W3C Recommendation

HTML 5.2

W3C Recommendation, 14 December 2017

This version:

https://www.w3.org/TR/2017/REC-html52-20171214/

Latest published version:

https://www.w3.org/TR/html52/

Latest published version of HTML:

https://www.w3.org/TR/html/

Previous version:

https://www.w3.org/TR/2017/PR-html52-20171102/

Editor's Draft:

https://w3c.github.io/html/

Editors:

Steve Faulkner (The Paciello Group)
Arron Eicholz (Microsoft)
Travis Leithead (Microsoft)
Alex Danilo (Google)
Sangwhan Moon (Invited Expert)

Former Editors:

Erika Doyle Navara (Microsoft)
Theresa O'Connor (Apple Inc.)
Robin Berjon (W3C)

Test Suite:

https://w3c-test.org/html/

Implementation Report:

https://w3c.github.io/test-results/html52/implementation-report.html

Participate:

File an issue (open issues)

Others:

Single page version

Errata for this document are recorded as Github issues.
The English version of this specification is the only normative version. Non-normative translations may also be available.
Copyright © 2017 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and document use rules apply.

Abstract

This specification defines the 5th major version, second minor revision of the core language of the World Wide Web: the Hypertext Markup Language (HTML). In this version, new features continue to be introduced to help Web application authors, new elements continue to be introduced based on research into prevailing authoring practices, and special attention continues to be given to defining clear conformance criteria for user agents in an effort to improve interoperability.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This document was published by the Web Platform Working Group as a W3C Recommendation for HTML 5.2 that would obsolete the HTML 5.1 Recommendation.

All interested parties are invited to provide implementation and bug reports and other comments through the Working Group's Issue tracker. These will generally be considered in the development of HTML 5.3.

The implementation report produced for this version demonstrates that in almost every case changes are matched by interoperable implementation.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 March 2017 W3C Process Document.

Table of Contents

1 **Introduction**
1.1 Background
1.2 Audience
1.3 Scope
1.4 History
1.5 Design notes
1.5.1 Serializability of script execution
1.5.2 Compliance with other specifications
1.5.3 Extensibility
1.6 HTML vs XML Syntax
1.7 Structure of this specification
1.7.1 How to read this specification
1.7.2 Typographic conventions
1.8 Privacy concerns
1.9 A quick introduction to HTML
1.9.1 Writing secure applications with HTML
1.9.2 Common pitfalls to avoid when using the scripting APIs
1.9.3 How to catch mistakes when writing HTML: validators and conformance checkers
1.10 Conformance requirements for authors
1.10.1 Presentational markup
1.10.2 Syntax errors
1.10.3 Restrictions on content models and on attribute values

1.11 Suggested reading

2 Common infrastructure

2.1 Terminology

2.1.1 Resources

2.1.2 XML compatibility

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

2.2.2 Dependencies

2.2.3 Extensibility

2.2.4 Interactions with XPath and XSLT

2.3 Case-sensitivity and string comparison

2.4 Common microsyntaxes

2.4.1 Common parser idioms

2.4.2 Boolean attributes

2.4.3 Keywords and enumerated attributes

2.4.4 Numbers

2.4.4.1 Signed integers

2.4.4.2 Non-negative integers

2.4.4.3 Floating-point numbers

2.4.4.4 Percentages and lengths

2.4.4.5 Non-zero percentages and lengths

2.4.4.6 Lists of floating-point numbers

2.4.4.7 Lists of dimensions

2.4.5 Dates and times

2.4.5.1 Months

2.4.5.2 Dates

2.4.5.3 Yearless dates

2.4.5.4 Times

2.4.5.5 Floating dates and times

2.4.5.6 Time zones

2.4.5.7 Global dates and times
2.4.5.8 Weeks
2.4.5.9 Durations
2.4.5.10 Vaguer moments in time
2.4.6 Colors
2.4.7 Space-separated tokens
2.4.8 Comma-separated tokens
2.4.9 References
2.4.10 Media queries
2.5 URLs
2.5.1 Terminology
2.5.2 Parsing URLs
2.5.3 Dynamic changes to base URLs
2.6 Fetching resources
2.6.1 Terminology
2.6.2 Processing model
2.6.3 Encrypted HTTP and related security concerns
2.6.4 Determining the type of a resource
2.6.5 Extracting character encodings from `meta` elements
2.6.6 CORS settings attributes
2.6.7 Referrer policy attributes
2.7 Common DOM interfaces
2.7.1 Reflecting content attributes in IDL attributes
2.7.2 Collections
2.7.2.1 The `HTMLAllCollection` interface
2.7.2.2 The `HTMLFormControlsCollection` interface
2.7.2.3 The `HTMLOptionsCollection` interface
2.7.3 The `DOMStringList` interface
2.7.4 Garbage collection
2.8 Namespaces
2.9 Safe passing of structured data
2.9.1 Serializable objects
2.9.2 Transferable objects
2.9.3 StructuredSerializeInternal (*value* , *forStorage* [, *memory*])

2.9.4 StructuredSerialize (*value*)
2.9.5 StructuredSerializeForStorage (*value*)
2.9.6 StructuredDeserialize (*serialized* , *targetRealm* [, *memory*])
2.9.7 StructuredSerializeWithTransfer (*value* , *transferList*)
2.9.8 StructuredDeserializeWithTransfer (*serializeWithTransferResult* , *targetRealm*)
2.9.9 Performing serialization and transferring from other specifications
2.9.10 Monkey patch for `Blob` and `FileList` objects

3 Semantics, structure, and APIs of HTML documents
3.1 Documents
3.1.1 The `Document` object
3.1.2 Resource metadata management
3.1.3 DOM tree accessors
3.2 Elements
3.2.1 Semantics
3.2.2 Elements in the DOM
3.2.3 Element definitions
3.2.3.1 Attributes
3.2.4 Content models
3.2.4.1 The "nothing" content model
3.2.4.2 Kinds of content
3.2.4.2.1 Metadata content
3.2.4.2.2 Flow content
3.2.4.2.3 Sectioning content
3.2.4.2.4 Heading content
3.2.4.2.5 Phrasing content
3.2.4.2.6 Embedded content
3.2.4.2.7 Interactive content
3.2.4.2.8 Palpable content
3.2.4.2.9 Script-supporting elements
3.2.4.3 Transparent content models
3.2.4.4 Paragraphs
3.2.5 Global attributes
3.2.5.1 The `title` attribute
3.2.5.2 The `lang` and `xml:lang` attributes

3.2.5.3 The `translate` attribute
3.2.5.4 The `xml:base` attribute (XML only)
3.2.5.5 The `dir` attribute
3.2.5.6 The `style` attribute
3.2.5.7 Embedding custom non-visible data with the `data-*` attributes
3.2.6 The `innerText` IDL attribute
3.2.7 Requirements relating to the bidirectional algorithm
3.2.7.1 Authoring conformance criteria for bidirectional-algorithm formatting characters
3.2.7.2 User agent conformance criteria
3.2.8 WAI-ARIA and HTML Accessibility API Mappings
3.2.8.1 ARIA Authoring Requirements
3.2.8.2 Conformance Checker Implementation Requirements
3.2.8.3 User Agent Implementation Requirements
3.2.8.3.1 ARIA Role Attribute
3.2.8.3.2 State and Property Attributes
3.2.8.4 Allowed ARIA roles, states and properties

4 The elements of HTML
4.1 The document 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
4.2.4.1 Processing the `media` attribute
4.2.4.2 Processing the `type` attribute
4.2.4.3 Obtaining a resource from a `link` element
4.2.4.4 Processing `Link` headers
4.2.4.5 Providing users with a means to follow hyperlinks created using the `link` element
4.2.4.6 The `LinkStyle` interface
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
4.2.7 Interactions of styling and scripting
4.3 Sections
4.3.1 The `body` element
4.3.2 The `article` element
4.3.3 The `section` element
4.3.4 The `nav` element
4.3.5 The `aside` element
4.3.6 The `h1`, `h2`, `h3`, `h4`, `h5`, and `h6` elements
4.3.7 The `header` element
4.3.8 The `footer` element
4.3.9 Headings and sections
4.3.9.1 Creating an outline
4.3.10 Usage summary
4.3.10.1 Article or section?
4.4 Grouping content
4.4.1 The `p` element
4.4.2 The `address` element
4.4.3 The `hr` element
4.4.4 The `pre` element
4.4.5 The `blockquote` element
4.4.6 The `ol` element
4.4.7 The `ul` element
4.4.8 The `li` element
4.4.9 The `dl` element
4.4.10 The `dt` element
4.4.11 The `dd` element
4.4.12 The `figure` element
4.4.13 The `figcaption` element
4.4.14 The `main` element
4.4.15 The `div` element

4.5 Text-level semantics
4.5.1 The `a` element
4.5.2 The `em` element
4.5.3 The `strong` element
4.5.4 The `small` element
4.5.5 The `s` element
4.5.6 The `cite` element
4.5.7 The `q` element
4.5.8 The `dfn` element
4.5.9 The `abbr` element
4.5.10 The `ruby` element
4.5.11 The `rb` element
4.5.12 The `rt` element
4.5.13 The `rtc` element
4.5.14 The `rp` element
4.5.15 The `data` element
4.5.16 The `time` element
4.5.17 The `code` element
4.5.18 The `var` element
4.5.19 The `samp` element
4.5.20 The `kbd` element
4.5.21 The `sub` and `sup` elements
4.5.22 The `i` element
4.5.23 The `b` element
4.5.24 The `u` element
4.5.25 The `mark` element
4.5.26 The `bdi` element
4.5.27 The `bdo` element
4.5.28 The `span` element
4.5.29 The `br` element
4.5.30 The `wbr` element
4.5.31 Usage summary
4.6 Edits

4.6.1 The `ins` element
4.6.2 The `del` element
4.6.3 Attributes common to `ins` and `del` elements
4.6.4 Edits and paragraphs
4.6.5 Edits and lists
4.6.6 Edits and tables
4.7 Embedded content
4.7.1 Introduction
4.7.2 Dependencies
4.7.3 The `picture` element
4.7.4 The `source` element
4.7.5 The `img` element
4.7.5.1 Requirements for providing text to act as an alternative for images
4.7.5.1.1 Examples of scenarios where users benefit from text alternatives for images
4.7.5.1.2 General guidelines
4.7.5.1.3 A link or button containing nothing but an image
4.7.5.1.4 Graphical Representations: Charts, diagrams, graphs, maps, illustrations
4.7.5.1.5 Images of text
4.7.5.1.6 Images that include text
4.7.5.1.7 Images that enhance the themes or subject matter of the page content
4.7.5.1.8 A graphical representation of some of the surrounding text
4.7.5.1.9 A purely decorative image that doesn’t add any information
4.7.5.1.10 Inline images
4.7.5.1.11 A group of images that form a single larger picture with no links
4.7.5.1.12 Image maps
4.7.5.1.13 A group of images that form a single larger picture with links
4.7.5.1.14 Images of Pictures
4.7.5.1.15 Webcam images
4.7.5.1.16 When a text alternative is not available at the time of publication
4.7.5.1.17 An image not intended for the user
4.7.5.1.18 Icon Images
4.7.5.1.19 Logos, insignia, flags, or emblems
4.7.5.1.20 CAPTCHA Images
4.7.5.1.21 An image in a `picture` element

4.7.5.1.22 Guidance for markup generators

4.7.5.1.23 Guidance for conformance checkers

4.7.6 The `iframe` element

4.7.7 The `embed` element

4.7.8 The `object` element

4.7.9 The `param` element

4.7.10 The `video` element

4.7.11 The `audio` element

4.7.12 The `track` element

4.7.13 Media elements

4.7.13.1 Error codes

4.7.13.2 Location of the media resource

4.7.13.3 MIME types

4.7.13.4 Network states

4.7.13.5 Loading the media resource

4.7.13.6 Offsets into the media resource

4.7.13.7 Ready states

4.7.13.8 Playing the media resource

4.7.13.9 Seeking

4.7.13.10 Media resources with multiple media tracks

4.7.13.10.1 `AudioTrackList` and `VideoTrackList` objects

4.7.13.10.2 Selecting specific audio and video tracks declaratively

4.7.13.11 Timed text tracks

4.7.13.11.1 Text track model

4.7.13.11.2 Sourcing in-band text tracks

4.7.13.11.3 Sourcing out-of-band text tracks

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

4.7.13.11.5 Text track API

4.7.13.11.6 Text tracks exposing in-band metadata

4.7.13.11.7 Text tracks describing chapters

4.7.13.11.8 Event handlers for objects of the text track APIs

4.7.13.11.9 Best practices for metadata text tracks

4.7.13.12 Identifying a track kind through a URL

4.7.13.13 User interface

4.7.13.14 Time ranges
4.7.13.15 The `TrackEvent` interface
4.7.13.16 Event summary
4.7.13.17 Security and privacy considerations
4.7.13.18 Best practices for authors using media elements
4.7.13.19 Best practices for implementors of media elements
4.7.14 The `map` element
4.7.15 The `area` element
4.7.16 Image maps
4.7.16.1 Authoring
4.7.16.2 Processing model
4.7.17 MathML
4.7.18 SVG
4.7.19 Dimension attributes
4.8 Links
4.8.1 Introduction
4.8.2 Links created by `a` and `area` elements
4.8.3 API for `a` and `area` elements
4.8.4 Following hyperlinks
4.8.5 Downloading resources
4.8.6 Link types
4.8.6.1 Link type "`alternate`"
4.8.6.2 Link type "`author`"
4.8.6.3 Link type "`bookmark`"
4.8.6.4 Link type "`help`"
4.8.6.5 Link type "`icon`"
4.8.6.6 Link type "`license`"
4.8.6.7 Link type "`nofollow`"
4.8.6.8 Link type "`noopener`"
4.8.6.9 Link type "`noreferrer`"
4.8.6.10 Link type "`search`"
4.8.6.11 Link type "`stylesheet`"
4.8.6.12 Link type "`tag`"
4.8.6.13 Sequential link types

4.8.6.13.1 Link type "`next`"
4.8.6.13.2 Link type "`prev`"
4.8.6.14 Other link types
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 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
4.10.1.5 Enabling client-side automatic filling of form controls
4.10.1.6 The difference between the field type, the autofill field name, and the input modality
4.10.1.7 Date, time, and number formats
4.10.2 Categories
4.10.3 The `form` element
4.10.4 The `label` element
4.10.5 The `input` element

4.10.5.1 States of the `type` attribute

4.10.5.1.1 `Hidden` state (`type=hidden`)

4.10.5.1.2 `Text` (`type=text`) state and `Search` state (`type=search`)

4.10.5.1.3 `Telephone` state (`type=tel`)

4.10.5.1.4 `URL` state (`type=url`)

4.10.5.1.5 `E-mail` state (`type=email`)

4.10.5.1.6 `Password` state (`type=password`)

4.10.5.1.7 `Date` state (`type=date`)

4.10.5.1.8 `Month` state (`type=month`)

4.10.5.1.9 `Week` state (`type=week`)

4.10.5.1.10 `Time` state (`type=time`)

4.10.5.1.11 `Local Date and Time` state (`type=datetime-local`)

4.10.5.1.12 `Number` state (`type=number`)

4.10.5.1.13 `Range` state (`type=range`)

4.10.5.1.14 `Color` state (`type=color`)

4.10.5.1.15 `Checkbox` state (`type=checkbox`)

4.10.5.1.16 `Radio Button` state (`type=radio`)

4.10.5.1.17 `File Upload` state (`type=file`)

4.10.5.1.18 `Submit Button` state (`type=submit`)

4.10.5.1.19 `Image Button` state (`type=image`)

4.10.5.1.20 `Reset Button` state (`type=reset`)

4.10.5.1.21 `Button` state (`type=button`)

4.10.5.2 Implementation notes regarding localization of form controls

4.10.5.3 Common `input` element attributes

4.10.5.3.1 The `maxlength` and `minlength` attributes

4.10.5.3.2 The `size` attribute

4.10.5.3.3 The `readonly` attribute

4.10.5.3.4 The `required` attribute

4.10.5.3.5 The `multiple` attribute

4.10.5.3.6 The `pattern` attribute

4.10.5.3.7 The `min` and `max` attributes

4.10.5.3.8 The `step` attribute

4.10.5.3.9 The `list` attribute

4.10.5.3.10 The `placeholder` attribute

4.10.5.4 Common `input` element APIs
4.10.5.5 Common event behaviors
4.10.6 The `button` element
4.10.7 The `select` element
4.10.8 The `datalist` element
4.10.9 The `optgroup` element
4.10.10 The `option` element
4.10.11 The `textarea` element
4.10.12 The `output` element
4.10.13 The `progress` element
4.10.14 The `meter` element
4.10.15 The `fieldset` element
4.10.16 The `legend` element
4.10.17 Form control infrastructure
4.10.17.1 A form control value
4.10.17.2 Mutability
4.10.17.3 Association of controls and forms
4.10.18 Attributes common to form controls
4.10.18.1 Naming form controls: the `name` attribute
4.10.18.2 Submitting element directionality: the `dirname` attribute
4.10.18.3 Limiting user input length: the `maxlength` attribute
4.10.18.4 Setting minimum input length requirements: the `minlength` attribute
4.10.18.5 Enabling and disabling form controls: the `disabled` attribute
4.10.18.6 Form submission
4.10.18.6.1 Autofocusing a form control: the `autofocus` attribute
4.10.18.7 Autofill
4.10.18.7.1 Autofilling form controls: the `autocomplete` attribute
4.10.18.7.2 Processing model
4.10.19 APIs for text field selections
4.10.20 Constraints
4.10.20.1 Definitions
4.10.20.2 Constraint validation
4.10.20.3 The constraint validation API
4.10.20.4 Security

4.10.21 Form submission
4.10.21.1 Introduction
4.10.21.2 Implicit submission
4.10.21.3 Form submission algorithm
4.10.21.4 Constructing the form data set
4.10.21.5 Selecting a form submission encoding
4.10.21.6 URL-encoded form data
4.10.21.7 Multipart form data
4.10.21.8 Plain text form data
4.10.22 Resetting a form
4.11 Interactive elements
4.11.1 The `details` element
4.11.2 The `summary` element
4.11.3 Commands
4.11.3.1 Facets
4.11.3.2 Using the `a` element to define a command
4.11.3.3 Using the `button` element to define a command
4.11.3.4 Using the `input` element to define a command
4.11.3.5 Using the `option` element to define a command
4.11.3.6 Using the `accesskey` attribute on a `label` element to define a command
4.11.3.7 Using the `accesskey` attribute on a `legend` element to define a command
4.11.3.8 Using the `accesskey` attribute to define a command on other elements
4.11.4 The `dialog` element
4.11.4.1 Anchor points
4.12 Scripting
4.12.1 The `script` element
4.12.1.1 Processing model
4.12.1.2 Scripting languages
4.12.1.3 Restrictions for contents of `script` elements
4.12.1.4 Inline documentation for external scripts
4.12.1.5 Interaction of `script` elements and XSLT
4.12.2 The `noscript` element
4.12.3 The `template` element
4.12.3.1 Interaction of `template` elements with XSLT and XPath

4.12.4 The `canvas` element
4.12.4.1 Color spaces and color correction
4.12.4.2 Serializing bitmaps to a file
4.12.4.3 Security with `canvas` elements
4.13 Common idioms without dedicated elements
4.13.1 Subheadings, subtitles, alternative titles and taglines
4.13.2 Bread crumb navigation
4.13.3 Tag clouds
4.13.4 Conversations
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 User interaction
5.1 The `hidden` attribute
5.2 Inert subtrees
5.3 Activation
5.4 Focus
5.4.1 Introduction
5.4.2 Data model
5.4.3 The `tabindex` attribute
5.4.4 Processing model
5.4.5 Sequential focus navigation
5.4.6 Focus management APIs
5.4.7 Clipboard actions and focus
5.5 Assigning keyboard shortcuts
5.5.1 Introduction
5.5.2 The `accesskey` attribute
5.5.3 Processing model
5.6 Editing
5.6.1 Making document regions editable: The `contenteditable` content attribute
5.6.2 Making entire documents editable: The `designMode` IDL attribute

5.6.3 Best practices for in-page editors
5.6.4 Editing APIs
5.6.5 Spelling and grammar checking
5.7 Drag and drop
5.7.1 Introduction
5.7.2 The drag data store
5.7.3 The `DataTransfer` interface
5.7.3.1 The `DataTransferItemList` interface
5.7.3.2 The `DataTransferItem` interface
5.7.4 The `DragEvent` interface
5.7.5 Drag-and-drop processing model
5.7.6 Events summary
5.7.7 The `draggable` attribute
5.7.8 Security risks in the drag-and-drop model

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 Security
6.1.4 Groupings of browsing contexts
6.1.5 Browsing context names
6.1.6 Script settings for browsing contexts
6.2 Security infrastructure for `Window`, `WindowProxy`, and `Location` objects
6.2.1 Integration with IDL
6.2.2 Shared internal slot: [[CrossOriginPropertyDescriptorMap]]
6.2.3 Shared abstract operations
6.2.3.1 CrossOriginProperties (*O*)
6.2.3.2 IsPlatformObjectSameOrigin (*O*)
6.2.3.3 CrossOriginGetOwnPropertyHelper (*O* , *P*)
6.2.3.3.1 CrossOriginPropertyDescriptor (*crossOriginProperty* , *originalDesc*)
6.2.3.3.2 CrossOriginFunctionWrapper (*needsWrapping* , *functionToWrap*)
6.2.3.4 CrossOriginGet (*O* , *P* , *Receiver*)

6.2.3.5 CrossOriginSet (*O* , *P* , V, *Receiver*)
6.2.3.6 CrossOriginOwnPropertyKeys (*O*)
6.3 The `Window` object
6.3.1 APIs for creating and navigating browsing contexts by name
6.3.2 Accessing other browsing contexts
6.3.3 Named access on the `Window` object
6.3.4 Garbage collection and browsing contexts
6.3.5 Closing browsing contexts
6.3.6 Browser interface elements
6.3.7 The `WindowProxy` object
6.3.7.1 The `WindowProxy` internal methods
6.3.7.1.1 [[GetPrototypeOf]] ( )
6.3.7.1.2 [[SetPrototypeOf]] (V)
6.3.7.1.3 [[IsExtensible]] ( )
6.3.7.1.4 [[PreventExtensions]] ( )
6.3.7.1.5 [[GetOwnProperty]] (*P*)
6.3.7.1.6 [[DefineOwnProperty]] (*P* , *Desc*)
6.3.7.1.7 [[Get]] (*P* , *Receiver*)
6.3.7.1.8 [[Set]] (*P* , V, *Receiver*)
6.3.7.1.9 [[Delete]] (*P*)
6.3.7.1.10 [[OwnPropertyKeys]] ( )
6.4 Origin
6.4.1 Relaxing the same-origin restriction
6.5 Sandboxing
6.6 Session history and navigation
6.6.1 The session history of browsing contexts
6.6.2 The `History` interface
6.6.3 Implementation notes for session history
6.6.4 The `Location` interface
6.6.4.1 The `Location` internal methods
6.6.4.1.1 [[GetPrototypeOf]] ( )
6.6.4.1.2 [[SetPrototypeOf]] (*V*)
6.6.4.1.3 [[IsExtensible]] ( )
6.6.4.1.4 [[PreventExtensions]] ( )

6.6.4.1.5 [[GetOwnProperty]] (*P*)
6.6.4.1.6 [[DefineOwnProperty]] (*P* , *Desc*)
6.6.4.1.7 [[Get]] (*P* , *Receiver*)
6.6.4.1.8 [[Set]] (*P* , *V* , *Receiver*)
6.6.4.1.9 [[Delete]] (*P*)
6.6.4.1.10 [[OwnPropertyKeys]] ( )
6.7 Browsing the Web
6.7.1 Navigating across documents
6.7.2 Page load processing model for HTML files
6.7.3 Page load processing model for XML files
6.7.4 Page load processing model for text files
6.7.5 Page load processing model for `multipart/x-mixed-replace` resources
6.7.6 Page load processing model for media
6.7.7 Page load processing model for content that uses plugins
6.7.8 Page load processing model for inline content that doesn’t have a DOM
6.7.9 Navigating to a fragment
6.7.10 History traversal
6.7.10.1 Persisted user state restoration
6.7.10.2 The `PopStateEvent` interface
6.7.10.3 The `HashChangeEvent` interface
6.7.10.4 The `PageTransitionEvent` interface
6.7.11 Unloading documents
6.7.11.1 The `BeforeUnloadEvent` interface
6.7.12 Aborting a document load
6.7.13 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 Fetching scripts
7.1.3.3 Creating scripts
7.1.3.4 Calling scripts

7.1.3.5 Realms, settings objects, and global objects
7.1.3.5.1 Entry
7.1.3.5.2 Incumbent
7.1.3.5.3 Current
7.1.3.5.4 Relevant
7.1.3.6 Killing scripts
7.1.3.7 Integration with the JavaScript job queue
7.1.3.7.1 EnqueueJob(*queueName* , *job* , *arguments*)
7.1.3.8 Integration with the JavaScript module system
7.1.3.8.1 HostResolveImportedModule(*referencingModule* , *specifier*)
7.1.3.9 Runtime script errors
7.1.3.9.1 Runtime script errors in documents
7.1.3.9.2 The `ErrorEvent` interface
7.1.3.10 Unhandled promise rejections
7.1.3.10.1 The HostPromiseRejectionTracker implementation
7.1.3.10.2 The `PromiseRejectionEvent` interface
7.1.3.11 HostEnsureCanCompileStrings(*callerRealm* , *calleeRealm*)
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 Events
7.1.5.1 Event handlers
7.1.5.2 Event handlers on elements, `Document` objects, and `Window` objects
7.1.5.2.1 IDL definitions
7.1.5.3 Event firing
7.1.5.4 Events and the `Window` object
7.2 The WindowOrWorkerGlobalScope mixin
7.3 Base64 utility methods
7.4 Dynamic markup insertion
7.4.1 Opening the input stream
7.4.2 Closing the input stream
7.4.3 `document.write()`
7.4.4 `document.writeln()`

7.5 Timers
7.6 User prompts
7.6.1 Simple dialogs
7.6.2 Printing
7.7 System state and capabilities
7.7.1 The `Navigator` object
7.7.1.1 Client identification
7.7.1.2 Language preferences
7.7.1.3 Custom scheme and content handlers: the `registerProtocolHandler()` and `registerContentHandler()` methods
7.7.1.3.1 Security and privacy
7.7.1.3.2 Sample user interface
7.7.1.4 Cookies
7.8 Images
7.9 Animation Frames

8 The HTML syntax
8.1 Writing HTML documents
8.1.1 The DOCTYPE
8.1.2 Elements
8.1.2.1 Start tags
8.1.2.2 End tags
8.1.2.3 Attributes
8.1.2.4 Optional tags
8.1.2.5 Restrictions on content models
8.1.2.6 Restrictions on the contents of raw text and escapable raw text elements
8.1.3 Text
8.1.3.1 Newlines
8.1.4 Character references
8.1.5 CDATA sections
8.1.6 Comments
8.2 Parsing HTML documents
8.2.1 Overview of the parsing model
8.2.2 The input byte stream
8.2.2.1 Parsing with a known character encoding

8.2.2.2 Determining the character encoding
8.2.2.3 Character encodings
8.2.2.4 Changing the encoding while parsing
8.2.2.5 Preprocessing the input stream
8.2.3 Parse state
8.2.3.1 The insertion mode
8.2.3.2 The stack of open elements
8.2.3.3 The list of active formatting elements
8.2.3.4 The element pointers
8.2.3.5 Other parsing state flags
8.2.4 Tokenization
8.2.4.1 Data state
8.2.4.2 RCDATA state
8.2.4.3 RAWTEXT state
8.2.4.4 Script data state
8.2.4.5 PLAINTEXT state
8.2.4.6 Tag open state
8.2.4.7 End tag open state
8.2.4.8 Tag name state
8.2.4.9 RCDATA less-than sign state
8.2.4.10 RCDATA end tag open state
8.2.4.11 RCDATA end tag name state
8.2.4.12 RAWTEXT less-than sign state
8.2.4.13 RAWTEXT end tag open state
8.2.4.14 RAWTEXT end tag name state
8.2.4.15 Script data less-than sign state
8.2.4.16 Script data end tag open state
8.2.4.17 Script data end tag name state
8.2.4.18 Script data escape start state
8.2.4.19 Script data escape start dash state
8.2.4.20 Script data escaped state
8.2.4.21 Script data escaped dash state
8.2.4.22 Script data escaped dash dash state
8.2.4.23 Script data escaped less-than sign state

8.2.4.24	Script data escaped end tag open state
8.2.4.25	Script data escaped end tag name state
8.2.4.26	Script data double escape start state
8.2.4.27	Script data double escaped state
8.2.4.28	Script data double escaped dash state
8.2.4.29	Script data double escaped dash dash state
8.2.4.30	Script data double escaped less-than sign state
8.2.4.31	Script data double escape end state
8.2.4.32	Before attribute name state
8.2.4.33	Attribute name state
8.2.4.34	After attribute name state
8.2.4.35	Before attribute value state
8.2.4.36	Attribute value (double-quoted) state
8.2.4.37	Attribute value (single-quoted) state
8.2.4.38	Attribute value (unquoted) state
8.2.4.39	After attribute value (quoted) state
8.2.4.40	Self-closing start tag state
8.2.4.41	Bogus comment state
8.2.4.42	Markup declaration open state
8.2.4.43	Comment start state
8.2.4.44	Comment start dash state
8.2.4.45	Comment state
8.2.4.46	Comment less-than sign state
8.2.4.47	Comment less-than sign bang state
8.2.4.48	Comment less-than sign bang dash state
8.2.4.49	Comment less-than sign bang dash dash state
8.2.4.50	Comment end dash state
8.2.4.51	Comment end state
8.2.4.52	Comment end bang state
8.2.4.53	DOCTYPE state
8.2.4.54	Before DOCTYPE name state
8.2.4.55	DOCTYPE name state
8.2.4.56	After DOCTYPE name state
8.2.4.57	After DOCTYPE public keyword state

8.2.4.58 Before DOCTYPE public identifier state
8.2.4.59 DOCTYPE public identifier (double-quoted) state
8.2.4.60 DOCTYPE public identifier (single-quoted) state
8.2.4.61 After DOCTYPE public identifier state
8.2.4.62 Between DOCTYPE public and system identifiers state
8.2.4.63 After DOCTYPE system keyword state
8.2.4.64 Before DOCTYPE system identifier state
8.2.4.65 DOCTYPE system identifier (double-quoted) state
8.2.4.66 DOCTYPE system identifier (single-quoted) state
8.2.4.67 After DOCTYPE system identifier state
8.2.4.68 Bogus DOCTYPE state
8.2.4.69 CDATA section state
8.2.4.70 CDATA section bracket state
8.2.4.71 CDATA section end state
8.2.4.72 Character reference state
8.2.4.73 Numeric character reference state
8.2.4.74 Hexadecimal character reference start state
8.2.4.75 Decimal character reference start state
8.2.4.76 Hexadecimal character reference state
8.2.4.77 Decimal character reference state
8.2.4.78 Numeric character reference end state
8.2.4.79 Character reference end state
8.2.5 Tree construction
8.2.5.1 Creating and inserting nodes
8.2.5.2 Parsing elements that contain only text
8.2.5.3 Closing elements that have implied end tags
8.2.5.4 The rules for parsing tokens in HTML content
8.2.5.4.1 The "initial" insertion mode
8.2.5.4.2 The "before html" insertion mode
8.2.5.4.3 The "before head" insertion mode
8.2.5.4.4 The "in head" insertion mode
8.2.5.4.5 The "in head noscript" insertion mode
8.2.5.4.6 The "after head" insertion mode
8.2.5.4.7 The "in body" insertion mode

8.2.5.4.8 The "text" insertion mode
8.2.5.4.9 The "in table" insertion mode
8.2.5.4.10 The "in table text" insertion mode
8.2.5.4.11 The "in caption" insertion mode
8.2.5.4.12 The "in column group" insertion mode
8.2.5.4.13 The "in table body" insertion mode
8.2.5.4.14 The "in row" insertion mode
8.2.5.4.15 The "in cell" insertion mode
8.2.5.4.16 The "in select" insertion mode
8.2.5.4.17 The "in select in table" insertion mode
8.2.5.4.18 The "in template" insertion mode
8.2.5.4.19 The "after body" insertion mode
8.2.5.4.20 The "in frameset" insertion mode
8.2.5.4.21 The "after frameset" insertion mode
8.2.5.4.22 The "after after body" insertion mode
8.2.5.4.23 The "after after frameset" insertion mode
8.2.5.5 The rules for parsing tokens in foreign content
8.2.6 The end
8.2.7 Coercing an HTML DOM into an infoset
8.2.8 An introduction to error handling and strange cases in the parser
8.2.8.1 Misnested tags: <b><i></b></i>
8.2.8.2 Misnested tags: <b><p></b></p>
8.2.8.3 Unexpected markup in tables
8.2.8.4 Scripts that modify the page as it is being parsed
8.2.8.5 The execution of scripts that are moving across multiple documents
8.2.8.6 Unclosed formatting elements
8.3 Serializing HTML fragments
8.4 Parsing HTML fragments
8.5 Named character references

9 The XML syntax
9.1 Writing documents in the XML syntax
9.2 Parsing XML documents
9.3 Serializing XML fragments
9.4 Parsing XML fragments

10 Rendering
10.1 Introduction
10.2 The CSS user agent style sheet and presentational hints
10.3 Non-replaced elements
10.3.1 Hidden elements
10.3.2 The page
10.3.3 Flow content
10.3.4 Phrasing content
10.3.5 Bidirectional text
10.3.6 Quotes
10.3.7 Sections and headings
10.3.8 Lists
10.3.9 Tables
10.3.10 Margin collapsing quirks
10.3.11 Form controls
10.3.12 The `hr` element
10.3.13 The `fieldset` and `legend` elements
10.4 Replaced elements
10.4.1 Embedded content
10.4.2 Images
10.4.3 Attributes for embedded content and images
10.4.4 Image maps
10.5 Widgets
10.5.1 Introduction
10.5.2 The `button` element
10.5.3 The `details` and `summary` elements
10.5.4 The `input` element as a text entry widget.
10.5.5 The `input` element as domain-specific widgets
10.5.6 The `input` element as a range control
10.5.7 The `input` element as a color well
10.5.8 The `input` element as a checkbox and radio button widgets
10.5.9 The `input` element as a file upload control
10.5.10 The `input` element as a button
10.5.11 The `marquee` element

10.5.12 The `meter` element
10.5.13 The `progress` element
10.5.14 The `select` element
10.5.15 The `textarea` element
10.6 Frames and framesets
10.7 Interactive media
10.7.1 Links, forms, and navigation
10.7.2 The `title` attribute
10.7.3 Editing hosts
10.7.4 Text rendered in native user interfaces
10.8 Print media
10.9 Unstyled XML documents

11 Obsolete features
11.1 Obsolete but conforming features
11.1.1 Warnings for obsolete but conforming features
11.2 Non-conforming features
11.3 Requirements for implementations
11.3.1 The `applet` element
11.3.2 The `marquee` element
11.3.3 Frames
11.3.4 Other elements, attributes and APIs
11.3.4.1 Plugins

12 IANA considerations
12.1 `text/html`
12.2 `multipart/x-mixed-replace`
12.3 `application/xhtml+xml`
12.4 `web+` scheme prefix

Index
Terms defined by this specification
Terms defined by reference
Elements
Element content categories

Attributes

Element Interfaces

Events

Property Index

IDL Index

References

Normative References

Informative References

Changes

New features

Features removed

Fixing bugs and matching reality better

New concepts

Editorial clarifications

Acknowledgements

People who have contributed to this version of HTML

People who have contributed to previous revisions of HTML 5.x