

Designing Web Applications

HTML 5



Francesco Marcelloni

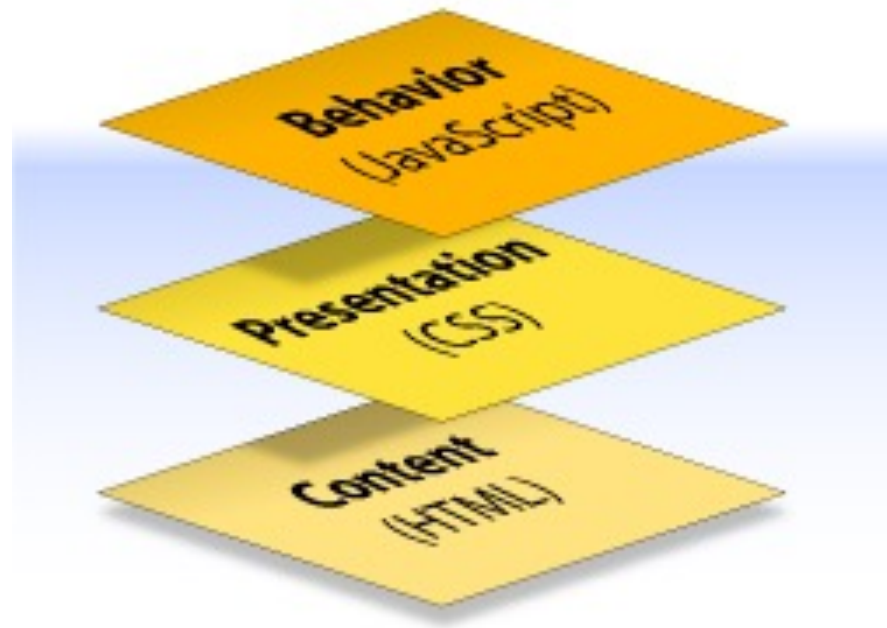
Department of Information Engineering
University of Pisa
ITALY

World Wide Web

- **World Wide Web (WWW)**: network of information resources.
- **Information resources**: documents or pages which can contain link to other documents or pages with correlated information
- Documents with only text: **Hypertext**
- Documents with also images, voice and so on: **Hypermedia**
- Users access documents, independently of their location, through interactive programs, called **browsers**, or other multimedia programs.
- Such programs allow **transferring** these documents, **showing** the contents, **navigating** through the hypertext and hypermedia network.

Web documents

- A web document can consist of up to three layers
 - Content
 - Presentation
 - Behaviour



Content Layer

- The *content layer* is always present.
- It comprises the information the author wishes to convey to his or her audience, and is embedded within HTML markup that defines its structure and semantics.
- Most of the content on the Web today is text, images, animations, sound, video, and whatever else an author wants to publish.

Presentation Layer

- The *presentation layer* defines how the content will appear to a human being who accesses the document in one way or another.
- The conventional way to view a web page is with a regular web browser, of course, but that is only one of many possible access methods.
 - For example, content can also be converted to synthetic speech for users who have impaired vision or reading difficulties.

Behavior Layer

- The *behavior layer* involves real-time user interaction with the document.
- This task is normally handled by JavaScript.
- The interaction can be anything from a trivial validation that ensures a required field is filled in before an order form can be submitted, to sophisticated web applications that work much like ordinary desktop programs.

Basic concept

- HTML -> structure and content
- CSS -> style and appearance
- Javascript -> behavior

Example:

```
<h2>Via Diotisalvi 2</h2>
```

```
!!!
```

```
<h2>Address</h2>
```

```
<p>Via Diotisalvi 2</p>
```

```
<font size="3" color="red">This is some text!</font>
```

Use CSS to define the font face, font size, and font color of text.

WWW – Three basic mechanisms

- WWW relies on three basic mechanisms:
 - A **uniform naming scheme** for locating resources on the Web (e.g., URIs)
 - **Protocols**, for access to named resources over the Web (e.g., HTTP), followed by “://”
 - **Hypertext**, for easy navigation among the resources (e.g., HTML)

Uniform Resource Identifier (URI)

- Each information resource on the Web has an address which can be codified by a **URI**.

- A URI is defined as

<scheme>: <scheme-specific-part>

- The <scheme> is the **name** of the mechanism used to access the resource (for instance, http protocol)

- A colon character (:)
- A **scheme-specific part**.

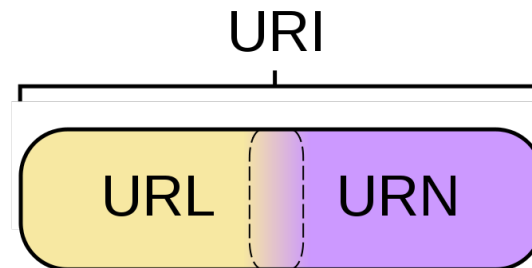
For instance

- The **name of the machine** hosting the resource (for instance, www.w3.org)
- The **name of the resource itself**, given as path

Universal Resource Identifier (URI)

- A **URI** may be a locator (URL) or a name (URN), or both.
- A Uniform Resource Locator (URL) is a URI that, in addition to identifying a resource, specifies the means of acting upon and obtaining the representation.

For example, the URL `http://www.w3.org/TR`



Universal Resource Identifier (URI)

- A Uniform Resource Name (URN) is a URI that identifies a resource by name, in a particular namespace. The resource does not need to necessarily be network homed.

For example, the URN *urn:isbn:0-395-36341-1* is a URI that specifies the identifier system, i.e. International Standard Book Number (ISBN), as well as the unique reference within that system and allows one to talk about a book, but does not suggest where and how to obtain an actual copy of it.

- URN - person's name
- URL – person`s street address.

Universal Resource Identifier (URI)

- **Fragment Identifier**

Some URIs refer to a location within a resource. The URI ends with “#” followed by an anchor identifier (called the *fragment identifier*).

`http://somesite.com/html/top.html#section_2`

- **Relative URI**

A *relative URI* contains no naming scheme information. Relative URIs are resolved to full URIs using a base URI.

``

`http://somesite.com/icons/logo.gif`

Universal Resource Identifier (URI)

In HTML, URIs are used to:

- **Link** to another document or resource,
- **Link** to an external style sheet or script,
- **Include** an image, object, or applet in a page,
- **Create** an image map,
- **Submit** a form,
- **Create** an iframe document,
- **Cite** an external reference,
- **Refer** to metadata conventions describing a document

Protocols

- [http](#) - Hypertext transfer Protocol
- [ftp](#) - File Transfer protocol
- [mailto](#) - Electronic mail address
- [news](#) - USENET news
- [file](#) - Host-specific file names

Protocol HTTP

- HTTP is a request-response standard typical of client-server computing: web browsers typically act as clients, while an application running on the computer hosting the web site acts as a server.
 - The client establishes a connection with the server and sends a request for a document
 - The server replies to the request by using the connection opened by the client
- Since the protocol is stateless (the server does not recognise the client and does not record the requests), if the connection falls, the request has to be repeated.

HTML

- HyperText Markup Language (HTML) is the language used to publish information on the Web
- HTML 4.01 is an Standard Generalized Markup Language (SGML) application conforming to International Standard ISO 8879.
 - **Publish online** documents with headings, text, tables, lists, photos, etc.
 - **Retrieve online** information via hypertext links, at the click of a button.
 - **Design forms** for conducting transactions with remote services.
 - **Searching for information**, making reservations, ordering products, etc.
 - **Include spread-sheets**, video clips, sound clips, and other applications directly in their documents.

HTML

- HTML 5
 - Compatibility
 - Utility
 - The user is king
 - Interoperability simplification
 - Native browser ability instead of complex JavaScript code
 - Powerful yet simple HTML5 APIs
 - HTML5 specification is also more detailed than previous ones to prevent misinterpretation: the specification is over 900 pages long!
 - HTML5 is also designed to handle errors well, with a variety of improved and ambitious error handling plans. It prefers graceful error recovery to hard failure, again giving top priority to the interest of the end user.
 - Universal Access
 - Accessibility: support users with disabilities
 - Media Independence
 - Support for all world languages

HTML Syntax

HTML 5.0 Document Structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title> My first HTML document </title>
    <!-- A simple html document -->
  </head>
  <body>
    <p> Hello world! </p>
  </body>
</html>
```

Constructs used in HTML

- Elements

- Everything from the start tag to the end tag
`<h1>Sample page</h1>`

- Attributes

- Provide additional information about HTML elements
`<p>This is a simple sample.</p>`

- Text

- Text is allowed inside elements, attribute values and comments

- Character references

- Examples: `´`; (é) ```; (è)

- Comments

- Examples: `<!-- comment -->`

HTML Elements

```
<start_tag attribute_list>  
    element_content  
<end_tag>
```

- The attribute list may contain the name and the identifier of the element
- The element names and identifiers are always case-insensitive.
- Some HTML elements have **empty content**. Empty elements are closed in the start
 - `
` is an empty element without a closing tag (it defines a line break).
- Most HTML elements can have **attributes**
- Most HTML elements can be **nested**.

HTML Elements

- Note: Most browsers will display HTML correctly even if you forget the end tag:

`<p>This is a paragraph`

- The example above will work in most browsers, but **don't rely on it**. Forgetting the end tag can produce unexpected results or errors.
- **Elements are not tags**. Some people refer to elements as tags (e.g., "the p tag"). For instance, the head element is always present, even though both start and end head tags may be missing in the markup.

HTML Elements

- HTML tags are not case sensitive: <P> means the same as <p>.
- Plenty of web sites use uppercase HTML tags in their pages.
- The World Wide Web Consortium (W3C) recommends lowercase in HTML 4, and demands lowercase tags in future versions of (X)HTML

Element definitions

An example

4.4.1 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](#)

Tag omission in text/html:

A `p` element's [end tag](#) may be omitted if the `p` element is immediately followed by an [address](#), [article](#), [aside](#), [blockquote](#), [div](#), [dl](#), [fieldset](#), [footer](#), [form](#), [h1](#), [h2](#), [h3](#), [h4](#), [h5](#), [h6](#), [header](#), [hgroup](#), [hr](#), [main](#), [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.

Allowed [ARIA role attribute](#) values:

[Any role value](#).

Allowed [ARIA state and property attributes](#):

[Global aria-* attributes](#)

Any [aria-* attributes](#) [applicable to the allowed roles](#).

DOM interface:

```
IDL interface HTMLParagraphElement : HTMLElement {};
```


Element definitions

Category

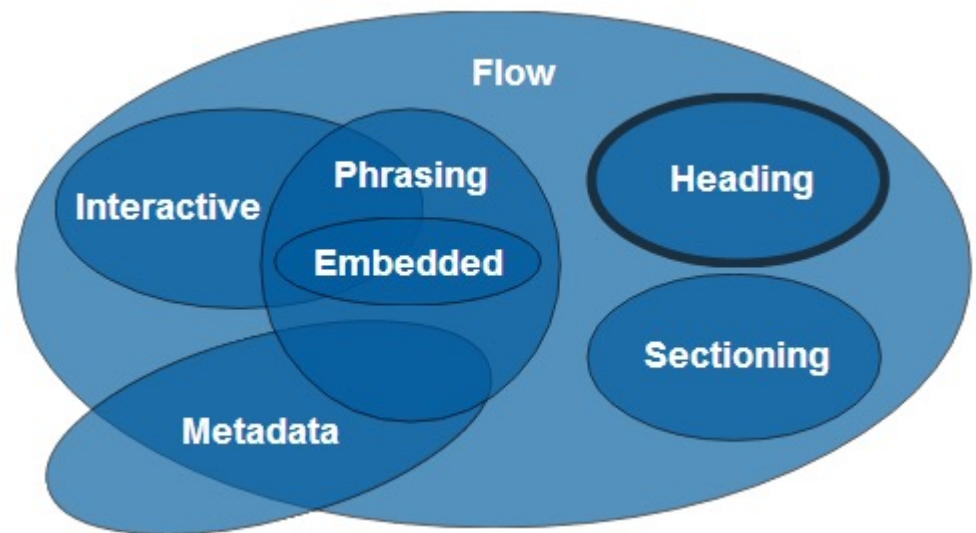
- Categories

- Each element in HTML falls into zero or more categories that group elements with similar characteristics together.

Categories:

Flow content.

Palpable content.



Element definitions

Category

Content Type	Description
Embedded	Content that imports other resources into the document, for example <code>audio</code> , <code>video</code> , <code>canvas</code> , and <code>iframe</code>
Flow	Elements used in the body of documents and applications, for example <code>form</code> , <code>h1</code> , and <code>small</code>
Heading	Section headers, for example <code>h1</code> , <code>h2</code> , and <code>hgroup</code>
Interactive	Content that users interact with, for example <code>audio</code> or <code>video</code> controls, <code>button</code> , and <code>textarea</code>
Metadata	Elements—commonly found in the <code>head</code> section—that set up the presentation or behavior of the rest of the document, for example <code>script</code> , <code>style</code> , and <code>title</code>
Phrasing	Text and text markup elements, for example <code>mark</code> , <code>kbd</code> , <code>sub</code> , and <code>sup</code>
Sectioning	Elements that define sections in the document, for example <code>article</code> , <code>aside</code> , and <code>title</code>

Element definitions

Context

- **Context in which the element can be used**
 - A non-normative description of where the element can be used
 - 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.

Contexts in which this element can be used:

Where flow content is expected.

Element definitions

Content

- **Content model**
 - A normative description of what content must be included as children and descendants of the element.

Content model:

Phrasing content.

Element definitions

Attributes

- **Content attributes**

- A normative list of attributes that may be specified on the element (except where otherwise disallowed), along with non-normative descriptions of those attributes.

Content attributes:

Global attributes

Element definitions

Tag omission

- Tag omission in text/html
 - A non-normative description of whether, in the text/html syntax, the start and end tags can be omitted.

Tag omission in text/html:

A `p` element's end tag may be omitted if the `p` element is immediately followed by an `address`, `article`, `aside`, `blockquote`, `div`, `dl`, `fieldset`, `footer`, `form`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `header`, `hgroup`, `hr`, `main`, `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.

Element definitions

DOM Interface

- DOM (Document Object Model) interface
 - A normative definition of a DOM interface that such elements must implement.
- What is DOM?
 - Let us consider the following example

Element definitions

DOM Interface

```
<!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>
```


Element definitions

DOM Interface

- HTML user agents (e.g. Web browsers) parse this markup, turning it into a DOM (Document Object Model) tree.
- A DOM tree is an in-memory representation of a document.
- DOM trees contain several kinds of nodes, in particular a DocumentType node, Element nodes, Text nodes, Comment nodes, and in some cases ProcessingInstruction nodes.

DOM interface:

IDL

```
interface HTMLParagraphElement : HTMLElement {};
```

Element definitions

DOM Interface

<!DOCTYPE html>

<html>

<head>

<title>Sample page</title>

</head>

<body>

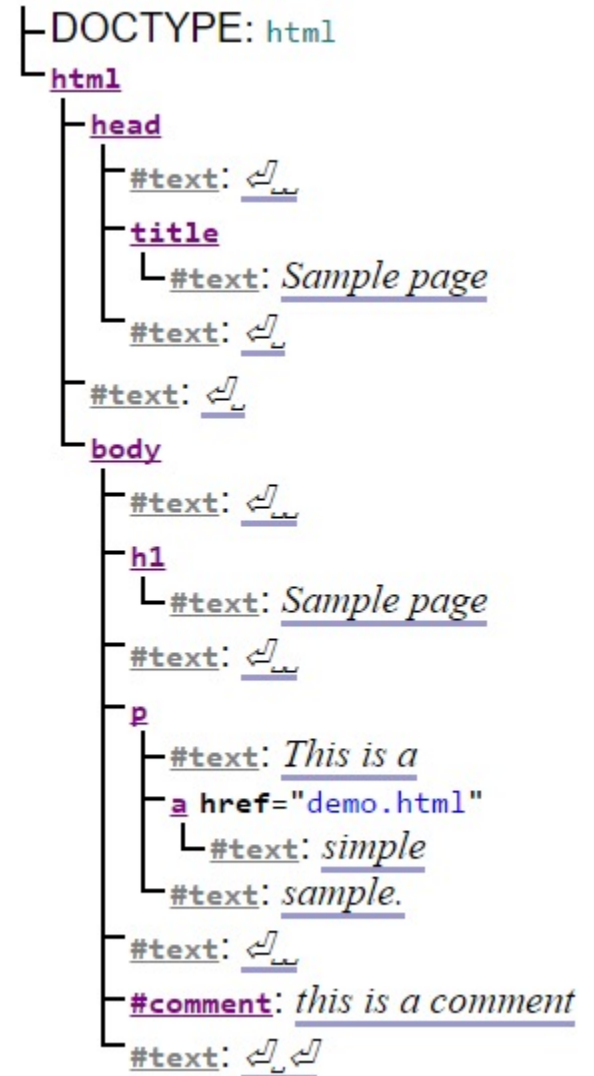
<h1>Sample page</h1>

<p>This is a simple
sample.</p>

<!-- this is a comment -->

</body>

</html>



Element definitions

WAI-ARIA

- **WAI-ARIA (Web Accessibility Initiative -Accessible Rich Internet Applications specification)**
- Client-side scripting causes accessibility problems when they do not convey the information necessary to Assistive Technologies (AT) – the devices and user systems typically used by people with disabilities. In simple terms assistive technology needs to know:
 1. **What things are:** The Role. (E.g., I am a checkbox)
 2. **What they are doing:** The state. (E.g., I am now checked)
 3. **What the relationships are.** (E.g., I am labeled by that text, I am part of this group, etc.)
 4. **Also the focus needs to be accessible from the keyboard and the interaction needs to be predictable.** (E.g., you can tab to this checkbox and press enter, then I am checked)

Element definitions

- **ARIA role attributes**

- The attribute describes the role(s) the current element plays in the context of the document.
 - This could allow a user to make informed decisions on which actions may be taken on an element and activate the selected action in a device independent way.
- The document “Using WAI-ARIA” is a practical guide for developers on how to add accessibility information to HTML elements, which defines a way to make Web content and Web applications more accessible to people with disabilities.
- This document demonstrates how to use WAI-ARIA, which especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies

Element definitions

- **ARIA State and Properties attributes**
 - Every HTML element may have ARIA state and property attributes specified. These attributes are defined by [ARIA] in Section 6.6, Definitions of States and Properties (all aria-* attributes).
 - ARIA State and Property attributes can be used on any element.

Allowed ARIA role attribute values:

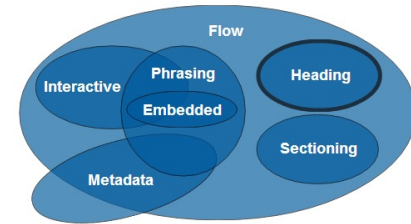
Any role value.

Allowed ARIA state and property attributes:

Global aria-* attributes

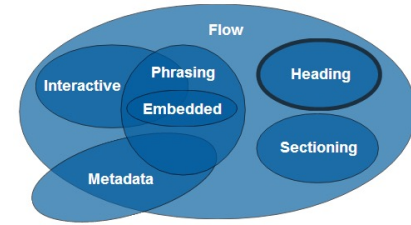
Any **aria-*** attributes applicable to the allowed roles.

Metadata Content



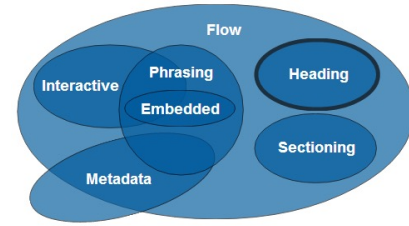
- Metadata content is content that sets up the presentation or behavior of the rest of the content, or that sets up the relationship of the document with other documents, or that conveys other "out of band" information.
 - base link meta noscript script style template title

Flow Content



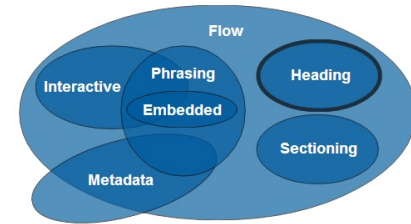
- Most elements that are used in the body of documents and applications are categorized as flow content.
- a abbr address area (if it is a descendant of a map element) article aside audio b bdi bdo blockquote br button canvas cite code data datalist del dfn div dl em embed fieldset figure footer form h1 h2 h3 h4 h5 h6 header hr i iframe img input ins kbd keygen label main map mark math meter nav noscript object ol output p pre progress q ruby s samp script section select small span strong sub sup svg table template textarea time u ul var video wbr

Sectioning Content



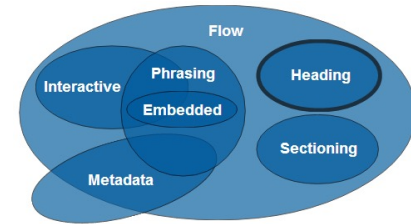
- Sectioning content is content that defines the scope of headings and footers
- article aside nav section

Heading Content



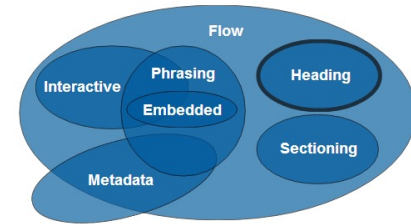
- Heading content defines the header of a section (whether explicitly marked up using sectioning content elements, or implied by the heading content itself)
- h1 h2 h3 h4 h5 h6

Phrasing Content



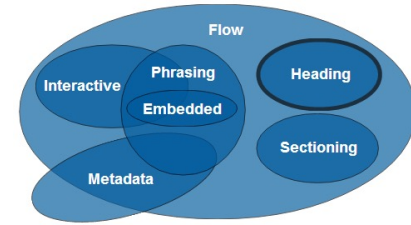
- Phrasing content is the text of the document, as well as elements that mark up that text at the intra-paragraph level. Runs of phrasing content form paragraphs
- a abbr area (if it is a descendant of a map element) audio b bdi bdo br button canvas cite code data datalist del dfn em embed i iframe img input ins kbd keygen label map mark math meter noscript object output progress q ruby s samp script select small span strong sub sup svg template textarea time u var video wbr text
- 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).

Embedded Content



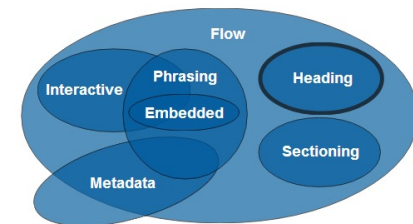
- Embedded content is content that imports another resource into the document, or content from another vocabulary that is inserted into the document.
- audio canvas embed iframe img math object svg video

Interactive Content



- **Interactive content is content that is specifically intended for user interaction.**
- a audio (if the controls attribute is present) button 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)

Palpable Content



- As a general rule, elements whose content model allows any flow content or phrasing content should have at least one node in its contents that is palpable content and that does not have the hidden attribute specified.
- a abbr address article aside audio (if the controls attribute is present) b bdi bdo blockquote button canvas cite code data 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 i iframe img input (if the type attribute is not in the hidden state) ins kbd keygen label main map mark math 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 svg table textarea time u ul (if the element's children include at least one li element) var video text that is not inter-element whitespace

Script-supporting Elements

- Script-supporting elements are those that do not represent anything themselves (i.e. they are not rendered), but are used to support scripts, e.g. to provide functionality for the user.
- script template

Transparent content models

- Some elements are described as transparent. The content model of a transparent element is derived from the content model of its parent element: the elements required in the part of the content model that is "transparent" are the same elements as required in the part of the content model of the parent of the transparent element in which the transparent element finds itself.
- 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.

HTML Attributes

- Attribute/value pairs appear before the final ">" of an element's start tag.
- Any number of (legal) attribute value pairs, separated by spaces, may appear in an element's start tag.
- Attributes may appear in any order.
- The attribute value can remain unquoted if it does not contain space characters or any of " ' ` = < or >. Otherwise, it must be delimited using either double quotation marks (ASCII decimal 34) or single quotation marks (ASCII decimal 39)

Attributes

- Example

```
<a href="http://www.w3schools.com">This is a link</a>
```

HTML links are defined with the <a> tag.

The link address is provided as an attribute

- Attribute names are always case-insensitive.
- Attribute values are generally case-insensitive.

Attributes

- Attribute values **may only contain**:
 - letters (a-z and A-Z),
 - digits (0-9),
 - hyphens (ASCII decimal 45),
 - periods (ASCII decimal 46),
 - underscores (ASCII decimal 95),
 - colons (ASCII decimal 58).
- The W3C recommends lowercase attributes/attribute values

Global Attributes

- Global Attributes

May be specified on all the HTML elements

- core attributes
- event-handler attributes
- xml attributes

Global Attributes

- Core Attributes

- **accesskey**: specifies a shortcut key to activate/focus an element
- **class**: Specifies one or more classnames for an element (refers to a class in a style sheet)
- **contenteditable**: Specifies whether the content of an element is editable or not
- **dir**: Specifies the text direction for the content in an element
- **draggable**: Specifies whether an element is draggable or not
- **dropzone**: Specifies whether the dragged data is copied, moved, or linked, when dropped (not implemented in the browsers)

Global Attributes

- Core Attributes

- **hidden**: Specifies that an element is not yet, or is no longer, relevant
- **id**: Specifies a unique id for an element
- **lang**: Specifies the language of the element's content
- **spellcheck**: Specifies whether the element represents an element whose contents are subject to spell checking and grammar checking.
- **style**: Specifies an inline CSS style for an element
- **tabindex**: Specifies the tabbing order of an element
- **title**: Specifies extra information about an element
- **translate**: specifies whether the content of an element should be translated or not (not properly supported in any of the major browsers)

Global Attributes

- Event-handler Attributes

An event handler content attribute is a content attribute for a specific event handler. The name of the content attribute is the same as the name of the event handler.

Some common events attributes are

- **onabort** - Load of element was aborted by the user.
- **onblur** - Element lost focus.
- **onchange** - User committed a change to the value of element (form control).
- **onclick** - User pressed pointer button down and released pointer button over element
- **ondblclick** - User clicked pointer button twice over element
- **ondrag** - User is continuing to drag element.

Global Attributes

- Event-handler Attributes (continued)

- **ondragend** - User ended dragging element.
- **ondragenter** - User's drag operation entered element.
- **ondragleave** - User's drag operation left element.
- **ondragover** - User is continuing drag operation over element.
- **ondragstart** - User started dragging element.
- **onerror** - element failed to load properly.
- **onfocus** - element received focus.
- **oninput** - user changed the value of element (form control).
- **oninvalid** - element (form control) did not meet validity constraints.
- **onkeydown** - user pressed down a key.

Global Attributes

- Event-handler Attributes (continued)

- **onkeypress** - User pressed down a key that is associated with a character value.
- **onkeyup** - User released a key.
- **onLoad** - Element finished loading.
- **onmousedown** - User pressed down pointer button over element.
- **onmousemove** - User moved mouse.
- **onmouseout** - User moved pointer off boundaries of element.
- **onmouseover** - User moved pointer into boundaries of element or one of its descendant elements.
- **onmouseup** - User released pointer button over element.

Global Attributes

- Event-handler Attributes (continued)
 - **onmousewheel** - User rotated wheel of mouse or other device in a manner that emulates such an action.
 - **onreset** - the form element was reset.
 - **onscroll** - Element or document view was scrolled.
 - **onselect** – User selected some text.
 - **onsubmit** – The form element was submitted.
- Other event-handler attributes can be found in the HTML5 specification.

Character references

- *Character references* are a form of markup for representing single individual characters. There are three types of character references:
 - named character references
 - decimal numeric character references
 - hexadecimal numeric character references

Character references

- **Named character references**
 - an "&" character.
 - one of the names listed in the “Named character references” section of the HTML5 specification [HTML5], using the same case.
 - a ";" character.
- Example:
 - † for the character +;

Character references

- **Decimal Numeric Character Reference**
 - an "&" character.
 - a "#" character.
 - one or more digits in the range 0–9, representing a base-ten integer that itself is a Unicode code point that is not U+0000, U+000D, in the range U+0080–U+009F, or in the range 0xD800–0xDFFF (surrogates).
 - a ";" character.
- Example:
 - `†` for the character +;

Character references

- **Hexadecimal Numeric Character Reference**

- an "&" character.
- a "#" character.
- either a "x" character or a "X" character.
- one or more digits in the range 0–9, a–f, and A–F, representing a base-sixteen integer that itself is a Unicode code point that is not U+0000, U+000D, in the range U+0080–U+009F, or in the range 0xD800–0xDFFF (surrogates).
- a ";" character.
- Example:
 - `†` for the character +;

Comments

- HTML comments have the following syntax:
 - <!-- this is a comment -->
 - <!-- and so is this one,
which occupies more than one line -->
- The text part of comments has the following restrictions:
 - must not start with a ">" character
 - must not start with the string "->"
 - must not contain the string "--"
 - must not end with a "-" character

Authoring HTML documents

- **Separate structure and presentation**
 - reduces the cost of serving a wide range of platforms, media, etc., and facilitates document revisions.
- **Consider universal accessibility to the Web**
 - authors should consider how their documents may be rendered on a variety of platforms
 - in order for documents to be interpreted correctly, authors should include in their documents information about the natural language and direction of the text, how the document is encoded, and other issues related to internationalization.
- **Help browsers with incremental rendering**
 - Allows browsers to render documents more quickly. For instance, to design tables for incremental rendering

The Elements of HTML

HTML 5.0 Document Structure

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title> My first HTML document </title>
```

```
</head>
```

```
<body>
```

```
<p> Hello world! </p>
```

```
</body>
```

```
</html>
```

Root element

HTML Element

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 `head` element followed by a `body` element.

Content attributes:

Global attributes

`manifest` — Application cache manifest

Tag omission in text/html:

An `html` element's start tag can be omitted if the first thing inside the `html` element is not a comment.

An `html` element's end tag can be omitted if the `html` element is not immediately followed by a comment.

Allowed ARIA role attribute values:

none

Allowed ARIA state and property attributes:

Global aria-* attributes

DOM interface:

IDL

```
interface HTMLHtmlElement : HTMLElement {};
```

Root element

HTML Element

- Cache Manifest basic:

- HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.

`<html manifest="demo.appcache">`

- The manifest file is a simple text file, which tells the browser what to cache (and what to never cache). The manifest file has three sections:
- **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

Root element

HTML Element

- Example of manifest file

CACHE MANIFEST

2012-02-21 v1.0.0

/theme.css

/logo.gif

/main.js

NETWORK:

login.asp

FALLBACK:

/html/ /offline.html

Document Metadata

Head Element

- The **head element** contains information about the current document, such as its **title**, **keywords** that may be useful to search engines, and other data that is not considered document content.
- Every HTML document **must** have a **title element** in the head section.

```
<head>  
<title>HTML course</title>  
</head>
```

Document Metadata

Title

- Use the title element to identify the contents of a document.
- Since users often consult documents out of context, authors should provide context-rich titles. Thus, instead of a title such as “A study”, authors should supply a title such as “A study of population dynamics” instead.
- User agents must always make the content of the title element available to users
- Titles may contain character entities (for accented characters, special characters, etc.), but may not contain other markup (including comments).

Document Metadata

Meta Data

- Meta Data -- information about a document

```
<head>
```

```
<meta charset="utf-8">
```

```
<meta name="author" content="Mario Rossi">
```

```
<meta name="description" content="A tutorial about HTML">
```

```
<meta name="generator" content="Notepad++">
```

```
<meta name="keywords" content="HTML,CSS,XML,JavaScript">
```

```
<title>HTML5 Example</title>
```

```
</head>
```

- Possible values for name: **application-name**, **author**, **description**, **generator**, **keywords**

Document Metadata

Meta Data

- The `http-equiv` attribute integrates an HTTP header with the information in the content attribute.
- The value of the `http-equiv` attribute depends on the value of the content attribute.

```
<meta http-equiv="Refresh" content="10">
```

Defines a time interval for the document to refresh itself

Document Metadata

Meta Data

- When **several meta elements** provide language-dependent information, search engines may filter on the lang attribute to display search results using the language preferences of the user.

```
<!-- For speakers of US English -->
```

```
<meta name="keywords" lang="en-us"  
content="vacation, Greece, sunshine">
```

```
<!-- For speakers of British English -->
```

```
<meta name="keywords" lang="en"  
content="holiday, Greece, sunshine">
```

```
<!-- For speakers of French -->
```

```
<meta name="keywords" lang="fr"  
content="vacances, Gr&egrave;ce, soleil">
```

Document Metadata

Base element

- 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.

```
<!DOCTYPE html>
```

```
<html> <head>
```

```
  <meta charset="utf-8">
```

```
  <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".

Document Metadata

Link element

- The **link element** allows authors to link their document to other resources.
- A link element must have a rel attribute. Possible values for rel are “alternate”, “author”, “help”, ..., “licence”, “stylesheet”,...

```
<link rel="author license" href="/about">
```

```
<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 title="English HTML (for printing)">
```

```
<link rel=alternate href="/fr/html/print" hreflang=fr type=text/html media=print title="French HTML (for printing)">
```

```
<link rel=alternate href="/en/pdf" hreflang=en type=application/pdf title="English PDF">
```

```
<link rel=alternate href="/fr/pdf" hreflang=fr type=application/pdf title="French PDF">
```

Document Metadata

Style element

- 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.
- The **media** attribute says which media the styles apply to. If the **media attribute is omitted**, is "all", meaning that by default styles apply to all media.

```
<html lang="en-US">
<head>
  <title>My favorite book</title>
  <style>
    body { color: black; background: white; }
    em { font-style: normal; color: red; }
  </style>
</head>
```

Sections

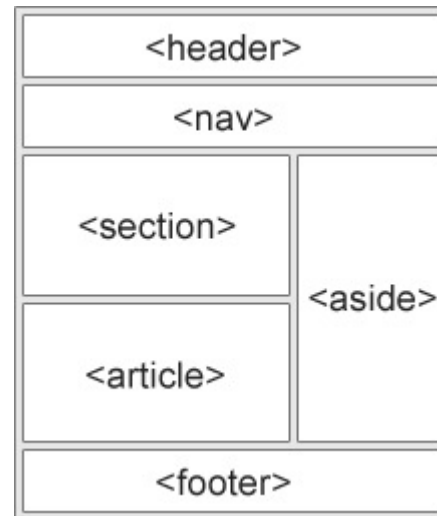
Body Element

- The body of a document contains the **document's content**.
- There is only one body element
- The onblur, onerror, onfocus, onload, onresize, and onscroll event handlers of the Window object, exposed on the body element, replace the generic event handlers with the same names normally supported by HTML elements

Sections

Semantic and non-semantic elements

- A **semantic element** clearly describes its meaning to both the browser and the developer.
- Examples of non-semantic elements: `<div>` and `` - Tells nothing about their content.
- Examples of semantic elements: `<form>`, `<table>`, and `` - Clearly defines their content.



Sections

Document Organization



↑
Rest of content
↓



↑
Rest of content
↓

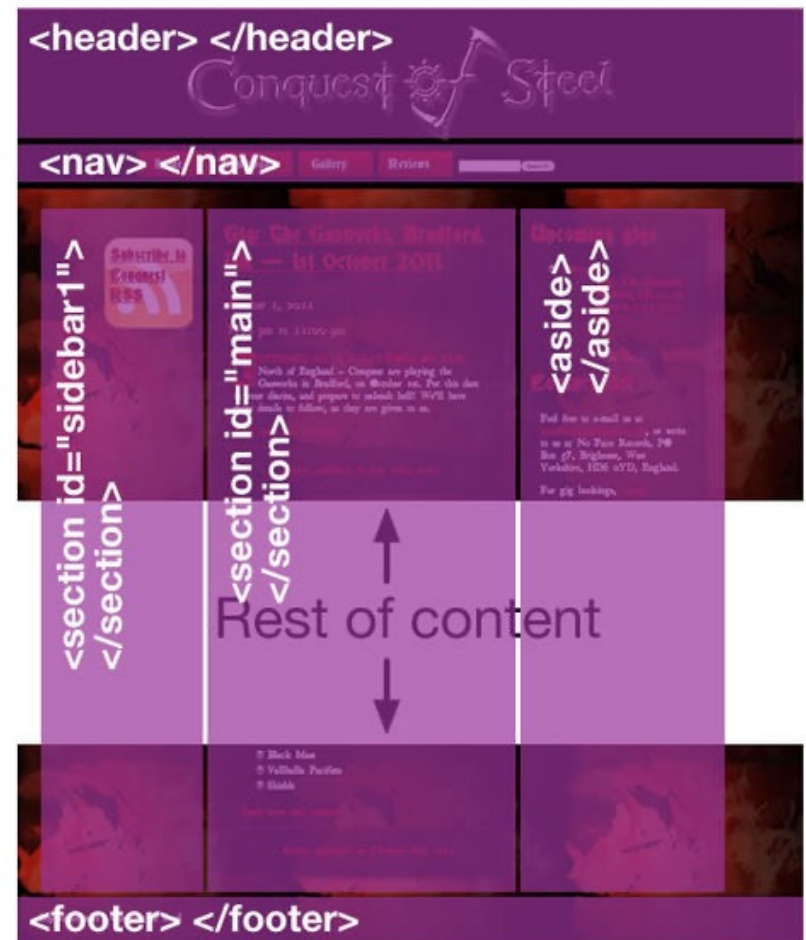


Sections Document Organization

HTML 4



HTML 5



Sections

Document Organization



Header Element

- The header element **represents introductory content for its nearest ancestor sectioning content or sectioning root element**. A header typically contains a group of introductory or navigational aids.
- When the nearest ancestor sectioning content or sectioning root element is the body element, then it applies to the whole page.
- The header element is not sectioning content; it doesn't introduce a new section.

`<header>`

`<!-- header content goes in here -->`

`</header>`

Sections

Document Organization



Nav Element

- The nav element **represents a section of a page that links to other pages or to parts within the page**: a section with navigation links.
- When the nearest ancestor sectioning content or sectioning root element is the body element, then it applies to the whole page.
- The header element is not sectioning content; it doesn't introduce a new section.

`<nav>`

`<!-- navigation menu goes in here -->`

`</nav>`

Sections

Document Organization



Section Element

- The section element represents **a generic section of a document or application**.
- A section, in this context, is a thematic grouping of content. The theme of each section should be identified, typically by including a heading (h1-h6 element) as a child of the section element.
- Examples of sections would be chapters, the various tabbed pages in a tabbed dialog box, or the numbered sections of a thesis.

`<section id="sidebar1">`

`<!-- sidebar content goes in here -->`

`</section>`

Sections

Document Organization



Article Element

- The article element **represents a complete, or 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 related to the contents of the outer article.

`<article>`

`<!-- article content goes in here -->`

`</article>`

Sections

Document Organization



Aside Element

- 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.

`<aside>`

`<!-- aside content goes in here -->`

`</aside>`

Sections

Document Organization



Footer Element

- Represents **a footer for its nearest ancestor sectioning content or sectioning root element.**
 - contains information about its section such as who wrote it, links to related documents, copyright data, and the like.
- When the footer element contains entire sections, they represent appendices, indexes, long colophons, verbose license agreements, and other such content.
 - Contact information for the author or editor of a section belongs in an address element, possibly itself inside a footer.

`<footer>`

`<!-- footer content goes in here -->`

`</footer>`

Sections

Document Organization

Heading Elements

- A heading element briefly describes the topic of the section it introduces. Heading information may be used by user agents, for example, to construct a table of contents for a document automatically.
- There are six levels of headings in HTML with h1 as the most important and h6 as the least.
- Use HTML headings for headings only. Don't use headings to make text BIG or bold.
- Search engines use your headings to index the structure and content of your web pages. Since users may skim your pages by its headings, it is important to use headings to show the document structure.
- h1 headings should be used as main headings, followed by h2 headings, then less important h3 headings, and so on.

<h1>This is a heading 1</h1>

Sections

Document Organization

Address Element

- The address element **represents the contact information for its nearest article or body element ancestor**. If that is the body element, then the contact information applies to the document as a whole.
- The **address element must not be used to represent arbitrary addresses** (e.g. postal addresses), unless those addresses are in fact the relevant contact information.
- Typically, the address element would be included along with other information in a footer element.

<address>

For more details, contact

John Smith.

</address>

Sections

Example

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>HTML5</title>
</head>
<body>
  <header>
    <h1>Header</h1>
    <h2>Subtitle</h2>
    <h4>HTML5 Rocks!</h4>
  </header>
```

Sections Example

```
<div id="container">  
  <nav>  
    <h3>Nav</h3>  
    <a href="">Link 1</a>  
    <a href="">Link 2</a>  
    <a href="">Link 3</a>  
  </nav>  
  
  <section>  
    <article>  
      <header>  
        <h1>Article Header</h1>  
      </header>
```

Sections

Example

```
<p>HTML5: "Lorem ipsum dolor nunc aut nunquam sit amet,  
consectetur adipiscing elit. Vivamus at est eros, vel fringilla urna.  
Pellentesque odio</p>
```

```
<footer>
```

```
  <h2>Article Footer</h2>
```

```
</footer>
```

```
</article>
```

```
</section>
```

```
<aside>
```

```
  <h3>Aside</h3>
```

```
  <p>HTML5: "Lorem ipsum dolor nunc aut nunquam sit amet,  
consectetur adipiscing elit. Vivamus at est eros, vel fringilla urna.  
Pellentesque odio rhoncus</p>
```

```
</aside>
```

Sections Example

```
<footer>
  <h2>Footer</h2>
  <address>
    For more details, contact <a id="mail"
    href="mailto:js@example.com">John Smith</a>.
  </address>
  <p><small>© copyright 2014 Example Corp.</small></p>
</footer>
</div>
</body>

</html>
```

Sections

Example with Styles

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>HTML5</title>
  <link rel="stylesheet" href="html5.css">

</head>
<body>
  <header>
    <h1>Header</h1>
    <h2>Subtitle</h2>
    <h4>HTML5 Rocks!</h4>
  </header>
```

Grouping content elements

Grouping Content

Element p

- The **p** element represents a paragraph.
- The p element should not be used when a more specific element is more appropriate.

```
<section>
```

```
<!-- ... -->
```

```
<p>Last modified: 2001-04-23</p>
```

```
<p>Author: fred@example.com</p>
```

```
</section>
```

Technically correct,
but not appropriate

```
<section>
```

```
<!-- ... -->
```

```
<footer>Last modified: 2001-04-23</footer>
```

```
<address>Author: fred@example.com</address>
```

```
</section>
```

Better

Grouping Content

Element `hr`

- 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
- There is no need for an `hr` element between the sections themselves, since the section elements and the `h1` elements imply thematic changes themselves.

Grouping Content

Element pre

- The **pre** element represents a block of preformatted text, in which structure is represented by typographic conventions rather than by elements.
- Some examples of cases where the pre element could be used:
 - 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.

<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>
```

Grouping Content

Element `blockquote`

- The `blockquote` element represents content that is quoted from another source, optionally with a citation which must be within a footer or cite element, and optionally with in-line changes such as annotations and abbreviations.

`<blockquote>`

The people recognize themselves in their commodities; they find their soul in their automobile, hi-fi set, split-level home, kitchen equipment.

— `<cite>Herbert Marcuse</cite>`

`</blockquote>`

Grouping Content

Element figure

- The **figure** element represents some flow content, optionally with a caption, that is self-contained (like a complete sentence) and is typically referenced as a single unit from the main flow of the document.
- The element can thus be used to annotate illustrations, diagrams, photos, code listings, etc.
- The first **figcaption** element child of the element, if any, represents **the caption of the figure element's contents**. If there is no child figcaption element, then there is no caption.
- A figure element's contents **are part of the surrounding flow**. If the purpose of the page is to display the figure, for example a photograph on an image sharing site, the figure and figcaption elements can be used to explicitly provide a caption for that figure.

Grouping Content

Element figure

Some uses

`<figure>`

`<p>'Twas brillig, and the slithy toves
`

`Did gyre and gimble in the wabe;
`

`All mimsy were the borogoves,
`

`And the mome raths outgrabe.</p>`

`<figcaption><cite>Jabberwocky</cite> (first verse). Lewis Carroll, 1832-98</figcaption>`

`</figure>`

`<figure>`

`<figcaption>Oil-based paint on canvas. Maria Towle, 1858.</figcaption>`

``

`</figure>`



Grouping Content

Element div

- 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.
- Example: div used to set the language of two paragraphs at once

```
<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>
```

Grouping Content

Element main

- The **main** element represents the main content of the body of a document or application. The main content area consists of content that is directly related to or expands upon the central topic of a document or central functionality of an application.
- The **main content area of a document includes content that is unique to that document and excludes content that is repeated across a set of documents such as site navigation links, copyright information, site logos and banners and search forms** (unless the document or applications main function is that of a search form).
- Authors must not include more than one main element in a document.

`<main>`

`<h1>Skateboards</h1>`

`<p>The skateboard is the way cool kids get around</p>`

`<article> </article>`

`<article> </article>`

`</main>`

Grouping Content

Unordered Lists

- HTML supports **ordered**, **unordered** and **definition** lists.
- **Unordered lists**
 - An unordered list starts with the `` tag.
 - Each list item starts with the `` tag.
 - The list items are marked with bullets (typically small black circles).

`<p>I have lived in the following countries:</p>`

``

`Norway`

`Switzerland`

`United Kingdom`

`United States`

``

Grouping Content

Ordered Lists

- **Ordered lists**

- An ordered list starts with the `` tag.
- Each list item starts with the `` tag.
- The list items are marked with numbers.

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 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.

Grouping Content

Ordered Lists

- **Ordered lists**

- The ol element has three specific attributes
 - **reversed** – number the list backwards
 - **start** – ordinal value of the first item
 - **type** – kind of list marker

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

Keywords	State	Description
1 (U+0031)	decimal	Decimal numbers
a (U+0061)	lower-alpha	Lowercase latin alphabet
A (U+0041)	upper-alpha	uppercase latin alphabet
i (U+0069)	lower-roman	lowercase roman numerals
I (U+0049)	upper-roman	uppercase roman numerals

Grouping Content

Ordered Lists

- **Ordered lists (example 1)**

```
<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 мачка, бели мачор</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>
```

Grouping Content

Ordered Lists

- **Ordered lists (example 2)**

<figure> <figcaption>The top 10 movies of all time</figcaption>

<ol reversed type="a">

<cite>Josie and the Pussycats</cite>, 2001

<cite lang="sh">Црна мачка, бели мачор</cite>, 1998

<cite>A Bug's Life</cite>, 1998

<cite>Toy Story</cite>, 1995

<cite>Monsters, Inc</cite>, 2001

<cite>Cars</cite>, 2006

<cite>Toy Story 2</cite>, 1999

<cite>Finding Nemo</cite>, 2003

<cite>The Incredibles</cite>, 2004

<cite>Ratatouille</cite>, 2007

</figure>

Grouping Content

Ordered Lists

- **Ordered lists (example 2)**

```
<figure> <figcaption>The top 10 movies of all time</figcaption>
<ol reversed type="a" start="10">
  <li><cite>Josie and the Pussycats</cite>, 2001</li>
  <li><cite lang="sh">Црна мачка, бели мачор</cite>, 1998</li>
  <li><cite>A Bug's Life</cite>, 1998</li>
  <li><cite>Toy Story</cite>, 1995</li>
  <li><cite>Monsters, Inc</cite>, 2001</li>
  <li><cite>Cars</cite>, 2006</li>
  <li><cite>Toy Story 2</cite>, 1999</li>
  <li><cite>Finding Nemo</cite>, 2003</li>
  <li><cite>The Incredibles</cite>, 2004</li>
  <li><cite>Ratatouille</cite>, 2007</li>
</ol></figure>
```

Grouping Content

Definition Lists

- **Definition lists**

- Lists of items (terms), with a description of each item (term).
- A definition list starts with a `<dl>` tag (definition list).
- Each term starts with a `<dt>` tag (definition term).
- Each description starts with a `<dd>` tag (definition description).

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.

Grouping Content

Definition Lists

- **Definition lists (Example)**

```
<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 of the  
fine structure of the eye to distinguish three differently  
filtered analyses of a view. </dd>
```

```
</dl>
```

Text-level semantics

Text-level semantics

Element a

- 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.
- Attributes
 - **href** - Address of the hyperlink
 - **target** - Default browsing context for hyperlink navigation and form submission
 - **download** - Whether to download the resource instead of navigating to it, and its file name if so
 - **rel** - Relationship between the document containing the hyperlink and the destination resource
 - **hreflang** - Language of the linked resource
 - **type** - Hint for the type of the referenced resource

Text-level semantics (Target)

Keyword	Ordinary effect	Effect in an <code>iframe</code> with...	
		<code>sandbox=""</code>	<code>sandbox="allow-top-navigation"</code>
none specified, for links and form submissions	current	current	current
empty string	current	current	current
<code>_blank</code>	new	maybe new	maybe new
<code>_self</code>	current	current	current
<code>_parent</code> if there isn't a parent	current	current	current
<code>_parent</code> if parent is also top	parent/top	none	parent/top
<code>_parent</code> if there is one and it's not top	parent	none	none
<code>_top</code> if top is current	current	current	current
<code>_top</code> if top is not current	top	none	top
name that doesn't exist	new	maybe new	maybe new
name that exists and is a descendant	specified descendant	specified descendant	specified descendant
name that exists and is current	current	current	current
name that exists and is an ancestor that is top	specified ancestor	none	specified ancestor/top
name that exists and is an ancestor that is not top	specified ancestor	none	none
other name that exists with common top	specified	none	none
name that exists with different top, if <code>familiar</code> and <code>one permitted sandboxed navigator</code>	specified	specified	specified
name that exists with different top, if <code>familiar</code> but not <code>one permitted sandboxed navigator</code>	specified	none	none
name that exists with different top, not <code>familiar</code>	new	maybe new	maybe new

Text-level semantics

Element a

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. Remove developer-view styles
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.



Text-level semantics

Element a

- Example

```
<nav>
```

```
<ul>
```

```
<li> <a href="http://www.unipi.it" target="_blank">New</a> </li>
```

```
<li> <a href="http://www.unipi.it" target="_self">Self</a> </li>
```

```
<li> <a href="http://www.unipi.it" target="_top" rel="tag">Top</a> </li>
```

```
</ul>
```

```
</nav>
```

Text-level semantics

Elements `em`, `strong`, `small`

- The `em` element represents stress emphasis of its contents.
- The `strong` element represents strong importance, seriousness, or urgency (contents that the user needs to see sooner than other parts of the document) for its contents.
- The `small` element represents side comments such as small print.

Example

`<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>`

Text-level semantics

Elements s and cite

- The **s** element represents contents that are no longer accurate or no longer relevant.

`<p>Buy our Iced Tea and Lemonade!</p>`

`<p><s>Recommended retail price: $3.99 per bottle</s></p>`

`<p>Now selling for just $2.99 a bottle!</p>`

- The **cite** element represents a reference to a creative work. It must include the title of the work or the name of the author(person, people or organization) or an URL reference, which may be in an abbreviated form as per the conventions used for the addition of citation metadata.

`<p>In the words of <cite>Charles Bukowski</cite> -`

Text-level semantics

Element q

- The **q** element represents some phrasing content quoted from another source.

<p>The W3C page <cite>About W3C</cite> says the W3C's mission is <q cite="http://www.w3.org/Consortium/">To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web</q>. I disagree with this mission.</p>

Text-level semantics

Elements dfn and abbr

- 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.
- If the dfn element has a *title* attribute, then the exact value of that attribute is the term being defined.
- 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.

Text-level semantics

Elements dfn and abbr

- Example

```
<p>The <dfn id=gdo><abbr title="Garage Door  
Opener">GDO</abbr></dfn>
```

is a device that allows off-world teams to open the iris.</p>

```
<!-- ... later in the document: -->
```

```
<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>

Text-level semantics

Elem code, var, samp, kbd, sup, sub

- The **code** element represents a fragment of computer code.
- The **var** element represents a variable.
- The **samp** element represents (sample) output from a program or computing system.
- The **kbd** element represents user input (typically keyboard input, although it may also be used to represent other input, such as voice commands).
- The **sup** element represents a superscript and the **sub** element represents a subscript.

<p>The coordinate of the <var>i</var>th point is

<code>(<var>x_{<var>i</var>}</var>,
<var>y_{<var>i</var>}</var>)</code>.

For example, the 10th point has coordinate

<code>(<var>x₁₀</var>,
<var>y₁₀</var>)</code>.</p>

Text-level semantics

Elements i, b and mark

- 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.
- 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 **mark** element represents a **run of text in one document marked or highlighted for reference purposes**, due to its relevance in another context.

Text-level semantics

Elements span, br, wbr

- 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**.
- The **br** element represents a line break.
- The **wbr** element represents a line break opportunity.

`<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>`

Edits

Edits

Elements ins, del

- The **ins** element represents an addition to the document.
- The **del** element represents a removal from the document.
 - The **cite** attribute may be used to specify the address of a document that explains the change.
 - The **datetime** attribute may be used to specify the time and date of the change.

```
<h1>List of <del>fruits</del><ins>colors</ins></h1>
```

```
<ul>
```

```
<li><del datetime="2009-10-10T23:38-07:00">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><ins>Olive</ins></li>
```

```
</ul>
```

Embedded content

Embedded Content

The `img` element

- An `img` element represents an image.
- The `` element is empty, which means that it contains attributes only and it has no closing tag.
- To display an image on a page, you need to use the `src` attribute. The value of the `src` attribute is the URL of the image you want to display on your page.
- The syntax of defining an image:

```

```

- The browser puts the image where the image tag occurs in the document.
- Except where otherwise specified, the `alt` attribute must be specified and its value must not be empty; the value must be an appropriate functional replacement for the image.

Embedded Content

The img element

```
<p> <img src= "tower.jpg" width="40" height="40" alt="Tower of Pisa">  
</p>
```

```
<p> <img src= "tower.jpg" width="90" height="90" alt="Tower of Pisa">  
</p>
```

```
<p>
```

You can make an image smaller or larger by changing the values of the "height" and "width" attributes.

```
</p>
```


Embedded Content

The img element

- An **img** is always in one of the following states:
 - **Unavailable**
 - The user agent has not 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).

Embedded Content

The image map

- **Image maps** allow authors to specify regions of an image and assign a specific action to each region (e.g., retrieve a document, run a program, etc.). When the region is activated by the user, the action is executed.
- An image map is created by associating an image with a specification of sensitive geometric areas on the image.
- The **map** element, in conjunction with an **img** element and 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.

Embedded Content

The image map

<p>Click on the sun or on one of the planets to get more information:</p>

<div>

<map name="planetmap">

<area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.html">

<area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.html">

<area shape="circle" coords="124,58,8" alt="Venus" href="venus.html">

</map>

</div>

Embedded Content

The image map

- The `<area>` tag defines an area inside an image-map (an image-map is an image with clickable areas).
- The **area element is always nested** inside a `<map>` tag
- Note: The `usemap` attribute in the `` tag is associated with the map element's name attribute, and creates a relationship between the image and the map.

Embedded Content

The image map

- Area attributes

Attribute	Value	Description
<u>alt</u>	<i>text</i>	Specifies an alternate text for an area
<u>coords</u>	<i>coordinates</i>	Specifies the coordinates of an area
<u>href</u>	<i>URL</i>	Specifies the destination of a link in an area
<u>download</u>	download	hyperlink is used for downloading a resource
<u>shape</u>	default rect circle poly	Specifies the shape of an area
<u>target</u>	_blank _parent _self _top	Specifies where to open the linked page specified in the href attribute

Embedded Content

The image map

- Coords attribute

This attribute specifies the position and shape on the screen. The number and order of values depends on the shape being defined. Possible combinations:

- rect: left-x, top-y, right-x, bottom-y.
- circle: center-x, center-y, radius.
- poly: x1, y1, x2, y2, ..., xN, yN.

The first x and y coordinate pair and the last should be the same to close the polygon. When these coordinate values are not the same, user agents should infer an additional coordinate pair to close the polygon.

Embedded Content

The iframe element

- The **iframe** element represents a nested browsing context.
- The **src** attribute gives the address of a document that the nested browsing context is to contain.
- The **srcdoc** attribute gives the HTML content that the nested browsing context is to contain. The value of the attribute is the source of an iframe srcdoc document.

```
<iframe srcdoc="<p>Hello world!</p>" </iframe>
```

- 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.

Embedded Content

The iframe element

```
<body>
```

```
<iframe src="html25.html" name="iframe_a">
```

```
<p>Your browser does not support iframes.</p>
```

```
</iframe>
```

```
<a href= "html26.html" target="iframe_a">Planets</a>
```

```
<p><b>Note:</b> Because the target of the link matches the name of the  
iframe, the link will open in the iframe.</p>
```

```
</body>
```

```
</html>
```


Embedded Content

The iframe element

- The **sandbox** attribute, when specified, enables a set of extra restrictions on any content hosted by the iframe.
- The **width** and **height** attributes determine the horizontal and vertical dimensions, respectively.

<p>Example</p>

```
<iframe sandbox="allow-same-origin allow-forms allow-scripts" src="html26.html" width="400" height="500">
```

[Your user agent does not support frames or is currently configured not to display frames.

However, you may visit the related document.]

Embedded Content

The embed element

- The `embed` element provides 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.
- The `type` attribute, if present, gives the MIME type by which the plugin to instantiate is selected.

```
<embed src="vowels.swf" width="720" height="500">
```

```
<embed src="movie.mp4" width="320" height="240" title="Title of my video">
```

Embedded Content

The object element

- 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.
- The **type** attribute, if present, specifies the type of the resource. If present, the attribute must be a valid MIME type.
- The **typemustmatch** attribute is a boolean attribute whose presence indicates that the resource specified by the data attribute is only to be used if the value of the type attribute and the Content-Type of the aforementioned resource match.

```
<object id="vowels" type="application/x-shockwave-flash"
      data="./vowels.swf" width="720" height="500">
  <param name="movie" value="./vowels.swf">
</object>
```

```
<object type="application/pdf"
  data="00-introduzione-al-corso.pdf"
  width="500"
  height="800">
</object>
```



Embedded Content

The param element

- 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.

Embedded Content

The video element

- A **video** element is used for playing videos or movies, and audio files with captions.
- The video element is a media element whose media data is ostensibly video data, possibly with associated audio data.

```
<video width="320" height="240" src="movie.mp4" type="video/mp4"
controls>
```

Your browser does not support the video tag.

```
</video>
```

Embedded Content

The video element

Attribute	Value	Description
<u>autoplay</u>	autoplay	Specifies that the video will start playing as soon as it is ready
<u>controls</u>	controls	Specifies that video controls should be displayed (such as a play/pause button etc).
<u>height</u>	pixels	Sets the height of the video player
<u>loop</u>	loop	Specifies that the video will start over again, every time it is finished
<u>muted</u>	muted	Specifies that the audio output of the video should be muted
<u>poster</u>	URL	Specifies an image to be shown while the video is downloading, or until the user hits the play button
<u>preload</u>	auto metadata none	Specifies if and how the author thinks the video should be loaded when the page loads. Metadata: only metadata. None: the browser should not load the video.
<u>src</u>	URL	Specifies the URL of the video file
<u>width</u>	pixels	Sets the width of the video player

Embedded Content

The source element

- 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.

```
<video width="320" height="240" controls>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
  Your browser does not support the video tag.  
</video>
```

Embedded Content

The audio element

- 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.

<audio controls>

<source src= "applause.ogg" type="audio/ogg">

<source src= "applause.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

HTML Links

Links

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**
 - These are links to other resources that are generally exposed to the user by the user agent so that the user can cause the user agent to navigate to those resources, e.g. to visit them in a browser or download them.

Links

Introduction

- For **link** elements with an **href** attribute and a **rel** attribute, links must be created for the keywords of the rel attribute, as defined for those keywords in the link types section.
- Similarly, for **a** and **area** elements with an **href** attribute and a **rel** attribute, links must be created for the keywords of the rel attribute as defined for those keywords in the link types section.

Links

Elements <a> and <area>

Attribute	Value	Description
<u>download</u>	filename	Specifies that the target will be downloaded when a user clicks on the hyperlink
<u>href</u>	URL	Specifies the URL of the page the link goes to
<u>hreflang</u>	language_code	Specifies the language of the linked document
<u>rel</u>	alternate author bookmark help license next nofollow noreferrer prefetch prev search tag	Specifies the relationship between the current document and the linked document
<u>target</u>	_blank _parent _self _top frameName	Specifies where to open the linked document
<u>type</u>	Mime_type	Specifies the Mime type of the linked document

Links

Link Types (attribute rel)

Rel type	Effect on... <u>link</u>	<u>a</u> and <u>area</u>	Brief description
<u>alternate</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Gives alternate representations of the current document.
<u>author</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Gives a link to the author of the current document or article.
<u>bookmark</u>	<i>not allowed</i>	<u>Hyperlink</u>	Gives the permalink for the nearest ancestor section.
<u>help</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Provides a link to context-sensitive help.
<u>icon</u>	<u>External Resource</u>	<i>not allowed</i>	Imports an icon to represent the current document.
<u>license</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Indicates that the main content of the current document is covered by the copyright license described by the referenced document.
<u>next</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Indicates that the current document is a part of a series, and that the next document in the series is the referenced document.

Links

Link Types (attribute rel)

Rel type	Effect on...	Brief description	
	<u>link</u>	<u>a</u> and <u>area</u>	
<u>nofollow</u>	<i>not allowed</i>	<u>Annotation</u>	Indicates that the current document's original author or publisher does not endorse the referenced document.
<u>noreferrer</u>	<i>not allowed</i>	<u>Annotation</u>	Requires that the user agent not send an HTTP Referer (sic) header if the user follows the hyperlink.
<u>prefetch</u>	<u>External Resource</u>	<u>External Resource</u>	Specifies that the target resource should be preemptively cached.
<u>prev</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Indicates that the current document is a part of a series, and that the previous document in the series is the referenced document.
<u>search</u>	<u>Hyperlink</u>	<u>Hyperlink</u>	Gives a link to a resource that can be used to search through the current document and its related pages.
<u>stylesheet</u>	<u>External Resource</u>	<i>not allowed</i>	Imports a stylesheet.
<u>tag</u>	<i>not allowed</i>	<u>Hyperlink</u>	Gives a tag (identified by the given address) that applies to the current document.

Links

Example 1

```
<h1>Table of Contents</h1>
```

```
<p><a href="#section1">Introduction</a><br>
```

```
<a href="#section2">Some background</a><br>
```

```
<a href="#section2.1">On a more personal note</a><br>
```

```
...the rest of the table of contents... ...the document body...</p>
```

```
<h2><a id="section1">Introduction</a></h2>
```

```
<p>...section 1...</p>
```

```
<h2><a id="section2">Some background</a></h2>
```

```
<p>...section 2...</p>
```

```
<h3><a id="section2.1">On a more personal  
note</a></h3>
```

```
<p>...section 2.1...</p>
```

Links

Example 2

We may achieve the same effect by making the header elements themselves the anchors:

```
<h1> Table of Contents</h1>
```

```
  <p>  <a href="#section1">Introduction</a><br>
```

```
    <a href="#section2">Some background</a><br>
```

```
    <a href="#section2.1">On a more personal note</a><br>
```

```
    ...the rest of the table of contents... ...the document body...
```

```
</p>
```

```
<h2 id="section1"> Introduction</h2>
```

```
  <p>...section 1...</p>
```

```
<h2 id="section2">Some background</h2>
```

```
  <p>...section 2...</p>
```

```
<h3 id="section2.1"> On a more personal note </h3>
```

```
  <p>...section 2.1...</p>
```


Links

Element <link>

Attribute	Value	Description
<u>href</u>	URL	Specifies the URL of the page the link goes to
<u>hreflang</u>	language_code	Specifies the language of the linked document
<u>media</u>	media_query	Specifies what media/device the linked document is optimized for
<u>rel</u>	alternate author bookmark help license next nofollow norereferrer prefetch prev search tag	Specifies the relationship between the current document and the linked document
<u>sizes</u>	value	Sizes of the icons (for rel="icon")
<u>type</u>	Mime_type	Specifies the Mime type of the linked document

Links

Example 3

```
<head>
```

```
<link rel="stylesheet" type="text/css" href="theme.css">
```

```
<link rel="stylesheet" type="text/css" href="print.css" media="print">
```

```
</head>
```

HTML Tables

HTML Table

- Tables are defined with the `<table>` tag
 - A table is divided into **rows** (with the `<tr>` tag), and each row is divided into data cells (with the `<td>` tag).
 - The `<caption>` element defines a **table caption**.
 - The `<caption>` element must be inserted immediately after the `<table>` tag.
 - You can specify only one caption per table.
 - Usually the caption will be centered above the table.
 - **Headings** in a table are defined with the `<th>` tag
 - The **letters td** stands for "table data" which is the content of a data cell.

HTML Table

- A **data cell** can contain text, images, lists, paragraphs, forms, tables, etc.
- If **you do not specify a border attribute** the table will be displayed without any borders
 - To display a table with borders, you will have to use the border attribute

Content Model:

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), optionally intermixed with one or more script-supporting elements.

HTML Table

```
<head>
```

```
  <meta charset="utf-8" />
```

```
  <title>The a element</title>
```

```
  <style>
```

```
    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>
```

```
</head>
```

HTML Table

```
<body>
<h1>Today's Sudoku</h1>
<table>
  <colgroup><col><col><col>
  <colgroup><col><col><col>
  <colgroup><col><col><col>
  <tbody>
    <tr><td> 1 <td>  <td> 3 <td> 6 <td>  <td> 4 <td> 7 <td>  <td> 9
    <tr><td>  <td> 2 <td>  <td>  <td> 9 <td>  <td>  <td> 1 <td>
    <tr><td> 7 <td>  <td>  <td>  <td>  <td>  <td>  <td> 6
  </tbody>
  <tbody>
    <tr><td> 2 <td>  <td> 4 <td>  <td> 3 <td>  <td> 9 <td>  <td> 8
    <tr><td>  <td>  <td>  <td>  <td>  <td>  <td>  <td>  <td>
    <tr><td> 5 <td>  <td>  <td> 9 <td>  <td> 7 <td>  <td>  <td> 1
```

HTML Table

```
<tbody>
```

```
  <tr> <td> 6 <td>  <td>  <td>  <td> 5 <td>  <td>  <td>  <td> 2
```

```
  <tr> <td>  <td>  <td>  <td>  <td> 7 <td>  <td>  <td>  <td>
```

```
  <tr> <td> 9 <td>  <td>  <td> 8 <td>  <td> 2 <td>  <td>  <td> 5
```

```
</table>
```

```
</body>
```


HTML Table

The **colgroup** element

- The **colgroup** element represents a group of one or more columns in the table that is its parent.
- 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.

HTML Table

The **colgroup** element

```
<table>
  <colgroup>
    <col span="2" style="background-color:red">
    <col style="background-color:yellow">
  </colgroup>
  <tr> <th>ISBN</th> <th>Title</th> <th>Price</th> </tr>
  <tr> <td>3476896</td> <td>My first HTML</td>
    <td>$53</td> </tr>
  <tr> <td>5869207</td> <td>My first CSS</td>
    <td>$49</td> </tr>
</table>
```

HTML Table (thead, tbody, tfoot)

- The **thead**, **tbody** and **tfoot** elements are used to group the header content, the body content and the footer content in an HTML table.
- Note: **<tfoot>** must appear before **<tbody>** within a table, so that a browser can render the foot before receiving all the rows of data.
- This division enables **user agents to support scrolling of table bodies independently of the table head and foot.**
 - When long tables are printed, the table head and foot information may be repeated on each page that contains table data.

HTML Table (thead, tbody, tfoot)

```
<body>
<table>
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>
```

HTML Table (thead, tbody, tfoot)

```
<tbody>
```

```
<tr>
```

```
<td>January</td>
```

```
<td>$100</td>
```

```
</tr>
```

```
<tr>
```

```
<td>February</td>
```

```
<td>$80</td>
```

```
</tr>
```

```
</tbody>
```

```
</table>
```

```
</body>
```




HTML Table (rowspan and colspan)

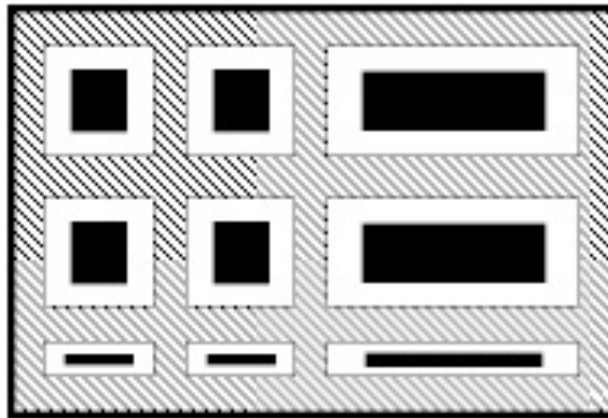
- The td and th elements may have
 - rowspan
defines the number of rows a cell should span
 - colspan
defines the number of columns a cell should span


```
<table width="100%" border="1">  
  <tr> <th>Month</th> <th>Amount</th> <th>Percentage</th></tr>  
  <tr> <td> January </td> <td colspan="2">-</td> </tr>  
  <tr> <td> February </td> <td>100</td> <td>10</td> </tr>  
</table>
```


HTML Table

- Defining the style of a table

Table border 



Cellspacing 

Cellpadding 

Cell content 

HTML Forms

HTML Forms

- A **form** is a section of a document containing normal content, markup, special elements called **controls** and labels on those controls.
- Controls:
 - Buttons
 - Checkboxes
 - Radio buttons
 - Menus
 - Text input
 - File select
 - Hidden control
 - Object controls

HTML Forms

- Users generally "complete" a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form to an agent for processing (e.g., to a Web server, to a mail server, etc.)
- A control's "control name" is given by its `name` attribute.
 - The scope of the name attribute for a control within a form element is the form element.

HTML Forms

Example:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="utf-8">
```

```
  <title>Pizza Ordering Form</title>
```

```
</head>
```

```
<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>
```

HTML Forms

```
<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>
```

HTML Forms

```
<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>
```

```
<p><button>Submit order</button></p>
```

```
</form>
```

```
</html>
```

HTML Forms

Server Communication

Example:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="utf-8">
```

```
  <title>Pizza Ordering Form</title>
```

```
</head>
```

```
<form method="post" enctype="application/x-www-form-urlencoded"
      action="https://pizza.example.com/order.php">
```

```
<p><label>Customer name: <input name="custname"></label></p>
```

```
<p><label>Telephone: <input type=tel name="custtel"></label></p>
```

```
<p><label>E-mail address: <input type=email name="custemail"></label></p>
```

HTML Forms

Server Communication

```
<fieldset>
```

```
  <legend> Pizza Size </legend>
```

```
  <p><label> <input type=radio name=size value="small"> Small </label></p>
```

```
  <p><label> <input type=radio name=size value="medium"> Medium  
  </label></p>
```

```
  <p><label> <input type=radio name=size value="large"> Large </label></p>
```

```
</fieldset>
```

HTML Forms

Server Communication

```
<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>
```


HTML Forms

Server Communication

```
<p><label>Preferred delivery time: <input type=time min="11:00" max="21:00"
step="900" name="delivery"></label></p>
<p><label>Delivery instructions: <textarea
name="comments"></textarea></label></p>
<p><button>Submit order</button></p>
</form>
```

HTML Forms

Server Communication

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&deliv  
ery=19%3A00&comments=
```

HTML Forms

Client-side form validation

- Forms can be annotated in such a way that the user agent will check the user's input before the form is submitted.
- 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.

```
<p><label>Customer name: <input name="custname" required></label></p>
```

...

```
<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>
```

..

```
<p><label>Preferred delivery time: <input type="time" min="11:00" max="21:00" step="900" name="delivery" required></label></p>
```

HTML Forms

Client-side form validation

- It is also possible to limit the length of the input, using the maxlength attribute.

```
<p><label>Delivery instructions: <textarea name="comments"
maxlength=1000></textarea></label></p>
```

HTML Forms

- The form element acts as a container for controls, by specifying:
 - The layout of the form (given by the contents of the element).
 - The program that will handle the completed and submitted form (the action attribute). The receiving program must be able to parse name/value pairs in order to make use of them.
 - The method by which user data will be sent to the server (the method attribute).
 - A character encoding that must be accepted by the server in order to handle this form (the accept-charset attribute). User agents may advise the user of the value of the accept-charset attribute and/or restrict the user's ability to enter unrecognized characters.

HTML Forms

- In general, a control's "initial value" may be specified with the control element's **value** attribute.
- However, the initial value of a **textarea element** is given by its contents
- **A control's initial value does not change.** Thus, when a form is reset, each control's current value is reset to its initial value

HTML Forms

- Attributes of the form element

Attribute	Value	Description
action	<i>URL</i>	Specifies where to send the form-data when a form is submitted
accept-charset	<i>charset</i>	Specifies the character-sets the server can handle for form-data
autocomplete	<i>on,off</i>	Specifies if the autofill is on or off
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Specifies how form-data should be encoded before sending it to a server (multipart/form-data text/plain is only for POST)
method	get post	Specifies how to send form-data
name	<i>name</i>	Specifies the name for a form
novalidate	<i>on,off</i>	Specifies whether the form has to be validated or not
target	<i>_blank</i> new window <i>_self</i> same frame <i>_top</i> full body of the window <i>iframe</i> in a named <i>iframe</i>	The target attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.



HTML Forms

- If the method is "get", the user agent takes the value of action, appends a ? to it, then appends the form data set, encoded using the application/x-www-form-urlencoded content type. The user agent then traverses the link to this URI. In this scenario, form data are restricted to ASCII codes.
- If the method is "post", the user agent conducts an HTTP post transaction using the value of the action attribute and a message created according to the content type specified by the enctype attribute.

HTML Forms: Control Types

- **Buttons types**
 - **submit:** When activated, a submit button submits a form. A form may contain more than one submit button.
 - **reset:** When activated, a reset button resets all controls to their initial values.
 - **buttons:** Buttons have no default behavior.
 - Each button may have client-side scripts associated with the element's event attributes.

HTML Forms: Control Types

- Checkboxes

- Checkboxes (and radio buttons) are on/off switches that may be toggled by the user. A switch is "on" when the control element's checked attribute is set.
- When a form is submitted, only "on" checkbox controls can become successful.
- Several checkboxes in a form may share the same control name. Thus, for example, checkboxes allow users to select several values for the same property.
- The input element is used to create a checkbox control.

HTML Forms: Control Types

- Radio Buttons

- Radio buttons are like checkboxes **except that when several share the same control name, they are mutually exclusive:** when one is switched "on", all others with the same name are switched "off".
- **The input element** is used to create a radio button control.

- Menus

- Menus offer users options from which to choose.
- **The select element** creates a menu, in combination with the **optgroup** and **option** elements.

HTML Forms: Control Types

- Text input
 - Authors may create two types of controls that allow users to input text.
 - The `input` element creates a single-line input control and the `textarea` element creates a multi-line input control.
 - In both cases, the input text becomes the control's current value
- File Select
 - This control type allows the user to select files so that their contents may be submitted with a form.
 - The `input` element is used to create a file select control.

HTML Forms: Control Types

- How are the control types implemented?
 - `<input>` element
 - `<button>` element
 - `<select>` element
 - `<datalist>` element
 - `<optgroup>` element
 - `<option>` element
 - `<textarea>` element
 - `<keygen>` element
 - `<output>` element
 - `<progress>` element
 - `<meter>` element
 - `<fieldset>` element
 - `<legend>` element

HTML Forms

Input Element

The `type` attribute of the Input Element

Keyword	State	Data type	Control type
hidden	<u>Hidden</u>	An arbitrary string	n/a
text	<u>Text</u>	Text with no line breaks	A text field
search	<u>Search</u>	Text with no line breaks	Search field
tel	<u>Telephone</u>	Text with no line breaks	A text field
url	<u>URL</u>	An absolute URL	A text field

HTML Forms

Input Element

The type attribute

Keyword	State	Data type	Control type
email	<u>E-mail</u>	An e-mail address or list of e-mail addresses	A text field
password	<u>Password</u>	Text with no line breaks (sensitive information)	A text field that obscures data entry
date	<u>Date</u>	A date (year, month, day) with no time zone	A date control
time	<u>Time</u>	A time (hour, minute, seconds, fractional seconds) with no time zone	A time control

HTML Forms

Input Element

The type attribute

Keyword	State	Data type	Control type
number	<u>Number</u>	A numerical value	A text field or spinner control
range	<u>Range</u>	A numerical value, with the extra semantic that the exact value is not important	A slider control or similar
color	<u>Color</u>	An sRGB color with 8-bit red, green, and blue components	A color well
checkbox	<u>Checkbox</u>	A set of zero or more values from a predefined list	A checkbox

HTML Forms

Input Element

The type attribute

Keyword	State	Data type	Control type
radio	Radio Button	An enumerated value	A radio button
file	File Upload	Zero or more files each with a MIME type and optionally a file name	A label and a button
submit	Submit Button	An enumerated value, with the extra semantic that it must be the last value selected and initiates form submission	A button
image	Image Button	A coordinate, relative to a particular image's size, with the extra semantic that it must be the last value selected and initiates form submission	Either a clickable image, or a button

HTML Forms

Input Element

The type attribute

Keyword	State	Data type	Control type
reset	<u>Reset Button</u>	n/a	A button
button	<u>Button</u>	n/a	A button

HTML Forms

Input Element

Input element attributes.

Different attributes depending on the specific value of attribute type.

List of attributes and their use can be found in the HTML5 specification

<http://www.w3.org/TR/html5/forms.html#the-input-element>

HTML Forms

Input Element

Some examples of specific attributes

- The `list` attribute is used to identify an element that lists predefined options suggested to the user.
- If present, its value must be the ID of a `datalist` element in the same document.

```
<label>Homepage: <input name=hp type=url list=hpurls></label>  
<datalist id=hpurls>  
  <option value="http://www.google.com/" label="Google">  
  <option value="http://www.reddit.com/" label="Reddit">  
</datalist>
```

HTML Forms

Input Element

Some examples of specific attributes

- The **placeholder** attribute represents a short hint (a word or short phrase) intended to aid the user with data entry when the control has no value.

```
<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>
```

HTML Forms

Input Element

Some examples of specific attributes

- The **readonly** attribute is a boolean attribute that controls whether or not the user can edit the form control.
- The **required** attribute is a boolean attribute. When specified, the element is required.
- The **pattern** attribute specifies a regular expression against which the control's value, or, when the multiple attribute applies and is set, the control's values, are to be checked.
- The **min** and **max** attributes indicate the allowed range of values for the element.
- The **step** attribute indicates the granularity that is expected (and required) of the value, by limiting the allowed values.

HTML Forms

Input Element

Example:

```
<form action="products.php" 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>
```

HTML Forms

Input Element

```
<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>
```


HTML Forms

Input Element

```
<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>
```

HTML Forms

Input Element

Some examples of specific attributes

- The **multiple** attribute is a boolean attribute that indicates whether the user is to be allowed to specify more than one value.
 - The multiple attribute works with the following input types: email, and file.
 - For `<input type="file">`: to select multiple files, hold down the CTRL or SHIFT key while selecting.
 - For `<input type="email">`: separate each email with a comma, like: mail@example.com, mail2@example.com, mail3@example.com in the email field.

```
<form action="demo_form.php">
```

```
  Select images: <input type="file" name="img" multiple>
```

```
  <input type="submit">
```

```
</form>
```

HTML Forms

Button Element

- Buttons created with the **button element** function just like buttons created with the input element, but they **offer richer rendering possibilities**:
 - **the button element may have content**. For example, a button element that contains an image functions like and may resemble an input element whose type is set to "image", but the button element type allows content.
- Always specify the type attribute for the button: submit, reset, button (does nothing).

HTML Forms

Button Element

Main attributes of the button element

- **autofocus**: allows the author to indicate that a control is to be focused as soon as the page is loaded, allowing the user to just start typing without having to manually focus the main control.
- **disabled**
- **form**: to explicitly associate the button element with its form owner (the value is the ID of the form)
- **formnovalidate**: indicates that the form is not to be validated during submission
- **value**: 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.

HTML Forms

Button Element

```
<form action="http://somesite.com/prog/adduser" method="post">
  <p>
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname"><br>
    email: <input type="text" name="email"><br>
    <input type="radio" name="sex" value="Male"> Male<br>
    <input type="radio" name="sex" value="Female"> Female<br>
    <button name="submit" value="submit" type="submit"> Send
    </button>
    <button name="reset" type="reset">Reset
    </button>
  </p>
</form>
```

HTML Forms

select, optgroup and option elements

- The **select** element creates a menu.
 - Each choice offered by the menu is represented by an **option** element.
 - A select element must contain at least one option element.
- The **optgroup** element allows authors to group choices logically. The label value gives the name of the group.
 - Helpful when the user must choose from a long list of options; groups of related choices are easier to grasp and remember than a single long list of options.
- The **option** element represents an option in a select element or as part of a list of suggestions in a datalist element. The selected attribute represents the default selectedness of the element

HTML Forms

select, optgroup and option elements

<select> attributes

- **disabled** – specifies that a drop-down list should be disabled
- **multiple** – specifies that multiple options can be selected
- **size** – gives the number of options to show to the user
- **required** – the user is required to select a value

HTML Forms

select, optgroup and option elements

- When rendering a menu choice, user agents should use the value of the label attribute of the option element as the choice.
 - If this attribute is not specified, user agents should use the contents of the option element.
- The label attribute of the optgroup element specifies the label for a group of choices
- Only selected options will be successful (using the control name "c" in the following example).
- When no options are selected, the control is not successful and neither the name nor any values are submitted to the server when the form is submitted.

HTML Forms

select, optgroup and option elements

- Example of the <optgroup> element

```
<form action="http://www.watch.com/watch.php" method="get">  
<p>Which course would you like to watch today?  
<p><label>Course:  
<select name="c">  
  <optgroup label="8.01 Physics I: Classical Mechanics">  
    <option value="8.01.1">Lecture 01: Powers of Ten  
    <option disabled value="8.01.2">Lecture 02: 1D Kinematics  
    <option value="8.01.3">Lecture 03: Vectors  
  <optgroup disabled label="8.02 Electricity and Magnetism">
```

HTML Forms

select, optgroup and option elements

```
<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  
</select>  
</label>  
<p><input type="submit" value="Play">  
</form>
```

HTML Forms

Textarea element

- The textarea element **creates a multi-line text input control**. User agents should use the contents of this element as the initial value of the control and should render this text initially.
- The **cols** and **rows** attribute specify the expected maximum number of characters per line and the number of lines to show, respectively.
- The **maxlength** attribute specifies the allowed maximum length.

HTML Forms

Textarea element

```
<form action="http://www.elaborate.com/elab.php"  
  method="get">
```

```
<p>
```

```
<textarea name="thetext" rows="10" cols="80"  
  maxlength="20">
```

```
</textarea>
```

```
<input type="submit" value="Send">
```

```
<input type="reset" value="Reset"></p>
```

```
</form>
```

HTML Forms

Label element

- Some form controls automatically have labels associated with them (press buttons) while most do not (text fields, checkboxes and radio buttons, and menus).
- For those controls that have implicit labels, user agents should use the value of the “value” attribute as the label string.
- The label element is used to specify labels for controls that do not have implicit labels.

HTML Forms

Label element

- The **label** element 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 the **for** attribute, or **by putting the form control inside the label element itself**.
- The **for** attribute explicitly **associates the label being defined with another control**. **When present**, the value of this attribute must be the same as the value of the **id** attribute of some other control in the same document. **When absent**, the label being defined is associated with the element's contents.
- More than one label may be associated with the same control by creating multiple references via the **for** attribute.
- When a **LABEL** element receives focus, it passes the focus on to its associated control.

HTML Forms

Label element

```
<form action="http://somesite.com/prog/adduser" method="post">  
<p><label for="firstname">First name:</label> <input type="text"  
id="firstname"><br>  
<label for="lastname">Last name:</label> <input type="text" id=  
"lastname"><br>  
<label for="email">email:</label> <input type="text" id=  
"email"><br>  
<input type="radio" name="sex" value="Male"> Male<br>  
<input type="radio" name="sex" value="Female"> Female<br>  
<input type="submit" value="Send"> <input type="reset"></p>  
</form>
```

HTML Forms

Label element

- To associate a label with another control implicitly, the control element must be within the contents of the label element.
 - In this case, the label may only contain one control element.
 - The label itself may be positioned before or after the associated control.

```
<form action="..." method="post">  
<p><label>First Name <input type="text"  
  name="firstname"></label>  
<label><input type="text" name="lastname"> Last  
  Name</label></p>  
</form>
```


HTML Forms

Focus to an element

- There are several ways to give focus to an element:
 - Designate the element with a pointing device.
 - Navigate from one element to the next with the keyboard.
 - The document's author may define a *tabbing order* that specifies the order in which elements will receive focus if the user navigates the document with the keyboard. Once selected, an element may be activated by some other key sequence.
 - Select an element through an *access key* (sometimes called "keyboard shortcut" or "keyboard accelerator").

HTML Forms

Tabbing navigation

`tabindex = number`

- The `tabindex` attribute specifies the position of the current element in the tabbing order for the current document.
 - Number is between 0 and 32767.
 - Values **need not** be **sequential** nor must they begin with any particular value.
 - The **tabbing order may include elements nested within other elements**.
 - The following elements support the `tabindex` attribute: `a`, `area`, `button`, `input`, `object`, `select`, and `textarea`.

HTML Forms

Tabbing navigation

- Elements should be navigated by user agents according to the following rules:
 1. First, elements that support the **tabindex** attribute and assign a positive value to it are navigated:
 - from the element with the lowest **tabindex** value to the element with the highest value in the order they appear in the character stream if have identical **tabindex** values
 2. Then, elements that do not support the **tabindex** attribute or support it and assign it a value of "0" are navigated next. These elements are navigated in the order they appear in the character stream.
 3. Elements that are disabled do not participate in the tabbing order.

HTML Forms

Tabbing navigation

```
<p>Go to the <a tabindex="20"
  href="http://www.w3.org/">W3C Web site. </a> ...some
  more... <button type="button" name=
  "get-database" tabindex="18" onclick="get-
  database">Get the
current database.</button> ...some more...</p>
<form action="..." method="post">
<p><input tabindex="1" type="text" name="field1">
<input tabindex="1" type="text" name="field2"> <input
  tabindex=
"2" type="submit" name="submit"></p>
</form>
```

HTML Forms

Access key

`accesskey = character`

- This attribute assigns an access key to an element. An access key is a single character from the document character set.
- Pressing an access key assigned to an element gives focus to the element. The action that occurs when an element receives focus depends on the element.
- The following elements support the accesskey attribute: a, area, button, input, label, legend, and textarea

HTML Forms

Access key

```
<form action="..." method="post">  
<p><label for="fuser" accesskey="u">User  
  Name</label>  
<input type="text" name="user" id="fuser"></p>  
</form>
```

- The invocation of access keys depends on the underlying system. For instance, on machines running MS Windows, one generally has to press the "alt" key in addition to the access key. On Apple systems, one generally has to press the "cmd" key in addition to the access key.

HTML Forms

Disabled controls

Disabled

- When set for a form control, **this boolean attribute disables the control for user input.**
 - Disabled controls do not receive focus
 - Disabled controls are skipped in tabbing navigation
 - Disabled controls cannot be successful.
- The following elements support the disabled attribute: **button, input, optgroup, option, select and textarea.**

`<input disabled name="fred" value="stone">`

HTML Forms

readonly controls

Readonly

- When set for a form control, **this boolean attribute prohibits changes to the control.**
 - Read-only elements receive focus but cannot be modified by the user.
 - Read-only elements are included in tabbing navigation
 - Read-only elements may be successful
- The following elements support the readonly attribute: **input and textarea.**

```
<input readonly name="fred" value="stone">
```


HTML Forms

Form submission

- A *successful control* is "valid" for submission. Every successful control has its control name paired with its current value as part of the submitted form data set.
- Note that
 - Controls that are disabled cannot be successful.
 - **Submit button:** If a form contains more than one submit button, only the activated submit button is successful.
 - **Checkboxes:** All "on" checkboxes may be successful.
 - **Radio Buttons:** For radio buttons that share the same value of the name attribute, **only the "on" radio button may be successful.**

HTML Forms

Form submission

- **Menus:** the control name is provided by a select element and values are provided by option elements. Only selected options may be successful. When no options are selected, the control is not successful and neither the name nor any values are submitted to the server when the form is submitted.
- **File select:** The current value of a file select is a list of one or more file names. Upon submission of the form, the *contents* of each file are submitted with the rest of the form data. The file contents are packaged according to the form's content type.
- **Object control:** The current value of an object control is determined by the object's implementation

HTML Forms

Form submission

- If a control does not have a current value when the form is submitted, user agents are not required to treat it as a successful control.
- Furthermore, user agents should not consider the following controls successful:
 - Reset buttons.
- Hidden controls and controls that are not rendered because of style sheet settings may still be successful. For example

HTML Forms

Form submission

```
<form action="..." method="post">  
<p><input type="password"  
  style="display:none"  
  name="invisible-password"  
  value="mypassword"></p>  
</form>
```

will still cause a value to be paired with the name "invisible-password" and submitted with the form

- Note: this mechanism affords only light security protection. Although the password is masked by user agents from casual observers, **it is transmitted to the server in clear text.**

HTML Forms

Processing form data

The user agent processes a form as follows:

- 1. Identify the successful controls**
- 2. Build a form data set**

A form data set is a sequence of control-name /current-value pairs constructed from successful controls

- 3. Encode the form data set**

The form data set is then **encoded according to the content type** specified by the enctype attribute of the FORM element.

- 4. Submit the encoded form data set**
- 5. Send to the Processing agent**

The encoded data is sent to the processing agent designated by the action attribute using the protocol specified by the method attribute.

HTML Forms

Processing form data

HTML 5 user agents must support the established conventions in the following cases:

- **Method = "get" and action = HTTP URI**
 - The user agent
 - takes the value of action,
 - appends a '?' to it,
 - appends the form data set, encoded using the "application/x-www-form-urlencoded" content type.
 - The user agent then traverses the link to the specified URI.
 - In this scenario, form data are restricted to ASCII codes.

HTML Forms

Processing form data

Example method “get”

```
<form action="/find.php" method=get>  
  <input type=text name=t>  
  <input type=search name=q>  
  <input type=submit>  
</form>
```

- The user agent will load
 /find.php?t=cats&q=fur.

HTML Forms

Processing form data

- Method = "post" and action = HTTP URI
 - The user agent conducts an HTTP "post" transaction using the value of the action attribute and a message created according to the content type specified by the enctype attribute.
- For any other value of action or method, behavior is unspecified.

HTML Forms

Form content type

- The **enctype** attribute of the **FORM** element specifies the **content type** used to **encode** the form data set for submission to the server.

- **application/x-www-form-urlencoded**

This is the default content type. Forms submitted with this content type must be encoded as follows:

- Control names and values are escaped. Space characters are replaced by '+', and then reserved characters are escaped: Non-alphanumeric characters are replaced by '%HH', a percent sign and two hexadecimal digits representing the ASCII code of the character. Line breaks are represented as "CR LF" pairs (i.e., '%0D%0A').
- The **control names/values** are listed in the order they appear in the document. The name is separated from the value by '=' and name/value pairs are separated from each other by '&'.

HTML Forms

Form content type

- **multipart/form-data**

This content type should be used for submitting forms that contain **files, non-ASCII data, and binary data**.

A "multipart/form-data" message **contains a series of parts, each representing a successful control**. The parts are sent to the processing agent in the same order the corresponding controls appear in the document stream.

Each part has an optional "Content-Type" header that defaults to "text/plain". User agents should supply the "Content-Type" header, accompanied by a "charset" parameter.

HTML Forms

Form content type

- **multipart/form-data** (continued)

Each part is expected to contain:

- a "Content-Disposition" header whose value is "form-data".
- a name attribute specifying the control name of the corresponding control.

Thus, for example, for a control named "mycontrol", the corresponding part would be specified:

Content-Disposition: form-data; name="mycontrol"

As with all MIME transmissions, "CR LF" (i.e., '%0D%0A') is used to separate lines of data.

HTML Forms

Form content type

Suppose we have the following form:

```
<form action="http://server.com/cgi/handle" enctype="multipart/form-data" method="post">
```

```
<p>What is your name? <input type="text" name="submit-name"><br>
```

```
What files are you sending? <input type="file" name="files"><br>
```

```
<input type="submit" value="Send"> <input type="reset"></p>
```

```
</form>
```

HTML Forms

Form content type

The user agent might send back:

Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x

Content-Disposition: form-data; name="submit-name"

Larry

--AaB03x

Content-Disposition: form-data; name="files"; filename="file1.txt"

Content-Type: text/plain

... contents of file1.txt ...

--AaB03x--

HTML Forms

Form content type

If the user selected a second (image) file "file2.gif":

Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x

Content-Disposition: form-data; name="submit-name"

Larry

--AaB03x

Content-Disposition: form-data; name="files"

Content-Type: multipart/mixed; boundary=BbC04y

HTML Forms

Form content type

--BbC04y

Content-Disposition: file; filename="file1.txt"

Content-Type: text/plain

... contents of file1.txt ...

--BbC04y

Content-Disposition: file; filename="file2.gif"

Content-Type: image/gif

Content-Transfer-Encoding: binary

...contents of file2.gif...

--BbC04y--

--AaB03x--

Checking Forms with validation

- **Form validation** is an optimization because it alone is not sufficient to guarantee that forms submitted to the server are correct and valid.
- It is an optimization because it is designed to help a **web application fail fast**. In other words, it is better to notify a user that a page contains invalid form controls right inside the page, using the browser's built-in processing.

Checking Forms using HTML 5.0 and CSS

- Example from MDN

```
<style type="text/css">
form {
  font: 1em sans-serif;
  max-width: 320px;
}

p > label {
  display: block;
}

input:invalid {
  box-shadow: 0 0 5px 1px red;
}
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN

```
input[type=text],  
input[type=email],  
input[type=number],  
textarea,  
fieldset {  
  width : 100%;  
  border: 1px solid #333;  
  box-sizing: border-box;  
}  
  
input:focus:invalid {  
  box-shadow: none;  
}  
  
</style>
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN

```
<p>
  <fieldset>
    <legend>Title<abbr title="This field is mandatory">*</abbr></legend>
    <input type="radio" required name="title" id="r1" value="Mr"><label
for="r1">Mr.</label>
    <input type="radio" required name="title" id="r2" value="Ms"><label
for="r2">Ms.</label>
  </fieldset>
</p>
<p>
  <label for="n1">How old are you?</label>
  <input type="number" min="12" max="120" step="1" id="n1" name="age"
pattern="\d+">
</p>
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN

```
<p>
```

```
<label for="t1">What's your favorite fruit?<abbr title="This field is  
mandatory">*</abbr></label>
```

```
<input type="text" id="t1" name="fruit" list="l1" required  
pattern="[Bb]anana|[Cc]herry|[Aa]pple|[Ss]trawberry|[Ll]emon|[Oo]range">
```

```
<datalist id="l1">
```

```
<option>Banana</option>
```

```
<option>Cherry</option>
```

```
<option>Apple</option>
```

```
<option>Strawberry</option>
```

```
<option>Lemon</option>
```

```
<option>Orange</option>
```

```
</datalist>
```

```
</p>
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN

```
<p>
```

```
<label for="t2">What's your e-mail?</label>
```

```
<input type="email" id="t2" name="email">
```

```
</p>
```

```
<p>
```

```
<label for="t3">Leave a short message</label>
```

```
<textarea id="t3" name="msg" maxlength="140" rows="5"></textarea>
```

```
</p>
```

```
<p>
```

```
<button>Submit</button>
```

```
</p>
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN: Customised error messages

```
<script type="text/javascript">
  function begin()
  {
    var email = document.getElementById("t2");
    email.addEventListener("input", function (event) {
      if (email.validity.typeMismatch) {
        email.setCustomValidity("I expect an e-mail, darling!");
      } else {
        email.setCustomValidity("");
      }
    });
  }
</script>
```

Checking Forms with validation

- HTML5 introduces eight ways to enforce correctness on form controls.
- Object `ValidityState` allows accessing their validity status

```
var valCheck = document.aForm.anInput.validity;
```

`valCheck` contains a reference to the `ValidityState` object of the form element named `myInput`.

`valCheck.valid`

a Boolean value which informs us whether or not all validity constraints are currently met on this form control.

If all eight constraints are passing, the valid flag will be true. False otherwise.

Checking Forms

Validity constraints (1)

The `valCheck.valueMissing` constraint

- Ensures that some value is set on this form control
- Set the `required` attribute on the form control to add this check
- Example: `<input type="text" name="myText" required>`
- If the `required` attribute is set on a form control, the control will be in an invalid state unless the user or a programmatic call puts a value into the field.
- For example, a blank text field will fail a required check, but will pass as soon as any text is entered. When blank, the `valueMissing` will return true.

Checking Forms

Validity constraints (2)

The `valCheck.typeMismatch` constraint

- The type of the value matches expectations (number, email, URL, and so on)
- Specify one of the appropriate type attributes on the form control
- Example: `<input type="email" name="myEmail">`
- Special form control types customize a phone keyboard. If the value entered into a form control doesn't conform to the rules for that type (e.g. an email address without an @ symbol) the browser can flag this control as having a type mismatch. Another example would be a number field that cannot parse to a valid number. In either case, the `typeMismatch` will return true.

Checking Forms

Validity constraints (3)

The `valCheck.patternMismatch` constraint

- Checks that the value corresponds to the pattern rule
- Add the `pattern` attribute on the form control to enable this check
- Example: `<input type="text" name="creditcardnumber" pattern="[0-9]{16}" title="A credit card number is 16 digits, no spaces">`
- To assist users and assistive technology, you should set the title on any pattern-controlled field to make clear the rules of the format.

Checking Forms

Validity constraints (4)

The `valCheck.maxLength` constraint

- Checks that a value does not contain too many characters
- Add a `maxLength` attribute on the form control to enable this check
- Example: `<input type="text" name="sms" maxLength="140">`
- true if the value length is greater than `maxLength`. Form controls will generally try to enforce the maximum length during user entry, certain situations including programmatic settings can cause the value to exceed the maximum.

Checking Forms

Validity constraints (5)

The `valCheck.rangeUnderflow` constraint

- Limits the minimum value of a numeric control
- Add a `min` attribute with the minimum allowed value to enable this check
- Example: `<input type="range" name="quantity" min="1">`
- It is possible for the value to get temporarily set below the allowable range. In these cases, the `ValidityState` will return true for the `rangeUnderflow` field.

Checking Forms

Validity constraints (6)

The `valCheck.rangeOverflow` constraint

- Limits the maximum value of a numeric control
- Set a `max` attribute with the maximum allowed value to enable this check
- Example: `<input type="range" name="maxTickets" max="4">`
- This validity constraint will return true if the value of a form control is greater than the max attribute.

Checking Forms

Validity constraints (7)

The `valCheck.stepMismatch` constraint

- Guarantee that a value conforms to the specified min, max, and step
- Set a step attribute to enable this check
- Usage example: `<input type="range" name="money" min="0" max="100" step="10">`
- The value must be a multiple of the step added to the minimum value. For the above example a value of 33 makes `stepMismatch` return true.

Checking Forms

Validity constraints (8)

The `valCheck.customError` constraint

- Indicates if the element's custom validity message has been set to a non-empty string by calling the element's `setCustomValidity()` method
- Call `setCustomValidity(message)` to put a form control into the `customError` state
- Example: `pwdChangeField.setCustomValidity("Password values are different.")`
- Application code should set a custom validity message whenever a field does not conform to the application rules.

Checking Forms

Validity fields and functions

The willValidate Attribute of the HTMLInputElement

- Indicates whether validation will be checked on this form control. If any of the above constraints—e.g. the required attribute, pattern attribute, etc.—are set on the control, the willValidate field will let you know that validation checking will be enforced.

The checkValidity Function of the HTMLInputElement

- The checkValidity function allows you to check validation on the form without any explicit user input. In general, a form's validation is done whenever the user or script code submits the form. This function allows validation to be done at any time.

Checking Forms

Validity fields and functions

The `validationMessage` attribute of the `HTMLInputElement`

- a localized error message that the browser would display based on the current state of validation. For example, if a required field has no value, the browser might present an error message to the user that “This field requires a value.”

Checking Forms

Validity feedback

- The specification does not dictate the terms of how the user interface is updated to present an error message, and existing implementations differ significantly.
- Any form in an invalid state will deliver an invalid event.
- This event can be ignored, observed, or even cancelled.
- To add an event handler to a field which will receive this notification

Checking Forms

Event Handler for Invalid Events

```
function invalidHandler(evt) {  
    var validity = evt.target.validity;  
    // check the validity to see if a particular constraint failed  
    if (validity.valueMissing) {  
        // present a UI to the user indicating  
        //that the field is missing a value  
    }  
    // If you do not want the browser to provide  
    // default validation feedback,  
    // cancel the event as shown here  
    evt.preventDefault();  
}  
  
// register an event listener for "invalid" events  
myField.addEventListener("invalid", invalidHandler, false);
```

Checking Forms

Event Handler for Invalid Events (Ex.)

```
function invalidHandler(evt) {  
    // find the label for this form control  
    var label =  
        evt.target.parentElement.getElementsByTagName("label")[0];  
    // set the label's text color to red  
    label.style.color = 'red';  
    // stop the event from propagating higher  
    evt.stopPropagation();  
    // stop the browser's default handling  
    // of the validation error  
    evt.preventDefault();  
}  
  
// register an event listener for "invalid" events  
myField.addEventListener("invalid", invalidHandler, false);
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN

<form novalidate>

- The `novalidate` attribute turns off the browser's automatic validation
- This lets the script take control over validation.
- This doesn't disable support for the constraint validation API nor the application of the CSS pseudo-class `:valid`, `:invalid`, `:in-range` and `:out-of-range` classes.
- That means that even though the browser doesn't automatically check the validity of the form before sending its data, you can still do it yourself and style the form accordingly.

Checking Forms using HTML 5.0 and CSS

- Example from MDN: Javascript handling

```
<style>
  body {
    font: 1em sans-serif;
    padding: 0;
    margin: 0;
  }
  form {
    max-width: 200px;
  }
  p * {
    display: block;
  }
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN: Javascript handling

```
input[type=email] {  
    width: 100%;  
    border: 1px solid #333;  
    margin: 0;
```

```
    font-family: inherit;  
    font-size: 90%;
```

```
    box-sizing: border-box;  
}
```

```
/* This is our style for the invalid fields */  
input:invalid {  
    border-color: #900;  
    background-color: #FDD;  
}
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN: Javascript handling

```
/* This is the style of our error messages */  
.error {  
    width: 100%;  
    padding: 0;  
  
    font-size: 80%;  
    color: white;  
    background-color: #900;  
    border-radius: 0 0 5px 5px;  
    box-sizing: border-box;  
}  
  
.error.active {  
    padding: 0.3em;  
}  
</style>
```


Checking Forms using HTML 5.0 and CSS

- Example from MDN: Javascript handling

```
<script>
```

```
function begin() {
```

```
    // The form itself and the email input box
```

```
    const form = document.getElementsByTagName('form')[0];
```

```
    const email = document.getElementById('mail');
```

```
    // the span element into which we will place the error message
```

```
    const error = document.querySelector('.error');
```

```
    email.addEventListener("input", function (event) {
```

```
        // Each time the user types something, we check if the
```

```
        // email field is valid.
```

```
        if (email.validity.valid) {
```

```
            // In case there is an error message visible, if the field
```

```
            // is valid, we remove the error message.
```

```
            error.innerHTML = ""; // Reset the content of the message
```

```
            error.className = "error"; // Reset the visual state of the mess
```

```
        }
```

```
    }, false);
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN: Javascript handling

```
form.addEventListener("submit", function (event) {  
    // Each time the user tries to send the data, we check  
    // if the email field is valid.  
    if (!email.validity.valid) {  
        // If the field is not valid, we display a custom  
        // error message.  
        error.innerHTML = "I expect an e-mail, darling!";  
        error.className = "error active";  
        // Prevent the form from being sent by canceling the event  
        event.preventDefault();  
    }  
}, false);  
</script>
```

Checking Forms using HTML 5.0 and CSS

- Example from MDN: Javascript handling

```
<body onload="begin()">
  <form novalidate>
    <p>
      <label for="mail">
        <span>Please enter an email address:</span>
        <input type="email" id="mail" name="mail">
        <span class="error" aria-live="polite"></span>
      </label>
    </p>
    <button>Submit</button>
  </form>
</body>
```