extension frameworks (Java, .NET, Flash, etc) are handled in a consistent manner.

When the ele

- The [Too slow? Try reading the multipage copy of the spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)
- The [spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)
- No [spec instead: https://web.archive.org/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)
- A Ja

: flag set.

Otherwise, the user agent should instantiate a Java Language runtime plugin, and should pass the names and values of all the attributes on the element, in the order they were added to the element, with the attributes added by the parser being ordered in source order, and then a parameter named "PARAM" whose value is null, and then all the names and values of parameters given by `param` elements that are children of the `applet` element, in tree order, to the plugin used. If the plugin supports a scriptable interface, the `HTMLAppletElement` object representing the element should expose that interface. The `applet` element represents the plugin.

**Note:** *The `applet` element is unaffected by the CSS 'display' property. The Java Language runtime is instantiated even if the element is hidden with a 'display:none' CSS style.*

The `applet` element must implement the `HTMLAppletElement` interface.

```
interface HTMLAppletElement : HTMLElement {
 attribute DOMString align;
 attribute DOMString alt;
 attribute DOMString archive;
 attribute DOMString code;
 attribute DOMString codeBase;
 attribute DOMString height;
 attribute unsigned long hspace;
 attribute DOMString name;
 attribute DOMString _object; // the underscore is not part of the identifier
 attribute unsigned long vspace;
 attribute DOMString width;
}
```

The `align`, `alt`, `archive`, `code`, `height`, `hspace`, `name`, `object`, `vspace`, and `width` IDL attributes must reflect the respective content attributes of the same name. For the purposes of reflection, the `applet` element's `object` content attribute is defined as containing a URL.

The `codeBase` IDL attribute must reflect the `codebase` content attribute, which for the purposes of reflection is defined as containing a URL.

### 15.3.2 The marquee element

The `marquee` element is a presentational element that animates content. CSS transitions and animations are a more appropriate mechanism. [CSSANIMATIONS] [CSSTRANSITIONS]

The task source for tasks mentioned in this section is the DOM manipulation task source.

The `marquee` element must implement the `HTMLMarqueeElement` interface.

```
interface HTMLMarqueeElement : HTMLElement {
 attribute DOMString behavior;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

```
attribute unsigned long scrollAmount;
attribute unsigned long scrollDelay;
attribute boolean trueSpeed;
attribute unsigned long vspace;
attribute DOMString width;

attribute EventHandler onbounce;
attribute EventHandler onfinish;
attribute EventHandler onstart;
```



vo Too slow? Try reading the multipage copy of  
 vo the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

A `marquee` When the `s`: [/whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

When the `stop()` method is called, the `marquee` element must be turned off.

When a `marquee` element is created, the user agent must queue a task to fire a simple event named `start` at the element.

The `behavior` content attribute on `marquee` elements is an enumerated attribute with the following keywords (all non-conforming):

Keyword	State
scroll	scroll
slide	slide
alternate	alternate

The *missing value default* is the scroll state.

The `direction` content attribute on `marquee` elements is an enumerated attribute with the following keywords (all non-conforming):

Keyword	State
left	left
right	right
up	up
down	down

The *missing value default* is the left state.

The `trueSpeed` content attribute on `marquee` elements is a boolean attribute.

A `marquee` element has a **marquee scroll interval**, which is obtained as follows:

1. If the element has a `scrollDelay` attribute, and parsing its value using the rules for parsing non-negative integers does not return an error, then let `delay` be the parsed value. Otherwise, let `delay` be 85.
2. If the element does not have a `trueSpeed` attribute, and the `delay` value is less than 60, then let `delay` be 60 instead.
3. The marquee scroll interval is `delay`, interpreted in milliseconds.

A `marquee` element has a **marquee scroll distance**, which, if the element has a `scrollAmount` attribute, and parsing its value using the rules for parsing non-negative integers does not return an error, is the parsed value interpreted in CSS pixels, and otherwise is 6 CSS pixels.

A `marquee` element has a **marquee loop count**, which, if the element has a `loop` attribute, and parsing its value using the rules for parsing integers does not return an error or a number less than 1, is the parsed value, and otherwise is -1.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The rendering layer will occasionally **increment the marquee current loop index**, which must cause the following steps to be run:

1. If the marquee loop count is  $-1$ , then abort these steps.
2. Increment the marquee current loop index by one.
3. If the marquee current loop index is now equal to or greater than the element's marquee loop count, turn off the `marquee` element and queue a task to fire a simple event named `finish` at the `marquee` element.

Other

mar Too slow? Try reading the multipage copy of

Oth the spec instead: <https://web.archive.org>

The following attributes, b<sup>o</sup> ported, as content and IDL  
attributes, b<sup>o</sup>

[/web/20130103030411/http://whatwg.org/html](https://web/20130103030411/http://whatwg.org/html)

#### Event handler

Event handler	
<code>onbounce</code>	bounce
<code>onfinish</code>	finish
<code>onstart</code>	start

The `behavior`, `direction`, `height`, `hspace`, `vspace`, and `width` IDL attributes must reflect the respective content attributes of the same name.

The `bgColor` IDL attribute must reflect the `bgcolor` content attribute.

The `scrollAmount` IDL attribute must reflect the `scrollamount` content attribute. The default value is 6.

The `scrollDelay` IDL attribute must reflect the `scrolldelay` content attribute. The default value is 85.

The `trueSpeed` IDL attribute must reflect the `truespeed` content attribute.

### 15.3.3 Frames

The `frameset` element acts as the `body` element in documents that use frames.

The `frameset` element must implement the `HTMLFrameSetElement` interface.

```
interface HTMLFrameSetElement : HTMLElement {
 attribute DOMString cols;
 attribute DOMString rows;
 attribute EventHandler onafterprint;
 attribute EventHandler onbeforeprint;
 attribute EventHandler onbeforeunload;
 attribute EventHandler onblur;
 attribute EventHandler onerror;
 attribute EventHandler onfocus;
 attribute EventHandler onfullscreenchange;
 attribute EventHandler onfullscreenerror;
 attribute EventHandler onhashchange;
 attribute EventHandler onload;
 attribute EventHandler onmessage;
 attribute EventHandler onoffline;
 attribute EventHandler ononline;
 attribute EventHandler onpagehide;
 attribute EventHandler onpageshow;
 attribute EventHandler onpopstate;
 attribute EventHandler onresize;
 attribute EventHandler onscroll;
 attribute EventHandler onstorage;
```

Go

NOV JAN FEB



2012 2013 2014

▼ About this capture

The `frameset` element must support the following event handler content attributes exposing the event handlers of the `Window` object:

- onafterprint
  - onbeforeprint
  - onbeforeunload
  - onblur
  - onerror
  - onfocus
  - onfullscreenchange
  - onfullscreenerror
  - onhead
  - onl
  - onm
  - onc
  - onc
  - onp
  - onp
  - onp
  - onr
  - ons
  - onstorage
  - onunload

Too slow  
the spec  
[/web/201](#)  
[/whatwg](#)

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The DOM interface also exposes event handler IDL attributes that mirror those on the `Window` element.

The `onblur`, `onerror`, `onfocus`, `onload`, and `onscroll` event handlers of the `Window` object, exposed on the `frameset` element, shadow the generic event handlers with the same names normally supported by HTML elements. With the exception of `onerror`, this is actually implemented by just using the properties of the `HTMLElement` interface. The `onerror` attribute has a different type.

The **frame** element defines a nested browsing context similar to the **iframe** element, but rendered within a **frameset** element.

A frame element is said to be an **active frame element** when it is in a Document and its parent element, if any, is a frameset element.

When a `frame` element is created as an active `frame` element, or becomes an active `frame` element after not having been one, the user agent must create a nested browsing context, and then process the `frame` attributes for the first time.

When a `frame` element stops being an active `frame` element, the user agent must discard the nested browsing context.

Whenever a `frame` element with a nested browsing context has its `src` attribute set, changed, or removed, the user agent must process the `frame` attributes.

When the user agent is to process the **frame** attributes, it must run the first appropriate steps from the following list:

↪ If the element has no `src` attribute specified, and the user agent is processing the frame's attributes for the first time

Queue a task to fire a simple event named `load` at the `frame` element

↳ Otherwise

1. If the value of the `src` attribute is the empty string, let `url` be the string "about:blank".

Otherwise, resolve the value of the `src` attribute, relative to the `frame` element.

If that is not successful, then let *url* be the string "about:blank". Otherwise, let *url* be the resulting absolute URI.

2. Navigate the element's child browsing context to *url*

Any navigation required of the user agent in the process the `frame` attributes algorithm must be completed as an explicit self-navigation override and with the `frame` element's document's browsing context as the source browsing context.

Furthermore, if the active document of the element's child browsing context before such a navigation was not completely loaded at the time of the new navigation, then the navigation must be completed with replacement enabled.

Similarly, if the child browsing context's session history contained only one Document when the process the frame attributes algorithm was invoked, and that was the about:blank Document created when the child browsing context was created, then any navigation required of the user agent in that algorithm must be completed with replacement enabled.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

When content loads in a `frame`, after any `load` events are fired within the content itself, the user agent must queue a task to fire a simple event named `load` at the `frame` element. When content fails to load (e.g. due to a network error), then the user agent must queue a task to fire a simple event named `error` at the element instead.

The task source for the tasks above is the DOM manipulation task source.

When there is an active parser in the `frame`, and when anything in the `frame` is delaying the `load` event of the `frame`'s browsing context's active document, the `frame` must delay the `load` event of its document.

The `frame`<sup>1</sup>

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

```
intended interface Document {
 readonly attribute Document? contentDocument;
 readonly attribute WindowProxy? contentWindow;

 [TreatNullAs=EmptyString] attribute DOMString marginHeight;
 [TreatNullAs=EmptyString] attribute DOMString marginWidth;
};
```

The `name`, `scrolling`, and `src` IDL attributes of the `frame` element must reflect the respective content attributes of the same name.

The `frameBorder` IDL attribute of the `frame` element must reflect the element's `frameborder` content attribute.

The `longDesc` IDL attribute of the `frame` element must reflect the element's `longdesc` content attribute, which for the purposes of reflection is defined as containing a URL.

The `noResize` IDL attribute of the `frame` element must reflect the element's `noresize` content attribute.

The `contentDocument` IDL attribute of the `frame` element must return the `Document` object of the active document of the `frame` element's nested browsing context.

The `contentWindow` IDL attribute must return the `WindowProxy` object of the `frame` element's nested browsing context.

The `marginHeight` IDL attribute of the `frame` element must reflect the element's `marginheight` content attribute.

The `marginWidth` IDL attribute of the `frame` element must reflect the element's `marginwidth` content attribute.

#### 15.3.4 Other elements, attributes and APIs

User agents must treat `acronym` elements in a manner equivalent to `abbr` elements in terms of semantics and for purposes of rendering.

```
partial interface HTMLAnchorElement {
 attribute DOMString coords;
 attribute DOMString charset;
 attribute DOMString name;
 attribute DOMString rev;
 attribute DOMString shape;
};
```

The `coords`, `charset`, `name`, `rev`, and `shape` IDL attributes of the `a` element must reflect the respective content attributes of the same name.

```
partial interface HTMLAreaElement {
```

Go

NOV JAN FEB



3,445 captures

5 Jun 2004 - 22 Aug 2020

2012 2013 2014

The `basefont` element must implement the `HTMLBaseFontElement` interface

```
interface HTMLBaseFontElement : HTMLElement {
 attribute DOMString color;
 attribute DOMString face;
 attribute long size;
};
```

The color, `color`, `background-color`, `border-color`, etc., attributes of the same name.

Too slow? Try reading the multipage copy of

part the spec instead: <https://web.archive.org>

[T]  
[T] /web/20130103030411/http:  
[T]  
[T] //whatwg.org/html

The `text` IDL attribute of the `body` element must reflect the element's `textContent` attribute.

The `link` IDL attribute of the `body` element must reflect the element's `link` content attribute.

The `aLink` IDL attribute of the `body` element must reflect the element's `alink` content attribute.

The `vlink` IDL attribute of the `body` element must reflect the element's `vlink` content attribute.

The `bacolor` IDL attribute of the `body` element must reflect the element's `bacolor` content attribute.

The `background` IDL attribute of the `body` element must reflect the element's `background` content attribute. (The `background` content is *not* defined to contain a URL, despite rules regarding its handling in the rendering section above.)

```
partial interface HTMLBRElement {
 attribute DOMString clear;
};
```

The `clear` IDL attribute of the `br` element must reflect the content attribute of the same name.

```
partial interface HTMLTableCaptionElement : Node {
 attribute DOMString align;
};
```

The `align` IDL attribute of the `caption` element must reflect the content attribute of the same name.

```
partial interface HTMLTableColElement {
 attribute DOMString align;
 attribute DOMString ch;
 attribute DOMString chOff;
 attribute DOMString vAlign;
 attribute DOMString width;
};
```

The `align` and `width` IDL attributes of the `col` element must reflect the respective content attributes of the same name.

The `ch` IDL attribute of the `col` element must reflect the element's `char` content attribute.

The `choff` IDL attribute of the `col` element must reflect the element's `charoff` content attribute.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `dir` element must implement the `HTMLDirectoryElement` interface.

```
interface HTMLDirectoryElement : HTMLElement {
 attribute boolean compact;
};
```

The `compact` IDL attribute of the `dir` element must reflect the content attribute of the same name.

part:  
};  
Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)  
The `align` attribute must reflect the content attribute of the same name.  
part  
//whatwg.org/html  
};

The `compact` IDL attribute of the `dl` element must reflect the content attribute of the same name.

```
partial interface HTMLEmbedElement {
 attribute DOMString align;
 attribute DOMString name;
};
```

The `name` and `align` IDL attributes of the `embed` element must reflect the respective content attributes of the same name.The `font` element must implement the `HTMLFontElement` interface.

```
interface HTMLFontElement : HTMLElement {
 [TreatNullAs=EmptyString] attribute DOMString color;
 attribute DOMString face;
 attribute DOMString size;
};
```

The `color`, `face`, and `size` IDL attributes of the `font` element must reflect the respective content attributes of the same name.

```
partial interface HTMLHeadingElement {
 attribute DOMString align;
};
```

The `align` IDL attribute of the `h1-h6` elements must reflect the content attribute of the same name.

**Note:** The `profile` IDL attribute on `head` elements (with the `HTMLHeadElement` interface) is intentionally omitted. Unless so required by another applicable specification, implementations would therefore not support this attribute. (It is mentioned here as it was defined in a previous version of the DOM specifications.)

```
partial interface HTMLHRElement {
 attribute DOMString align;
 attribute DOMString color;
 attribute boolean noShade;
 attribute DOMString size;
 attribute DOMString width;
};
```

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

```
partial interface HTMLHtmlElement {
 attribute DOMString version;
};
```

The **version** IDL attribute of the `html` element must reflect the `content` attribute of the same name.

```
partial interface HTMLIFrameElement {
 attribute DOMString align;
```



**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

The `align` attribute must reflect the `content` attribute of the same name.

The **frameBorder** IDL attribute of the `iframe` element must reflect the element's `frameborder` `content` attribute.

The **longDesc** IDL attribute of the `iframe` element must reflect the element's `longdesc` `content` attribute, which for the purposes of reflection is defined as containing a URL.

The **marginHeight** IDL attribute of the `iframe` element must reflect the element's `marginheight` `content` attribute.

The **marginWidth** IDL attribute of the `iframe` element must reflect the element's `marginwidth` `content` attribute.

```
partial interface HTMLImageElement {
 attribute DOMString name;
 attribute DOMString align;
 attribute unsigned long hspace;
 attribute unsigned long vspace;
 attribute DOMString longDesc;

 [TreatNullAs=EmptyString] attribute DOMString border;
};
```

The `name`, `align`, `border`, `hspace`, and `vspace` IDL attributes of the `img` element must reflect the respective `content` attributes of the same name.

The **longDesc** IDL attribute of the `img` element must reflect the element's `longdesc` `content` attribute, which for the purposes of reflection is defined as containing a URL.

```
partial interface HTMLInputElement {
 attribute DOMString align;
 attribute DOMString useMap;
};
```

The `align` IDL attribute of the `input` element must reflect the `content` attribute of the same name.

The `useMap` IDL attribute of the `input` element must reflect the element's `usemap` `content` attribute.

```
partial interface HTMLLegendElement {
 attribute DOMString align;
};
```

The `align` IDL attribute of the `legend` element must reflect the `content` attribute of the same name.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `type` IDL attribute of the `li` element must reflect the content attribute of the same name.

```
partial interface HTMLLinkElement {
 attribute DOMString charset;
 attribute DOMString rev;
 attribute DOMString target;
};
```

The `charse'`

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

User agents rendering.

tributes of the same name.

and for purposes of

The `compact` IDL attribute of the `menu` element must reflect the content attribute of the same name.

```
partial interface HTMLMetaElement {
 attribute DOMString scheme;
};
```

User agents may treat the `scheme` content attribute on the `meta` element as an extension of the element's `name` content attribute when processing a `meta` element with a `name` attribute whose value is one that the user agent recognizes as supporting the `scheme` attribute.

User agents are encouraged to ignore the `scheme` attribute and instead process the value given to the metadata name as if it had been specified for each expected value of the `scheme` attribute.

For example, if the user agent acts on `meta` elements with `name` attributes having the value "eGMS.subject.keyword", and knows that the `scheme` attribute is used with this metadata name, then it could take the `scheme` attribute into account, acting as if it was an extension of the `name` attribute. Thus the following two `meta` elements could be treated as two elements giving values for two different metadata names, one consisting of a combination of "eGMS.subject.keyword" and "LGCL", and the other consisting of a combination of "eGMS.subject.keyword" and "ONLY":

```
<!-- this markup is invalid -->
<meta name="eGMS.subject.keyword" scheme="LGCL" content="Abandoned vehicles">
<meta name="eGMS.subject.keyword" scheme="ONLY" content="Mah car: kthxbye">
```

The suggested processing of this markup, however, would be equivalent to the following:

```
<meta name="eGMS.subject.keyword" content="Abandoned vehicles">
<meta name="eGMS.subject.keyword" content="Mah car: kthxbye">
```

The `scheme` IDL attribute of the `meta` element must reflect the content attribute of the same name.

```
partial interface HTMLObjectElement {
 attribute DOMString align;
 attribute DOMString archive;
 attribute DOMString code;
 attribute boolean declare;
 attribute unsigned long hspace;
 attribute DOMString standby;
 attribute unsigned long vspace;
 attribute DOMString codeBase;
 attribute DOMString codeType;

 [TreatNullAs=EmptyString] attribute DOMString border;
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `codeBase` IDL attribute of the `object` element must reflect the element's `codebase` content attribute, which for the purposes of reflection is defined as containing a URL.

The `codeType` IDL attribute of the `object` element must reflect the element's `codetype` content attribute.

```
partial interface HTMLListElement {
 attribute boolean compact;
};
```



The `compac` Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

The `align` IDL

```
partial interface HTMLParamElement {
 attribute DOMString type;
 attribute DOMString valueType;
};
```

The `type` IDL attribute of the `param` element must reflect the content attribute of the same name.

The `valueType` IDL attribute of the `param` element must reflect the element's `valuetype` content attribute.

User agents must treat `plaintext` elements in a manner equivalent to `pre` elements in terms of semantics and for purposes of rendering. (The parser has special behavior for this element, though.)

```
partial interface HTMLPreElement {
 attribute long width;
};
```

The `width` IDL attribute of the `pre` element must reflect the content attribute of the same name.

```
partial interface HTMLScriptElement {
 attribute DOMString event;
 attribute DOMString htmlFor;
};
```

The `event` and `htmlFor` IDL attributes of the `script` element must return the empty string on getting, and do nothing on setting.

```
partial interface HTMLTableElement {
 attribute DOMString align;
 attribute DOMString border;
 attribute DOMString frame;
 attribute DOMString rules;
 attribute DOMString summary;
 attribute DOMString width;

 [TreatNullAs=EmptyString] attribute DOMString bgColor;
 [TreatNullAs=EmptyString] attribute DOMString cellPadding;
 [TreatNullAs=EmptyString] attribute DOMString cellSpacing;
};
```

The `align`, `border`, `frame`, `summary`, `rules`, and `width`, IDL attributes of the `table` element must reflect the respective content

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `cellspacing` IDL attribute of the `table` element must reflect the element's `cellspacing` content attribute.

```
partial interface HTMLTableSectionElement {
 attribute DOMString align;
 attribute DOMString ch;
 attribute DOMString chOff;
 attribute DOMString vAlign;
};
```



The `align` IDL attribute of the `table` element must reflect the elements' `align` content attributes. The `ch` IDL attribute of the `table` element must reflect the elements' `ch` content attributes. The `chOff` IDL attribute of the `table` element must reflect the elements' `chOff` content attributes. The `vAlign` IDL attribute of the `table` element must reflect the elements' `vAlign` content attributes.

e same name.

ttributes.

ontent attributes.

ntent attributes.

```
partial interface HTMLTableCellElement {
 attribute DOMString align;
 attribute DOMString axis;
 attribute DOMString height;
 attribute DOMString width;

 attribute DOMString ch;
 attribute DOMString chOff;
 attribute boolean nowrap;
 attribute DOMString vAlign;

 [TreatNullAs=EmptyString] attribute DOMString bgColor;
};
```

The `align`, `axis`, `height`, and `width` IDL attributes of the `td` and `th` elements must reflect the respective content attributes of the same name.

The `ch` IDL attribute of the `td` and `th` elements must reflect the elements' `char` content attributes.

The `chOff` IDL attribute of the `td` and `th` elements must reflect the elements' `charoff` content attributes.

The `nowrap` IDL attribute of the `td` and `th` elements must reflect the elements' `nowrap` content attributes.

The `vAlign` IDL attribute of the `td` and `th` element must reflect the elements' `valign` content attributes.

The `bgColor` IDL attribute of the `td` and `th` elements must reflect the elements' `bgcolor` content attributes.

```
partial interface HTMLTableDataCellElement {
 attribute DOMString abbr;
};
```

The `abbr` IDL attribute of the `td` element must reflect the respective content attributes of the same name.

```
partial interface HTMLTableRowElement {
 attribute DOMString align;
 attribute DOMString ch;
 attribute DOMString chOff;
 attribute DOMString vAlign;

 [TreatNullAs=EmptyString] attribute DOMString bgColor;
};
```

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

The `vAlign` IDL attribute of the `tr` element must reflect the element's `valign` content attribute.

The `bgColor` IDL attribute of the `tr` element must reflect the element's `bgcolor` content attribute.

```
partial interface HTMLUListElement {
 attribute boolean compact;
 attribute DOMString type;
};
```

The `compact`

**Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**

ie same name.

User agents  
(The parser

or purposes of rendering.

The `bgsoun`

gment interface.

```
partial interface Document {
 [TreatNullAs=EmptyString] attribute DOMString fgColor;
 [TreatNullAs=EmptyString] attribute DOMString linkColor;
 [TreatNullAs=EmptyString] attribute DOMString vlinkColor;
 [TreatNullAs=EmptyString] attribute DOMString alinkColor;
 [TreatNullAs=EmptyString] attribute DOMString bgColor;

 readonly attribute HTMLCollection anchors;
 readonly attribute HTMLCollection applets;

 void clear();

 readonly attribute HTMLAllCollection all;
};
```

The attributes of the `Document` object listed in the first column of the following table must reflect the content attribute on the `body` element with the name given in the corresponding cell in the second column on the same row, if the `body` element is a `body` element (as opposed to a `frameset` element). When there is no `body` element or if it is a `frameset` element, the attributes must instead return the empty string on getting and do nothing on setting.

IDL attribute	Content attribute
<code>fgColor</code>	<code>text</code>
<code>linkColor</code>	<code>link</code>
<code>vlinkColor</code>	<code>vlink</code>
<code>alinkColor</code>	<code>alink</code>
<code>bgColor</code>	<code>bgcolor</code>

The `anchors` attribute must return an `HTMLCollection` rooted at the `Document` node, whose filter matches only `a` elements with `name` attributes.

The `applets` attribute must return an `HTMLCollection` rooted at the `Document` node, whose filter matches only `applet` elements.

The `clear()` method must do nothing.

The `all` attribute must return an `HTMLAllCollection` rooted at the `Document` node, whose filter matches all elements.

The object returned for `all` has several unusual behaviors:

- The user agent must act as if the `ToBoolean()` operator in JavaScript converts the object returned for `all` to the `false` value.
- The user agent must act as if, for the purposes of the `==` and `!=` operators in JavaScript, the object returned for `all` compares

GoNOV JAN FEB

**3,445 captures**  
5 Jun 2004 - 22 Aug 2020

◀ **03** ▶  
**2012** **2013** **2014**

▼ About this capture

*Note: These requirements are a willful violation of the JavaScript specification current at the time of writing (ECMAScript edition 5). The JavaScript specification requires that the ToBoolean() operator convert all objects to the true value, and does not have provisions for objects acting as if they were undefined for the purposes of certain operators. This violation is motivated by a desire for compatibility with two classes of legacy content: one that uses the presence of document.all as a way to detect legacy user agents, and one that only supports those legacy user agents and uses the document.all object without testing for its presence first. [ECMA262]*



Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http:  
//whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

Go

NOV JAN FEB

03

2012 2013 2014



## 3,445 captures

5 Jun 2004 - 22 Aug 2020

This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

**Type name:**

text

**Subtype name:**

html

**Required parameters:**

No required parameters

**Optional parameters:**

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

encoding, overriding any  
of the character encoding  
case-insensitive match for

**Encoding:**

8bit (see the section on character encoding declarations)

**Security considerations:**

Entire novels have been written about the security considerations that apply to HTML documents. Many are listed in this document, to which the reader is referred for more details. Some general concerns bear mentioning here, however:

HTML is a scripted language, and has a large number of APIs (some of which are described in this document). Script can expose the user to potential risks of information leakage, credential leakage, cross-site scripting attacks, cross-site request forgeries, and a host of other problems. While the designs in this specification are intended to be safe if implemented correctly, a full implementation is a massive undertaking and, as with any software, user agents are likely to have security bugs.

Even without scripting, there are specific features in HTML which, for historical reasons, are required for broad compatibility with legacy content but that expose the user to unfortunate security problems. In particular, the `img` element can be used in conjunction with some other features as a way to effect a port scan from the user's location on the Internet. This can expose local network topologies that the attacker would otherwise not be able to determine.

HTML relies on a compartmentalization scheme sometimes known as the *same-origin policy*. An origin in most cases consists of all the pages served from the same host, on the same port, using the same protocol.

It is critical, therefore, to ensure that any untrusted content that forms part of a site be hosted on a different origin than any sensitive content on that site. Untrusted content can easily spoof any other page on the same origin, read data from that origin, cause scripts in that origin to execute, submit forms to and from that origin even if they are protected from cross-site request forgery attacks by unique tokens, and make use of any third-party resources exposed to or rights granted to that origin.

**Interoperability considerations:**

Rules for processing both conforming and non-conforming content are defined in this specification.

**Published specification:**

This document is the relevant specification. Labeling a resource with the `text/html` type asserts that the resource is an HTML document using the HTML syntax.

**Applications that use this media type:**

Web browsers, tools for processing Web content, HTML authoring tools, search engines, validators.

**Additional information:****Magic number(s):**

No sequence of bytes can uniquely identify an HTML document. More information on detecting HTML documents is available in the Media Type Sniffing specification. [MIMESNIFF]

**File extension(s):**

"`html`" and "`htm`" are commonly, but certainly not exclusively, used as the extension for HTML documents.

**Macintosh file type code(s):**

TEXT

**Person & email address to contact for further information:**

Ian Hickson <[ian@hixie.ch](mailto:ian@hixie.ch)>

**Intended usage:**

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Author:**

Ian Hickson &lt;ian@hixie.ch&gt;

**Change controller:**

W3C

Fragment identifiers used with `text/html` resources either refer to the indicated part of the document or provide state information for in-page scripts.

**16.2 `mul`** Too slow? Try reading the multipage copy of

This registration is available for download from the spec instead: <https://web.archive.org/web/20130103030411/http://www.whatwg.org/specs/>

ation with IANA.

Type name: `mul`

Subtype name: `//whatwg.org/html`

`x-mixed-replace`**Required parameters:**

- `boundary` (defined in RFC2046) [RFC2046]

**Optional parameters:**

No optional parameters.

**Encoding considerations:**`binary`**Security considerations:**

Subresources of a `multipart/x-mixed-replace` resource can be of any type, including types with non-trivial security implications such as `text/html`.

**Interoperability considerations:**

None.

**Published specification:**

This specification describes processing rules for Web browsers. Conformance requirements for generating resources with this type are the same as for `multipart/mixed`. [RFC2046]

**Applications that use this media type:**

This type is intended to be used in resources generated by Web servers, for consumption by Web browsers.

**Additional information:****Magic number(s):**

No sequence of bytes can uniquely identify a `multipart/x-mixed-replace` resource.

**File extension(s):**

No specific file extensions are recommended for this type.

**Macintosh file type code(s):**

No specific Macintosh file type codes are recommended for this type.

**Person & email address to contact for further information:**

Ian Hickson &lt;ian@hixie.ch&gt;

**Intended usage:**`Common`**Restrictions on usage:**

No restrictions apply.

**Author:**

Ian Hickson &lt;ian@hixie.ch&gt;

**Change controller:**

W3C

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

## 16.3 application/xhtml+xml

This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

**Type name:**

application

**Subtype name:**

xhtml+xml

**Required parameters:**

San **Too slow? Try reading the multipage copy of**

**the spec instead: https://web.archive.org**

San **/web/20130103030411/http:**

San **//whatwg.org/html**

**Security considerations:**

Same as for application/xml [RFC3023]

**Interoperability considerations:**

Same as for application/xml [RFC3023]

**Published specification:**

Labeling a resource with the application/xhtml+xml type asserts that the resource is an XML document that likely has a root element from the HTML namespace. Thus, the relevant specifications are the XML specification, the Namespaces in XML specification, and this specification. [XML] [XMLNS]

**Applications that use this media type:**

Same as for application/xml [RFC3023]

**Additional information:****Magic number(s):**

Same as for application/xml [RFC3023]

**File extension(s):**

"xhtml" and "xht" are sometimes used as extensions for XML resources that have a root element from the HTML namespace.

**Macintosh file type code(s):**

TEXT

**Person & email address to contact for further information:**

Ian Hickson <ian@hixie.ch>

**Intended usage:**

Common

**Restrictions on usage:**

No restrictions apply.

**Author:**

Ian Hickson <ian@hixie.ch>

**Change controller:**

W3C

Fragment identifiers used with application/xhtml+xml resources have the same semantics as with any XML MIME type. [RFC3023]

## 16.4 application/x-www-form-urlencoded

This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

**Type name:**

[Go](#)

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

## [3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**Required parameters:**

No parameters

**Optional parameters:**

No parameters

**Encoding considerations:**

7bit (US-ASCII encoding of octets that themselves can be encoding text using any ASCII-compatible character encoding)

**Security considerations:**ever, as this type is usually  
the context of this type.

In is  
use **Too slow? Try reading the multipage copy of  
the spec instead: https://web.archive.org**

Interoperab Rule  
Published s /web/20130103030411/http:

This //whatwg.org/html

**Applications**

Web browsers and servers.

**Additional information:****Magic number(s):**

There is no reliable mechanism for recognising application/x-www-form-urlencoded payloads.

**File extension(s):**

Not applicable.

**Macintosh file type code(s):**

Not applicable.

**Person & email address to contact for further information:**

Ian Hickson &lt;ian@hixie.ch&gt;

**Intended usage:**

Common

**Restrictions on usage:**

This type is only intended to be used to describe HTML form submission payloads.

**Author:**

Ian Hickson &lt;ian@hixie.ch&gt;

**Change controller:**

W3C

Fragment identifiers have no meaning with the application/x-www-form-urlencoded type.

## 16.5 text/cache-manifest

This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

**Type name:**

text

**Subtype name:**

cache-manifest

**Required parameters:**

No parameters

**Optional parameters:**

charset

The charset parameter may be provided. The parameter's value must be "utf-8". This parameter serves no purpose; it is only allowed for compatibility with legacy servers.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Implementations, however, are required to follow specific rules when populating a cache based on a cache manifest, to ensure that certain origin-based restrictions are honored. Failure to correctly implement these rules can result in information leakage, cross-site scripting attacks, and the like.

**Interoperability considerations:**

Rules for processing both conforming and non-conforming content are defined in this specification.

**Published specification:**

This document is the relevant specification.

**Application**

Web

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

SPACE character, a U+0009  
+000D CARRIAGE

**Additional information**

Mac

/web/20130103030411/http://whatwg.org/html

File extension

`"appcache"`**Macintosh file type code(s):**

No specific Macintosh file type codes are recommended for this type.

**Person & email address to contact for further information:**

Ian Hickson <ian@hixie.ch>

**Intended usage:**

Common

**Restrictions on usage:**

No restrictions apply.

**Author:**

Ian Hickson <ian@hixie.ch>

**Change controller:**

W3C

Fragment identifiers have no meaning with `text/cache-manifest` resources.

## 16.6 `text/ping`

This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

**Type name:**

`text`

**Subtype name:**

`ping`

**Required parameters:**

No parameters

**Optional parameters:**

`charset`

The `charset` parameter may be provided. The parameter's value must be "`utf-8`". This parameter serves no purpose; it is only allowed for compatibility with legacy servers.

**Encoding considerations:**

Not applicable.

**Security considerations:**

If used exclusively in the fashion described in the context of hyperlink auditing, this type introduces no new security concerns.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**Applications that use this media type:**

Web browsers.

**Additional information:****Magic number(s):**

text/ping resources always consist of the four bytes 0x50 0x49 0x4E 0x47 (ASCII 'PING').

**File extension(s):**

No specific file extension is recommended for this type.



Mac

**Too slow? Try reading the multipage copy of****Person & el  
ian** **the spec instead: <https://web.archive.org>****Intended us** /web/20130103030411/http:**Cor** //whatwg.org/html**Restrictions**

Only intended for use with HTTP POST requests generated as part of a Web browser's processing of the ping attribute.

**Author:**

Ian Hickson &lt;ian@hixie.ch&gt;

**Change controller:**

W3C

Fragment identifiers have no meaning with text/ping resources.

## 16.7 application/microdata+json

This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

**Type name:**

application

**Subtype name:**

microdata+json

**Required parameters:**

Same as for application/json [JSON]

**Optional parameters:**

Same as for application/json [JSON]

**Encoding considerations:**

8bit (always UTF-8)

**Security considerations:**

Same as for application/json [JSON]

**Interoperability considerations:**

Same as for application/json [JSON]

**Published specification:**

Labeling a resource with the application/microdata+json type asserts that the resource is a JSON text that consists of an object with a single entry called "items" consisting of an array of entries, each of which consists of an object with an entry called "id" whose value is a string, an entry called "type" whose value is another string, and an entry called "properties" whose value is an object whose entries each have a value consisting of an array of either objects or strings, the objects being of the same form as the objects in the aforementioned "items" entry. Thus, the relevant specifications are the JSON specification and this specification. [JSON]

**Applications that use this media type:**

Applications that transfer data intended for use with HTML's microdata feature, especially in the context of drag-and-drop, are

3,445 captures  
5 Jun 2004 - 22 Aug 2020

Go NOV JAN FEB  
◀ 03 ▶  
2012 2013 2014

About this capture

**File extension(s):**  
Same as for application/json [JSON]

**Macintosh file type code(s):**  
Same as for application/json [JSON]

**Person & email address to contact for further information:**  
Ian Hickson <ian@hixie.ch>

**Intended usage:**   
**Restriction:** Cor  
Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>  
**Author:** Ian  
**Change controller:** /web/20130103030411/http:  
W3C

Fragment identifiers used with application/microdata+json resources have the same semantics as when used with application/json (namely, at the time of writing, no semantics at all). [JSON]

## 16.8 Ping-From

This section describes a header field for registration in the Permanent Message Header Field Registry. [RFC3864]

**Header field name:**

Ping-From

**Applicable protocol:**

http

**Status:**

standard

**Author/Change controller:**

W3C

**Specification document(s):**

This document is the relevant specification.

**Related information:**

None.

## 16.9 Ping-To

This section describes a header field for registration in the Permanent Message Header Field Registry. [RFC3864]

**Header field name:**

Ping-To

**Applicable protocol:**

http

**Status:**

standard

**Author/Change controller:**

W3C

**Specification document(s):**

This document is the relevant specification.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

## 16.10 web+ scheme prefix

This section describes a convention for use with the IANA URI scheme registry. It does not itself register a specific scheme. [RFC4395]

**URI scheme name:**

Schemes starting with the four characters "web+" followed by one or more letters in the range a-z.

**Status:**

permanent

**URI scheme**

Sch Too slow? Try reading the multipage copy of

URI scheme the spec instead: <https://web.archive.org>

Sch

Encoding c /web/20130103030411/http:

All " //whatwg.org/html

**Applications:**

Scheme-specific.

**Interoperability considerations:**

The scheme is expected to be used in the context of Web applications.

**Security considerations:**

Any Web page is able to register a handler for all "web+" schemes. As such, these schemes must not be used for features intended to be core platform features (e.g. network transfer protocols like HTTP or FTP). Similarly, such schemes must not store confidential information in their URLs, such as usernames, passwords, personal information, or confidential project names.

**Contact:**

Ian Hickson <ian@hixie.ch>

**Author/Change controller:**

Ian Hickson <ian@hixie.ch>

**References:**

*Custom scheme and content handlers*, HTML Living Standard: <http://www.whatwg.org/specs/web-apps/current-work/#custom-handlers>

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

*This section is non-normative.*

List of elements						
Element	Description	Categories	Parents†	Children	Attributes	Interface
<b>a</b>	Hyperlink	flow; phrasing*; interactive	phrasing	transparent*	globals; href; target; download; ping; rel; hreflang; type	HTMLAnchorElement
<b>abbr</b>	Abbreviation	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>address</b>	Contact	flow	flow	flow*	globals	HTMLElement

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

<b>area</b>	composition					
<b>aside</b>	Sidebar for tangentially related content	flow; sectioning	flow	flow	globals	HTMLElement
<b>audio</b>	Audio player	flow; phrasing; embedded; interactive	phrasing	source*; transparent*	globals; src; crossorigin; preload; autoplay; mediagroup; loop; muted; controls	HTMLAudioElement
<b>b</b>	Keywords	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>base</b>	Base URL and default target browsing context for hyperlinks and forms	metadata	head	empty	globals; href; target	HTMLBaseElement
<b>bdi</b>	Text directionality isolation	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>bdo</b>	Text directionality formatting	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>blockquote</b>	A section quoted from another source	flow; sectioning root	flow	flow	globals; cite	HTMLQuoteElement
<b>body</b>	Document body	sectioning root	html	flow	globals; onafterprint; onbeforeprint; onbeforeunload; onblur; onerror; onfocus; onfullscreenchange; onfullscreenerror; onhashchange; onload; onmessage; onoffline; ononline; onpagehide; onpageshow; onpopstate; onresize; onscroll; onstorage; onunload	HTMLBodyElement
<b>br</b>	Line break, e.g. in poem or postal address	flow; phrasing	phrasing	empty	globals	HTMLBRElement
<b>button</b>	Button control	flow; phrasing; interactive; listed; labelable; submittable; form-associated	phrasing	phrasing*	globals; autofocus; disabled; form; formaction; formenctype; formmethod; formnovalidate; formtarget; name; type; value	HTMLButtonElement
<b>canvas</b>	Scriptable bitmap canvas	flow; phrasing; embedded	phrasing	transparent	globals; width; height	HTMLCanvasElement
<b>caption</b>	Table caption	none	table	flow*	globals	HTMLTableCaptionElement
<b>cite</b>	Title of a work	flow; phrasing	phrasing	phrasing	globals	HTMLElement

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

Columns in a table						
<b>menuitem</b>	Menu command	metadata; flow; phrasing	head; phrasing	empty	globals; type; label; icon; disabled; checked; radiogroup; command	HTMLMenuItemElement
<b>data</b>	Machine-readable equivalent	flow; phrasing	phrasing	phrasing	globals; value	HTMLDataElement
<b>datalist</b>	Container for options for	flow; phrasing	phrasing	phrasing; option	globals	HTMLDataListElement

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

<b>dd</b>						HTMLElement
<b>del</b>						HTMLModElement
<b>details</b>						HTMLDetailsElement
<b>dfn</b>	Defining instance	flow; phrasing	phrasing	phrasing*	globals	HTMLElement
<b>dialog</b>	Dialog box or window	flow; sectioning root	dt; th; flow	flow	globals; open	HTMLDialogElement
<b>div</b>	Generic flow container	flow	flow	flow	globals	HTMLDivElement
<b>d1</b>	Association list consisting of zero or more name-value groups	flow	flow	dt*; dd*	globals	HTMLDListElement
<b>dt</b>	Legend for corresponding <b>dd</b> element(s)	none	d1	flow*	globals	HTMLElement
<b>em</b>	Stress emphasis	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>embed</b>	Plugin	flow; phrasing; embedded; interactive	phrasing	empty	globals; src; type; width; height; any*	HTMLEmbedElement
<b>fieldset</b>	Group of form controls	flow; sectioning root; listed; form-associated	flow	legend*; flow	globals; disabled; form; name	HTMLFieldSetElement
<b>figcaption</b>	Caption for figure	none	figure	flow	globals	HTMLElement
<b>figure</b>	Figure with optional caption	flow; sectioning root	flow	figcaption*; flow	globals	HTMLElement
<b>footer</b>	Footer for a page or section	flow	flow	flow*	globals	HTMLElement
<b>form</b>	User-submittable form	flow	flow	flow*	globals; accept-charset; action; autocomplete; enctype; method; name; novalidate; target	HTMLFormElement
<b>h1, h2, h3, h4, h5, h6</b>	Section heading	flow; heading	hgroup; flow	phrasing	globals	HTMLHeadingElement
<b>head</b>	Container for document metadata	none	html	metadata content*	globals	HTMLHeadElement
<b>header</b>	Introductory or navigational aids for a page or section	flow	flow	flow*	globals	HTMLElement
<b>hgroup</b>	heading group	flow; heading	flow	One or more h1, h2, h3, h4, h5, and h6	globals	HTMLElement
<b>hr</b>	Thematic break	flow	flow	empty	globals	HTMLHRElement
<b>html</b>	Root element	none	none*	head*; body*	globals; manifest	HTMLHtmlElement

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

<b>img</b>	Image	flow; phrasing; embedded; interactive*	phrasing	empty	globals; alt; src; srcset; crossorigin; usemap; ismap; width; height	HTMLElement
<b>input</b>	Form control	flow; phrasing; interactive*, listed; labelable; submittable; resettable;	phrasing	empty	globals; accept; alt; autocomplete; autofocus; checked; dirname; disabled; form; formaction; formencotype; formmethod; formnovalidate; formtarget; height; list-	HTMLInputElement
<b>ins</b>						HTMLModElement
<b>kbd</b>						HTMLElement
<b>keygen</b>						HTMLKeygenElement
<b>control</b>		labelable; submittable; resettable; form-associated				
<b>label</b>	Caption for a form control	flow; phrasing; interactive; form-associated	phrasing	phrasing*	globals; form; for	HTMLLabelElement
<b>legend</b>	Caption for fieldset	none	fieldset	phrasing	globals	HTMLLegendElement
<b>li</b>	List item	none	ol; ul; menu*	flow	globals; value*	HTMLListElement
<b>link</b>	Link metadata	metadata; flow*; phrasing*	head; noscript*; phrasing*	empty	globals; href; crossorigin; rel; media; hreflang; type; sizes	HTMLLinkElement
<b>map</b>	Image map	flow; phrasing*	phrasing	transparent; area*	globals; name	HTMLMapElement
<b>mark</b>	Highlight	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>menu</b>	Menu of commands	flow	flow; menu*	li*; flow*; menuitem*; hr*; menu*	globals; type; label	HTMLMenuElement
<b>meta</b>	Text metadata	metadata; flow*; phrasing*	head; noscript*; phrasing*	empty	globals; name; http-equiv; content; charset	HTMLMetaElement
<b>meter</b>	Gauge	flow; phrasing; labelable	phrasing	phrasing*	globals; value; min; max; low; high; optimum	HTMLMeterElement
<b>nav</b>	Section with navigational links	flow; sectioning	flow	flow	globals	HTMLElement
<b>noscript</b>	Fallback content for script	metadata; flow; phrasing	head*; phrasing*	varies*	globals	HTMLElement
<b>object</b>	Image, nested browsing context, or plugin	flow; phrasing; embedded; interactive*, listed; submittable; form-associated	phrasing	param*; transparent	globals; data; type; typemustmatch; name; usemap; form; width; height	HTMLObjectElement
<b>ol</b>	Ordered list	flow	flow	li	globals; reversed; start	HTMLListElement
<b>optgroup</b>	Group of options in a list box	none	select	option	globals; disabled; label	HTMLOptGroupElement
<b>option</b>	Option in a list box or combo box control	none	select; datalist; optgroup	text*	globals; disabled; label; selected; value	HTMLOptionElement

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

		form-associated				
<b>p</b>	Paragraph	flow	flow	phrasing	globals	HTMLElement
<b>param</b>	Parameter for object	none	object	empty	globals; name; value	HTMLParamElement
<b>pre</b>	Block of preformatted text	flow	flow	phrasing	globals	HTMLPreElement
<b>progress</b>	Progress bar	flow; phrasing;	phrasing	phrasing*	globals; value; max	HTMLProgressElement
<b>q</b>						HTMLElement
<b>rp</b>						HTMLElement
<b>rt</b>						HTMLElement
<b>ruby</b>						HTMLElement
<b>s</b>						HTMLElement
<b>samp</b>	Computer output	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>script</b>	Embedded script	metadata; flow; phrasing	head; phrasing	script, data, or script documentation*	globals; src; type; charset; async; defer; crossorigin	HTMLScriptElement
<b>section</b>	Generic document or application section	flow; sectioning	flow	flow	globals	HTMLElement
<b>select</b>	List box control	flow; phrasing; interactive; listed; labelable; submittable; resettable; form-associated	phrasing	option, optgroup	globals; autofocus; disabled; form; multiple; name; required; size	HTMLSelectElement
<b>small</b>	Side comment	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>source</b>	Media source for video or audio	none	video; audio	empty	globals; src; type; media	HTMLSourceElement
<b>span</b>	Generic phrasing container	flow; phrasing	phrasing	phrasing	globals	HTMLSpanElement
<b>strong</b>	Importance	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>style</b>	Embedded styling information	metadata; flow	head; noscript*; flow*	varies*	globals; media; type; scoped	HTMLStyleElement
<b>sub</b>	Subscript	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>summary</b>	Caption for details	none	details	phrasing	globals	HTMLElement
<b>sup</b>	Superscript	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>table</b>	Table	flow	flow	caption*; colgroup*; thead*; tbody*; tfoot*, tr*	globals	HTMLTableElement
<b>tbody</b>	Group of rows in a table	none	table	tr	globals	HTMLTableSectionElement
<b>td</b>	Table cell	sectioning root	tr	flow	globals; colspan; rowspan; headers	HTMLTableDataCellElement

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

		submittable; resettable; form-associated				
<b>tfoot</b>	Group of footer rows in a table	none	table	tr	globals	HTMLTableSectionElement
<b>th</b>	Table header cell	none	tr	flow*	globals; colspan; rowspan; headers; scope; abbr	HTMLTableHeaderCellElement
<b>thead</b>	Group of heading rows in	none	table	tr	globals	HTMLTableSectionElement

**time** Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

**title** [HTMLTitleElement](#)

**tr** [HTMLTableRowElement](#)

			tfoot			
<b>track</b>	Timed text track	none	audio; video	empty	globals; default; kind; label; src; srclang	HTMLTrackElement
<b>u</b>	Keywords	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>ul</b>	List	flow	flow	li	globals	HTMLULListElement
<b>var</b>	Variable	flow; phrasing	phrasing	phrasing	globals	HTMLElement
<b>video</b>	Video player	flow; phrasing; embedded; interactive	phrasing	source*; transparent*	globals; src; crossorigin; poster; preload; autoplay; mediagroup; loop; muted; controls; width; height	HTMLVideoElement
<b>wbr</b>	Line breaking opportunity	flow; phrasing	phrasing	empty	globals	HTMLElement

An asterisk (\*) in a cell indicates that the actual rules are more complicated than indicated in the table above.

† Categories in the "Parents" column refer to parents that list the given categories in their content model, not to elements that themselves are in those categories. For example, the **a** element's "Parents" column says "phrasing", so any element whose content model contains the "phrasing" category could be a parent of an **a** element. Since the "flow" category includes all the "phrasing" elements, that means the **th** element could be a parent to an **a** element.

## Element content categories

This section is non-normative.

List of element content categories

Category	Elements	Elements with exceptions
Metadata content	base; link; meta; noscript; script; style; title	—
Flow content	a; abbr; address; article; aside; audio; b; bdi; bdo; blockquote; br; button; canvas; cite; code; data; datalist; del; details; dfn; dialog; div; dl; em; embed; fieldset; figure; footer; form; h1; h2; h3; h4; h5; h6; header; hgroup; hr; i; iframe; img; input; ins; kbd; keygen; label; map; mark; math; menu; meter; nav; noscript; object; ol; output; p; pre; progress; q; ruby; s; samp; script; section; select; small; span; strong; sub; sup; svg; table; textarea; time; u; ul; var; video; wbr; Text	area (if it is a descendant of a <b>map</b> element); link (if the <b> itemprop</b> attribute is present); meta (if the <b> itemprop</b> attribute is present); style (if the <b> scoped</b> attribute is present)
Sectioning content	article; aside; nav; section	—
Heading content	h1; h2; h3; h4; h5; h6; hgroup	—
Phrasing content	a; abbr; audio; b; bdi; bdo; br; button; canvas; cite; code; data; datalist; del; dfn; em; embed; i; iframe; img; input; ins; kbd; keygen; label; map; mark; math; meter; noscript; object; output; progress; q; ruby; s; samp; script; select; small; span; strong; sub; sup; svg; textarea; time; u; var; video; wbr; Text	area (if it is a descendant of a <b>map</b> element); link (if the <b> itemprop</b> attribute is present); meta (if the <b> itemprop</b> attribute is present)

Go

NOV JAN FEB

03



2012 2013 2014

▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Sectioning roots	blockquote; body; details; dialog; fieldset; figure; td	not in the Hidden state); object (if the usemap attribute is present); video (if the controls attribute is present)
Form-associated elements	button; fieldset; input; keygen; label; object; output; select; textarea	—
Listed elements	button; fieldset; input; keygen; object; output; select; textarea	—
Submittable elements	—	—
Resettable elements	—	—
Labelable elements	—	—
Palpable content	progress; q; ruby; s; samp; section; select; small; span; strong; sub; sup; svg; table; textarea; time; u; var; video	attribute is present); dl (if the element's children include at least one name-value group); attribute is not in the Hidden state); attribute is in the toolbar state); ol (if the element's children include at least one li element); ul (if the element's children include at least one li element); Text that is not inter-element whitespace

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

## Attributes

This section is non-normative.

List of attributes (excluding event handler content attributes)

Attribute	Element(s)	Description	Value
abbr	th	Alternative label to use for the header cell when referencing the cell in other contexts	Text*
accept	input	Hint for expected file type in file upload controls	Set of comma-separated tokens* consisting of valid MIME types with no parameters or audio/*, video/*, or image/*
accept-charset	form	Character encodings to use for form submission	Ordered set of unique space-separated tokens, ASCII case-insensitive, consisting of preferred MIME names of ASCII-compatible character encodings*
accesskey	HTML elements	Keyboard shortcut to activate or focus element	Ordered set of unique space-separated tokens, case-sensitive, consisting of one Unicode code point in length
action	form	URL to use for form submission	Valid non-empty URL potentially surrounded by spaces
allowfullscreen	iframe	Whether to allow the iframe's contents to use requestFullScreen()	Boolean attribute
alt	area; img; input	Replacement text for use when images are not available	Text*
async	script	Execute script asynchronously	Boolean attribute
autocomplete	form	Default setting for autofill feature for controls in the form	"on"; "off"
autocompleter	input; select; textarea	Hint for form autofill feature	Autofill field name and related tokens*
autofocus	button; input; keygen; select; textarea	Automatically focus the form control when the page is loaded	Boolean attribute
autoplay	audio; video	Hint that the media resource can be started automatically when the page is loaded	Boolean attribute
challenge	keygen	String to package with the generated and signed public key	Text
charset	meta	Character encoding declaration	Preferred MIME name of a character encoding*
charsets	script	Character encoding of the external script resource	Preferred MIME name of a character encoding*
checked	menuitem; input	Whether the command or control is checked	Boolean attribute
cite	blockquote; del; ins; q	Link to the source of the quotation or more information about the edit	Valid URL potentially surrounded by spaces
class	HTML elements	Classes to which the element belongs	Set of space-separated tokens
cols	textarea	Maximum number of characters per line	Valid non-negative integer greater than zero

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

<code>contenteditable</code>	HTML elements	Whether the element is editable	"true"; "false"
<code>contextmenu</code>	HTML elements	The element's context menu	ID*
<code>controls</code>	audio; video	Show user agent controls	Boolean attribute
<code>coords</code>	area	Coordinates for the shape to be created in an image map	Valid list of integers*
<code>crossorigin</code>	audio; img; link; script; video	How the element handles crossorigin requests.	"anonymous"; "use-credentials"
<code>data</code>	object	Address of the resource	Valid non-empty URL potentially surrounded by spaces
<code>datetime</code>			string, valid yearless date string, date and time string, valid time-zone and time string, valid week string, valid duration string
<code>datetime</code>			
<code>default</code>			
<code>defer</code>			
<code>dir</code>			

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

<code>dirname</code>	input, textarea	Name of form field to use for sending the element's directionality in form submission	Text
<code>disabled</code>	button; menuitem; fieldset; input; keygen; optgroup; option; select; textarea	Whether the form control is disabled	Boolean attribute
<code>download</code>	a; area	Whether to download the resource instead of navigating to it, and its filename if so	Text
<code>draggable</code>	HTML elements	Whether the element is draggable	"true"; "false"
<code>dropzone</code>	HTML elements	Accepted item types for drag-and-drop	Unordered set of unique space-separated tokens, ASCII case-insensitive, consisting of accepted types and drag feedback*
<code>enctype</code>	form	Form data set encoding type to use for form submission	"application/x-www-form-urlencoded"; "multipart/form-data"; "text/plain"
<code>for</code>	label	Associate the label with form control	ID*
<code>for</code>	output	Specifies controls from which the output was calculated	Unordered set of unique space-separated tokens, case-sensitive, consisting of IDs*
<code>form</code>	button; fieldset; input; keygen; label; object; output; select; textarea	Associates the control with a <code>form</code> element	ID*
<code>formaction</code>	button; input	URL to use for form submission	Valid non-empty URL potentially surrounded by spaces
<code>formenctype</code>	button; input	Form data set encoding type to use for form submission	"application/x-www-form-urlencoded"; "multipart/form-data"; "text/plain"
<code>formmethod</code>	button; input	HTTP method to use for form submission	"GET"; "POST"
<code>formnovalidate</code>	button; input	Bypass form control validation for form submission	Boolean attribute
<code>formtarget</code>	button; input	Browsing context for form submission	Valid browsing context name or keyword
<code>headers</code>	td; th	The header cells for this cell	Unordered set of unique space-separated tokens, case-sensitive, consisting of IDs*
<code>height</code>	canvas; embed; iframe; img; input; object; video	Vertical dimension	Valid non-negative integer
<code>hidden</code>	HTML elements	Whether the element is relevant	Boolean attribute
<code>high</code>	meter	Low limit of high range	Valid floating-point number*
<code>href</code>	a; area	Address of the hyperlink	Valid URL potentially surrounded by spaces
<code>href</code>	link	Address of the hyperlink	Valid non-empty URL potentially surrounded by spaces
<code>href</code>	base	Document base URL	Valid URL potentially surrounded by spaces
<code>hreflang</code>	a; area; link	Language of the linked resource	Valid BCP 47 language tag
<code>http-equiv</code>	meta	Pragma directive	Text*
<code>icon</code>	menuitem	Icon for the command	Valid non-empty URL potentially surrounded by spaces
<code>id</code>	HTML elements	The element's ID	Text*
<code>inert</code>	HTML elements	Whether the element is inert	Boolean attribute
<code>inputmode</code>	input; textarea	Hint for selecting an input modality	verbatim; latin; latin-name; latin-prose; full-width-latin; kana; katakana; numeric; tel; email; url

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

<b>itemprop</b>	HTML elements	Property names of a microdata item	Unordered set of unique space-separated tokens, case-sensitive, consisting of valid absolute URLs, defined property names, or text*
<b>itemref</b>	HTML elements	Referenced elements	Unordered set of unique space-separated tokens, case-sensitive, consisting of IDs*
<b>itemscope</b>	HTML elements	Introduces a microdata item	Boolean attribute
<b>itemtype</b>	HTML elements	Item types of a microdata item	Unordered set of unique space-separated tokens, case-sensitive, consisting of valid absolute URL*
<b>keytype</b>	keygen	The type of cryptographic key to generate	Text*
<b>kind</b>			; "descriptions"; "chapters";

label      Too slow? Try reading the multipage copy of  
 lang      the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

list      · the empty string

loop     

low     

manifest      .nally surrounded by spaces

<b>max</b>	input	Maximum value	Varies*
<b>max</b>	meter; progress	Upper bound of range	Valid floating-point number*
<b>maxlength</b>	input; textarea	Maximum length of value	Valid non-negative integer
<b>media</b>	link; source; style	Applicable media	Valid media query
<b>mediagroup</b>	audio; video	Groups media elements together with an implicit MediaController	Text
<b>method</b>	form	HTTP method to use for form submission	"GET"; "POST"; "dialog"
<b>min</b>	input	Minimum value	Varies*
<b>min</b>	meter	Lower bound of range	Valid floating-point number*
<b>multiple</b>	input; select	Whether to allow multiple values	Boolean attribute
<b>muted</b>	audio; video	Whether to mute the media resource by default	Boolean attribute
<b>name</b>	button;fieldset;input;keygen;output;select;textarea	Name of form control to use for form submission and in the <code>form.elements</code> API	Text*
<b>name</b>	form	Name of form to use in the <code>document.forms</code> API	Text*
<b>name</b>	iframe; object	Name of nested browsing context	Valid browsing context name or keyword
<b>name</b>	map	Name of image map to reference from the <code>usemap</code> attribute	Text*
<b>name</b>	meta	Metadata name	Text*
<b>name</b>	param	Name of parameter	Text
<b>novalidate</b>	form	Bypass form control validation for form submission	Boolean attribute
<b>open</b>	details	Whether the details are visible	Boolean attribute
<b>open</b>	dialog	Whether the dialog box is showing	Boolean attribute
<b>optimum</b>	meter	Optimum value in gauge	Valid floating-point number*
<b>pattern</b>	input	Pattern to be matched by the form control's value	Regular expression matching the JavaScript <i>Pattern</i> production
<b>ping</b>	a; area	URLs to ping	Set of space-separated tokens consisting of valid non-empty URLs
<b>placeholder</b>	input; textarea	User-visible label to be placed within the form control	Text*
<b>poster</b>	video	Poster frame to show prior to video playback	Valid non-empty URL potentially surrounded by spaces
<b>preload</b>	audio; video	Hints how much buffering the media resource will likely need	"none"; "metadata"; "auto"
<b>radiogroup</b>	menuitem	Name of group of commands to treat as a radio button group	Text
<b>readonly</b>	input; textarea	Whether to allow the value to be edited by the user	Boolean attribute

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

<b>reversed</b>	ol	submission	Boolean attribute
<b>rows</b>	textarea	Number of lines to show	Valid non-negative integer greater than zero
<b rowspan="2"><b>rowspan</b></b>	td; th	Number of rows that the cell is to span	Valid non-negative integer
iframe	Security rules for nested content	Unordered set of unique space-separated tokens, ASCII case-insensitive, consisting of "allow-forms", "allow-pointer-lock", "allow-popups", "allow-same-origin", "allow-scripts" and "allow-top-navigation"	
<b>spellcheck</b>	HTML elements	Whether the element is to have its spelling checked	Boolean attribute

scope      Too slow? Try reading the multipage copy of  
 scoped     the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

<b>selected</b>			
<b>shape</b>		map	...ly"; "rect"
<b>size</b>	input; select	Size of the control	Valid non-negative integer greater than zero
<b>sizes</b>	link	Sizes of the icons (for rel="icon")	Unordered set of unique space-separated tokens, ASCII case-insensitive, consisting of sizes*
<b>span</b>	col; colgroup	Number of columns spanned by the element	Valid non-negative integer greater than zero
<b>src</b>	audio; embed; iframe; img; input; script; source; track; video	Address of the resource	Valid non-empty URL potentially surrounded by spaces
<b>srcdoc</b>	iframe	A document to render in the iframe	The source of an iframe srcdoc document*
<b>srclang</b>	track	Language of the text track	Valid BCP 47 language tag
<b>srcset</b>	img	Images to use in different situations (e.g. high-resolution displays, small monitors, etc)	Comma-separated list of image candidate strings
<b>start</b>	ol	Ordinal value of the first item	Valid integer
<b>step</b>	input	Granularity to be matched by the form control's value	Valid floating-point number greater than zero, or "any"
<b>style</b>	HTML elements	Presentational and formatting instructions	CSS declarations*
<b>tabindex</b>	HTML elements	Whether the element is focusable, and the relative order of the element for the purposes of sequential focus navigation	Valid integer
<b>target</b>	a; area	Browsing context for hyperlink navigation	Valid browsing context name or keyword
<b>target</b>	base	Default browsing context for hyperlink navigation and form submission	Valid browsing context name or keyword
<b>target</b>	form	Browsing context for form submission	Valid browsing context name or keyword
<b>title</b>	HTML elements	Advisory information for the element	Text
<b>title</b>	abbr; dfn	Full term or expansion of abbreviation	Text
<b>title</b>	menuitem	Hint describing the command	Text
<b>title</b>	link	Title of the link	Text
<b>title</b>	link; style	Alternative style sheet set name	Text
<b>translate</b>	HTML elements	Whether the element is to be translated when the page is localized	"yes"; "no"
<b>type</b>	a; area; link	Hint for the type of the referenced resource	Valid MIME type
<b>type</b>	button	Type of button	"submit"; "reset"; "button"; "menu"
<b>type</b>	menuitem	Type of command	"command"; "checkbox"; "radio"
<b>type</b>	embed; object; script; source; style	Type of embedded resource	Valid MIME type
<b>type</b>	input	Type of form control	input type keyword
<b>type</b>	menu	Type of menu	"popup"; "toolbar"
<b>typemustmatch</b>	object	Whether the type attribute and the Content-Type value need to match for the resource to be used	Boolean attribute
<b>usemap</b>	img; object	Name of image map to use	Valid hash-name reference*

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

<b>value</b>	li	Ordinal value of the list item	Valid integer
<b>value</b>	meter; progress	Current value of the element	Valid floating-point number
<b>value</b>	param	Value of parameter	Text
<b>width</b>	canvas; embed; iframe; img; input; object; video	Horizontal dimension	Valid non-negative integer
<b>wrap</b>	textarea	How the value of the form control is to be wrapped for form submission	"soft"; "hard"

An asterisk (\*) in a cell indicates that the actual rules are more complicated than indicated in the table above.

Attribut Value

**onabort** event handler content attribute

**onafterpri:** event handler content attribute

**onbeforepri:** event handler content attribute

**onbeforeunl:** event handler content attribute

**onblur** event handler content attribute

**onblur** HTML elements blur event handler

**oncancel** HTML elements cancel event handler

**oncanplay** HTML elements canplay event handler

**oncanplaythrough** HTML elements canplaythrough event handler

**onchange** HTML elements change event handler

**onclick** HTML elements click event handler

**onclose** HTML elements close event handler

**oncontextmenu** HTML elements contextmenu event handler

**oncuechange** HTML elements cuechange event handler

**ondblclick** HTML elements dblclick event handler

**ondrag** HTML elements drag event handler

**ondragend** HTML elements dragend event handler

**ondragenter** HTML elements dragenter event handler

**ondragleave** HTML elements dragleave event handler

**ondragover** HTML elements dragover event handler

**ondragstart** HTML elements dragstart event handler

**ondrop** HTML elements drop event handler

**ondurationchange** HTML elements durationchange event handler

**onemptied** HTML elements emptied event handler

**onended** HTML elements ended event handler

**onerror** body error event handler for Window object, and handler for script error notifications

**onerror** HTML elements error event handler

**onfocus** body focus event handler for Window object

**onfocus** HTML elements focus event handler

**onfullscreenchange** body fullscreenchange event handler for Window object

**onfullscreenerror** body fullscreenerror event handler for Window object

**onhashchange** body hashchange event handler for Window object

**oninput** HTML elements input event handler

**oninvalid** HTML elements invalid event handler

**onkeydown** HTML elements keydown event handler

**onkeypress** HTML elements keypress event handler

**onkeyup** HTML elements keyup event handler

**onload** body load event handler for Window object

**onload** HTML elements load event handler

**onloadeddeodata** HTML elements loadeddeodata event handler

**onloadedmetadata** HTML elements loadedmetadata event handler

**onloadstart** HTML elements loadstart event handler

**onmessage** body message event handler for Window object

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

<code>onmouseover</code>	HTML elements	mouseover event handler	Event handler content attribute
<code>onmouseup</code>	HTML elements	mouseup event handler	Event handler content attribute
<code>onmousewheel</code>	HTML elements	mousewheel event handler	Event handler content attribute
<code>onoffline</code>	body	offline event handler for Window object	Event handler content attribute
<code>ononline</code>	body	online event handler for Window object	Event handler content attribute
<code>onpagehide</code>	body	pagehide event handler for Window object	Event handler content attribute
<code>onpageshow</code>	body	pageshow event handler for Window object	Event handler content attribute
<code>onpause</code>	HTML elements	pause event handler	Event handler content attribute
<code>onplay</code>			Event handler content attribute
<code>onplaying</code>			Event handler content attribute
<code>onpopstate</code>			Event handler content attribute
<code>onprogress</code>			Event handler content attribute
<code>onratechange</code>			Event handler content attribute
<code>onreset</code>			Event handler content attribute
<code>onresize</code>			Event handler content attribute
<code>onscroll</code>			Event handler content attribute
<code>onscroll</code>	HTML elements	scroll event handler	Event handler content attribute
<code>onseeked</code>	HTML elements	seeked event handler	Event handler content attribute
<code>onseeking</code>	HTML elements	seeking event handler	Event handler content attribute
<code>onselect</code>	HTML elements	select event handler	Event handler content attribute
<code>onshow</code>	HTML elements	show event handler	Event handler content attribute
<code>onsort</code>	HTML elements	sort event handler	Event handler content attribute
<code>onstalled</code>	HTML elements	stalled event handler	Event handler content attribute
<code>onstorage</code>	body	storage event handler for Window object	Event handler content attribute
<code>onsubmit</code>	HTML elements	submit event handler	Event handler content attribute
<code>onsuspend</code>	HTML elements	suspend event handler	Event handler content attribute
<code>ontimeupdate</code>	HTML elements	timeupdate event handler	Event handler content attribute
<code>onunload</code>	body	unload event handler for Window object	Event handler content attribute
<code>onvolumechange</code>	HTML elements	volumechange event handler	Event handler content attribute
<code>onwaiting</code>	HTML elements	waiting event handler	Event handler content attribute

## Element Interfaces

This section is non-normative.

List of interfaces for elements

Element(s)	Interface(s)
a	HTMLAnchorElement : HTMLElement
abbr	HTMLElement
address	HTMLElement
area	HTMLAreaElement : HTMLElement
article	HTMLElement
aside	HTMLElement
audio	HTMLAudioElement : HTMLMediaElement : HTMLElement
b	HTMLElement
base	HTMLBaseElement : HTMLElement
bdi	HTMLElement
bdo	HTMLElement
blockquote	HTMLQuoteElement : HTMLElement
body	HTMLBodyElement : HTMLElement
br	HTMLBRElement : HTMLElement
button	HTMLButtonElement : HTMLElement
canvas	HTMLCanvasElement : HTMLElement
caption	HTMLTableCaptionElement : HTMLElement

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

colgroup	HTMLTableColElement : HTMLElement
menuitem	HTMLMenuItemElement : HTMLElement
data	HTMLDataElement : HTMLElement
datalist	HTMLDataListElement : HTMLElement
dd	HTMLElement
del	HTMLModElement : HTMLElement
details	HTMLDetailsElement : HTMLElement
dfn	HTMLElement
dialog	
div	
dl	
dt	
em	
embed	
fieldset	
figcaption	h.
figure	HTMLElement
footer	HTMLElement
form	HTMLFormElement : HTMLElement
h1	HTMLHeadingElement : HTMLElement
h2	HTMLHeadingElement : HTMLElement
h3	HTMLHeadingElement : HTMLElement
h4	HTMLHeadingElement : HTMLElement
h5	HTMLHeadingElement : HTMLElement
h6	HTMLHeadingElement : HTMLElement
head	HTMLHeadElement : HTMLElement
header	HTMLElement
hgroup	HTMLElement
hr	HTMLHRElement : HTMLElement
html	HTMLHtmlElement : HTMLElement
i	HTMLElement
iframe	HTMLIFrameElement : HTMLElement
img	HTMLImageElement : HTMLElement
input	HTMLInputElement : HTMLElement
ins	HTMLModElement : HTMLElement
kbd	HTMLElement
keygen	HTMLKeygenElement : HTMLElement
label	HTMLLabelElement : HTMLElement
legend	HTMLLegendElement : HTMLElement
li	HTMLLIElement : HTMLElement
link	HTMLLinkElement : HTMLElement
map	HTMLMapElement : HTMLElement
mark	HTMLElement
menu	HTMLMenuElement : HTMLElement
meta	HTMLMetaElement : HTMLElement
meter	HTMLMeterElement : HTMLElement
nav	HTMLElement
noscript	HTMLElement
object	HTMLObjectElement : HTMLElement
ol	HTMLListElement : HTMLElement
optgroup	HTMLOptGroupElement : HTMLElement
option	HTMLOptionElement : HTMLElement
output	HTMLOutputElement : HTMLElement
p	HTMLParagraphElement : HTMLElement

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

q	HTMLQuoteElement : HTMLElement
rp	HTMLElement
rt	HTMLElement
ruby	HTMLElement
s	HTMLElement
samp	HTMLElement
script	HTMLScriptElement : HTMLElement
section	HTMLElement
select	
small	
source	
span	
strong	
style	
sub	
summary	h.
sup	HTMLElement
table	HTMLTableElement : HTMLElement
tbody	HTMLTableSectionElement : HTMLElement
td	HTMLTableDataCellElement : HTMLTableCellElement : HTMLElement
textarea	HTMLTextAreaElement : HTMLElement
tfoot	HTMLTableSectionElement : HTMLElement
th	HTMLTableHeaderCellElement : HTMLTableCellElement : HTMLElement
thead	HTMLTableSectionElement : HTMLElement
time	HTMLTimeElement : HTMLElement
title	HTMLTitleElement : HTMLElement
tr	HTMLTableRowElement : HTMLElement
track	HTMLTrackElement : HTMLElement
u	HTMLElement
ul	HTMLULListElement : HTMLElement
var	HTMLElement
video	HTMLVideoElement : HTMLMediaElement : HTMLElement
wbr	HTMLElement



Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

## All Interfaces

*This section is non-normative.*

- AbstractWorker
- ApplicationCache
- AudioTrack
- AudioTrackList
- BarProp
- BeforeUnloadEvent
- CanvasDrawingStyles
- CanvasGradient
- CanvasPathMethods
- CanvasPattern
- CanvasProxy
- CanvasRenderingContext2D
- CloseEvent
- DOMElementMap
- DOMStringMap
- DataTransfer
- DataTransferItem
- DataTransferItemList
- DedicatedWorkerGlobalScope
- Document
- DragEvent

Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- HTMLAllCollection
- HTMLAnchorElement
- HTMLAppletElement
- HTMLAreaElement
- HTMLAudioElement
- HTMLBRElement
- HTMLBaseElement
- HTMLBaseFontElement
- HTMLBodyElement
- HTMLButtonElement
- HTMLCaptionElement
- HTMLCanvasElement
- HTMLFormElement
- HTMLFrameElement
- HTMLFrameSetElement
- HTMLHRElement
- HTMLHeadElement
- HTMLHeadingElement
- HTMLHtmlElement
- HTMLIFrameElement
- HTMLImageElement
- HTMLInputElement
- HTMLKeygenElement
- HTMLLIElement
- HTMLLabelElement
- HTMLLegendElement
- HTMLLinkElement
- HTMLMapElement
- HTMLMarqueeElement
- HTMLMediaElement
- HTMLMenuElement
- HTMLMenuItemElement
- HTMLMetaElement
- HTMLMeterElement
- HTMLModElement
- HTMLOLListElement
- HTMLObjectElement
- HTMLOptGroupElement
- HTMLOptionElement
- HTMLOptionsCollection
- HTMLOutputElement
- HTMLParagraphElement
- HTMLParamElement
- HTMLPreElement
- HTMLProgressElement
- HTMLPropertiesCollection
- HTMLQuoteElement
- HTMLScriptElement
- HTMLSelectElement
- HTMLSourceElement
- HTMLSpanElement
- HTMLStyleElement
- HTMLTableCaptionElement
- HTMLTableCellElement
- HTMLTableColElement
- HTMLTableDataCellElement
- HTMLTableElement
- HTMLTableHeaderCellElement
- HTMLTableRowElement
- HTMLTableSectionElement

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



Go

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

- HTMLElement
- HTMLVideoElement
- HashChangeEvent
- History
- ImageBitmap
- ImageBitmapFactories
- ImageData
- Location
- MediaController
- MediaError
- MessageEvent
- MessagePort
- Nav
- Nav
- Nav
- Nav
- Nav
- Pag
- Path
- PopStateEvent
- PortCollection
- PropertyNodeList
- RadioNodeList
- SharedWorker
- SharedWorkerGlobalScope
- Storage
- StorageEvent
- TextMetrics
- TextTrack
- TextTrackCue
- TextTrackCueList
- TextTrackList
- TimeRanges
- TrackEvent
- Transferable
- ValidityState
- VideoTrack
- VideoTrackList
- WebSocket
- Window
- WindowBase64
- WindowLocalStorage
- WindowModal
- WindowSessionStorage
- WindowTimers
- Worker
- WorkerGlobalScope
- WorkerLocation
- WorkerNavigator
- WorkerUtils
- XMLDocument, partial

Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>



Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

About this capture

*Augmented BNF for Syntax Specifications: ABNF*, D. Crocker, P. Overell. IETF.**[ABOUT]***The 'about' URI scheme*, S. Moonesamy. IETF.**[AGIF]***(Non-normative) GIF Application Extension: NETSCAPE2.0*. R. Frazier.**[APNG]**(Non-**[ARIA]** Too slow? Try reading the multipage copy of  
the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>**[ARIAIMPL]**WA /[whatwg.org/html](http://whatwg.org/html)**[ATAG]**(Noi /[whatwg.org/html](http://whatwg.org/html)**[ATOM]**(Non-normative) *The Atom Syndication Format*, M. Nottingham, R. Sayre. IETF.**[BCP47]***Tags for Identifying Languages; Matching of Language Tags*, A. Phillips, M. Davis. IETF.**[BECSS]***Behavioral Extensions to CSS*, I. Hickson. W3C.**[BEZIER]***Courbes à poles*, P. de Casteljau. INPI, 1959.**[BIDI]***UAX #9: Unicode Bidirectional Algorithm*, M. Davis. Unicode Consortium.**[BOCU1]**(Non-normative) *UTN #6: BOCU-1: MIME-Compatible Unicode Compression*, M. Scherer, M. Davis. Unicode Consortium.**[CESU8]**(Non-normative) *UTR #26: Compatibility Encoding Scheme For UTF-16: 8-BIT (CESU-8)*, T. Phipps. Unicode Consortium.**[CHARMOD]**(Non-normative) *Character Model for the World Wide Web 1.0: Fundamentals*, M. Dürst, F. Yergeau, R. Ishida, M. Wolf, T. Texin. W3C.**[CLDR]***Unicode Common Locale Data Repository*. Unicode.**[COMPUTABLE]**(Non-normative) *On computable numbers, with an application to the Entscheidungsproblem*, A. Turing. In *Proceedings of the London Mathematical Society*, series 2, volume 42, pages 230-265. London Mathematical Society, 1937.**[COOKIES]***HTTP State Management Mechanism*, A. Barth. IETF.**[CORS]***Cross-Origin Resource Sharing*, A. van Kesteren. WHATWG.**[CP50220]**

CP50220, Y. Naruse. IANA.

**[CP51932]**

CP51932, Y. Naruse. IANA.

**[CSP]**(Non-normative) *Content Security Policy*, B. Sterne, A. Barth. W3C.**[CSS]**

[Go](#)

NOV JAN FEB

◀ 03 ▶

2012 2013 2014



▼ About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**[CSSATTR]***CSS Styling Attribute Syntax*, T. Çelik, E. Etemad. W3C.**[CSSCOLOR]***CSS Color Module Level 3*, T. Çelik, C. Lilley, L. Baron. W3C.**[CSSFONTS]***CSS Fonts Module Level 3*, J. Daggett. W3C.**[CSSIMAGES]**

CSS

**[CSSOM]** Too slow? Try reading the multipage copy ofthe spec instead: <https://web.archive.org>**[CSSOMVIE]**CS /<http://web/20130103030411/http://whatwg.org/html>**[CSSRUBY]**

CSS

**[CSSTRANSITIONS]**(Non-normative) *CSS Transitions*, D. Jackson, D. Hyatt, C. Marrin, L. Baron. W3C.**[CSSUI]***CSS3 Basic User Interface Module*, T. Çelik. W3C.**[CSSVALUES]***CSS3 Values and Units*, H. Lie, T. Atkins, E. Etemad. W3C.**[DASH]***Dynamic adaptive streaming over HTTP (DASH)*. ISO.**[DOM]***DOM*, A. van Kesteren, A. Gregor, Ms2ger. WHATWG.**[DOMEVENTS]***Document Object Model (DOM) Level 3 Events Specification*, T. Leithead, J. Rossi, D. Schepers, B. Höhrmann, P. Le Hégaret, T. Pixley. W3C.**[DOMPARSING]***DOM Parsing and Serialization*, Ms2ger. WHATWG.**[DOT]**(Non-normative) *The DOT Language*. Graphviz.**[E163]***Recommendation E. 163 — Numbering Plan for The International Telephone Service*, CCITT Blue Book, Fascicle II.2, pp. 128-134, November 1988.**[ECMA262]***ECMAScript Language Specification*. ECMA.**[ECMA357]**(Non-normative) *ECMAScript for XML (E4X) Specification*. ECMA.**[EDITING]***HTML Editing APIs*, A. Gregor. W3C Editing APIs CG.**[EUCKR]***Hangul Unix Environment*. Korea Industrial Standards Association. Ref. No. KS C 5861-1992.**[EUCJP]***Definition and Notes of Japanese EUC*. UI-OSF-USLP. In English in the abridged translation of the UI-OSF Application Platform Profile for Japanese Environment, Appendix C.**[FILEAPI]***File API*, A. Ranganathan. W3C.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

▼ About this capture

**[GBK]***Chinese Internal Code Specification.* Chinese IT Standardization Technical Committee.**[GIF]**(Non-normative) *Graphics Interchange Format.* CompuServe.**[GRAPHICS]**(Non-normative) *Computer Graphics: Principles and Practice in C*, Second Edition, J. Foley, A. van Dam, S. Feiner, J. Hughes. Addison Wesley, ISBN 0-201-81240-9.**[GREGORI]** Too slow? Try reading the multipage copy of

(No

the spec instead: <https://web.archive.org>**[HTMLDIFF]**

(No /web/20130103030411/http:

**[HTTP]**Hyp //whatwg.org/html  
IETF.

P. Leach, T. Berners-Lee.

**[HTTPS]**(Non-normative) *HTTP Over TLS*, E. Rescorla. IETF.**[IANACHARSET]***Character Sets.* IANA.**[IANAPERMHEADERS]***Permanent Message Header Field Names.* IANA.**[ISO8601]**(Non-normative) *ISO8601: Data elements and interchange formats — Information interchange — Representation of dates and times.* ISO.**[ISO885911]***ISO-8859-11: Information technology — 8-bit single-byte coded graphic character sets — Part 11: Latin/Thai alphabet.* ISO.**[JLREQ]***Requirements for Japanese Text Layout.* W3C.**[JPEG]***JPEG File Interchange Format.* E. Hamilton.**[JSON]***The application/json Media Type for JavaScript Object Notation (JSON).* D. Crockford. IETF.**[JSURL]***The 'javascript' resource identifier scheme.* B. Höhrmann. IETF.**[MAILTO]**(Non-normative) *The 'mailto' URI scheme.* M. Duerst, L. Masinter, J. Zawinski. IETF.**[MATHML]***Mathematical Markup Language (MathML).* D. Carlisle, P. Ion, R. Miner, N. Poppelier. W3C.**[MEDIAFRAG]***Media Fragments URI.* R. Troncy, E. Mannens, S. Pfeiffer, D. Van Deursen. W3C.**[MFREL]***Microformats Wiki: existing rel values.* Microformats.**[MIMESNIFF]***MIME Sniffing.* G. Hemsley. WHATWG.**[MNG]***MNG (Multiple-image Network Graphics) Format.* G. Randers-Pehrson.**[MPEG2]**

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**[MQ]***Media Queries*, H. Lie, T. Çelik, D. Glazman, A. van Kesteren. W3C.**[NPAPI]**(Non-normative) *Gecko Plugin API Reference*. Mozilla.**[OGGSKELETONHEADERS]***SkeletonHeaders*. Xiph.Org.**[OPENSEAR<sup>r</sup>]***Aut<sup>r</sup> Too slow? Try reading the multipage copy of***[ORIGIN]***The spec instead: <https://web.archive.org>***[PAGEVIS]** /web/20130103030411/http:*(No //whatwg.org/html***[PDF]**(Non-normative) *Document management — Portable document format — Part 1: PDF*. ISO.**[PINGBACK]***Pingback 1.0*, S. Langridge, I. Hickson.**[PNG]***Portable Network Graphics (PNG) Specification*, D. Duce. W3C.**[POINTERLOCK]***Pointer Lock*, V. Scheib. W3C.**[POLYGLOT]**(Non-normative) *Polyglot Markup: HTML-Compatible XHTML Documents*, E. Graff. W3C.**[PORTERDUFF]***Compositing Digital Images*, T. Porter, T. Duff. In *Computer graphics*, volume 18, number 3, pp. 253-259. ACM Press, July 1984.**[PPUTF8]**(Non-normative) *The Properties and Promises of UTF-8*, M. Dürst. University of Zürich. In *Proceedings of the 11th International Unicode Conference*.**[PSL]***Public Suffix List*. Mozilla Foundation.**[RFC1034]***Domain Names - Concepts and Facilities*, P. Mockapetris. IETF, November 1987.**[RFC1345]***Character Mnemonics and Character Sets*, K. Simonsen. IETF.**[RFC1468]***Japanese Character Encoding for Internet Messages*, J. Murai, M. Crispin, E. van der Poel. IETF.**[RFC1554]***ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP*, M. Ohta, K. Handa. IETF.**[RFC1557]***Korean Character Encoding for Internet Messages*, U. Choi, K. Chon, H. Park. IETF.**[RFC1842]***ASCII Printable Characters-Based Chinese Character Encoding for Internet Messages*, Y. Wei, Y. Zhang, J. Li, J. Ding, Y. Jiang. IETF.**[RFC1922]***Chinese Character Encoding for Internet Messages*, HF. Zhu, DY. Hu, ZG. Wang, TC. Kao, WCH. Chang, M. Crispin. IETF.**[RFC2046]***Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, N. Freed, N. Borenstein. IETF.

Go

NOV JAN FEB

03

2012 2013 2014



About this capture

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

**[RFC2313]***PKCS #1: RSA Encryption*, B. Kaliski. IETF.**[RFC2318]***The text/css Media Type*, H. Lie, B. Bos, C. Lilley. IETF.**[RFC2388]***Returning Values from Forms: multipart/form-data*, L. Masinter. IETF.**[RFC2397]***The Too slow? Try reading the multipage copy of***[RFC2445]***the spec instead: https://web.archive.org**son. IETF.**Inte***[RFC2483]***/web/20130103030411/http:**URI**//whatwg.org/html***[RFC2781]***UTF-16, an encoding of ISO 10646*, P. Hoffman, F. Yergeau. IETF.**[RFC3676]***The Text/Plain Format and DelSp Parameters*, R. Gellens. IETF.**[RFC3023]***XML Media Types*, M. Murata, S. St. Laurent, D. Kohn. IETF.**[RFC3279]***Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, W. Polk, R. Housley, L. Bassham. IETF.**[RFC3490]***Internationalizing Domain Names in Applications (IDNA)*, P. Faltstrom, P. Hoffman, A. Costello. IETF.**[RFC3629]***UTF-8, a transformation format of ISO 10646*, F. Yergeau. IETF.**[RFC3864]***Registration Procedures for Message Header Fields*, G. Klyne, M. Nottingham, J. Mogul. IETF.**[RFC3986]***Uniform Resource Identifier (URI): Generic Syntax*, T. Berners-Lee, R. Fielding, L. Masinter. IETF.**[RFC3987]***Internationalized Resource Identifiers (IRIs)*, M. Dürst, M. Suignard. IETF.**[RFC4281]***The Codecs Parameter for "Bucket" Media Types*, R. Gellens, D. Singer, P. Frojdh. IETF.**[RFC4329]***(Non-normative) Scripting Media Types*, B. Höhrmann. IETF.**[RFC4395]***Guidelines and Registration Procedures for New URI Schemes*, T. Hansen, T. Hardie, L. Masinter. IETF.**[RFC4648]***The Base16, Base32, and Base64 Data Encodings*, S. Josefsson. IETF.**[RFC5280]***Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk. IETF.**[RFC5322]***Internet Message Format*, P. Resnick. IETF.**[RFC5724]***URI Scheme for Global System for Mobile Communications (GSM) Short Message Service (SMS)*, E. Wilde, A. Vaha-Sipila.

[Go](#)

NOV JAN FEB

03

2012 2013 2014

[About this capture](#)[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[RFC3550]

*vCard Format Specification*, S. Perreault. IETF.

[SCSU]

(Non-normative) *UTR #6: A Standard Compression Scheme For Unicode*, M. Wolf, K. Whistler, C. Wicksteed, M. Davis, A. Freytag, M. Scherer. Unicode Consortium.

[SELECTORS]

*Selectors*, E. Etemad, T. Çelik, D. Glazman, I. Hickson, P. Linss, J. Williams. W3C.

[SHIFTJIS]

*JIS Cor* Too slow? Try reading the multipage copy of Industrial Standards[SRGB] the spec instead: <https://web.archive.org>*IEC mai* /web/20130103030411/http: Part 2-1: Colour

[SVG] //whatwg.org/html

*Scal.* A. Emmons, J. Ferraiolo, A. Grasso, V. Hardy, S. Hayman, D. Jackson, C. Lilley, C. McCormack, A. Neumann, C. Northway, A. Quint, N. Ramani, D. Schepers, A. Shellshear. W3C.

[TIS620]

*UDC 681.3.04:003.62*. Thai Industrial Standards Institute, Ministry of Industry, Royal Thai Government. ISBN 974-606-153-4.

[TYPEDARRAY]

*Typed Array Specification*, D. Herman, K. Russell. Khronos.

[UAAG]

(Non-normative) *User Agent Accessibility Guidelines (UAAG) 2.0*, J. Allan, K. Ford, J. Richards, J. Spellman. W3C.

[UNDO]

*UndoManager and DOM Transaction*, R. Niwa.

[UNICODE]

*The Unicode Standard*. Unicode Consortium.

[UNIVCHARDET]

(Non-normative) *A composite approach to language/encoding detection*, S. Li, K. Momoi. Netscape. In *Proceedings of the 19th International Unicode Conference*.

[UTF7]

*UTF-7: A Mail-Safe Transformation Format of Unicode*, D. Goldsmith, M. Davis. IETF.

[UTF8DET]

(Non-normative) *Multilingual form encoding*, M. Dürst. W3C.

[UTR36]

(Non-normative) *UTR #36: Unicode Security Considerations*, M. Davis, M. Suignard. Unicode Consortium.

[WCAG]

(Non-normative) *Web Content Accessibility Guidelines (WCAG) 2.0*, B. Caldwell, M. Cooper, L. Reid, G. Vanderheiden. W3C.

[WEBGL]

*WebGL Specification*, C. Marrin. Khronos Group.

[WEBIDL]

*Web IDL*, C. McCormack. W3C.

[WEBLINK]

*Web Linking*, M. Nottingham. IETF.

[WEBMCG]

*WebM Container Guidelines*. The WebM Project.

[WEBVTT]

*WebVTT*, I. Hickson. W3C.

[ ] [Go](#) **NOV JAN FEB**  
3,445 captures **03** [◀](#) [▶](#)  
5 Jun 2004 - 22 Aug 2020 **2012 2013 2014** [▼ About this capture](#)

**[WIN1254]**

*Windows 1254.* Microsoft.

**[WIN31J]**

*Windows Codepage 932.* Microsoft.

**[WIN874]**

*Windows 874.* Microsoft.

**[WIN949]**

*Win* Too slow? Try reading the multipage copy of

**[WSP]**

*The* the spec instead: <https://web.archive.org>

**[X121]**

/web/20130103030411/http:

*Rec* //whatwg.org/html

ook, Fascicle VIII.3, pp.

317

**[X690]**

*Recommendation X.690 — Information Technology — ASN.1 Encoding Rules — Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER).* International Telecommunication Union.

**[XFN]**

*XFN 1.1 profile,* T. Çelik, M. Mullenweg, E. Meyer. GMPG.

**[XHR]**

*XMLHttpRequest,* A. van Kesteren. WHATWG.

**[XML]**

*Extensible Markup Language,* T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau. W3C.

**[XMLBASE]**

*XML Base,* J. Marsh, R. Tobin. W3C.

**[XMLNS]**

*Namespaces in XML,* T. Bray, D. Hollander, A. Layman, R. Tobin. W3C.

**[XPath10]**

*XML Path Language (XPath) Version 1.0,* J. Clark, S. DeRose. W3C.

**[XSLT10]**

(Non-normative) *XSL Transformations (XSLT) Version 1.0,* J. Clark. W3C.

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

[About this capture](#)

Adele Peterson, Adrian Bateman, Adrian Sutton, Agustín Fernández, Aharon (Vladimir) Lanin, Ajai Tirumali, Akatsuki Kitamura, Alan Plum, Alastair Campbell, Alejandro G. Castro, Alex Bishop, Alex Nicolaou, Alex Rousskov, Alexander J. Vincent, Alexandre Morgaut, Alexey Feldgendler, Алексей Проскуряков (Alexey Proskuryakov), Alexis Deveria, Allan Clements, Amos Jeffries, Anders Carlsson, Andreas, Andreas Kling, Andrei Popescu, André E. Veltstra, Andrew Barfield, Andrew Clover, Andrew Gove, Andrew Grieve, Andrew Oakley, Andrew Sidwell, Andrew Simons, Andrew Smith, Andrew W. Hagen, Andrey V. Lukyanov, Andy Heydon, Andy Palay, Anne van Kesteren, Anthony Boyd, Anthony Bryan, Anthony Hickson, Anthony Ricaud, Antti Koivisto, Arne Thomassen, Aron Spohr, Arphen Lin, Arun Patole, Aryeh Gregor, Asbjørn Ulsberg, Ashley Gullen, Ashley Sheridan, Atsushi Takayama, Aurelien Levy, Ave Wrigley, Ben Boyle, Ben Godfrey, Ben Lerner, Ben Leslie, Ben Meadowcroft, Ben Millard, Benjamin Carl Wiley Sittler, Benjamin Hawkes-Lewis, Benoit Ren, Bert Ros, Rian Parsia, Ril Corry, Bill Mason, Bill McCay, Billie Wong, Biartur Thorlund, Björn Höhrmann, Blake Frantz, Bob Lund, B Zamir, Brian D'Arcus, Br Carols Perel, Apers, Chris Schmidt, Ch Corpew Re Daniel Brum Steinberg, D

Too slow? Try reading the multipage copy of the spec instead: <https://web.archive.org/web/20130103030411/http://whatwg.org/html>

Townsend, David Baron, David Bloom, David Bruant, David Carlisle, David E. Cleary, David Egan Evans, David Flanagan, David Gerard, David Häsäther, David Hyatt, David I. Lehn, David John Burrowes, David Matja, David Remahl, David Smith, David Woolley, DeWitt Clinton, Dean Edridge, Dean Edwards, Debi Orton, Derek Featherstone, Devdatta, Dimitri Glazkov, Dmitry Golubovsky, Dirk Pranke, Divya Manian, Dmitry Titov, dolphinling, Dominique Hazaël-Massieux, Don Brutzman, Doron Rosenberg, Doug Kramer, Doug Simpkinson, Drew Wilson, Edmund Lai, Eduard Pascual, Eduardo Vela, Edward O'Connor, Edward Welbourne, Edward Z. Yang, Ehsan Akhgari, Eira Monstad, Eitan Adler, Eliot Graff, Elisabeth Robson, Elizabeth Castro, Elliott Sprehn, Elliotte Harold, Eric Carlson, Eric Lawrence, Eric Rescorla, Eric Semling, Erik Arvidsson, Erik Rose, Evan Martin, Evan Prodromou, Evert, fantasai, Felix Sasaki, Francesco Schwarz, Francis Brosnan Blazquez, Franck 'Shift' Quélain, Frank Barchard, 鶴飼文敏 (Fumitoshi Ukai), Futomi Hatano, Gavin Carothers, Gavin Kistner, Gareth Rees, Garrett Smith, Geoff Richards, Geoffrey Garen, Geoffrey Sneddon, George Lund, Gianmarco Armellin, Giovanni Campagna, Giuseppe Pascale, Glenn Adams, Glenn Maynard, Graham Klyne, Greg Botten, Greg Houston, Greg Wilkins, Gregg Tavares, Gregory J. Rosmaita, Grey, Guilherme Johansson Tramontina, Gytis Jakutonis, Hákon Wium Lie, Hallvard Reiari Michaelsen Steen, Hans S. Tømmerhalt, Hans Stimer, Harald Alvestrand, Henri Sivonen, Henrik Lied, Henry Mason, Hugh Guiney, Hugh Winkler, Ian Bickling, Ian Clelland, Ian Davis, Ian Fette, Ido Green, Ignacio Javier, Ivan Enderlin, Ivo Emanuel Gonçalves, J. King, Jacob Davies, Jacques Distler, Jake Verbaten, James Craig, James Graham, James Justin Harrell, James Kozianski, James M Snell, James Perrett, James Robinson, Jamie Lokier, Janusz Majnert, Jan-Klaas Kollhof, Jason Duell, Jason Kersey, Jason Lustig, Jason White, Jasper Bryant-Greene, Jatinder Mann, Jed Hartman, Jeff Balogh, Jeff Cuttsinger, Jeff Schiller, Jeff Walden, Jeffrey Zeldman, 胡慧鋒 (Jennifer Braithwaite), Jens Bannmann, Jens Fendler, Jens Lindström, Jens Meiert, Jeremy Hustman, Jeremy Keith, Jeremy Orlow, Jeroen van der Meer, Jian Li, Jim Jewett, Jim Ley, Jim Meehan, Jirka Kosek, Jigod Jiang, João Eiras, Joe Clark, Joe Gregorio, Joel Spolsky, Johan Herland, John Boyer, John Bussjaeger, John Carpenter, John Daggett, John Fallows, John Foliot, John Harding, John Keiser, John Snyders, John Stockton, John-Mark Bell, Johnny Stenback, Jon Ferraiolo, Jon Gibbins, Jon Perlow, Jonas Sicking, Jonathan Cook, Jonathan Rees, Jonathan Worent, Jonny Axelsson, Jordan Tucker, Jorgen Horstink, Jorunn Danielsen Newth, Joseph Kesselman, Joseph Mansfield, Joseph Pecoraro, Josh Aas, Josh Hart, Josh Levenberg, Joshua Bell, Joshua Randall, Jukka K. Korpiela, Jules Clément-Ripoche, Julian Reschke, Jürgen Jeka, Justin Lebar, Justin Novosad, Justin Schuh, Justin Sinclair, Kai Hendry, 呂康豪 (KangHao Lu), Kartikaya Gupta, Kathy Walton, Kelly Ford, Kelly Norton, Kevin Benson, Kevin Gadd, Kevin Cole, Kornél Pál, Kornel Lesinski, Kris Northfield, Kristof Zelechovski, Krzysztof Maczyński, 黒澤剛志 (Kurosawa Takeshi), Kyle Barnhart, Kyle Hofmann, Kyle Huey, Léonard Bouchet, Léonie Watson, Lachlan Hunt, Larry Masinter, Larry Page, Lars Gunther, Lars Solberg, Laura Carlson, Laura Granka, Laura L. Carlson, Laura Wisewell, Laurens Holst, Lee Kowalkowski, Leif Halvard Silli, Lenny Domnitser, Leonard Rosenthal, Leons Petrazickis, Lobotom Dysmon, Logan, Loune, Luke Kenneth Casson Leighton, Maciej Stachowiak, Magnus Kristiansen, Maik Merten, Malcolm Rowe, Manish Tripathi, Mark Birbeck, Mark Davis, Mark Miller, Mark Nottingham, Mark Pilgrim, Mark Rowe, Mark Schenk, Mark Vickers, Mark Wilton-Jones, Martijn Wargers, Martin Atkins, Martin Dürst, Martin Honnen, Martin Janecke, Martin Kutschker, Martin Nilsson, Martin Thomson, Masataka Yakura, Mathias Bynens, Mathieu Henri, Matias Larsson, Matt Schmidt, Matt Wright, Matthew Gregan, Matthew Mastracci, Matthew Raymond, Matthew Thomas, Mattias Waldau, Max Romantschuk, Menno van Slooten, Micah Dubinko, Michael 'Ratt' Iannarelli, Michael A. Nachbaur, Michael A. Puls II, Michael Carter, Michael Daskalov, Michael Day, Michael Enright, Michael Gratton, Michael Nordman, Michael Powers, Michael Rakowski, Michael(tm) Smith, Michal Zalewski, Michel Fortin, Michelangelo De Simone, Michiel van der Blonk, Mihai Şucan, Mihai Parparita, Mike Brown, Mike Dierken, Mike Dixon, Mike Schinkel, Mike Shaver, Mikko Rantalainen, Mohamed Zergaoui, Mounir Lamouri, Ms2ger, NARUSE Yui, Neil Deakin, Neil Rashbrook, Neil Soiffer, Nicholas Shanks, Nicholas Stimpson, Nicholas Zakas, Nickolay Ponomarev, Nicolas Gallagher, Noah Mendelsohn, Noah Slater, Noel Gordon, NoozNooz42, Norbert Lindenbergs, Ojan Vafai, Olaf Hoffmann, Olav Junker Kjær, Oldřich Vetešník, Oli Studholme, Oliver Hunt, Oliver Rigby, Olivier Gendrin, Olli Pettay, oSand, Pablo Flouret, Patrick H. Lauke, Patrik Persson, Paul Adenot, Paul Norman, Per-Erik Brodin, Perry Smith, Peter Beverloo, Peter Karlsson, Peter Kasting, Peter Moulder, Peter Stark, Peter Van der Beken, Peter-Paul Koch, Phil Pickering, Philip Jägenstedt, Philip Taylor, Philip TAYLOR, Philippe De Ryck, Prateek Rungra, Pravir Gupta, 李普君 (Pujun Li), Rachid Finge, Rafał Milecki, Raj Doshi, Rajas Moonka, Ralf Stoltze, Ralph Giles, Raphael Champeimont, Remci Mizkur, Remco, Remy Sharp, Rene Saarsoo, Rene Stach, Ric Hardacre, Rich Clark, Rich Doughty, Richard Ishida, Rigo Wenning, Rikkert Koppes, Rimantas Liubertas, Riona Macnamara, Rob Ennals, Rob Jellinghaus, Rob S, Robert Blaut, Robert Collins, Robert Kieffer, Robert Millan, Robert O'Callahan, Robert Sayre, Robin Berjon, Rodger

Go

NOV JAN FEB

03

2012 2013 2014

[3,445 captures](#)

5 Jun 2004 - 22 Aug 2020

Steve Comstock, Steve Faulkner, Steve Runyon, Steven Bennett, Steven Garrity, Steven Tate, Stewart Brodie, Stuart Ballard, Stuart Langridge, Stuart Parmenter, Subramanian Peruvemba, Sunava Dutta, Susan Borgink, Susan Lesch, Sylvain Pasche, T. J. Crowder, Tab Atkins, Takeshi Yoshino, Tantek Çelik, 田村健人 (TAMURA Kent), Ted Mielczarek, Terrence Wood, Thomas Broyer, Thomas Koetter, Thomas O'Connor, Tim Altman, Tim Johansson, Toby Inkster, Todd Moody, Tom Baker, Tom Pike, Tommy Thorsen, Travis Leithead, Tyler Close, Victor Carbune, Vladimir Katardjiev, Vladimir Vukićević, voracity, Wakaba, Wayne Carr, Wayne Pollock, Wellington Fernando de Macedo, Weston Ruter, Wilhelm Joys Andersen, Will Levine, William Swanson, Wladimir Palant, Wojciech Mach, Wolfram Kriesing, Xan Gregg, Yang Chen, Ye-Kui Wang, Yehuda Katz, Yi-An Huang, Yngve Nysaeter Pettersen, Yonathan Randolph, Yuzo Fujishima, Zhenbin Xu, Zoltan Herczeg, and Øistein E. Andersen, for their useful comments, both large and small, that have led to changes to this specification over the years



Thanks also  
the various 1

Too slow? Try reading the multipage copy of  
the spec instead: [https://web.archive.org/  
/web/20130103030411/http://whatwg.org/html](https://web.archive.org/web/20130103030411/http://whatwg.org/html)

cluding all the contributors to

the canvas feature was

Special than  
designed.

Special than  
contented

hanism,

Thanks to the participants of the microdata usability study for allowing us to use their mistakes as a guide for designing the microdata feature.

Special thanks and \$10,000 to David Hyatt who came up with a broken implementation of the adoption agency algorithm that the editor had to reverse engineer and fix before using it in the parsing section.

Thanks to the many sources that provided inspiration for the examples used in the specification.

The image in the introduction is based on a photo by Wonderlane. (CC BY 2.0)

Thanks also to the Microsoft blogging community for some ideas, to the attendees of the W3C Workshop on Web Applications and Compound Documents for inspiration, to the #mrt crew, the #mrt.no crew, and the #whatwg crew, and to Pillar and Hedral for their ideas and support.