HTML Headings

« Previous

Next Chapter »

Headings are important in HTML documents.

HTML Headings

Headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

Example

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

Try it Yourself »

Note: Browsers automatically add some empty space (a margin) before and after each heading.

Headings Are Important

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.

Users skim your pages by its headings. It is important to use headings to show the document structure.

h1 headings should be main headings, followed by h2 headings, then the less important h3, and so on.

HTML Horizontal Rules

The **<hr>** tag creates a horizontal line in an HTML page.

The hr element can be used to separate content:

Example

```
<p>This is a paragraph.</p>
<hr>
<p>This is a paragraph.</p>
<hr>
<p>This is a paragraph.</p>
```

Try it Yourself »

The HTML <head> Element

The HTML **<head>** element has nothing to do with HTML headings.

The HTML <head> element contains **meta data**. Meta data are not displayed.

The HTML <head> element is placed between the <html> tag and the <body> tag:

Example

```
<!DOCTYPE html>
<html>

<head>
  <title>My First HTML</title>
  <meta charset="UTF-8">
</head>

<body>
.
.
.
```

Try it Yourself »

Meta data means data **about** data. HTML meta data is data **about** the HTML document.

The HTML <title> Element

The HTML **<title>** element is meta data. It defines the HTML document's title.

The title will not be displayed in the document, but might be displayed in the browser tab.

The HTML <meta> Element

The HTML **<meta>** element is also meta data.

It can be used to define the character set, and other information about the HTML document.

More Meta Elements

In the chapter about HTML styles you discover more meta elements:

The HTML **<style>** element is used to define internal CSS style sheets.

The HTML **<link>** element is used to define external CSS style sheets.

HTML Tip - How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?"

To find out, right-click in the page and select "View Page Source" (in Chrome) or "View Source" (in IE), or similar in another browser. This will open a window containing the HTML code of the page.

Test Yourself with Exercises!

HTML Tag Reference

W3Schools' tag reference contains additional information about these tags and their attributes.

You will learn more about HTML tags and attributes in the next chapters of this tutorial.

Tag	Description
<html>	Defines an HTML document
<body>	Defines the document's body

HTML Paragraphs

« Previous

Next Chapter »

HTML documents are divided into paragraphs.

HTML Paragraphs

The HTML **<p>** element defines a **paragraph**.

Example

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Try it Yourself »

Browsers automatically add an empty line before and after a paragraph.

HTML Display

You cannot be sure how HTML will be displayed.

Large or small screens, and resized windows will create different results.

With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code.

The browser will remove extra spaces and extra lines when the page is displayed.

Any number of spaces, and any number of new lines, count as **only one space**.

Example

```
<p>
This paragraph
contains a lot of lines
in the source code,
but the browser
ignores it.
```

```
</p>

<p>
This paragraph
contains         a lot of spaces
in the source          code,
but the        browser
ignores it.
</p>
```

Try it Yourself »

Don't Forget the End Tag

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

Example

```
<p>This is a paragraph
<p>This is another paragraph
```

Try it Yourself »

The example above will work in most browsers, but do not rely on it.

Forgetting the end tag can produce unexpected results or errors.

Stricter versions of HTML, like XHTML, do not allow you to skip the end tag.

HTML Line Breaks

The HTML **
** element defines a **line break**.

Use <br> if you want a line break (a new line) without starting a new paragraph:

Example

```
<p>This is<br>a para<br>graph with line breaks</p>
```

Try it Yourself »

The <br> element is an empty HTML element. It has no end tag.

The Poem Problem

Example

```
<p>This poem will display as one line:</p>
<p>
  My Bonnie lies over the ocean.

  My Bonnie lies over the sea.

  My Bonnie lies over the ocean.

  Oh, bring back my Bonnie to me.
</p>
```

Try it Yourself »

The HTML <pre> Element

The HTML <pre> element defines preformatted text.

The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

Example

```
<pre>
  My Bonnie lies over the ocean.

  My Bonnie lies over the sea.

  My Bonnie lies over the ocean.

  Oh, bring back my Bonnie to me.
</pre>
```

Try it Yourself »

Test Yourself with Exercises!

HTML Tag Reference

W3Schools' tag reference contains additional information about HTML elements and their attributes.

Tag	Description
<p>	Defines a paragraph
 	Inserts a single line break
<pre>	Defines pre-formatted text

« Previous

Next Chapter »

HTML Styles

« Previous

Next Chapter »

I am Red

I am Blue

Try it Yourself »

HTML Styling

Every HTML element has a **default style** (background color is white and text color is black).

Changing the default style of an HTML element, can be done with the **style attribute**.

This example changes the default background color from white to lightgrey:

Example

```
<body style="background-color:lightgrey">

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

```
</body>
```

Try it Yourself »

The bgcolor attribute, supported in older versions of HTML, is not valid in HTML5.

The HTML Style Attribute

The HTML style attribute has the following **syntax**:

`style="property:value"`

The ***property*** is a CSS property. The ***value*** is a CSS value.

You will learn more about CSS later in this tutorial.

HTML Text Color

The **color** property defines the text color to be used for an HTML element:

Example

```
<h1 style="color:blue">This is a heading</h1>
<p style="color:red">This is a paragraph.</p>
```

Try it Yourself »

HTML Fonts

The **font-family** property defines the font to be used for an HTML element:

Example

```
<h1 style="font-family:verdana">This is a heading</h1>
<p style="font-family:courier">This is a paragraph.</p>
```

Try it Yourself »

The <font> tag, supported in older versions of HTML, is not valid in HTML5.

HTML Text Size

The **font-size** property defines the text size to be used for an HTML element:

Example

```
<h1 style="font-size:300%">This is a heading</h1>
<p style="font-size:160%">This is a paragraph.</p>
```

Try it Yourself »

HTML Text Alignment

The **text-align** property defines the horizontal text alignment for an HTML element:

Example

```
<h1 style="text-align:center">Centered Heading</h1>
<p>This is a paragraph.</p>
```

Try it Yourself »

The tag, supported in older versions of HTML, is not valid in HTML5.

Chapter Summary

- Use the **style** attribute for styling HTML elements
- Use **background-color** for background color
- Use **color** for text colors
- Use **font-family** for text fonts
- Use **font-size** for text sizes

- Use **text-align** for text alignment

Test Yourself with Exercises!

« Previous

Next Chapter »

HTML Text Formatting Elements

« Previous

Next Chapter »

Text Formatting

This text is bold

This text is italic

This is superscript

HTML Formatting Elements

In the previous chapter, you learned about HTML **styling**, using the HTML **style attribute**.

HTML also defines special **elements**, for defining text with a special **meaning**.

HTML uses elements like <b> and <i> for formatting output, like **bold** or *italic* text.

Formatting elements were designed to display special **types of text**:

- Bold text
- Important text
- Italic text
- Emphasized text
- Marked text
- Small text
- Deleted text
- Inserted text
- Subscripts
- Superscripts

HTML **Bold** and **Strong** Formatting

The HTML **<b>** element defines **bold** text, without any extra importance.

Example

```
<p>This text is normal.</p>

<p><b>This text is bold</b>.</p>
```

Try it Yourself »

The HTML **** element defines **strong** text, with added semantic "strong" importance.

Example

```
<p>This text is normal.</p>

<p><strong>This text is strong</strong>.</p>
```

Try it Yourself »

HTML *Italic* and *Emphasized* Formatting

The HTML **<i>** element defines *italic* text, without any extra importance.

Example

```
<p>This text is normal.</p>

<p><i>This text is italic</i>.</p>
```

Try it Yourself »

The HTML **** element defines *emphasized* text, with added semantic importance.

Example

```
<p>This text is normal.</p>

<p><em>This text is emphasized</em>.</p>
```

Try it Yourself »

Browsers display as <b>, and as <i>.

However, there is a difference in the meaning of these tags:< b> and <i> defines bold and italic text,
but< strong> and means that the text is "important".

HTML Small Formatting

The HTML **<small>** element defines **small** text:

Example

```
<h2>HTML <small>Small</small> Formatting</h2>
```

Try it Yourself »

HTML Marked Formatting

The HTML **<mark>** element defines **marked** or highlighted text:

Example

```
<h2>HTML <mark>Marked</mark> Formatting</h2>
```

Try it Yourself »

HTML ~~Deleted~~ Formatting

The HTML **<del>** element defines **deleted** (removed) of text.

Example

```
<p>My favorite color is <del>blue</del> red.</p>
```

Try it Yourself »

HTML Inserted Formatting

The HTML **<ins>** element defines **inserted** (added) text.

Example

```
<p>My favorite <ins>color</ins> is red.</p>
```

Try it Yourself »

HTML Subscript Formatting

The HTML **<sub>** element defines **subscripted** text.

Example

```
<p>This is <sub>subscripted</sub> text.</p>
```

Try it Yourself »

HTML Superscript Formatting

The HTML **<sup>** element defines **superscripted** text.

Example

```
<p>This is <sup>superscripted</sup> text.</p>
```

Try it Yourself »

Test Yourself with Exercises!

HTML Text Formatting Elements

Tag	Description
<b>	Defines bold text
	Defines emphasized text

<i>	Defines italic text
<small>	Defines smaller text
	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<mark>	Defines marked/highlighted text

« Previous

Next Chapter »

HTML Quotation and Citation Elements

« Previous

Next Chapter »

Quotation

Here is a quote from WWF's website:

> For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

HTML <q> for Short Quotations

The HTML **<q>** element defines a **short quotation**.

Browsers usually insert **quotation marks** around the <q> element.

Example

```
<p>WWF's goal is to: <q>Build a future where people live in harmony with nature.</q></p>
```

Try it Yourself »

HTML <blockquote> for Long Quotations

The HTML **<blockquote>** element defines a quoted section.

Browsers usually **indent** <blockquote> elements.

Example

```
<p>Here is a quote from WWF's website:</p>
<blockquote cite="http://www.worldwildlife.org/who/index.html">
For 50 years, WWF has been protecting the future of nature.
The world's leading conservation organization,
WWF works in 100 countries and is supported by
1.2 million members in the United States and
close to 5 million globally.
</blockquote>
```

Try it Yourself »

HTML <abbr> for Abbreviations

The HTML **<abbr>** element defines an **abbreviation** or an acronym.

Marking abbreviations can give useful information to browsers, translation systems and search-engines.

Example

```
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in
1948.</p>
```

Try it Yourself »

HTML <address> for Contact Information

The HTML **<address>** element defines contact information (author/owner) of a document or article.

The element is usually displayed in **italic**. Most browsers will add a line break before and after the element.

Example

```
<address>
Written by Jon Doe.<br>
```

```
Visit us at:<br>
Example.com<br>
Box 564, Disneyland<br>
USA
</address>
```

Try it Yourself »

HTML <cite> for Work Title

The HTML **<cite>** element defines the **title of a work**.

Browsers usually displays <cite> elements in *italic*.

Example

```
<p><cite>The Scream</cite> by Edward Munch. Painted in 1893.</p>
```

Try it Yourself »

HTML <bdo> for Bi-Directional Override

The HTML **<bdo>** element defines **bi-directional override**.

If your browser supports bdo, this text will be written from right to left:

Example

```
<bdo dir="rtl">This text will be written from right to left</bdo>
```

Try it Yourself »

Test Yourself with Exercises!

HTML Quotations, Citations, and Definition Elements

Tag	Description
<abbr>	Defines an abbreviation or acronym
<address>	Defines contact information for the author/owner of a document
<bdo>	Defines the text direction
<blockquote>	Defines a section that is quoted from another source
<dfn>	Defines the definition of a term or an abbreviation.
<q>	Defines a short inline quotation
<cite>	Defines the title of a work

« Previous

Next Chapter »

HTML Computer Code Elements

« Previous

Next Chapter »

Computer Code

```
var person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
}
```

HTML `Computer Code` Formatting

Normally, HTML uses **variable** letter size, and variable letter spacing.

This is not wanted when displaying examples of **computer code**.

The **<kbd>**, **<samp>**, and **<code>** elements all support **fixed** letter size and spacing.

HTML `Keyboard` Formatting

The HTML **<kbd>** element defines **keyboard input**:

Example

```
<p>To open a file, select:</p>

<p><kbd>File | Open...</kbd></p>
```

Try it Yourself »

HTML `Sample` Formatting

The HTML **<samp>** element defines a **computer output**:

Example

```
<samp>
demo.example.com login: Apr 12 09:10:17
Linux 2.6.10-grsec+gg3+e+fhs6b+nfs+gr0501+++p3+c4a+gr2b-reslog-v6.189
</samp>
```

Try it Yourself »

HTML `Code` Formatting

The HTML **<code>** element defines **programming code**:

Example

```
<code>
var person = { firstName:"John", lastName:"Doe", age:50, eyeColor:"blue" }
</code>
```

Try it Yourself »

The <code> element does **not** preserve extra **whitespace** and **line-breaks**:

Example

```
<p>Coding Example:</p>

<code>
var person = {
    firstName:"John",
    lastName:"Doe",
```

```
    age:50,
    eyeColor:"blue"
}
</code>
```

Try it Yourself »

To fix this, you must wrap the code in a <pre> element:

Example

```
<p>Coding Example:</p>

<code>
<pre>
var person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
}
</pre>
</code>
```

Try it Yourself »

HTML *Variable* Formatting

The HTML **<var>** element defines a **mathematical variable**:

Example

```
<p>Einstein wrote:</p>

<p><var>E = m c<sup>2</sup></var></p>
```

Try it Yourself »

HTML Computer Code Elements

Tag	Description
<code>	Defines programming code

<kbd>	Defines keyboard input
<samp>	Defines computer output
<var>	Defines a mathematical variable
<pre>	Defines preformatted text

HTML Comments

« Previous

Next Chapter »

Comment tags <!-- and --> are used to insert comments in HTML.

HTML Comment Tags

You can add comments to your HTML source by using the following syntax:

Example

```
<!-- Write your comments here -->
```

Note: There is an exclamation point (!) in the opening tag, but not in the closing tag.

Comments are not displayed by the browser, but they can help document your HTML.

With comments you can place notifications and reminders in your HTML:

Example

```
<!-- This is a comment -->

<p>This is a paragraph.</p>

<!-- Remember to add more information here -->
```

Try it Yourself »

Comments are also great for debugging HTML, because you can comment out HTML lines of code, one at a time, to search for errors:

Example

```
<!-- Do not display this at the moment
< img border="0" src="pic_mountain.jpg" alt="Mountain">
-->
```

Try it Yourself »

Conditional Comments

You might stumble upon conditional comments in HTML:

```
<!--[if IE 8]>
    .... some HTML here ....
<![endif]-->
```

Conditional comments defines HTML tags to be executed by Internet Explorer only.

Software Program Tags

HTML comments tags can also be generated by various HTML software programs.

For example <!--webbot bot--> tags wrapped inside HTML comments by FrontPage and Expression Web.

As a rule, let these tags stay, to help support the software that created them.

« Previous

Next Chapter »

HTML Styles - CSS

« Previous

Next Chapter »

CSS = Styles and Colors

Manipulate Text
Colors, Boxes

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color:lightgray}
h1   {color:blue}
p    {color:green}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Try it Yourself »

Styling HTML with CSS

CSS stands for **C**ascading **S**tyle **S**heets

Styling can be added to HTML elements in 3 ways:

- Inline - using a **style attribute** in HTML elements
- Internal - using a **<style> element** in the HTML <head> section
- External - using one or more **external CSS files**

The most common way to add styling, is to keep the styles in separate CSS files. But, in this tutorial, we use internal styling, because it is easier to demonstrate, and easier for you to try it yourself.

You can learn much more about CSS in our CSS Tutorial.

CSS Syntax

CSS styling has the following syntax:

element { property:value; property:value }

The **element** is an HTML element name. The ***property*** is a CSS property. The ***value*** is a CSS value.

Multiple styles are separated with semicolon.

Inline Styling (Inline CSS)

Inline styling is useful for applying a unique style to a single HTML element:

Inline styling uses the **style attribute**.

This inline styling changes the text color of a single heading:

Example

```
<h1 style="color:blue">This is a Blue Heading</h1>
```

Try it Yourself »

Internal Styling (Internal CSS)

An internal style sheet can be used to define a common style for all HTML elements on a page.

Internal styling is defined in the **<head>** section of an HTML page, using a **<style>** element:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color:lightgrey}
h1   {color:blue}
p    {color:green}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Try it Yourself »

External Styling (External CSS)

External style sheet are ideal when the style is applied to many pages.

With external style sheets, you can change the look of an entire web site by changing one file.

External styles are defined in an external CSS file, and then linked to in the **<head>** section of an HTML page:

Example

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Try it Yourself »

CSS Fonts

The CSS **color** property defines the text color to be used for the HTML element.

The CSS **font-family** property defines the font to be used for the HTML element.

The CSS **font-size** property defines the text size to be used for the HTML element.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    color:blue;
    font-family:verdana;
```

```
    font-size:300%;
}
p  {
    color:red;
    font-family:courier;
    font-size:160%;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Try it Yourself »

The CSS Box Model

Every HTML element has a box around it, even if you cannot see it.

The CSS **border** property defines a visible border around an HTML element:

Example

```
p {
    border:1px solid black;
}
```

Try it Yourself »

The CSS **padding** property defines a padding (space) inside the border:

Example

```
p {
    border:1px solid black;
    padding:10px;
}
```

Try it Yourself »

The CSS **margin** property defines a margin (space) outside the border:

Example

```
p {
    border:1px solid black;
    padding:10px;
    margin:30px;
}
```

Try it Yourself »

The CSS examples above use px to define sizes in pixels.

The id Attribute

All the examples above use CSS to style HTML elements in a general way.

To define a special style for one special element, first add an id attribute to the element:

Example

```
<p id="p01">I am different</p>
```

then define a different style for the (identified) element:

Example

```
p#p01 {
    color:blue;
}
```

Try it Yourself »

The class Attribute

To define a style for a special type (class) of elements, add a class attribute to the element:

Example

```
<p class="error">I am different</p>
```

Now you can define a different style for all elements with the specified class:

Example

```
p.error {
    color:red;
}
```

Try it Yourself »

Use **id** to address **single** elements. Use **class** to address **groups** of elements.

Deprecated Tags and Attributes in HTML5

In older HTML versions, several tags and attributes were used to style documents.

These tags and attributes are not supported in HTML5!

Avoid using the <font>, , and <strike> elements.

Avoid using the color and bgcolor attributes.

Chapter Summary

- Use the HTML **style** attribute for inline styling
- Use the HTML **<style>** element to define internal CSS
- Use the HTML **<link>** element to refer to an external CSS file
- Use the HTML **<head>** element to store <style> and <link> elements
- Use the CSS **color** property for text colors
- Use the CSS **font-family** property for text fonts
- Use the CSS **font-size** property for text sizes
- Use the CSS **border** property for visible element borders
- Use the CSS **padding** property for space inside the border
- Use the CSS **margin** property for space outside the border

---

Test Yourself with Exercises!

HTML Style Tags

Tag	Description
<style>	Defines style information for a document
<link>	Defines a link between a document and an external resource

« Previous

Next Chapter »

HTML Links

« Previous

Next Chapter »

Links are found in nearly all web pages. Links allow users to click their way from page to page.

HTML Links - Hyperlinks

HTML links are hyperlinks.

A hyperlink is a text or an image you can click on, and jump to another document.

HTML Links - Syntax

In HTML, links are defined with the **<a>** tag:

```
<a href="url">link text</a>
```

Example:

```
<a href="http://www.w3schools.com/html/">Visit our HTML tutorial</a>
```

Try it Yourself »

The **href** attribute specifies the destination address (http://www.w3schools.com/html/)

The **link text** is the visible part (Visit our HTML tutorial).

Clicking on the link text, will send you to the specified address.

The link text does not have to be text. It can be an HTML image or any other HTML element.

Local Links

The example above used an absolute URL (A full web address).

A local link (link to the same web site) is specified with a relative URL (without http://www....).

Example:

```
<a href="html_images.asp">HTML Images</a>
```

Try it Yourself »

HTML Links - Colors and Icons

When you move the mouse cursor over a link, two things will normally happen:

- The mouse arrow will turn into a little hand
- The color of the link element will change

By default, links will appear as this in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

You can change the defaults, using styles:

Example

```
<style>
a:link    {color:#000000; background-color:transparent; text-decoration:none}
a:visited {color:#000000; background-color:transparent; text-decoration:none}
a:hover   {color:#ff0000; background-color:transparent; text-
decoration:underline}
a:active  {color:#ff0000; background-color:transparent; text-
decoration:underline}
</style>
```

Try it Yourself »

HTML Links - The target Attribute

The **target** attribute specifies where to open the linked document.

This example will open the linked document in a new browser window or in a new tab:

Example

```
<a href="http://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

Try it Yourself »

Target Value	Description
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in a named frame

If your webpage is locked in a frame, you can use target="_top" to break out of the frame:

Example

```
<a href="http://www.w3schools.com/html/" target="_top">HTML5 tutorial!</a>
```

Try it Yourself »

HTML Links - Image as Link

It is common to use images as links:

Example

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial"
```

```
style="width:42px;height:42px;border:0">
</a>
```

Try it Yourself »

border:0 is added to prevent IE9 (and earlier) from displaying a border around the image.

HTML Links - The id Attribute

The **id** attribute can be used to create bookmarks inside HTML documents.

Bookmarks are not displayed in any special way. They are invisible to the reader.

Example

Add an id attribute to any <a> element:

```
<a id="tips">Useful Tips Section</a>
```

Then create a link to the <a> element (Useful Tips Section):

```
<a href="#tips">Visit the Useful Tips Section</a>
```

Or, create a link to the <a> element (Useful Tips Section) from another page:

```
<a href="http://www.w3schools.com/html_links.htm#tips">Visit the Useful Tips Section</a>
```

Try it Yourself »

Without a trailing slash on subfolder addresses, you might generate two requests to the server.
Many servers will automatically add a slash to the address, and then create a new request.

Chapter Summary

- Use the HTML **<a>** element to define a link

- Use the HTML **href** attribute to define the link address
- Use the HTML **target** attribute to define where to open the linked document
- Use the HTML **<img>** element (inside <a>) to use an image as a link
- Use the HTML **id** attribute (id="*value*") to define bookmarks in a page
- Use the HTML **href** attribute (href="#*value*") to address the bookmark

---

Test Yourself with Exercises!

Exercise 1 » **Exercise 2 »** **Exercise 3 »** **Exercise 4 »** **Exercise 5 »**

HTML Link Tags

Tag	Description
<a>	Defines a hyperlink

« Previous

HTML Images

« Previous

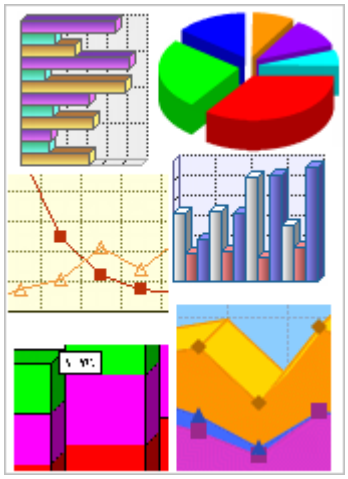
Next Chapter »

Example

GIF Images

JPG Images

PNG Images



```
<!DOCTYPE html>
<html>
<body>

<h2>Spectacular Mountains</h2>
<img src="pic_mountain.jpg" alt="Mountain View" style="width:304px;height:228px">

</body>
</html>
```

Try it Yourself »

Always specify the width and height of an image. If width and height are not specified, the page will flicker while the image loads.

HTML Images Syntax

In HTML, images are defined with the **<img>** tag.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The **src** attribute defines the url (web address) of the image:

```
<img src="url" alt="some_text">
```

The alt Attribute

The **alt** attribute specifies an alternate text for the image, if it cannot be displayed.

The value of the alt attribute should describe the image in words:

Example

```
<img src="html5.gif" alt="The official HTML5 Icon">
```

The alt attribute is **required**. A web page will not validate correctly without it.

HTML Screen Readers

Screen readers are software programs that can read what is displayed on a screen.

Used on the web, screen readers can "reproduce" HTML as text-to-speech, sound icons, or braille output.

Screen readers are used by people who are blind, visually impaired, or learning disabled.

Screen readers can read the **alt** attribute.

Image Size - Width and Height

You can use the **style** attribute to specify the **width** and **height** of an image.

The values are specified in pixels (use px after the value):

Example

```
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px">
```

Try it Yourself »

Alternatively, you can use width and height **attributes**.

The values are specified in pixels (without px after the value):

Example

```
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">
```

Try it Yourself »

Width and Height or Style?

Both the width, the height, and the style attributes, are valid in the latest HTML5 standard.

We suggest you use the style attribute. It prevents styles sheets from changing the default size of images:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
    width:100%;
}
</style>
</head>
<body>

<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px">
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">

</body>
</html>
```

Try it Yourself »

At W3schools we prefer to use the style attribute.

Images in Another Folder

If not specified, the browser expects to find the image in the same folder as the web page.

However, it is common on the web, to store images in a sub-folder, and refer to the folder in the image name:

Example

```
<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px">
```

Try it Yourself »

If a browser cannot find an image, it will display a broken link icon:

Example

```
<img src="wrongname.gif" alt="HTML5 Icon" style="width:128px;height:128px">
```

Try it Yourself »

Images on Another Server

Some web sites store their images on image servers.

Actually, you can access images from any web address in the world:

Example

```
<img src="http://www.w3schools.com/images/w3schools_green.jpg">
```

Try it Yourself »

Animated Images

The GIF standard allows animated images:

Example

```
<img src="programming.gif" alt="Computer Man" style="width:48px;height:48px">
```

Try it Yourself »

Note that the syntax of inserting animated images is no different from non-animated images.

Using an Image as a Link

It is common to use images as links:

Example

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial"
style="width:42px;height:42px;border:0">
</a>
```

Try it Yourself »

We have added border:0 to prevent IE9 (and earlier) from displaying a border around the image.

Image Maps

For an image, you can create an image map, with clickable areas:

Example

```
<img src="planets.gif" alt="Planets" usemap="#planetmap"
style="width:145px;height:126px">

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">
  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">
```

```
  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">
</map>
```

Try it Yourself »

Image Floating

You can let an image float to the left or right of a paragraph:

Example

```
<p>
  <img src="smiley.gif" alt="Smiley face"
style="float:left;width:42px;height:42px">
  A paragraph with an image. The image floats to the left of the text.
</p>
```

Try it Yourself »

Chapter Summary

- Use the HTML **<img>** element to define images
- Use the HTML **src** attribute to define the image file name
- Use the HTML **alt** attribute to define an alternative text
- Use the HTML **width** and **height** attributes to define the image size
- Use the CSS **width** and **height** properties to define the image size (alternatively)
- Use the CSS **float** property to let the image float
- Use the HTML **usemap** attribute to point to an image map
- Use the HTML **<map>** element to define an image map
- Use the HTML **<area>** element to define image map areas

Loading images takes time. Large images can slow down your page. Use images carefully.

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »
Exercise 6 »

HTML Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image-map
<area>	Defines a clickable area inside an image-map

« Previous

Next Chapter »

HTML Tables

« Previous

Next Chapter »

HTML Table Example

Number	First Name	Last Name	Points
1	Eve	Jackson	94
2	John	Doe	80
3	Adam	Johnson	67
4	Jill	Smith	50

Defining HTML Tables

Example

```
<table style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Try it Yourself »

Example explained:

Tables are defined with the **<table>** tag.

Tables are divided into **table rows** with the **<tr>** tag.

Table rows are divided into **table data** with the **<td>** tag.

A table row can also be divided into **table headings** with the **<th>** tag.

Table data <td> are the data containers of the table.
They can contain all sorts of HTML elements like text, images, lists, other tables, etc.

An HTML Table with a Border Attribute

If you do not specify a border for the table, it will be displayed without borders.

A border can be added using the border attribute:

Example

```
<table border="1" style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
```

```
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Try it Yourself »

The border attribute is on its way out of the HTML standard! It is better to use CSS.

To add borders, use the **CSS border** property:

Example

```
table, th, td {
    border: 1px solid black;
}
```

Try it Yourself »

Remember to define borders for both the table and the table cells.

An HTML Table with Collapsed Borders

If you want the borders to collapse into one border, add **CSS border-collapse**:

Example

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
```

Try it Yourself »

An HTML Table with Cell Padding

Cell padding specifies the space between the cell content and its borders.

If you do not specify a padding, the table cells will be displayed without padding.

To set the padding, use the **CSS padding** property:

Example

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 15px;
}
```

Try it Yourself »

HTML Table Headings

Table headings are defined with the **<th>** tag.

By default, all major browsers display table headings as bold and centered:

Example

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Try it Yourself »

To left-align the table headings, use the **CSS text-align** property:

Example

```
th {
    text-align: left;
}
```

Try it Yourself »

An HTML Table with Border Spacing

Border spacing specifies the space between the cells.

To set the border spacing for a table, use the **CSS border-spacing** property:

Example

```
table {
    border-spacing: 5px;
}
```

Try it Yourself »

If the table has collapsed borders, border-spacing has no effect.

Table Cells that Span Many Columns

To make a cell span more than one column, use the **colspan** attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>555 77 854</td>
    <td>555 77 855</td>
  </tr>
</table>
```

Try it Yourself »

Table Cells that Span Many Rows

To make a cell span more than one row, use the **rowspan** attribute:

Example

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>555 77 854</td>
  </tr>
  <tr>
    <td>555 77 855</td>
  </tr>
</table>
```

Try it Yourself »

An HTML Table With a Caption

To add a caption to a table, use the **<caption>** tag:

Example

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
```

```
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

Try it Yourself »

The <caption> tag must be inserted immediately after the <table> tag.

Different Styles for Different Tables

Most of the examples above use a style attribute (width="100%") to define the width of each table.

This makes it easy to define different widths for different tables.

The styles in the <head> section, however, define a style for all tables in a page.

To define a special style for a special table, add an **id attribute** to the table:

Example

```
<table id="t01">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Now you can define a different style for this table:

```
table#t01 {
    width: 100%;
    background-color: #f1f1c1;
}
```

Try it Yourself »

And add more styles:

```
table#t01 tr:nth-child(even) {
    background-color: #eee;
}
table#t01 tr:nth-child(odd) {
    background-color: #fff;
}
table#t01 th {
    color: white;
    background-color: black;
}
```

Try it Yourself »

Chapter Summary

- Use the HTML **<table>** element to define a table
- Use the HTML **<tr>** element to define a table row
- Use the HTML **<td>** element to define a table data
- Use the HTML **<th>** element to define a table heading
- Use the HTML **<caption>** element to define a table caption
- Use the CSS **border** property to define a border
- Use the CSS **border-collapse** property to collapse cell borders
- Use the CSS **padding** property to add padding to cells
- Use the CSS **text-align** property to align cell text
- Use the CSS **border-spacing** property to set the spacing between cells
- Use the **colspan** attribute to make a cell span many columns
- Use the **rowspan** attribute to make a cell span many rows
- Use the **id** attribute to uniquely define one table

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »
Exercise 6 »

HTML Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting
<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

« Previous

Next Chapter »

HTML Lists

« Previous

Next Chapter »

HTML can have Unordered lists, Ordered lists, or Description lists:

Unordered List

- The first item
- The second item
- The third item
- The fourth item

Ordered List

1. The first item
2. The second item
3. The third item
4. The fourth item

Description List

The first item

: Description of item

The second item

: Description of item

Unordered HTML Lists

An unordered list starts with the **<ul>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with bullets (small black circles).

Unordered List:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

Unordered HTML Lists - The Style Attribute

A **style** attribute can be added to an **unordered list**, to define the style of the marker:

Style	Description
list-style-type:disc	The list items will be marked with bullets (default)

list-style-type:circle	The list items will be marked with circles
list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

Disc:

```
<ul style="list-style-type:disc">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

Circle:

```
<ul style="list-style-type:circle">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

Square:

```
<ul style="list-style-type:square">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

None:

```
<ul style="list-style-type:none">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

Using a type attribute <ul type="disc">, instead of <ul style="list-style-type:disc">, also works. But in HTML5, the type attribute is not valid in unordered lists, only in ordered list.

Ordered HTML Lists

An ordered list starts with the **<ol>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with numbers.

Ordered List:

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Try it Yourself »

Ordered HTML Lists - The Type Attribute

A **type** attribute can be added to an **ordered list**, to define the type of the marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Numbers:

```
<ol type="1">
  <li>Coffee</li>
```

```
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Try it Yourself »

Upper Case:

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Try it Yourself »

Lower Case:

```
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Try it Yourself »

Roman Upper Case:

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Try it Yourself »

Roman Lower Case:

```
<ol type="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Try it Yourself »

HTML Description Lists

A description list, is a list of terms, with a description of each term.

The **<dl>** tag defines a description list.

The **<dt>** tag defines the term (name), and the **<dd>** tag defines the data (description).

Description List:

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

Try it Yourself »

Nested HTML Lists

List can be nested (lists inside lists).

Nested Lists:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Try it Yourself »

List items can contain new list, and other HTML elements, like images and links, etc.

Horizontal Lists

HTML lists can be styled in many different ways with CSS.

One popular way, is to style a list to display horizontally:

Horizontal List:

```
<!DOCTYPE html>
<html>

<head>
<style>
ul#menu li {
    display:inline;
}
</style>
</head>

<body>

<h2>Horizontal List</h2>

<ul id="menu">
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
  <li>PHP</li>
</ul>

</body>
</html>
```

Try it Yourself »

With a little extra style, you can make it look like a menu:

New Style:

```
ul#menu {
    padding: 0;
```

```
}

ul#menu li {
    display: inline;
}

ul#menu li a {
    background-color: black;
    color: white;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 4px 4px 0 0;
}

ul#menu li a:hover {
    background-color: orange;
}
```

Try it Yourself »

Chapter Summary

- Use the HTML **<ul>** element to define an unordered list
- Use the HTML **style** attribute to define the bullet style
- Use the HTML **<ol>** element to define an ordered list
- Use the HTML **type** attribute to define the numbering type
- Use the HTML **<li>** element to define a list item
- Use the HTML **<dl>** element to define a description list
- Use the HTML **<dt>** element to define the description term
- Use the HTML **<dd>** element to define the description data
- Lists can be nested inside lists
- List items can contain other HTML elements
- Use the CSS property **display:inline** to display a list horizontally

---

Test Yourself with Exercises!

Exercise 1 » **Exercise 2 »** **Exercise 3 »** **Exercise 4 »** **Exercise 5 »**

Exercise 6 »

HTML List Tags

Tag	Description
<ul>	Defines an unordered list
<ol>	Defines an ordered list
<li>	Defines a list item
<dl>	Defines a description list
<dt>	Defines the term in a description list
<dd>	Defines the description in a description list

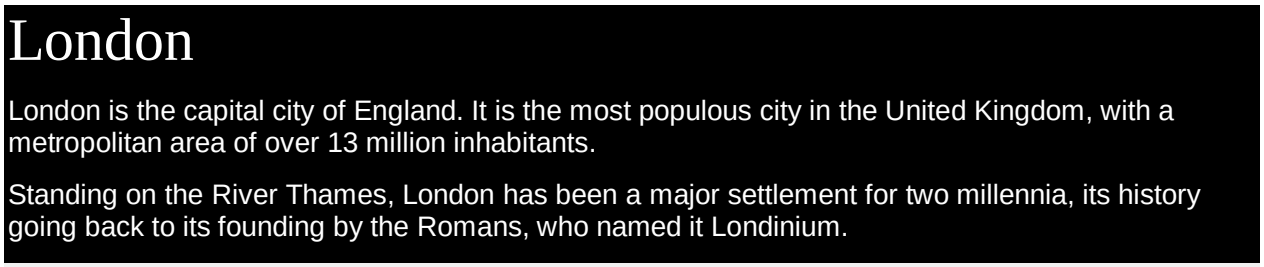
« Previous

Next Chapter »

HTML Block Elements

« Previous

Next Chapter »



Example

```
<div style="background-color:black; color:white; padding:20px;">

<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

</div>
```

Try it Yourself »

HTML Block Elements and Inline Elements

Most HTML elements are defined as **block level** elements or **inline** elements.

Block level elements normally start (and end) with a new line, when displayed in a browser.

Examples: <h1>, <p>, <ul>, <table>

Inline elements are normally displayed without line breaks.

Examples: <b>, <td>, <a>, <img>

The HTML Element

The HTML element is a **block level element** that can be used as a container for other HTML elements.

The element has no special meaning. It has no required attributes, but **style** and **class** are common.

Because it is a block level element, the browser will display line breaks before and after it.

When used together with CSS, the element can be used to style blocks of content.

The HTML Element

The HTML element is an **inline element** that can be used as a container for text.

The element has no special meaning. It has no required attributes, but **style** and **class** are common.

Unlike , which is formatted with line breaks, the element does not have any automatic formatting.

When used together with CSS, the element can be used to style parts of the text:

Example

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

Try it Yourself »

HTML Grouping Tags

Tag	Description
	Defines a section in a document (block-level)
	Defines a section in a document (inline)

« Previous

Next Chapter »

HTML Classes

« Previous

Next Chapter »

Classing HTML elements, makes it possible to define CSS styles for classes of elements.

Equal styles for equal classes, or different styles for different classes.

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.cities {
    background-color:black;
    color:white;
```

```
    margin:20px;
    padding:20px;
}
</style>
</head>
<body>

<div class="cities">
<h2>London</h2>
<p>
London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.
</p>
</div>

</body>
</html>
```

Try it Yourself »

Classing Block Elements

The HTML element is a **block level** element. It can be used as a container for other HTML elements.

Classing elements, makes it possible to define equal styles for equal < div> elements:

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Paris

Paris is the capital and most populous city of France.

Situated on the Seine River, it is at the heart of the Île-de-France region, also known as the région parisienne.

Within its metropolitan area is one of the largest population centers in Europe, with over 12 million inhabitants.

Tokyo

Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.

It is the seat of the Japanese government and the Imperial Palace, and the home of the Japanese Imperial Family.

The Tokyo prefecture is part of the world's most populous metropolitan area with 38 million people and the world's largest urban economy.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.cities {
    background-color:black;
    color:white;
    margin:20px;
    padding:20px;
}
</style>
</head>
<body>

<div class="cities">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="cities">
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="cities">
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>

</body>
</html>
```

Try it Yourself »

Classing Inline Elements

The HTML element is an inline element that can be used as a container for text.

Classing elements makes it possible to design equal styles for equal < span> elements.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
span.red {color:red;}
</style>
</head>
<body>

<h1>My <span class="red">Important</span> Heading</h1>

</body>
</html>
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 »

« Previous

Next Chapter »

- HTML Layouts

HTML Layouts

Websites often display content in multiple columns (like a magazine or newspaper).

City Gallery

London

Paris

Tokyo

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Copyright © W3Schools.com

HTML Layout Using Elements

The element is often used as a layout tool, because it can easily be positioned with CSS.

This example uses 4 elements to create a multiple column layout:

Example

```
<body>

<div id="header">
<h1>City Gallery</h1>
</div>

<div id="nav">
London<br>
Paris<br>
Tokyo<br>
</div>
```

```
<div id="section">
<h1>London</h1>
<p>
London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.
</p>
<p>
Standing on the River Thames, London has been a major settlement for two millennia,
its history going back to its founding by the Romans, who named it Londinium.
</p>
</div>

<div id="footer">
Copyright © W3Schools.com
</div>

</body>
```

Try it yourself »

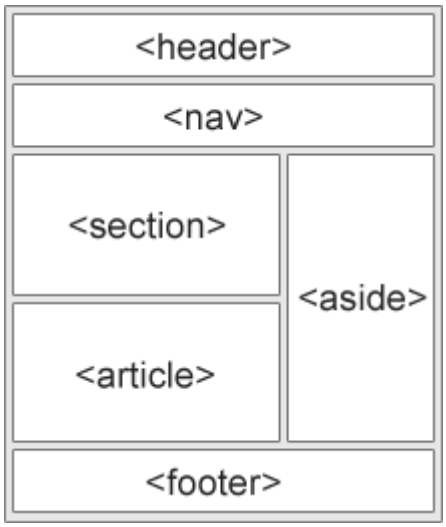
The CSS:

```
<style>
#header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
#nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding:5px;
}
#section {
    width:350px;
    float:left;
    padding:10px;
}
```

```
#footer {
    background-color:black;
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
}
</style>
```

Website Layout Using HTML5

HTML5 offers new semantic elements that define different parts of a web page:



header	Defines a header for a document or a section
nav	Defines a container for navigation links
section	Defines a section in a document
article	Defines an independent self-contained article
aside	Defines content aside from the content (like a sidebar)
footer	Defines a footer for a document or a section
details	Defines additional details
summary	Defines a heading for the details element

This example uses <header>, <nav>, <section>, and <footer> to create a multiple column layout:

Example

```
<body>

<header>
<h1>City Gallery</h1>
</header>

<nav>
London<br>
Paris<br>
Tokyo<br>
</nav>

<section>
<h1>London</h1>
```

```
<p>
London is the capital city of England. It is the most populous city in the United
Kingdom,
with a metropolitan area of over 13 million inhabitants.
</p>
<p>
Standing on the River Thames, London has been a major settlement for two
millennia,
its history going back to its founding by the Romans, who named it Londinium.
</p>
</section>

<footer>
Copyright © W3Schools.com
</footer>

</body>
```

Try it yourself »

The CSS

```
<style>
header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding:5px;
}
section {
    width:350px;
    float:left;
    padding:10px;
}
footer {
    background-color:black;
```

```
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
}
```

HTML Layout Using Tables

The <table> element was not designed to be a layout tool.
The purpose of the <table> element is to display tabular data.

Layout can be achieved using the <table> element, because table elements can be styled with CSS:

Example

```
<body>

<table class="lamp">
<tr>
  <th>
    <img src="/images/lamp.jpg" alt="Note" style="height:32px;width:32px">
  </th>
  <td>
    The table element was not designed to be a layout tool.
  </td>
</tr>
</table>

</body>
```

Try it yourself »

The CSS

```
<style>
table.lamp {
    width:100%;
    border:1px solid #d4d4d4;
}
table.lamp th, td {
    padding:10px;
}
```

```
table.lamp th {
    width:40px;
}
</style>
```

« Previous

Next Chapter »

HTML Responsive Web Design

- HTML Responsive Web Design

« Previous

Next Chapter »

What is Responsive Web Design?

- RWD stands for Responsive Web Design
- RWD can deliver web pages in variable sizes
- RWD is a must for tablets and mobile devices

Creating Your Own Responsive Design

One way to create a responsive design, is to create it yourself:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<style>
.city {
    float: left;
    margin: 5px;
    padding: 15px;
```

```
    width: 300px;
    height: 300px;
    border: 1px solid black;
}
</style>
</head>
<body>

<h1>W3Schools Demo</h1>
<h2>Resize this responsive page!</h2>

<div class="city">
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="city">
  <h2>Paris</h2>
  <p>Paris is the capital and most populous city of France.</p>
</div>

<div class="city">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.</p>
</div>

</body>
</html>
```

Try it yourself »

Using Bootstrap

Another way to create a responsive design, is to use an already existing CSS framework.

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive webs.

Bootstrap helps you to develop sites that look nice at any size; screen, laptop, tablet, or phone:

```
<!DOCTYPE html>
<html>
```

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
</head>
<body>

<div class="container">

<div class="jumbotron">
  <h1>W3Schools Demo</h1>
  <p>Resize this responsive page!</p>
</div>

<div class="row">
  <div class="col-md-4">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
    <p>It is the most populous city in the United Kingdom,
    with a metropolitan area of over 13 million inhabitants.</p>
  </div>
  <div class="col-md-4">
    <h2>Paris</h2>
    <p>Paris is the capital and most populous city of France.</p>
  </div>
  <div class="col-md-4">
    <h2>Tokyo</h2>
    <p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.</p>
  </div>
</div>

</div>

</body>
</html>
```

Try it yourself »

To learn more about Bootstrap read our Bootstrap Tutorial.

« Previous Next Chapter »

HTML Iframes

« Previous

Next Chapter »

An iframe is used to display a web page within a web page.

Iframe Syntax

The syntax for adding an iframe is:

```
<iframe src="URL"></iframe>
```

The **src** attribute specifies the URL (web address) of the iframe page.

Iframe - Set Height and Width

Use the **height** and **width** attributes to specify the size.

The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

Example

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Try it Yourself »

Iframe - Remove the Border

By default, an iframe has a black border around it.

To remove the border, add the style attribute and use the CSS border property:

Example

```
<iframe src="demo_iframe.htm" style="border:none"></iframe>
```

Try it Yourself »

With CSS, you can also change the size, style and color of the iframe's border:

Example

```
<iframe src="demo_iframe.htm" style="border:5px dotted red"></iframe>
```

Try it Yourself »

Use iframe as a Target for a Link

An iframe can be used as the target frame for a link.

The **target** attribute of the link must refer to the **name** attribute of the iframe:

Example

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

Try it Yourself »

HTML iframe Tag

Tag	Description
<iframe>	Defines an inline frame

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

« Previous Next Chapter »

HTML Color Names

« Previous

Next Chapter »

Colors are displayed combining RED, GREEN, and BLUE light.

140 Color Names are Supported by All Browsers

140 color names are defined in the HTML5 and CSS3 color specifications.

17 colors are from the HTML specification, 123 colors are from the CSS specification.

The table below lists them all, along with their hexadecimal values.

The 17 colors from the HTML specification are: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow.

Sorted by Color Name

Colors sorted by HEX values

Click on a color name (or a hex value) to view the color as the background-color along with different text colors:

Color Name	HEX	Color	Shades	Mix
AliceBlue	#F0F8FF		Shades	Mix
AntiqueWhite	#FAEBD7		Shades	Mix
Aqua	#00FFFF		Shades	Mix
Aquamarine	#7FFFD4		Shades	Mix
Azure	#F0FFFF		Shades	Mix
Beige	#F5F5DC		Shades	Mix
Bisque	#FFE4C4		Shades	Mix

Black	#000000		Shades	Mix
BlanchedAlmond	#FFEBCD		Shades	Mix
Blue	#0000FF		Shades	Mix
BlueViolet	#8A2BE2		Shades	Mix
Brown	#A52A2A		Shades	Mix
BurlyWood	#DEB887		Shades	Mix
CadetBlue	#5F9EA0		Shades	Mix
Chartreuse	#7FFF00		Shades	Mix
Chocolate	#D2691E		Shades	Mix
Coral	#FF7F50		Shades	Mix
CornflowerBlue	#6495ED		Shades	Mix
Cornsilk	#FFF8DC		Shades	Mix
Crimson	#DC143C		Shades	Mix
Cyan	#00FFFF		Shades	Mix
DarkBlue	#00008B		Shades	Mix
DarkCyan	#008B8B		Shades	Mix
DarkGoldenRod	#B8860B		Shades	Mix
DarkGray	#A9A9A9		Shades	Mix
DarkGreen	#006400		Shades	Mix
DarkKhaki	#BDB76B		Shades	Mix
DarkMagenta	#8B008B		Shades	Mix
DarkOliveGreen	#556B2F		Shades	Mix
DarkOrange	#FF8C00		Shades	Mix
DarkOrchid	#9932CC		Shades	Mix
DarkRed	#8B0000		Shades	Mix
DarkSalmon	#E9967A		Shades	Mix
DarkSeaGreen	#8FBC8F		Shades	Mix
DarkSlateBlue	#483D8B		Shades	Mix

DarkSlateGray	#2F4F4F		Shades	Mix
DarkTurquoise	#00CED1		Shades	Mix
DarkViolet	#9400D3		Shades	Mix
DeepPink	#FF1493		Shades	Mix
DeepSkyBlue	#00BFFF		Shades	Mix
DimGray	#696969		Shades	Mix
DodgerBlue	#1E90FF		Shades	Mix
FireBrick	#B22222		Shades	Mix
FloralWhite	#FFFAF0		Shades	Mix
ForestGreen	#228B22		Shades	Mix
Fuchsia	#FF00FF		Shades	Mix
Gainsboro	#DCDCDC		Shades	Mix
GhostWhite	#F8F8FF		Shades	Mix
Gold	#FFD700		Shades	Mix
GoldenRod	#DAA520		Shades	Mix
Gray	#808080		Shades	Mix
Green	#008000		Shades	Mix
GreenYellow	#ADFF2F		Shades	Mix
HoneyDew	#F0FFF0		Shades	Mix
HotPink	#FF69B4		Shades	Mix
IndianRed	#CD5C5C		Shades	Mix
Indigo	#4B0082		Shades	Mix
Ivory	#FFFFF0		Shades	Mix
Khaki	#F0E68C		Shades	Mix
Lavender	#E6E6FA		Shades	Mix
LavenderBlush	#FFF0F5		Shades	Mix
LawnGreen	#7CFC00		Shades	Mix
LemonChiffon	#FFFACD		Shades	Mix

LightBlue	#ADD8E6		Shades	Mix
LightCoral	#F08080		Shades	Mix
LightCyan	#E0FFFF		Shades	Mix
LightGoldenRodYellow	#FAFAD2		Shades	Mix
LightGray	#D3D3D3		Shades	Mix
LightGreen	#90EE90		Shades	Mix
LightPink	#FFB6C1		Shades	Mix
LightSalmon	#FFA07A		Shades	Mix
LightSeaGreen	#20B2AA		Shades	Mix
LightSkyBlue	#87CEFA		Shades	Mix
LightSlateGray	#778899		Shades	Mix
LightSteelBlue	#B0C4DE		Shades	Mix
LightYellow	#FFFFE0		Shades	Mix
Lime	#00FF00		Shades	Mix
LimeGreen	#32CD32		Shades	Mix
Linen	#FAF0E6		Shades	Mix
Magenta	#FF00FF		Shades	Mix
Maroon	#800000		Shades	Mix
MediumAquaMarine	#66CDAA		Shades	Mix
MediumBlue	#0000CD		Shades	Mix
MediumOrchid	#BA55D3		Shades	Mix
MediumPurple	#9370DB		Shades	Mix
MediumSeaGreen	#3CB371		Shades	Mix
MediumSlateBlue	#7B68EE		Shades	Mix
MediumSpringGreen	#00FA9A		Shades	Mix
MediumTurquoise	#48D1CC		Shades	Mix
MediumVioletRed	#C71585		Shades	Mix
MidnightBlue	#191970		Shades	Mix

MintCream	#F5FFFA		Shades	Mix
MistyRose	#FFE4E1		Shades	Mix
Moccasin	#FFE4B5		Shades	Mix
NavajoWhite	#FFDEAD		Shades	Mix
Navy	#000080		Shades	Mix
OldLace	#FDF5E6		Shades	Mix
Olive	#808000		Shades	Mix
OliveDrab	#6B8E23		Shades	Mix
Orange	#FFA500		Shades	Mix
OrangeRed	#FF4500		Shades	Mix
Orchid	#DA70D6		Shades	Mix
PaleGoldenRod	#EEE8AA		Shades	Mix
PaleGreen	#98FB98		Shades	Mix
PaleTurquoise	#AFEEEE		Shades	Mix
PaleVioletRed	#DB7093		Shades	Mix
PapayaWhip	#FFEFD5		Shades	Mix
PeachPuff	#FFDAB9		Shades	Mix
Peru	#CD853F		Shades	Mix
Pink	#FFC0CB		Shades	Mix
Plum	#DDA0DD		Shades	Mix
PowderBlue	#B0E0E6		Shades	Mix
Purple	#800080		Shades	Mix
RebeccaPurple	#663399		Shades	Mix
Red	#FF0000		Shades	Mix
RosyBrown	#BC8F8F		Shades	Mix
RoyalBlue	#4169E1		Shades	Mix
SaddleBrown	#8B4513		Shades	Mix
Salmon	#FA8072		Shades	Mix

SandyBrown	#F4A460		Shades	Mix
SeaGreen	#2E8B57		Shades	Mix
SeaShell	#FFF5EE		Shades	Mix
Sienna	#A0522D		Shades	Mix
Silver	#C0C0C0		Shades	Mix
SkyBlue	#87CEEB		Shades	Mix
SlateBlue	#6A5ACD		Shades	Mix
SlateGray	#708090		Shades	Mix
Snow	#FFFAFA		Shades	Mix
SpringGreen	#00FF7F		Shades	Mix
SteelBlue	#4682B4		Shades	Mix
Tan	#D2B48C		Shades	Mix
Teal	#008080		Shades	Mix
Thistle	#D8BFD8		Shades	Mix
Tomato	#FF6347		Shades	Mix
Turquoise	#40E0D0		Shades	Mix
Violet	#EE82EE		Shades	Mix
Wheat	#F5DEB3		Shades	Mix
White	#FFFFFF		Shades	Mix
WhiteSmoke	#F5F5F5		Shades	Mix
Yellow	#FFFF00		Shades	Mix
YellowGreen	#9ACD32		Shades	Mix

« Previous

Next Chapter »

HTML Color Values

« Previous

Next Chapter »

Colors are displayed combining RED, GREEN, and BLUE light.

Color Values

Colors are defined using a hexadecimal (hex) notation for the Red, Green, and Blue values (RGB).

The lowest value for each light source is 0 (hex 00). The highest value is 255 (hex FF).

Hex values are written as # followed by either three or six hex characters.

Three-digit notations (#rgb) are automatically converted to six digits (#rrggbb):

Color	Color 3 digit HEX	Color 6 digit HEX	Color RGB
	#F00	#FF0000	rgb(255,0,0)
	#0F0	#00FF00	rgb(0,255,0)
	#00F	#0000FF	rgb(0,0,255)

Try it Yourself »

Shades of grey (from black to white) are defined using equal values for all the 3 light sources:

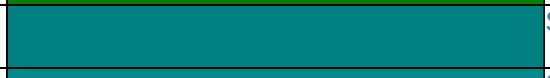
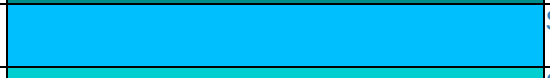
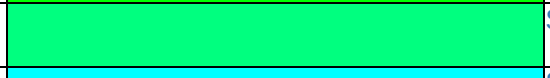
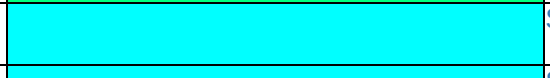
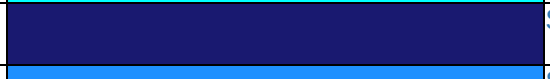
Color	Color 3 digit HEX	Color 6 digit HEX	Color RGB
	#000	#000000	rgb(0,0,0)
	#888	#888888	rgb(136,136,136)
	#FFF	#FFFFFF	rgb(255,255,255)

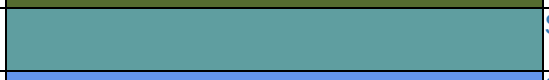
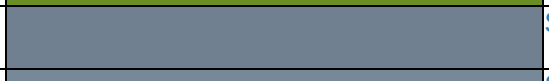
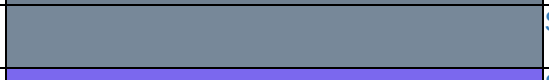
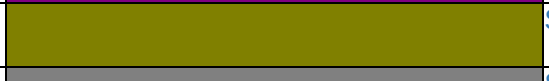
Try it Yourself »

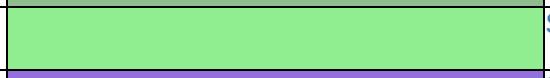
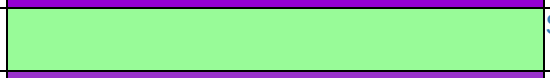
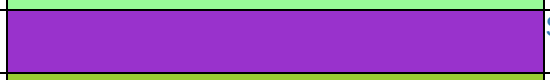
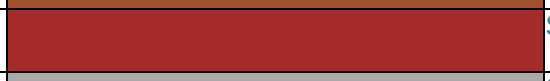
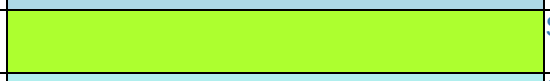
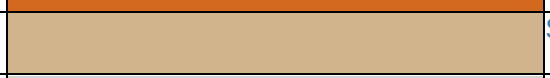
Colors Sorted by HEX Value

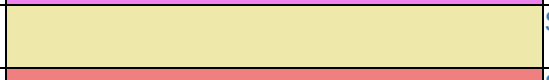
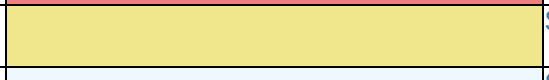
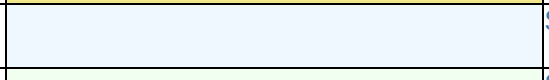
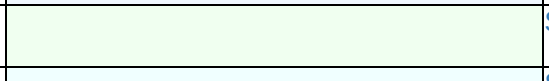
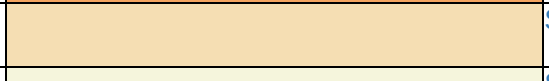
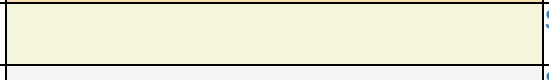
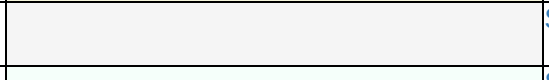
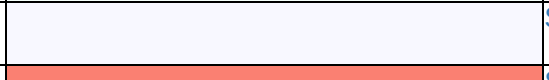
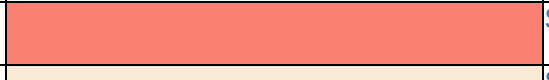
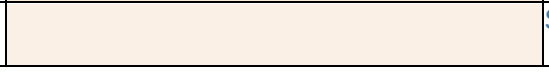
Colors sorted by color name

Color Name	HEX	Color	Shades	Mix

Black	#000000		Shades	Mix
Navy	#000080		Shades	Mix
DarkBlue	#00008B		Shades	Mix
MediumBlue	#0000CD		Shades	Mix
Blue	#0000FF		Shades	Mix
DarkGreen	#006400		Shades	Mix
Green	#008000		Shades	Mix
Teal	#008080		Shades	Mix
DarkCyan	#008B8B		Shades	Mix
DeepSkyBlue	#00BFFF		Shades	Mix
DarkTurquoise	#00CED1		Shades	Mix
MediumSpringGreen	#00FA9A		Shades	Mix
Lime	#00FF00		Shades	Mix
SpringGreen	#00FF7F		Shades	Mix
Aqua	#00FFFF		Shades	Mix
Cyan	#00FFFF		Shades	Mix
MidnightBlue	#191970		Shades	Mix
DodgerBlue	#1E90FF		Shades	Mix
LightSeaGreen	#20B2AA		Shades	Mix
ForestGreen	#228B22		Shades	Mix
SeaGreen	#2E8B57		Shades	Mix
DarkSlateGray	#2F4F4F		Shades	Mix
LimeGreen	#32CD32		Shades	Mix
MediumSeaGreen	#3CB371		Shades	Mix
Turquoise	#40E0D0		Shades	Mix
RoyalBlue	#4169E1		Shades	Mix
SteelBlue	#4682B4		Shades	Mix

DarkSlateBlue	#483D8B		Shades	Mix
MediumTurquoise	#48D1CC		Shades	Mix
Indigo	#4B0082		Shades	Mix
DarkOliveGreen	#556B2F		Shades	Mix
CadetBlue	#5F9EA0		Shades	Mix
CornflowerBlue	#6495ED		Shades	Mix
RebeccaPurple	#663399		Shades	Mix
MediumAquaMarine	#66CDAA		Shades	Mix
DimGray	#696969		Shades	Mix
SlateBlue	#6A5ACD		Shades	Mix
OliveDrab	#6B8E23		Shades	Mix
SlateGray	#708090		Shades	Mix
LightSlateGray	#778899		Shades	Mix
MediumSlateBlue	#7B68EE		Shades	Mix
LawnGreen	#7CFC00		Shades	Mix
Chartreuse	#7FFF00		Shades	Mix
Aquamarine	#7FFFD4		Shades	Mix
Maroon	#800000		Shades	Mix
Purple	#800080		Shades	Mix
Olive	#808000		Shades	Mix
Gray	#808080		Shades	Mix
SkyBlue	#87CEEB		Shades	Mix
LightSkyBlue	#87CEFA		Shades	Mix
BlueViolet	#8A2BE2		Shades	Mix
DarkRed	#8B0000		Shades	Mix
DarkMagenta	#8B008B		Shades	Mix
SaddleBrown	#8B4513		Shades	Mix

DarkSeaGreen	#8FBC8F		Shades	Mix
LightGreen	#90EE90		Shades	Mix
MediumPurple	#9370DB		Shades	Mix
DarkViolet	#9400D3		Shades	Mix
PaleGreen	#98FB98		Shades	Mix
DarkOrchid	#9932CC		Shades	Mix
YellowGreen	#9ACD32		Shades	Mix
Sienna	#A0522D		Shades	Mix
Brown	#A52A2A		Shades	Mix
DarkGray	#A9A9A9		Shades	Mix
LightBlue	#ADD8E6		Shades	Mix
GreenYellow	#ADFF2F		Shades	Mix
PaleTurquoise	#AFEEEE		Shades	Mix
LightSteelBlue	#B0C4DE		Shades	Mix
PowderBlue	#B0E0E6		Shades	Mix
FireBrick	#B22222		Shades	Mix
DarkGoldenRod	#B8860B		Shades	Mix
MediumOrchid	#BA55D3		Shades	Mix
RosyBrown	#BC8F8F		Shades	Mix
DarkKhaki	#BDB76B		Shades	Mix
Silver	#C0C0C0		Shades	Mix
MediumVioletRed	#C71585		Shades	Mix
IndianRed	#CD5C5C		Shades	Mix
Peru	#CD853F		Shades	Mix
Chocolate	#D2691E		Shades	Mix
Tan	#D2B48C		Shades	Mix
LightGray	#D3D3D3		Shades	Mix

Thistle	#D8BFD8		Shades	Mix
Orchid	#DA70D6		Shades	Mix
GoldenRod	#DAA520		Shades	Mix
PaleVioletRed	#DB7093		Shades	Mix
Crimson	#DC143C		Shades	Mix
Gainsboro	#DCDCDC		Shades	Mix
Plum	#DDA0DD		Shades	Mix
BurlyWood	#DEB887		Shades	Mix
LightCyan	#E0FFFF		Shades	Mix
Lavender	#E6E6FA		Shades	Mix
DarkSalmon	#E9967A		Shades	Mix
Violet	#EE82EE		Shades	Mix
PaleGoldenRod	#EEE8AA		Shades	Mix
LightCoral	#F08080		Shades	Mix
Khaki	#F0E68C		Shades	Mix
AliceBlue	#F0F8FF		Shades	Mix
HoneyDew	#F0FFF0		Shades	Mix
Azure	#F0FFFF		Shades	Mix
SandyBrown	#F4A460		Shades	Mix
Wheat	#F5DEB3		Shades	Mix
Beige	#F5F5DC		Shades	Mix
WhiteSmoke	#F5F5F5		Shades	Mix
MintCream	#F5FFFA		Shades	Mix
GhostWhite	#F8F8FF		Shades	Mix
Salmon	#FA8072		Shades	Mix
AntiqueWhite	#FAEBD7		Shades	Mix
Linen	#FAF0E6		Shades	Mix

LightGoldenRodYellow	#FAFAD2		Shades	Mix
OldLace	#FDF5E6		Shades	Mix
Red	#FF0000		Shades	Mix
Fuchsia	#FF00FF		Shades	Mix
Magenta	#FF00FF		Shades	Mix
DeepPink	#FF1493		Shades	Mix
OrangeRed	#FF4500		Shades	Mix
Tomato	#FF6347		Shades	Mix
HotPink	#FF69B4		Shades	Mix
Coral	#FF7F50		Shades	Mix
DarkOrange	#FF8C00		Shades	Mix
LightSalmon	#FFA07A		Shades	Mix
Orange	#FFA500		Shades	Mix
LightPink	#FFB6C1		Shades	Mix
Pink	#FFC0CB		Shades	Mix
Gold	#FFD700		Shades	Mix
PeachPuff	#FFDAB9		Shades	Mix
NavajoWhite	#FFDEAD		Shades	Mix
Moccasin	#FFE4B5		Shades	Mix
Bisque	#FFE4C4		Shades	Mix
MistyRose	#FFE4E1		Shades	Mix
BlanchedAlmond	#FFEBCD		Shades	Mix
PapayaWhip	#FFEFD5		Shades	Mix
LavenderBlush	#FFF0F5		Shades	Mix
SeaShell	#FFF5EE		Shades	Mix
Cornsilk	#FFF8DC		Shades	Mix
LemonChiffon	#FFFACD		Shades	Mix

FloralWhite	#FFFAF0		Shades	Mix
Snow	#FFFAFA		Shades	Mix
Yellow	#FFFF00		Shades	Mix
LightYellow	#FFFFE0		Shades	Mix
Ivory	#FFFFF0		Shades	Mix
White	#FFFFFF		Shades	Mix

« Previous

Next Chapter »

HTML Color Shades

« Previous

Next Chapter »

Colors are displayed combining RED, GREEN, and BLUE light.

Shades of Gray

Gray colors are displayed using an equal amount of power to all of the light sources.

To make it easy for you to select a gray color we have compiled a table of gray shades for you:

Gray Shades	HEX	RGB
	#000000	rgb(0,0,0)
	#080808	rgb(8,8,8)
	#101010	rgb(16,16,16)
	#181818	rgb(24,24,24)
	#202020	rgb(32,32,32)
	#282828	rgb(40,40,40)

#303030	rgb(48,48,48)
#383838	rgb(56,56,56)
#404040	rgb(64,64,64)
#484848	rgb(72,72,72)
#505050	rgb(80,80,80)
#585858	rgb(88,88,88)
#606060	rgb(96,96,96)
#686868	rgb(104,104,104)
#707070	rgb(112,112,112)
#787878	rgb(120,120,120)
#808080	rgb(128,128,128)
#888888	rgb(136,136,136)
#909090	rgb(144,144,144)
#989898	rgb(152,152,152)
#A0A0A0	rgb(160,160,160)
#A8A8A8	rgb(168,168,168)
#B0B0B0	rgb(176,176,176)
#B8B8B8	rgb(184,184,184)
#C0C0C0	rgb(192,192,192)
#C8C8C8	rgb(200,200,200)
#D0D0D0	rgb(208,208,208)
#D8D8D8	rgb(216,216,216)
#E0E0E0	rgb(224,224,224)
#E8E8E8	rgb(232,232,232)
#F0F0F0	rgb(240,240,240)
#F8F8F8	rgb(248,248,248)
#FFFFFF	rgb(255,255,255)

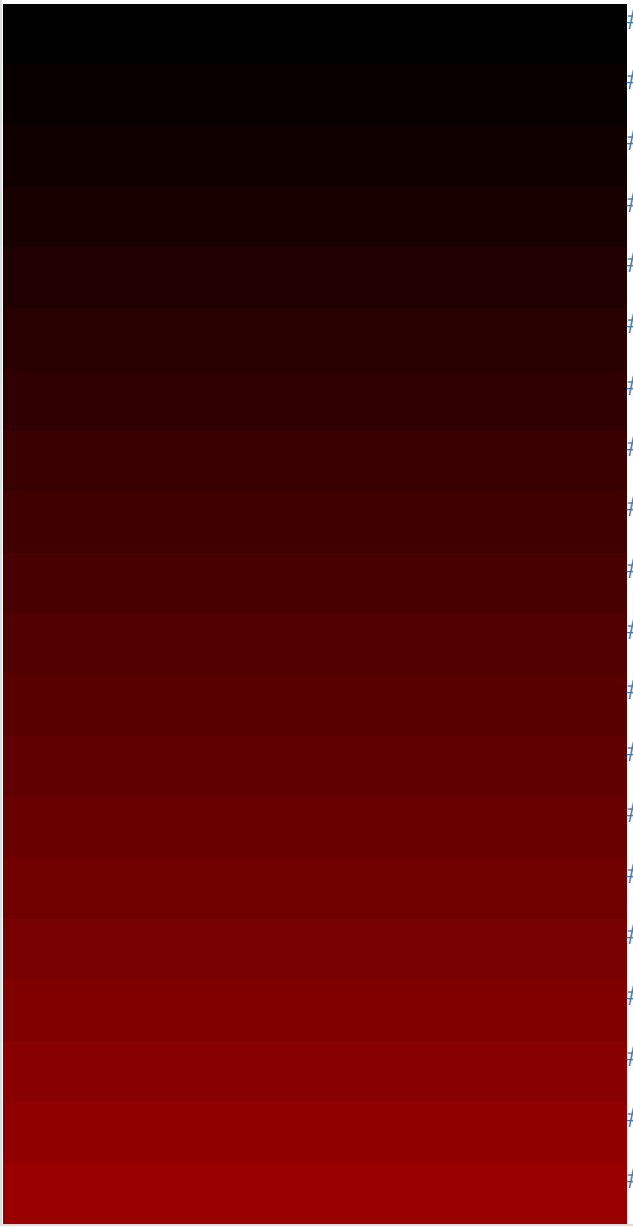
16 Million Different Colors

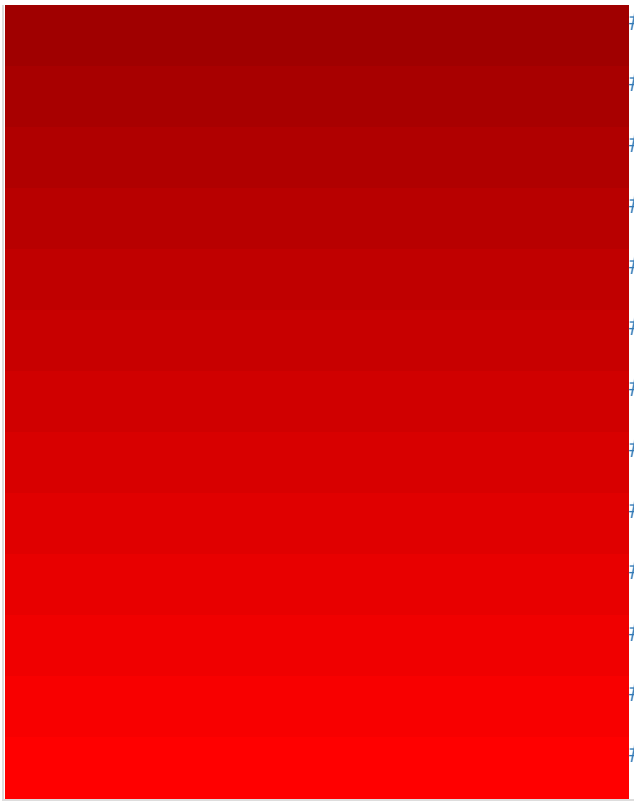
The combination of Red, Green and Blue values from 0 to 255 gives a total of more than 16 million different colors to play with (256 x 256 x 256).

Most modern monitors are capable of displaying at least 16384 different colors.

If you look at the color table below, you will see the result of varying the red light from 0 to 255, while keeping the green and blue light at zero.

To see a full list of color mixes when the red light varies from 0 to 255, click on one of the hex or rgb values below.

Red Light	HEX	RGB
	#000000	rgb(0,0,0)
	#080000	rgb(8,0,0)
	#100000	rgb(16,0,0)
	#180000	rgb(24,0,0)
	#200000	rgb(32,0,0)
	#280000	rgb(40,0,0)
	#300000	rgb(48,0,0)
	#380000	rgb(56,0,0)
	#400000	rgb(64,0,0)
	#480000	rgb(72,0,0)
	#500000	rgb(80,0,0)
	#580000	rgb(88,0,0)
	#600000	rgb(96,0,0)
	#680000	rgb(104,0,0)
	#700000	rgb(112,0,0)
	#780000	rgb(120,0,0)
	#800000	rgb(128,0,0)
	#880000	rgb(136,0,0)
	#900000	rgb(144,0,0)
	#980000	rgb(152,0,0)



#A00000	rgb(160,0,0)
#A80000	rgb(168,0,0)
#B00000	rgb(176,0,0)
#B80000	rgb(184,0,0)
#C00000	rgb(192,0,0)
#C80000	rgb(200,0,0)
#D00000	rgb(208,0,0)
#D80000	rgb(216,0,0)
#E00000	rgb(224,0,0)
#E80000	rgb(232,0,0)
#F00000	rgb(240,0,0)
#F80000	rgb(248,0,0)
#FF0000	rgb(255,0,0)

In the Stone Age

In the stone age, when computers only supported 256 different colors, a list of 216 "Web Safe Colors" was suggested as a Web standard, reserving 40 fixed system colors.

This 216 cross-browser color palette was created to ensure that all computers would display colors correctly:

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF

339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

HTML Scripts

« Previous

Next Chapter »

JavaScripts make HTML pages more dynamic and interactive.

Try it Yourself - Examples

Insert a script
How to insert a script into an HTML document.

Use of the <noscript> tag
How to handle browsers that do not support scripting, or have scripting disabled.

The HTML <script> Tag

The <script> tag is used to define a client-side script, such as a JavaScript.

The <script> element either contains scripting statements or it points to an external script file through the src attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

The script below writes Hello JavaScript! into an HTML element with id="demo":

Example

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

Try it Yourself »

To learn all about JavaScript, visit our JavaScript Tutorial!

The HTML <noscript> Tag

The <noscript> tag is used to provide an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support client-side scripting.

The <noscript> element can contain all the elements that you can find inside the <body> element of a normal HTML page.

The content inside the <noscript> element will only be displayed if scripts are not supported, or are disabled in the user's browser:

Example

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>

<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

Try it Yourself »

A Taste of JavaScript (From Our JavaScript Tutorial)

Here are some examples of what JavaScript can do:

JavaScript can change HTML content:

```
document.getElementById("demo").innerHTML = "Hello JavaScript!";
```

Try it Yourself »

JavaScript can change HTML styles:

```
document.getElementById("demo").style.fontSize = "25px";
```

Try it Yourself »

JavaScript can change HTML attributes:

```
document.getElementById("image").src = "picture.gif";
```

Try it Yourself »

HTML Script Tags

Tag	Description
<script>	Defines a client-side script
<noscript>	Defines an alternate content for users that do not support client-side scripts

---

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

« Previous

Next Chapter »

HTML Head

« Previous

Next Chapter »

The HTML <head> Element

The <head> element is a container for meta data (data about data).

HTML meta data is data about the HTML document. Metadata is not displayed.

Meta data typically define document title, styles, links, scripts, and other meta information.

The following tags describes meta data: <title>, <style>, < meta>, <link>, <script>, and <base>.

Omitting <html> and <body>

In the HTML5 standard, the <html> tag, the <body> tag, and the <head> tag can be omitted.

The following code will validate as HTML5:

Example

```
<!DOCTYPE html>
<head>
<title>Page Title</title>
</head>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

Try it Yourself »

W3Schools does not recommend omitting the <html> and <body> tags:

The <html> element is the document root. It is the recommended place for specifying the page language:

```
<!DOCTYPE html>
<html lang="en-US">
```

Declaring a language is important for accessibility applications (screen readers) and search engines.

Omitting <html> and <body> can crash badly written DOM and XML software.

Finally, omitting <body> can produce errors in older browsers (IE9).

Omitting <head>

In the HTML5 standard, the <head> tag can also be omitted.

By default, browsers will add all elements before <body>, to a default <head> element.

You can reduce the complexity of HTML, by omitting the <head> tag:

Example

```
<!DOCTYPE html>
<html>
<title>Page Title</title>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Try it Yourself »

Omitting tags is unfamiliar to web developers. It needs time to be established as a guideline.

The HTML <title> Element

The <title> element defines the title of the document.

The <title> element is required in all HTML/XHTML documents.

The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine results

A simplified HTML document:

Example

```
<!DOCTYPE html>
<html>
<title>Page Title</title>

<body>
The content of the document......
</body>
```

```
</html>
```

Try it Yourself »

The HTML <style> Element

The <style> element is used to define style information for an HTML document.

Inside the <style> element you specify how HTML elements should render in a browser:

Example

```
<style>
body {background-color:yellow;}
p {color:blue;}
</style>
```

Try it Yourself »

The HTML <link> Element

The <link> element defines the page relationship to an external resource.

The <link> element is most often used to link to style sheets:

Example

```
<link rel="stylesheet" href="mystyle.css">
```

Try it Yourself »

The HTML <meta> Element

The <meta> element is used to specify page description, keywords, author, and other metadata.

Meta data is used by browsers (how to display content), by search engines (keywords), and other web services.

Define keywords for search engines:

Example

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

Define a description of your web page:

Example

```
<meta name="description" content="Free Web tutorials on HTML and CSS">
```

Define the character set used:

Example

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

Define the author of a page:

Example

```
<meta name="author" content="Hege Refsnes">
```

Try it Yourself »

Refresh document every 30 seconds:

Example

```
<meta http-equiv="refresh" content="30">
```

The HTML <script> Element

The <script> element is used to define client-side JavaScripts.

The script below writes Hello JavaScript! into an HTML element with id="demo":

Example

```
<script>
function myFunction {
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
}
</script>
```

Try it Yourself »

To learn all about JavaScript, visit our JavaScript Tutorial!

The HTML <base> Element

The <base> element specifies the base URL and base target for all relative URLs in a page:

Example

```
<base href="http://www.w3schools.com/images/" target="_blank">
```

Try it Yourself »

HTML head Elements

Tag	Description
<head>	Defines information about the document
<title>	Defines the title of a document
<base>	Defines a default address or a default target for all links on a page
<link>	Defines the relationship between a document and an external resource
<meta>	Defines metadata about an HTML document
<script>	Defines a client-side script
<style>	Defines style information for a document

« Previous

Next Chapter »

HTML Forms

HTML FormsHTML Form ElementsHTML Input TypesHTML Input Attributes

HTML5

HTML5 IntroHTML5 SupportHTML5 ElementsHTML5 SemanticsHTML5 MigrationHTML5 Style Guide

HTML Graphics

HTML CanvasHTML SVG

HTML Media

HTML MediaHTML VideoHTML AudioHTML Plug-insHTML YouTube

HTML APIs

HTML GeolocationHTML Drag/DropHTML Local StorageHTML App CacheHTML Web WorkersHTML SSE

HTML Examples

HTML ExamplesHTML QuizHTML5 QuizHTML CertificateHTML5 CertificateHTML Summary

HTML References

HTML Tag ListHTML AttributesHTML EventsHTML CanvasHTML Audio/VideoHTML DoctypesHTML ColornamesHTML ColorpickerHTML ColormixerHTML Character SetsHTML URL EncodeHTML Lang CodesHTTP MessagesHTTP MethodsPX to EM ConverterKeyboard Shortcuts

HTML Entities

« Previous

Next Chapter »

Reserved characters in HTML must be replaced with character entities.

Characters, not present on your keyboard, can also be replaced by entities.

HTML Entities

Some characters are reserved in HTML.

If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.

Character entities are used to display reserved characters in HTML.

A character entity looks like this:

`&entity_name;`

OR

`&#entity_number;`
To display a less than sign we must write: **<** or **<**

The advantage of using an entity name, instead of a number, is that the name is easier to remember.
The disadvantage is that browsers may not support all entity names, but the support for numbers is good.

Non Breaking Space

A common character entity used in HTML is the non breaking space ( ).

Remember that browsers will always truncate spaces in HTML pages. If you write 10 spaces in your text, the browser will remove 9 of them. To add real spaces to your text, you can use the ** ** character entity.

Some Other Useful HTML Character Entities

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	<	<
>	greater than	>	>

&	ampersand	&	&
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥
€	euro	€	€
©	copyright	©	©
®	registered trademark	®	®

Entity names are case sensitive.

Combining Diacritical Marks

A diacritical mark is a "glyph" added to a letter.

Some diacritical marks, like grave (̀) and acute (́) are called accents.

Diacritical marks can appear both above and below a letter, inside a letter, and between two letters.

Diacritical marks can be used in combination with alphanumeric characters, to produce a character that is not present in the character set (encoding) used in the page.

Here are some examples:

Mark	Character	Construct	Result
̀	a	à	à
́	a	á	á
̂	a	â	â
̃	a	ã	ã
̀	O	Ò	Ò
́	O	Ó	Ó
̂	O	Ô	Ô
̃	O	Õ	Õ

You will see more HTML symbols in the next chapter of this tutorial.

« Previous

Next Chapter »

HTML Symbols

« Previous

Next Chapter »

HTML Symbol Entities

HTML entities were described in the previous chapter.

Many mathematical, technical, and currency symbols, are not present on a normal keyboard.

To add these symbols to an HTML page, you can use an HTML entity name.

If no entity name exists, you can use an entity number; a decimal (or hexadecimal) reference.

If you use an HTML entity name or a hexadecimal number, the character will always display correctly.
This is independent of what character set (encoding) your page uses!

Example

```
<p>I will display &euro;</p>
<p>I will display &#8364;</p>
<p>I will display &#x20AC;</p>
```

Will display as:

```
I will display €
I will display €
I will display €
```

Try it Yourself »

Some Mathematical Symbols Supported by HTML

Char	Number	Entity	Description
∀	∀	∀	FOR ALL
∂	∂	∂	PARTIAL DIFFERENTIAL
∃	∃	∃	THERE EXISTS
∅	∅	∅	EMPTY SETS
∇	∇	∇	NABLA
∈	∈	∈	ELEMENT OF
∉	∉	∉	NOT AN ELEMENT OF
∋	∋	∋	CONTAINS AS MEMBER
∏	∏	∏	N-ARY PRODUCT
∑	∑	∑	N-ARY SUMMATION

Full Math Reference

Some Greek Letters Supported by HTML

Char	Number	Entity	Description
Α	Α	Α	GREEK CAPITAL LETTER ALPHA
Β	Β	Β	GREEK CAPITAL LETTER BETA
Γ	Γ	Γ	GREEK CAPITAL LETTER GAMMA
Δ	Δ	Δ	GREEK CAPITAL LETTER DELTA
Ε	Ε	Ε	GREEK CAPITAL LETTER EPSILON
Ζ	Ζ	Ζ	GREEK CAPITAL LETTER ZETA

Full Greek Reference

Some Other Entities Supported by HTML

Char	Number	Entity	Description
©	©	©	COPYRIGHT SIGN
®	®	®	REGISTERED SIGN
€	€	€	EURO SIGN
™	™	™	TRADEMARK
←	←	←	LEFTWARDS ARROW
↑	↑	↑	UPWARDS ARROW
→	→	→	RIGHTWARDS ARROW
↓	↓	↓	DOWNWARDS ARROW
♠	♠	♠	BLACK SPADE SUIT
♣	♣	♣	BLACK CLUB SUIT
♥	♥	♥	BLACK HEART SUIT
♦	♦	♦	BLACK DIAMOND SUIT

Full Currency Reference

Full Arrows Reference

Full Symbols Reference

« Previous

Next Chapter »

HTML Encoding (Character Sets)

« Previous

Next Chapter »

To display an HTML page correctly, a web browser must know the character set (character encoding) to use.

What is Character Encoding?

ASCII was the first **character encoding standard** (also called character set). It defines 127 different alphanumeric characters that could be used on the internet.

ASCII supported numbers (0-9), English letters (A-Z), and some special characters like ! $ + - ( ) @ < > .

ANSI (Windows-1252) was the original Windows character set. It supported 256 different character codes.

ISO-8859-1 was the default character set for HTML 4. It also supported 256 different character codes.

Because ANSI and ISO was limited, the default character encoding was changed to UTF-8 in HTML5.

UTF-8 (Unicode) covers almost all of the characters and symbols in the world.

All HTML 4 processors also support UTF-8.

The HTML charset Attribute

To display an HTML page correctly, a web browser must know the character set used in the page.

This is specified in the <meta> tag:

For HTML4:

```
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
```

For HTML5:

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

If a browser detects ISO-8859-1 in a web page, it defaults to ANSI, because ANSI is identical to ISO-8859-1 except that ANSI has 32 extra characters.

Differences Between Character Sets

The following table displays the differences between the character sets described above:

Numb	ASCII	ANSI	8859	UTF-8	Description
32					space
33	!	!	!	!	exclamation mark
34	"	"	"	"	quotation mark
35	#	#	#	#	number sign
36	$	$	$	$	dollar sign
37	%	%	%	%	percent sign
38	&	&	&	&	ampersand
39	'	'	'	'	apostrophe
40	(	(	(	(	left parenthesis
41	)	)	)	)	right parenthesis
42	*	*	*	*	asterisk
43	+	+	+	+	plus sign
44	,	,	,	,	comma
45	-	-	-	-	hyphen-minus
46	.	.	.	.	full stop
47	/	/	/	/	solidus
48	0	0	0	0	digit zero
49	1	1	1	1	digit one
50	2	2	2	2	digit two
51	3	3	3	3	digit three
52	4	4	4	4	digit four
53	5	5	5	5	digit five
54	6	6	6	6	digit six

55	7	7	7	7	digit seven
56	8	8	8	8	digit eight
57	9	9	9	9	digit nine
58	:	:	:	:	colon
59	;	;	;	;	semicolon
60	<	<	<	<	less-than sign
61	=	=	=	=	equals sign
62	>	>	>	>	greater-than sign
63	?	?	?	?	question mark
64	@	@	@	@	commercial at
65	A	A	A	A	Latin capital letter A
66	B	B	B	B	Latin capital letter B
67	C	C	C	C	Latin capital letter C
68	D	D	D	D	Latin capital letter D
69	E	E	E	E	Latin capital letter E
70	F	F	F	F	Latin capital letter F
71	G	G	G	G	Latin capital letter G
72	H	H	H	H	Latin capital letter H
73	I	I	I	I	Latin capital letter I
74	J	J	J	J	Latin capital letter J
75	K	K	K	K	Latin capital letter K
76	L	L	L	L	Latin capital letter L
77	M	M	M	M	Latin capital letter M
78	N	N	N	N	Latin capital letter N
79	O	O	O	O	Latin capital letter O
80	P	P	P	P	Latin capital letter P
81	Q	Q	Q	Q	Latin capital letter Q
82	R	R	R	R	Latin capital letter R

83	S	S	S	S	Latin capital letter S
84	T	T	T	T	Latin capital letter T
85	U	U	U	U	Latin capital letter U
86	V	V	V	V	Latin capital letter V
87	W	W	W	W	Latin capital letter W
88	X	X	X	X	Latin capital letter X
89	Y	Y	Y	Y	Latin capital letter Y
90	Z	Z	Z	Z	Latin capital letter Z
91	[	[	[	[	left square bracket
92	\	\	\	\	reverse solidus
93	]	]	]	]	right square bracket
94	^	^	^	^	circumflex accent
95	_	_	_	_	low line
96	`	`	`	`	grave accent
97	a	a	a	a	Latin small letter a
98	b	b	b	b	Latin small letter b
99	c	c	c	c	Latin small letter c
100	d	d	d	d	Latin small letter d
101	e	e	e	e	Latin small letter e
102	f	f	f	f	Latin small letter f
103	g	g	g	g	Latin small letter g
104	h	h	h	h	Latin small letter h
105	i	i	i	i	Latin small letter i
106	j	j	j	j	Latin small letter j
107	k	k	k	k	Latin small letter k
108	l	l	l	l	Latin small letter l
109	m	m	m	m	Latin small letter m
110	n	n	n	n	Latin small letter n

111	o	o	o	o	Latin small letter o
112	p	p	p	p	Latin small letter p
113	q	q	q	q	Latin small letter q
114	r	r	r	r	Latin small letter r
115	s	s	s	s	Latin small letter s
116	t	t	t	t	Latin small letter t
117	u	u	u	u	Latin small letter u
118	v	v	v	v	Latin small letter v
119	w	w	w	w	Latin small letter w
120	x	x	x	x	Latin small letter x
121	y	y	y	y	Latin small letter y
122	z	z	z	z	Latin small letter z
123	{	{	{	{	left curly bracket
124	|	|	|	|	vertical line
125	}	}	}	}	right curly bracket
126	~	~	~	~	tilde
127	DEL				
128		€			euro sign
129					NOT USED
130		‚			single low-9 quotation mark
131		*ƒ*			Latin small letter f with hook
132		„			double low-9 quotation mark
133		…			horizontal ellipsis
134		†			dagger
135		‡			double dagger
136		ˆ			modifier letter circumflex accent
137		‰			per mille sign
138		Š			Latin capital letter S with caron

139	‹			single left-pointing angle quotation mark
140	Œ			Latin capital ligature OE
141				NOT USED
142	Ž			Latin capital letter Z with caron
143				NOT USED
144				NOT USED
145	‘			left single quotation mark
146	’			right single quotation mark
147	“			left double quotation mark
148	”			right double quotation mark
149	•			bullet
150	–			en dash
151	—			em dash
152	˜			small tilde
153	™			trade mark sign
154	š			Latin small letter s with caron
155	›			single right-pointing angle quotation mark
156	œ			Latin small ligature oe
157				NOT USED
158	ž			Latin small letter z with caron
159	Ÿ			Latin capital letter Y with diaeresis
160				no-break space
161	¡	¡	¡	inverted exclamation mark
162	¢	¢	¢	cent sign
163	£	£	£	pound sign
164	¤	¤	¤	currency sign
165	¥	¥	¥	yen sign
166	¦	¦	¦	broken bar

167	§	§	§	section sign
168	¨	¨	¨	diaeresis
169	©	©	©	copyright sign
170	ª	ª	ª	feminine ordinal indicator
171	«	«	«	left-pointing double angle quotation mark
172	¬	¬	¬	not sign
173				soft hyphen
174	®	®	®	registered sign
175	¯	¯	¯	macron
176	°	°	°	degree sign
177	±	±	±	plus-minus sign
178	²	²	²	superscript two
179	³	³	³	superscript three
180	´	´	´	acute accent
181	µ	µ	µ	micro sign
182	¶	¶	¶	pilcrow sign
183	·	·	·	middle dot
184	¸	¸	¸	cedilla
185	¹	¹	¹	superscript one
186	º	º	º	masculine ordinal indicator
187	»	»	»	right-pointing double angle quotation mark
188	¼	¼	¼	vulgar fraction one quarter
189	½	½	½	vulgar fraction one half
190	¾	¾	¾	vulgar fraction three quarters
191	¿	¿	¿	inverted question mark
192	À	À	À	Latin capital letter A with grave
193	Á	Á	Á	Latin capital letter A with acute
194	Â	Â	Â	Latin capital letter A with circumflex

195	Ã	Ã	Ã	Latin capital letter A with tilde
196	Ä	Ä	Ä	Latin capital letter A with diaeresis
197	Å	Å	Å	Latin capital letter A with ring above
198	Æ	Æ	Æ	Latin capital letter AE
199	Ç	Ç	Ç	Latin capital letter C with cedilla
200	È	È	È	Latin capital letter E with grave
201	É	É	É	Latin capital letter E with acute
202	Ê	Ê	Ê	Latin capital letter E with circumflex
203	Ë	Ë	Ë	Latin capital letter E with diaeresis
204	Ì	Ì	Ì	Latin capital letter I with grave
205	Í	Í	Í	Latin capital letter I with acute
206	Î	Î	Î	Latin capital letter I with circumflex
207	Ï	Ï	Ï	Latin capital letter I with diaeresis
208	Ð	Ð	Ð	Latin capital letter Eth
209	Ñ	Ñ	Ñ	Latin capital letter N with tilde
210	Ò	Ò	Ò	Latin capital letter O with grave
211	Ó	Ó	Ó	Latin capital letter O with acute
212	Ô	Ô	Ô	Latin capital letter O with circumflex
213	Õ	Õ	Õ	Latin capital letter O with tilde
214	Ö	Ö	Ö	Latin capital letter O with diaeresis
215	×	×	×	multiplication sign
216	Ø	Ø	Ø	Latin capital letter O with stroke
217	Ù	Ù	Ù	Latin capital letter U with grave
218	Ú	Ú	Ú	Latin capital letter U with acute
219	Û	Û	Û	Latin capital letter U with circumflex
220	Ü	Ü	Ü	Latin capital letter U with diaeresis
221	Ý	Ý	Ý	Latin capital letter Y with acute
222	Þ	Þ	Þ	Latin capital letter Thorn

223	ß	ß	ß	Latin small letter sharp s
224	à	à	à	Latin small letter a with grave
225	á	á	á	Latin small letter a with acute
226	â	â	â	Latin small letter a with circumflex
227	ã	ã	ã	Latin small letter a with tilde
228	ä	ä	ä	Latin small letter a with diaeresis
229	å	å	å	Latin small letter a with ring above
230	æ	æ	æ	Latin small letter ae
231	ç	ç	ç	Latin small letter c with cedilla
232	è	è	è	Latin small letter e with grave
233	é	é	é	Latin small letter e with acute
234	ê	ê	ê	Latin small letter e with circumflex
235	ë	ë	ë	Latin small letter e with diaeresis
236	ì	ì	ì	Latin small letter i with grave
237	í	í	í	Latin small letter i with acute
238	î	î	î	Latin small letter i with circumflex
239	ï	ï	ï	Latin small letter i with diaeresis
240	ð	ð	ð	Latin small letter eth
241	ñ	ñ	ñ	Latin small letter n with tilde
242	ò	ò	ò	Latin small letter o with grave
243	ó	ó	ó	Latin small letter o with acute
244	ô	ô	ô	Latin small letter o with circumflex
245	õ	õ	õ	Latin small letter o with tilde
246	ö	ö	ö	Latin small letter o with diaeresis
247	÷	÷	÷	division sign
248	ø	ø	ø	Latin small letter o with stroke
249	ù	ù	ù	Latin small letter u with grave
250	ú	ú	ú	Latin small letter u with acute

251	û	û	û	Latin small letter with circumflex
252	ü	ü	ü	Latin small letter u with diaeresis
253	ý	ý	ý	Latin small letter y with acute
254	þ	þ	þ	Latin small letter thorn
255	ÿ	ÿ	ÿ	Latin small letter y with diaeresis

The ASCII Character Set

ASCII uses the values from 0 to 31 (and 127) for control characters.

ASCII uses the values from 32 to 126 for letters, digits, and symbols.

ASCII does not use the values from 128 to 255.

The ANSI Character Set (Windows-1252)

ANSI is identical to ASCII for the values from 0 to 127.

ANSI has a proprietary set of characters for the values from 128 to 159.

ANSI is identical to UTF-8 for the values from 160 to 255.

The ISO-8859-1 Character Set

8859-1 is identical to ASCII for the values from 0 to 127.

8859-1 does not use the values from 128 to 159.

8859-1 is identical to UTF-8 for the values from 160 to 255.

The UTF-8 Character Set

UTF-8 is identical to ASCII for the values from 0 to 127.

UTF-8 does not use the values from 128 to 159.

UTF-8 is identical to both ANSI and 8859-1 for the values from 160 to 255.

UTF-8 continues from the value 256 with more than 10.000 different characters.

For a closer look, study our Complete HTML Character Set Reference.

« Previous

Next Chapter »

HTML Uniform Resource Locators

« Previous

Next Chapter »

A URL is another word for a web address.

A URL can be composed of words (w3schools.com), or an Internet Protocol (IP) address (192.68.20.50).

Most people enter the name when surfing, because names are easier to remember than numbers.

URL - Uniform Resource Locator

Web browsers request pages from web servers by using a URL.

When you click on a link in an HTML page, an underlying <a> tag points to an address on the web.

A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.

A web address, like http://www.w3schools.com/html/default.asp follows these syntax rules:

Example

```
scheme://host.domain:port/path/filename
```

Explanation:

- **scheme** - defines the **type** of Internet service (most common is **http**)
- **host** - defines the **domain host** (default host for http is **www**)
- **domain** - defines the Internet **domain name** (w3schools.com)
- **port** - defines the **port number** at the host (default for http is **80**)
- **path** - defines a **path** at the server (If omitted: the root directory of the site)
- **filename** - defines the name of a document or resource

Common URL Schemes

The table below lists some common schemes:

Scheme	Short for	Used for
http	HyperText Transfer Protocol	Common web pages. Not encrypted
https	Secure HyperText Transfer Protocol	Secure web pages. Encrypted
ftp	File Transfer Protocol	Downloading or uploading files
file		A file on your computer

URL Encoding

URLs can only be sent over the Internet using the ASCII character-set.

Since URLs often contain characters outside the ASCII set, the URL has to be converted into ASCII.

URL encoding converts characters into a format that can be transmitted over the Internet.

URL encoding replaces non ASCII characters with a "%" followed by hexadecimal digits.

URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign, or %20.

Try It Yourself

Hello Günter Submit

If you click "Submit", the browser will URL encode the input before it is sent to the server.

A page at the server will display the received input.

Try some other input and click Submit again.

ASCII Encoding Examples

Your browser will encode input, according to the character-set used in your page.

The default character-set in HTML5 is UTF-8.

Character	From Windows-1252	From UTF-8
€	%80	%E2%82%AC
£	%A3	%C2%A3
©	%A9	%C2%A9
®	%AE	%C2%AE
À	%C0	%C3%80
Á	%C1	%C3%81
Â	%C2	%C3%82
Ã	%C3	%C3%83
Ä	%C4	%C3%84
Å	%C5	%C3%85

For a complete reference of all URL encodings, visit our URL Encoding Reference.

« Previous

Next Chapter »

HTML and XHTML

« Previous

Next Chapter »

XHTML is HTML written as XML.

What Is XHTML?

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is almost identical to HTML
- XHTML is stricter than HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Why XHTML?

Many pages on the internet contain "bad" HTML.

This HTML code works fine in most browsers (even if it does not follow the HTML rules):

```
<html>
<head>
  <title>This is bad HTML</title>

<body>
  <h1>Bad HTML
  <p>This is a paragraph
</body>
```

Today's market consists of different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices. Smaller devices often lack the resources or power to interpret "bad" markup.

XML is a markup language where documents must be marked up correctly (be "well-formed").

If you want to study XML, please read our XML tutorial.

By combining the strengths of HTML and XML, XHTML was developed.

XHTML is HTML redesigned as XML.

The Most Important Differences from HTML:

Document Structure

- XHTML DOCTYPE is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**

XHTML Elements

- XHTML elements must be **properly nested**
- XHTML elements must always be **closed**

- XHTML elements must be in **lowercase**
- XHTML documents must have **one root element**

XHTML Attributes

- Attribute names must be in **lower case**
- Attribute values must be **quoted**
- Attribute minimization is **forbidden**

<!DOCTYPE ....> Is Mandatory

An XHTML document must have an XHTML DOCTYPE declaration.

A complete list of all the XHTML Doctypes is found in our HTML Tags Reference.

The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document.

This example shows an XHTML document with a minimum of required tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Title of document</title>
</head>

<body>
  some content
</body>

</html>
```

XHTML Elements Must Be Properly Nested

In HTML, some elements can be improperly nested within each other, like this:

```
<b><i>This text is bold and italic</b></i>
```

In XHTML, all elements must be properly nested within each other, like this:

```
<b><i>This text is bold and italic</i></b>
```

XHTML Elements Must Always Be Closed

This is wrong:

```
<p>This is a paragraph
<p>This is another paragraph
```

This is correct:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Empty Elements Must Also Be Closed

This is wrong:

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

This is correct:

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

XHTML Elements Must Be In Lower Case

This is wrong:

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

This is correct:

```
<body>
<p>This is a paragraph</p>
</body>
```

XHTML Attribute Names Must Be In Lower Case

This is wrong:

```
<table WIDTH="100%">
```

This is correct:

```
<table width="100%">
```

Attribute Values Must Be Quoted

This is wrong:

```
<table width=100%>
```

This is correct:

```
<table width="100%">
```

Attribute Minimization Is Forbidden

Wrong:

```
<input type="checkbox" name="vehicle" value="car" checked />
```

Correct:

```
<input type="checkbox" name="vehicle" value="car" checked="checked" />
```

Wrong:

```
<input type="text" name="lastname" disabled />
```

Correct:

```
<input type="text" name="lastname" disabled="disabled" />
```

How to Convert from HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page
2. Add an xmlns attribute to the html element of every page
3. Change all element names to lowercase
4. Close all empty elements
5. Change all attribute names to lowercase
6. Quote all attribute values

Validate XHTML With The W3C Validator

Put your web address in the box below:

http://www.w3schools.com/html/demo_xhtml.asp

Validate the page

« Previous

Next Chapter »

HTML Forms

« Previous

Next Chapter »

The <form> Element

HTML forms are used to collect user input.

The **<form>** element defines an HTML form:

Example

```
<form>
.
form elements
.
</form>
```

HTML forms contain **form elements**.

Form elements are different types of input elements, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

The **<input>** element is the most important **form element**.

The <input> element has many variations, depending on the **type** attribute.

Here are the types used in this chapter:

Type	Description
text	Defines normal text input
radio	Defines radio button input (for selecting one of many choices)
submit	Defines a submit button (for submitting the form)

You will learn a lot more about input types later in this tutorial.

Text Input

<input type="text"> defines a one-line input field for **text input**:

Example

```
<form>
First name:<br>
<input type="text" name="firstname">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

This is how it will look like in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Radio Button Input

<input type="radio"> defines a **radio button**.

Radio buttons let a user select ONE of a limited number of choices:

Example

```
<form>
<input type="radio" name="sex" value="male" checked>Male
<br>
<input type="radio" name="sex" value="female">Female
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

- Male
- Female

The Submit Button

<input type="submit"> defines a button for **submitting** a form to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's **action** attribute:

Example

```
<form action="action_page.php">
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit">
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

First name:

Mickey

Last name:

Mouse

Submit

The Action Attribute

The **action attribute** defines the action to be performed when the form is submitted.

The common way to submit a form to a server, is by using a submit button.

Normally, the form is submitted to a web page on a web server.

In the example above, a server-side script is specified to handle the submitted form:

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

If the action attribute is omitted, the action is set to the current page.

The Method Attribute

The **method attribute** specifies the HTTP method (**GET** or **POST**) to be used when submitting the forms:

```
<form action="action_page.php" method="GET">
```

or:

```
<form action="action_page.php" method="POST">
```

When to Use GET?

You can use GET (the default method):

If the form submission is passive (like a search engine query), and without sensitive information.

When you use GET, the form data will be visible in the page address:

```
action_page.php?firstname=Mickey&lastname=Mouse
```

GET is best suited to short amounts of data. Size limitations are set in your browser.

When to Use POST?

You should use POST:

If the form is updating data, or includes sensitive information (password).

POST offers better security because the submitted data is not visible in the page address.

The Name Attribute

To be submitted correctly, each input field must have a name attribute.

This example will only submit the "Last name" input field:

Example

```
<form action="action_page.php">
First name:<br>
<input type="text" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit">
</form>
```

Try it Yourself »

Grouping Form Data with <fieldset>

The **<fieldset>** element groups related data in a form.

The **<legend>** element defines a caption for the <fieldset> element.

Example

```
<form action="action_page.php">
<fieldset>
<legend>Personal information:</legend>
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit"></fieldset>
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

Personal information:First name:

Mickey

Last name:

Mouse

Submit

HTML Form Attributes

An HTML <form> element, with all possible attributes set, will look like this:

Example

```
<form action="action_page.php" method="GET" target="_blank" accept-charset="UTF-8"
enctype="application/x-www-form-urlencoded" autocomplete="off" novalidate>
.
form elements
.
</form>
```

Here is the list of <form> attributes:

Attribute	Description
accept-charset	Specifies the charset used in the submitted form (default: the page charset).

action	Specifies an address (url) where to submit the form (default: the submitting page).
autocomplete	Specifies if the browser should autocomplete the form (default: on).
enctype	Specifies the encoding of the submitted data (default: is url-encoded).
method	Specifies the HTTP method used when submitting the form (default: GET).
name	Specifies a name used to identify the form (for DOM usage: document.forms.name).
novalidate	Specifies that the browser should not validate the form.
target	Specifies the target of the address in the action attribute (default: _self).

You will learn more about attributes in the next chapters.

More Examples

Send e-mail from a form
How to send e-mail from a form.

« Previous

Next Chapter »

HTML Form Elements

« Previous

Next Chapter »

This chapter describes all HTML form elements.

The <input> Element

The most important form element is the **<input>** element.

The <input> element can vary in many ways, depending on the **type** attribute.

All HTML input types are covered in the next chapter.

The <select> Element (Drop-Down List)

The **<select>** element defines a **drop-down** list:

Example

```
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
```

Try it Yourself »

The **<option>** elements defines the options to select.

The list will normally show the first item as selected.

You can add a selected attribute to define a predefined option.

Example

```
<option value="fiat" selected>Fiat</option>
```

Try it Yourself »

The <textarea> Element

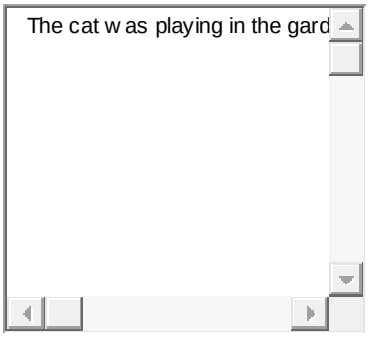
The **<textarea>** element defines a multi-line input field (**a text area**):

Example

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:



The <button> Element

The **<button>** element defines a clickable **button**:

Example

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

Click Me!

HTML5 Form Elements

HTML5 added the following form elements:

- <datalist>
- <keygen>
- <output>

By default, browsers do not display unknown elements. New elements will not destroy your page.

HTML5 <datalist> Element

The **<datalist>** element specifies a list of pre-defined options for an <input> element.

Users will see a drop-down list of pre-defined options as they input data.

The **list** attribute of the <input> element, must refer to the **id** attribute of the <datalist> element.

Example

An <input> element with pre-defined values in a <datalist>:

```
<form action="action_page.php">
<input list="browsers">
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
</form>
```

Try it Yourself »

HTML5 <keygen> Element

The purpose of the **<keygen>** element is to provide a secure way to authenticate users.

The <keygen> element specifies a key-pair generator field in a form.

When the form is submitted, two keys are generated, one private and one public.

The private key is stored locally, and the public key is sent to the server.

The public key could be used to generate a client certificate to authenticate the user in the future.

Example

A form with a keygen field:

```
<form action="action_page.php">
Username: <input type="text" name="user">
Encryption: <keygen name="security">
<input type="submit">
</form>
```

Try it Yourself »

HTML5 <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

Example

Perform a calculation and show the result in an <output> element:

```
<form action="action_page.asp"
oninput="x.value=parseInt(a.value)+parseInt(b.value)">
0
<input type="range"  id="a" name="a" value="50">
100 +
<input type="number" id="b" name="b" value="50">
=
<output name="x" for="a b"></output>
<br><br>
<input type="submit">
</form>
```

Try it Yourself »

HTML Form Elements

= new in HTML5.

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control

<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button
<datalist>	Specifies a list of pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

« Previous

Next Chapter »

HTML Input Types

« Previous

Next Chapter »

Input Types

This chapter describes the input types of the <input> element.

Input Type: text

<input type="text"> defines a one-line input field for **text input**:

Example

```
<form>
First name:<br>
<input type="text" name="firstname">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

First name:

Last name:

Input Type: password

<input type="password"> defines a **password field**:

Example

```
<form>
User name:<br>
<input type="text" name="username">
<br>
User password:<br>
<input type="password" name="psw">
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

User name:

User password:

The characters in a password field are masked (shown as asterisks or circles).

Input Type: submit

<input type="submit"> defines a button for **submitting** form input to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's action attribute:

Example

```
<form action="action_page.php">
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit">
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

First name:

Mickey

Last name:

Mouse

Submit

If you omit the submit button's value attribute, the button will get a default text:

Example

```
<form action="action_page.php">
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
```

```
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit">
</form>
```

Try it Yourself »

Input Type: radio

<input type="radio"> defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices:

Example

```
<form>
<input type="radio" name="sex" value="male" checked>Male
<br>
<input type="radio" name="sex" value="female">Female
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

- Male
- Female

Input Type: checkbox

<input type="checkbox"> defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike
<br>
```

```
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

I have a bike

I have a car

Input Type: button

<input type="button"> defines a **button**:

Example

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

Try it Yourself »

This is how the HTML code above will be displayed in a browser:

HTML5 Input Types

HTML5 added several new input types:

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Input types, not supported by old web browsers, will behave as input type text.

Input Type: number

The **<input type="number">** is used for input fields that should contain a numeric value.

You can set restrictions on the numbers.

Depending on browser support, the restrictions can apply to the input field.

Example

```
<form>
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">
</form>
```

Try it Yourself »

Input Restrictions

Here is a list of some common input restrictions (some are new in HTML5):

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field

step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

You will learn more about input restrictions in the next chapter.

Example

```
<form>
  Quantity:
  <input type="number" name="points" min="0" max="100" step="10" value="30">
</form>
```

Try it Yourself »

Input Type: date

The **<input type="date">** is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  Birthday:
  <input type="date" name="bday">
</form>
```

Try it Yourself »

You can add restrictions to the input:

Example

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```

Try it Yourself »

Input Type: color

The **<input type="color">** is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

Example

```
<form>
  Select your favorite color:
  <input type="color" name="favcolor">
</form>
```

Try it Yourself »

Input Type: range

The **<input type="range">** is used for input fields that should contain a value within a range.

Depending on browser support, the input field can be displayed as a slider control.

Example

```
<form>
  <input type="range" name="points" min="0" max="10">
</form>
```

Try it Yourself »

You can use the following attributes to specify restrictions: min, max, step, value.

Input Type: month

The **<input type="month">** allows the user to select a month and year.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  Birthday (month and year):
  <input type="month" name="bdaymonth">
</form>
```

Try it Yourself »

Input Type: week

The **<input type="week">** allows the user to select a week and year.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  Select a week:
  <input type="week" name="week_year">
</form>
```

Try it Yourself »

Input Type: time

The **<input type="time">** allows the user to select a time (no time zone).

Depending on browser support, a time picker can show up in the input field.

Example

```
<form>
  Select a time:
```

```
  <input type="time" name="usr_time">
</form>
```

Try it Yourself »

Input Type: datetime

The **<input type="datetime">** allows the user to select a date and time (with time zone).

Example

```
<form>
  Birthday (date and time):
  <input type="datetime" name="bdaytime">
</form>
```

Try it Yourself »

The input type datetime is removed from the HTML standard. Use datetime-local instead.

Input Type: datetime-local

The **<input type="datetime-local">** allows the user to select a date and time (no time zone).

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
  Birthday (date and time):
  <input type="datetime-local" name="bdaytime">
</form>
```

Try it Yourself »

Input Type: email

The **<input type="email">** is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.

Example

```
<form>
  E-mail:
  <input type="email" name="email">
</form>
```

Try it Yourself »

Input Type: search

The **<input type="search">** is used for search fields (a search field behaves like a regular text field).

Example

```
<form>
  Search Google:
  <input type="search" name="googlesearch">
</form>
```

Try it Yourself »

Input Type: tel

The **<input type="tel">** is used for input fields that should contain a telephone number.

The tel type is currently supported only in Safari 8.

Example

```
<form>
  Telephone:
  <input type="tel" name="usrtel">
</form>
```

Try it Yourself »

Input Type: url

The **<input type="url">** is used for input fields that should contain a URL address.

Depending on browser support, the url field can be automatically validated when submitted

Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

Example

```
<form>
  Add your homepage:
  <input type="url" name="homepage">
</form>
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »

« Previous

Next Chapter »

HTML Input Attributes

« Previous

Next Chapter »

The value Attribute

The **value** attribute specifies the initial value for an input field:

Example

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

The readonly Attribute

The **readonly** attribute specifies that the input field is read only (cannot be changed):

Example

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John" readonly>
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

The readonly attribute does not need a value. It is the same as writing readonly="readonly".

The disabled Attribute

The **disabled** attribute specifies that the input field is disabled.

A disabled element is un-usable and un-clickable.

Disabled elements will not be submitted.

Example

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John" disabled>
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

The disabled attribute does not need a value. It is the same as writing disabled="disabled".

The size Attribute

The **size** attribute specifies the size (in characters) for the input field:

Example

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John" size="40">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

The maxlength Attribute

The **maxlength** attribute specifies the maximum allowed length for the input field:

Example

```
<form action="">
First name:<br>
<input type="text" name="firstname" maxlength="10">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

Try it Yourself »

With a maxlength attribute, the input control will not accept more than the allowed number of characters.

The attribute does not provide any feedback. If you want to alert the user, you must write JavaScript code.

Input restrictions are not foolproof. JavaScript provides many ways to add illegal input.
To safely restrict input, restrictions must be checked by the receiver (the server) as well.

HTML5 Attributes

HTML5 added the following attributes for <input>:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

and the following attributes for <form>:

- autocomplete
- novalidate

The autocomplete Attribute

The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

Tip: It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

Example

An HTML form with autocomplete on (and off for one input field):

```
<form action="action_page.php" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email" autocomplete="off"><br>
  <input type="submit">
</form>
```

Try it Yourself »

Tip: In some browsers you may need to activate the autocomplete function for this to work.

The novalidate Attribute

The novalidate attribute is a <form> attribute

When present, novalidate specifies that form data should not be validated when submitted.

Example

Indicates that the form is not to be validated on submit:

```
<form action="action_page.php" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

Try it Yourself »

The autofocus Attribute

The autofocus attribute is a boolean attribute.

When present, it specifies that an <input> element should automatically get focus when the page loads.

Example

Let the "First name" input field automatically get focus when the page loads:

```
First name:<input type="text" name="fname" autofocus>
```

Try it Yourself »

The form Attribute

The form attribute specifies one or more forms an <input> element belongs to.

Tip: To refer to more than one form, use a space-separated list of form ids.

Example

An input field located outside the HTML form (but still a part of the form):

```
<form action="action_page.php" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

Last name: <input type="text" name="lname" form="form1">
```

Try it Yourself »

The formaction Attribute

The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.

The formaction attribute overrides the action attribute of the <form> element.

The formaction attribute is used with type="submit" and type="image".

Example

An HTML form with two submit buttons, with different actions:

```
<form action="action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="demo_admin.asp"
  value="Submit as admin">
</form>
```

Try it Yourself »

The formenctype Attribute

The formenctype attribute specifies how the form-data should be encoded when submitting it to the server (only for forms with method="post")

The formenctype attribute overrides the enctype attribute of the <form> element.

The formenctype attribute is used with type="submit" and type="image".

Example

Send form-data that is default encoded (the first submit button), and encoded as "multipart/form-data" (the second submit button):

```
<form action="demo_post_enctype.asp" method="post">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
```

```
  value="Submit as Multipart/form-data">
</form>
```

Try it Yourself »

The formmethod Attribute

The formmethod attribute defines the HTTP method for sending form-data to the action URL.

The formmethod attribute overrides the method attribute of the <form> element.

The formmethod attribute can be used with type="submit" and type="image".

Example

The second submit button overrides the HTTP method of the form:

```
<form action="action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post" formaction="demo_post.asp"
  value="Submit using POST">
</form>
```

Try it Yourself »

The formnovalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the <input> element should not be validated when submitted.

The formnovalidate attribute overrides the novalidate attribute of the <form> element.

The formnovalidate attribute can be used with type="submit".

Example

A form with two submit buttons (with and without validation):

```
<form action="action_page.php">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formnovalidate value="Submit without validation">
</form>
```

Try it Yourself »

The formtarget Attribute

The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

The formtarget attribute overrides the target attribute of the <form> element.

The formtarget attribute can be used with type="submit" and type="image".

Example

A form with two submit buttons, with different target windows:

```
<form action="action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit as normal">
  <input type="submit" formtarget="_blank"
  value="Submit to a new window">
</form>
```

Try it Yourself »

The height and width Attributes

The height and width attributes specify the height and width of an <input> element.

The height and width attributes are only used with <input type="image">.

Always specify the size of images. If the browser does not know the size, the page will flicker while images load.

Example

Define an image as the submit button, with height and width attributes:

```
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
```

Try it Yourself »

The list Attribute

The list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

Example

An <input> element with pre-defined values in a <datalist>:

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

Try it Yourself »

The min and max Attributes

The min and max attributes specify the minimum and maximum value for an <input> element.

The min and max attributes work with the following input types: number, range, date, datetime, datetime-local, month, time and week.

Example

<input> elements with min and max values:

```
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">
```

Try it Yourself »

The multiple Attribute

The multiple attribute is a boolean attribute.

When present, it specifies that the user is allowed to enter more than one value in the <input> element.

The multiple attribute works with the following input types: email, and file.

Example

A file upload field that accepts multiple values:

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

Try it Yourself »

The pattern Attribute

The pattern attribute specifies a regular expression that the <input> element's value is checked against.

The pattern attribute works with the following input types: text, search, url, tel, email, and password.

Tip: Use the global title attribute to describe the pattern to help the user.

Tip: Learn more about regular expressions in our JavaScript tutorial.

Example

An input field that can contain only three letters (no numbers or special characters):

```
Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}"
title="Three letter country code">
```

Try it Yourself »

The placeholder Attribute

The placeholder attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).

The hint is displayed in the input field before the user enters a value.

The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

Example

An input field with a placeholder text:

```
<input type="text" name="fname" placeholder="First name">
```

Try it Yourself »

The required Attribute

The required attribute is a boolean attribute.

When present, it specifies that an input field must be filled out before submitting the form.

The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Example

A required input field:

```
Username: <input type="text" name="usrname" required>
```

Try it Yourself »

The step Attribute

The step attribute specifies the legal number intervals for an <input> element.

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

Tip: The step attribute can be used together with the max and min attributes to create a range of legal values.

The step attribute works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

Example

An input field with a specified legal number intervals:

```
<input type="number" name="points" step="3">
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 »

« Previous

Next Chapter »

HTML5 Introduction

« Previous

Next Chapter »

What is New in HTML5?

The DOCTYPE declaration for HTML5 is very simple:

```
<!DOCTYPE html>
```

The character encoding (charset) declaration is also very simple:

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

HTML5 Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document......
</body>

</html>
```

The default character encoding in HTML5 is UTF-8.

New HTML5 Elements

The most interesting new elements are:

New **semantic** elements like <header>, <footer>, < article>, and <section>.

New form **control attributes** like number, date, time, calendar, and range.

New **graphic** elements: <svg> and <canvas>.

New **multimedia** elements: <audio> and < video>.

In the chapter HTML5 Support, you will learn how to "teach" old browsers to handle HTML5 semantic.

New HTML5 API's (Application Programming Interfaces)

The most interesting new API's are:

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers
- HTML SSE

Local storage is a powerful replacement for cookies.

Elements Removed in HTML5

The following HTML4 elements have been removed from HTML5:

Element	Use instead
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	<ul>
<font>	CSS

<frame>	
<frameset>	
<noframes>	
<strike>	CSS
<tt>	CSS

In the chapter HTML5 Migration, you will learn how to easily migrate from HTML4 to HTML5.

HTML History

Since the early days of the web, there have been many versions of HTML:

Version	Year
Tim Berners-Lee invented www	1989
Tim Berners-Lee invented HTML	1991
Dave Raggett drafted HTML+	1993
HTML Working Group defined HTML 2.0	1995
W3C Recommended HTML 3.2	1997
W3C Recommended HTML 4.01	1999
W3C Recommended XHTML 1.0	2000
HTML5 WHATWG First Public Draft	2008
HTML5 WHATWG Living Standard	2012
HTML5 W3C Final Recommendation	2014

Tim Berners-Lee invented the "World Wide Web" in 1989, and the Internet took off in the 1990s.

From 1991 to 1998, HTML developed from version 1 to version 4.

In 2000, the World Wide Web Consortium (W3C) recommended XHTML 1.0.

The XHTML syntax was strict, and the developers were forced to write valid and "well-formed" code.

In 2004, WHATWG (Web Hypertext Application Technology Working Group) was formed in response to slow W3C development, and W3C's decision to close down the development of HTML, in favor of XHTML.

WHATWG wanted to develop HTML, consistent with how the web was used, while being backward compatible with older versions of HTML.

In the period 2004-2006, the WHATWG initiative gained support by the major browser vendors.

In 2006, W3C announced that they would support WHATWG.

In 2008, the first HTML5 public draft was released

In 2012, WHATWG and W3C decided on a separation:

WHATWG will develop HTML as a "Living Standard".

A living standard is never fully complete, but always updated and improved. New features can be added, but old functionality can not be removed.

The WHATWG Living Standard was published in 2012, and is continuously updated.

W3C will develop a definitive HTML5 and XHTML5 standard, as a "snapshot" of WHATWG.

The W3C HTML5 recommendation was released 28. October 2014.

« Previous

Next Chapter »

HTML5 Browser Support

« Previous

Next Chapter »

You can teach old browsers to handle HTML5

HTML5 Browser Support

HTML5 is supported in all modern browsers.

In addition, all browsers, old and new, automatically handle unrecognized elements as inline elements.

Because of this, you can "teach" old browsers to handle "unknown" HTML elements.

You can even teach stone age IE6 (Windows XP 2001) how to handle unknown HTML elements.

Define HTML5 Elements as Block Elements

HTML5 defines 8 new **semantic** HTML elements. All these are **block level** elements.

To secure correct behavior in older browsers, you can set the CSS **display** property to **block**:

Example

```
header, section, footer, aside, nav, main, article, figure {
    display: block;
}
```

Adding New Elements to HTML

You can add any new element to HTML with a browser trick:

This example adds a new element called **<myHero>** to HTML, and defines a display style for it:

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Creating an HTML Element</title>
  <script>document.createElement("myHero")</script>
  <style>
  myHero {
      display: block;
      background-color: #ddd;
      padding: 50px;
      font-size: 30px;
  }
  </style>
</head>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>
```

```
<myHero>My First Hero</myHero>

</body>
</html>
```

Try it Yourself »

The JavaScript statement **document.createElement("myHero")** is added, only to satisfy IE.

Problem With Internet Explorer

You could use the solution described above, for all new HTML5 elements, but:

Internet Explorer 8 and earlier, does not allow styling of unknown elements.

Thankfully, Sjoerd Visscher created the "HTML5 Enabling JavaScript", "**the shiv**":

```
<!--[if lt IE 9]>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.2/html5shiv.js"></scrip
t>
< ![endif]-->
```

The code above is a comment, but versions previous to IE9 will read it (and understand it).

The Complete Shiv Solution

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Styling HTML5</title>
  <!--[if lt IE 9]>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.2/html5shiv.js"></scrip
t>
  < ![endif]-->
</head>
<body>
```

```
<h1>My First Article</h1>

<article>
London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.
</article>

</body>
</html>
```

Try it Yourself »

The link to the shiv code must be placed in the <head> element, because Internet Explorer needs to know about all new elements before reading them.

An HTML5 Skeleton

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Skeleton</title>
<meta charset="utf-8">

<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->

<style>
body {font-family: Verdana, sans-serif; font-size:0.8em;}
header,nav, section,article,footer
{border:1px solid grey; margin:5px; padding:8px;}
nav ul {margin:0; padding:0;}
nav ul li {display:inline; margin:5px;}
</style>
</head>
<body>

<header>
  <h1>HTML5 SKeleton</h1>
```

```
</header>

<nav>
<ul>
  <li><a href="html5_semantic_elements.asp">HTML5 Semantic</a></li>
  <li><a href="html5_geolocation.asp">HTML5 Geolocation</a></li>
  <li><a href="html5_canvas.asp">HTML5 Graphics</a></li>
</ul>
</nav>

<section>

<h1>Famous Cities</h1>

<article>
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</article>

<article>
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</article>

<article>
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</article>

</section>

<footer>
<p>&copy; 2014 W3Schools. All rights reserved.</p>
</footer>

</body>
</html>
```

Try it Yourself »

HTML5 New Elements

« Previous

Next Chapter »

New Elements in HTML5

Below is a list of the new HTML5 elements, and a description of what they are used for.

New Semantic/Structural Elements

HTML5 offers new elements for better document structure:

Tag	Description
<article>	Defines an article in the document
<aside>	Defines content aside from the page content
<bdi>	Defines a part of text that might be formatted in a different direction from other text
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for the document or a section
<header>	Defines a header for the document or a section
<main>	Defines the main content of a document
<mark>	Defines marked or highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu

<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links in the document
<progress>	Defines the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in the document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time
<wbr>	Defines a possible line-break

Read more about HTML5 Semantics.

New Form Elements

Tag	Description
<datalist>	Defines pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

Read all about old and new form elements in HTML Form Elements.

New Input Types

New Input Types

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel

New Input Attributes

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list

- time
- url
- week

- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

Learn all about old and new input types in HTML Input Types.

Learn all about input attributes in HTML Input Attributes.

HTML5 - New Attribute Syntax

HTML5 allows four different syntaxes for attributes.

This example demonstrates the different syntaxes used in an <input> tag:

Type	Example
Empty	<input type="text" value="John" **disabled**>
Unquoted	<input type="text" **value=John**>
Double-quoted	<input type="text" **value="John Doe"**>
Single-quoted	<input type="text" **value='John Doe'**>

In HTML5, all four syntaxes may be used, depending on what is needed for the attribute.

HTML5 Graphics

Tag	Description
<canvas>	Defines graphic drawing using JavaScript
<svg>	Defines graphic drawing using SVG

Read more about HTML5 Canvas.

Read more about HTML5 SVG.

New Media Elements

Tag	Description

<audio>	Defines sound or music content
<embed>	Defines containers for external applications (like plug-ins)
<source>	Defines sources for <video> and <audio>
<track>	Defines tracks for <video> and <audio>
<video>	Defines video or movie content

Read more about HTML5 Video.

Read more about HTML5 Audio.

« Previous

Next Chapter »

HTML5 Semantic Elements

« Previous

Next Chapter »

Semantics is the study of the meanings of words and phrases in language.

Semantic elements are elements with a meaning.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: and - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <img> - Clearly defines its content.

Browser Support

HTML5 semantic elements are supported in all modern browsers.

In addition, you can "teach" older browsers how to handle "unknown elements".

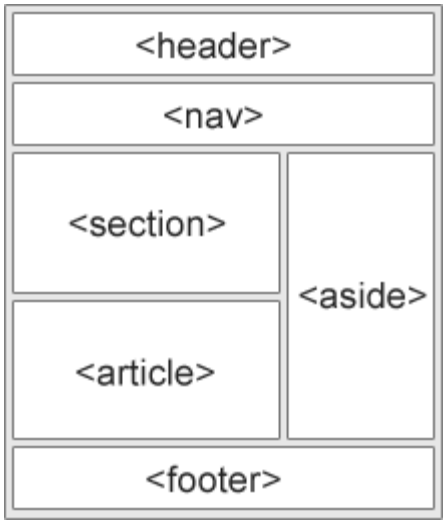
Read about it in HTML5 Browser Support.

New Semantic Elements in HTML5

Many web sites contain HTML code like:< div id="nav">   to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A Web site's home page could be split into sections for introduction, content, and contact information.

Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

Try it Yourself »

HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural
environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Try it Yourself »

Nesting Semantic Elements

In the HTML5 standard, the <article> element defines a complete, self-contained block of related elements.

The <section> element is defined as a block of related elements.

Can we use the definitions to decide how to nest elements? No, we cannot!

On the Internet, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <sections> elements.

You will also find pages with <section> elements containing <section> elements, and <article> elements containing <article> elements.

Newspaper: The sports **articles** in the sports **section**, have a technical **section** in each **article**.

HTML5 <header> Element

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document.

The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural
environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Try it Yourself »

HTML5 <footer> Element

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You can have several <footer> elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

Try it Yourself »

HTML5 <nav> Element

The <nav> element defines a set of navigation links.

The <nav> element is intended for large blocks of navigation links. However, not all links in a document should be inside a <nav> element!

Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

Try it Yourself »

HTML5 <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The aside content should be related to the surrounding content.

Example

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
```

```
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

Try it Yourself »

HTML5 <figure> and <figcaption> Elements

In books and newspapers, it is common to have captions with images.

The purpose of a caption is to add a visual explanation to an image.

With HTML5, images and captions can be grouped together in **<figure>** elements:

Example

```
<figure>
  <img src="pic_mountain.jpg" alt="The Pulpit Rock" width="304" height="228">
  <figcaption>Fig1. - The Pulpit Rock, Norway.</figcaption>
</figure>
```

Try it Yourself »

The **<img>** element defines the image, the **<figcaption>** element defines the caption.

Why Semantic HTML5 Elements?

With HTML4, developers used their own favorite attribute names to style page elements:

header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, ...

This made it impossible for search engines to identify the correct web page content.

With HTML5 elements like: <header> <footer> <nav> <section> <article>, this will become easier.

According to the W3C, a Semantic Web:

"Allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

The links goes to our complete HTML5 Reference.

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

« Previous

Next Chapter »

HTML5 Migration

« Previous

Next Chapter »

Migration from HTML4 to HTML5

This chapter is entirely about how to **migrate** from a typical **HTML4** page to a typical **HTML5** page.

This chapter demonstrates how to convert an existing HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

You can migrate to HTML5 from HTML4, and XHTML, using the same recipe.

Typical HTML4	Typical HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

A Typical HTML4 Page

Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<title>HTML4</title>
<style>
  body {font-family:Verdana,sans-serif;font-size:0.8em;}
  div#header,div#footer,div#content,div#post
  {border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-
color:white;}
  div#header,div#footer {color:white;background-color:#444;margin-bottom:5px;}
  div#content {background-color:#ddd;}
  div#menu ul {margin:0;padding:0;}
```

```
  div#menu ul li {display:inline; margin:5px;}
</style>
</head>
<body>

<div id="header">
  <h1>Monday Times</h1>
</div>

<div id="menu">
  <ul>
    <li>News</li>
    <li>Sports</li>
    <li>Weather</li>
  </ul>
</div>

<div id="content">
<h2>News Section</h2>

<div id="post">
  <h2>News Article</h2>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
</div>

<div id="post">
  <h2>News Article</h2>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
</div>

</div>

<div id="footer">
  <p>&amp;copy; 2014 Monday Times. All rights reserved.</p>
</div>
```

```
</body>
</html>
```

Try it Yourself »

Change to HTML5 Doctype

Change the **doctype**, from the HTML4 doctype:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

to the HTML5 doctype:

```
<!DOCTYPE html>
```

Try it Yourself »

Change to HTML5 Encoding

Change the **encoding** information, from HTML4:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

to HTML5:

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

Try it Yourself »

Add The Shiv

HTML5 semantic elements are supported in all modern browsers.

In addition, you can "teach" older browsers how to handle "unknown elements".

Add **the shiv** for Internet Explorer support:

```
<!--[if lt IE 9]>
  <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Try it Yourself »

Read about **the shiv** in HTML5 Browser Support.

Add CSS for HTML5 Semantic Elements

Look at your existing CSS styles:

```
div#header,div#footer,div#content,div#post {
    border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-color:white;
}
div#header,div#footer {
    color:white;background-color:#444;margin-bottom:5px;
}
div#content {
    background-color:#ddd;
}
div#menu ul {
    margin:0;padding:0;
}
div#menu ul li {
    display:inline; margin:5px;
}
```

Duplicate with equal CSS styles for HTML5 semantic elements:

```
header,footer,section,article {
    border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-color:white;
}
header,footer {
    color:white;background-color:#444;margin-bottom:5px;
}
section {
    background-color:#ddd;
}
nav ul  {
```

```
    margin:0;padding:0;
}
nav ul li {
    display:inline; margin:5px;
}
```

Try it Yourself »

Change to HTML5 <header> and <footer>

Change the elements with **id="header"** and **id="footer"**:

```
<div id="header">
  <h1>Monday Times</h1>
</div>
.
.
.
<div id="footer">
  <p>&amp;copy; 2014 W3Schools. All rights reserved.</p>
</div>
```

to HTML5 semantic **<header>** and **<footer>** elements:

```
<header>
  <h1>Monday Times</h1>
</header>
.
.
.
<footer>
  <p>&copy; 2014 W3Schools. All rights reserved.</p>
</footer>
```

Try it Yourself »

Change to HTML5 <nav>

Change the element with **id="menu"**:

```
<div id="menu">
  <ul>
    <li>News</li>
    <li>Sports</li>
    <li>Weather</li>
  </ul>
</div>
```

to an HTML5 semantic **<nav>** element:

```
<nav>
  <ul>
    <li>News</li>
    <li>Sports</li>
    <li>Weather</li>
  </ul>
</nav>
```

Try it Yourself »

Change to HTML5 <section>

Change the element with **id="content"**:

```
<div id="content">
.
.
.
</div>
```

to an HTML5 semantic **<section>** element:

```
<section>
.
.
.
</section>
```

Try it Yourself »

Change to HTML5 <article>

Change all element with **class="post"**:

```
<div class="post">
  <h2>News Article</h2>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
</div>
```

to HTML5 semantic **<article>** elements:

```
<article>
  <h2>News Article</h2>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
</article>
```

Try it Yourself »

Remove these "no longer needed" <style> elements:

```
div#header,div#footer,div#content,div#post {
    border:1px solid grey;margin:5px;margin-bottom:15px;padding:8px;background-
color:white;
}
div#header,div#footer {
    color:white;background-color:#444;margin-bottom:5px;
}
div#content {
    background-color:#ddd;
}
div#menu ul {
    margin:0;padding:0;
}
div#menu ul li {
    display:inline; margin:5px;
}
```

Try it Yourself »

A Typical HTML5 Page

Finally you can remove the <head> tags. They are not needed in HTML5:

Example

```
<!DOCTYPE html>
<html lang="en">
<title>HTML5</title>
<meta charset="utf-8">

<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->

<style>
body {
    font-family:Verdana,sans-serif;font-size:0.8em;
}
header,footer,section,article {
    border:1px solid grey;
    margin:5px;margin-bottom:15px;padding:8px;
    background-color:white;
}
header,footer {
    color:white;background-color:#444;margin-bottom:5px;
}
section {
    background-color:#ddd;
}
nav ul {
    margin:0;padding:0;
}
nav ul li {
    display:inline; margin:5px;
}
</style>
<body>

<header>
  <h1>Monday Times</h1>
</header>

<nav>
  <ul>
    <li>News</li>
    <li>Sports</li>
    <li>Weather</li>
  </ul>
</nav>
```

```
<section>
<h2>News Section</h2>

<article>
  <h2>News Article</h2>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
</article>

<article>
<h2>News Article</h2>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
  <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum turum
ipsum
  lurum hurum turum.</p>
</article>

</section>

<footer>
  <p>&amp;copy; 2014 Monday Times. All rights reserved.</p>
</footer>

</body>
</html>
```

Try it Yourself »

The Difference Between <article> <section> and

There is a confusing (lack of) difference in the HTML5 standard, between <article> <section> and .

In the HTML5 standard, the <section> element is defined as a block of related elements.

The <article> element is defined as a complete, self-contained block of related elements.

The element is defined as a block of children elements.

How to interpret that?

In the example above, we have used <section> as a container for related <articles>.

But, we could have used <article> as a container for articles as well.

Here are some different examples:

<article> in <article>:

```
<article>

<h2>Famous Cities</h2>

<article>
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</article>

<article>
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</article>

<article>
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</article>

</article>
```

Try it Yourself »

in <article>:

```
<article>

<h2>Famous Cities</h2>

<div class="city">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="city">
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="city">
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>

</article>
```

Try it Yourself »

in <section> in <article>:

```
<article>

<section>
<h2>Famous Cities</h2>

<div class="city">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="city">
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="city">
```

```
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>
</section>

<section>
<h2>Famous Countries</h2>

<div class="country">
<h2>England</h2>
<p>London is the capital city of England. It is the most populous city in the
United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="country">
<h2>France</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="country">
<h2>Japan</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>
</section>

</article>
```

Try it Yourself »

« Previous

Next Chapter »

HTML(5) Style Guide and Coding Conventions

« Previous

Next Chapter »

HTML Coding Conventions

Web developers are often uncertain about the coding style and syntax to use in HTML.

Between 2000 and 2010, many web developers converted from HTML to XHTML.

With XHTML, developers were forced to write valid and "well-formed" code.

HTML5 is a bit more sloppy when it comes to code validation.

With HTML5, you must create your own **Best Practice, Style Guide and Coding Conventions**.

Be Smart and Future Proof

A consequent use of style, makes it easier for others to understand and use your HTML.

In the future, programs like XML readers, may want to read your HTML.

Using a well-formed "close to XHTML" syntax, can be smart.

Always keep your style smart, tidy, clean, and well-formed.

Use Correct Document Type

Always declare the document type as the first line in your document:

```
<!DOCTYPE html>
```

If you want consistency with lower case tags, you can use:

```
<!doctype html>
```

Use Lower Case Element Names

HTML5 allows mixing uppercase and lowercase letters in element names.

We recommend using lowercase element names:

- Mixing uppercase and lowercase names is bad
- Developers are used to use lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write

Bad:

```
<SECTION>
  <p>This is a paragraph.</p>
</SECTION>
```

Very Bad:

```
<Section>
  <p>This is a paragraph.</p>
</SECTION>
```

Good:

```
<section>
  <p>This is a paragraph.</p>
</section>
```

Close All HTML Elements

In HTML5, you don't have to close all elements (for example the <p> element).

We recommend closing all HTML elements:

Looking bad:

```
<section>
  <p>This is a paragraph.
  <p>This is a paragraph.
</section>
```

Looking good:

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```

Close Empty HTML Elements

In HTML5, it is optional to close empty elements.

This is allowed:

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

This is also allowed:

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

The slash (/) is required in XHTML and XML.

If you expect XML software to access your page, it might be a good idea to keep it.

Use Lower Case Attribute Names

HTML5 allows mixing uppercase and lowercase letters in attribute names.

We recommend using lowercase attribute names:

- Mixing uppercase and lowercase names is bad
- Developers are used to use lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write

Looking bad:

```
<div CLASS="menu">
```

Looking good:

```
<div class="menu">
```

Quote Attribute Values

HTML5 allows attribute values without quotes.

We recommend quoting attribute values:

- You have to use quotes if the value contains spaces
- Mixing styles is never good
- Quoted values are easier to read

This will not work, because the value contains spaces:

```
<table class=table striped>
```

This will work:

```
<table class="table striped">
```

Image Attributes

Always use the **alt** attribute with images. It is important when the image cannot be viewed.

```
<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">
```

Always define image size. It reduces flickering because the browser can reserve space for images before they are loaded.

```
<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">
```

Spaces and Equal Signs

Spaces around equal signs is legal:

```
<link rel = "stylesheet" href = "styles.css">
```

But space-less is easier to read, and groups entities better together:

```
<link rel="stylesheet" href="styles.css">
```

Avoid Long Code Lines

When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code.

Try to avoid code lines longer than 80 characters.

Blank Lines and Indentation

Do not add blank lines without a reason.

For readability, add blank lines to separate large or logical code blocks.

For readability, add 2 spaces of indentation. Do not use TAB.

Do not use unnecessary blank lines and indentation. It is not necessary to use blank lines between short and related items. It is not necessary to indent every element:

Unnecessary:

```
<body>

  <h1>Famous Cities</h1>

  <h2>Tokyo</h2>

  <p>
    Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.
    It is the seat of the Japanese government and the Imperial Palace,
    and the home of the Japanese Imperial Family.
  </p>

</body>
```

Better:

```
<body>

<h1>Famous Cities</h1>
<h2>Tokyo</h2>

<p>
Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.
It is the seat of the Japanese government and the Imperial Palace,
and the home of the Japanese Imperial Family.
</p>

</body>
```

Table Example:

```
<table>
  <tr>
    <th>Name</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```

List Example:

```
<ol>
  <li>London</li>
  <li>Paris</li>
  <li>Tokyo</li>
</ol>
```

Omitting <html> and <body>?

In the HTML5 standard, the <html> tag and the <body> tag can be omitted.

The following code will validate as HTML5:

Example

```
<!DOCTYPE html>
<head>
  <title>Page Title</title>
</head>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

Try it Yourself »

We do not recommend omitting the <html> and <body> tags.

The <html> element is the document root. It is the recommended place for specifying the page language:

```
<!DOCTYPE html>
<html lang="en-US">
```

Declaring a language is important for accessibility applications (screen readers) and search engines.

Omitting <html> or <body> can crash DOM and XML software.

Omitting <body> can produce errors in older browsers (IE9).

Omitting <head>?

In the HTML5 standard, the <head> tag can also be omitted.

By default, browsers will add all elements before <body>, to a default <head> element.

You can reduce the complexity of HTML, by omitting the <head> tag:

Example

```
<!DOCTYPE html>
<html>
<title>Page Title</title>

<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>

</html>
```

Try it Yourself »

Omitting tags is unfamiliar to web developers. It needs time to be established as a guideline.

Meta Data

The <title> element is required in HTML5. Make the title as meaningful as possible:

```
<title>HTML5 Syntax and Coding Style</title>
```

To ensure proper interpretation, and correct search engine indexing, both the language and the character encoding should be defined as early as possible in a document:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Syntax and Coding Style</title>
</head>
```

HTML Comments

Short comments should be written on one line, with a space after <!-- and a space before -->:

```
<!-- This is a comment -->
```

Long comments, spanning many lines, should be written with <!-- and --> on separate lines:

```
<!--
  This is a long comment example. This is a long comment example. This is a long
comment example.
  This is a long comment example. This is a long comment example. This is a long
comment example.
-->
```

Long comments are easier to observe, if they are indented 2 spaces.

Style Sheets

Use simple syntax for linking style sheets (the type attribute is not necessary):

```
<link rel="stylesheet" href="styles.css">
```

Short rules can be written compressed, on one line, like this:

```
p.into {font-family:"Verdana"; font-size:16em;}
```

Long rules should be written over multiple lines:

```
body {
  background-color: lightgrey;
  font-family: "Arial Black", Helvetica, sans-serif;
  font-size: 16em;
  color: black;
}
```

- Place the opening bracket on the same line as the selector.
- Use one space before the opening bracket.
- Use 2 spaces of indentation.
- Use colon plus one space between each property and its value.
- Use space after each comma or semicolon.
- Use semicolon after each property-value pair, including the last.
- Only use quotes around values if the value contains spaces.
- Place the closing bracket on a new line, without leading spaces.
- Avoid lines over 80 characters.

Adding a space after a comma, or a semicolon, is a general rule in all types of writing.

Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js">
```

Accessing HTML Elements with JavaScript

A consequence of using "untidy" HTML styles, might result in JavaScript errors.

These two JavaScript statements will produce different results:

```
var obj = getElementById("Demo")

var obj = getElementById("demo")
```

Try it Yourself »

If possible, use the same naming convention (as JavaScript) in HTML.

Visit the JavaScript Style Guide.

Use Lower Case File Names

Most web servers (Apache, Unix) are case sensitive about file names:

london.jpg cannot be accessed as London.jpg.

Other web servers (Microsoft, IIS) are not case sensitive:

london.jpg can be accessed as London.jpg or london.jpg.

If you use a mix of upper and lower case, you have to be extremely consistent.

If you move from a case insensitive, to a case sensitive server, even small errors will break your web.

To avoid these problems, always use lower case file names (if possible).

File Extensions

HTML files should have a **.html** extension (not **.htm**).

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

« Previous

Next Chapter »

HTML5 Canvas

« Previous

Next Chapter »

Your browser does not support the <canvas> element.

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

What is HTML Canvas?

The HTML <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).

The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Browser Support

The numbers in the table specify the first browser version that fully supports the < canvas> element.

Element					
<canvas>	4.0	9.0	2.0	3.1	9.0

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas.

To add a border, use the style attribute:

Basic Canvas Example

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

Try it Yourself »

Drawing with JavaScript

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
```

Try it Yourself »

Draw a Line

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```

Try it Yourself »

Draw a Circle

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
```

```
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```

Try it Yourself »

Draw a Text

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
```

Try it Yourself »

Stroke Text

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
```

Try it Yourself »

Draw Linear Gradient

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```

Try it Yourself »

Draw Circular Gradient

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
```

```
// Create gradient
var grd = ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```

Try it Yourself »

Draw Image

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img,10,10);
```

Try it Yourself »

HTML Canvas Tutorial

To learn all about HTML <canvas>, Visit our full HTML Canvas Tutorial.

« Previous

Next Chapter »

HTML5 SVG

« Previous

Next Chapter »

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

The HTML <svg> Element

The HTML <svg> element (introduced in HTML5) is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Browser Support

The numbers in the table specify the first browser version that fully supports the < svg> element.

Element					
<svg>	4.0	9.0	3.0	3.2	10.1

SVG Circle

Example

```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
```

```
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

Try it Yourself »

SVG Rectangle

Example

```
<svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-
width:10;stroke:rgb(0,0,0)" />
</svg>
```

Try it Yourself »

SVG Rounded Rectangle

Sorry, your browser does not support inline SVG.

Example

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>
```

Try it Yourself »

SVG Star

Sorry, your browser does not support inline SVG.

Example

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

Try it Yourself »

SVG Logo

SVG Sorry, your browser does not support inline SVG.

Example

```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50"
y="86">SVG</text>
  Sorry, your browser does not support inline SVG.
</svg>
```

Try it Yourself »

Differences Between SVG and Canvas

SVG is a language for describing 2D graphics in XML.

Canvas draws 2D graphics, on the fly (with a JavaScript).

SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.

In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG:

Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

To learn more about SVG, please read our SVG Tutorial.

« Previous

Next Chapter »

HTML Multimedia

« Previous

Next Chapter »

Multimedia on the web, is sound, music, videos, movies, and animations.

What is Multimedia?

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Pictures, music, sound, videos, records, films, animations, and more.

Web pages often contains multimedia elements of different types and formats.

In this chapter you will learn about the different multimedia formats.

Browser Support

The first web browsers had support for text only, limited to a single font in a single color.

Later came browsers with support for colors and fonts, and even support for pictures!

The support for sounds, animations, and videos is handled differently by various browsers. Different types and formats are supported, and some formats requires extra helper programs (plug-ins) to work.

Hopefully this will become history. HTML5 multimedia promises an easier future for multimedia.

Multimedia Formats

Multimedia elements (like sounds or videos) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension. When a browser sees the file extension .htm or .html, it will treat the file as an HTML file. The .xml extension indicates an XML file, and the .css extension indicates a style sheet file. Pictures are recognized by extensions like .gif, .png and .jpg.

Multimedia files also have their own formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Common Video Formats

MP4 is the new and upcoming format for internet video.

MP4 is recommended by YouTube.

MP4 is supported by Flash Players

MP4 is supported by HTML5.

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5 (See MP4).
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers. (See MP4)
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
WebM	.webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube.

Only MP4, WebM, and Ogg video is supported by the newest HTML5 standard.

Sound Formats

MP3 is the newest format for compressed recorded music. The term MP3 has become synonymous with digital music.

If your website is about recorded music, MP3 is the choice.

Format	File	Description

MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Commonly used in music players. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML5.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. MP4 video is the upcoming video format on the internet. This leads to automatic support for MP4 audio by all browsers.

Only MP3, WAV, and Ogg audio is supported by the newest HTML5 standard.

« Previous

Next Chapter »

HTML5 Video

« Previous

Next Chapter »

HTML Video Example. Courtesy of Big Buck Bunny.

Your browser does not support HTML5 video.

Try it yourself »

Playing Videos in HTML

Before HTML5, there was no standard for showing videos on a web page.

Before HTML5, videos could only be played with a plug-in (like flash).

The HTML5 <video> element specifies a standard way to embed a video in a web page.

Browser Support

The numbers in the table specify the first browser version that fully supports the < video> element.

Element					
<video>	4.0	9.0	3.5	4.0	10.5

The HTML <video> Element

To show a video in HTML, use the **<video>** element:

Example

```
<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>
```

Try it yourself »

How it Works

The **controls** attribute adds video controls, like play, pause, and volume.

It is a good idea to always include **width** and **height** attributes.

If height and width are not set, the browser does not know the size of the video. The effect will be that the page will change (or flicker) while the video loads.

Text between the <video> and </video> tags will only display in browsers that do not support the <video> element.

Multiple **<source>** elements can link to different video files. The browser will use the first recognized format.

HTML <video> Autoplay

To start a video automatically use the **autoplay** attribute:

Example

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

Try it yourself »

The autoplay attribute does not work in Safari and Opera, or in mobile devices like iPad and iPhone.

HTML Video - Browser Support

Currently, there are 3 supported video formats for the <video> element: MP4, WebM, and Ogg:

Browser	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	NO	NO
Opera	YES (from Opera 25)	YES	YES

HTML Video - Media Types

File Format	Media Type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

HTML Video - Methods, Properties, and Events

HTML5 defines DOM methods, properties, and events for the <video> element.

This allows you to load, play, and pause videos, as well as setting duration and volume.

There are also DOM events that can notify you when a video begins to play, is paused, etc.

Example: Using JavaScript

Play/Pause Big Small Normal

Your browser does not support HTML5 video.

Video courtesy of Big Buck Bunny.

Try it yourself »

For a full DOM reference, go to our HTML5 Audio/Video DOM Reference.

HTML5 Video Tags

Tag	Description
<video>	Defines a video or movie
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	Defines text tracks in media players

« Previous

Next Chapter »

HTML5 Audio

« Previous

Next Chapter »

HTML5 provides a standard for playing audio files.

Audio on the Web

Before HTML5, there was no standard for playing audio files on a web page.

Before HTML5, audio files could only be played with a plug-in (like flash).

The HTML5 <audio> element specifies a standard way to embed audio in a web page.

Browser Support

The numbers in the table specify the first browser version that fully supports the < audio> element.

Element					
<audio>	4.0	9.0	3.5	4.0	10.5

The HTML <audio> Element

To play an audio file in HTML, use the **<audio>** element:

Example

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Try it yourself »

HTML Audio - How It Works

The **controls** attribute adds audio controls, like play, pause, and volume.

Text between the <audio> and </audio> tags will display in browsers that do not support the <audio> element.

Multiple **<source>** elements can link to different audio files. The browser will use the first recognized format.

HTML Audio - Browser Support

Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg:

Browser	MP3	Wav	Ogg

Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

HTML Audio - Media Types

File Format	Media Type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

HTML Audio - Methods, Properties, and Events

HTML5 defines DOM methods, properties, and events for the <audio> element.

This allows you to load, play, and pause audios, as well as setting duration and volume.

There are also DOM events that can notify you when an audio begins to play, is paused, etc.

For a full DOM reference, go to our HTML5 Audio/Video DOM Reference.

HTML5 Audio Tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and < audio>

« Previous

Next Chapter »

HTML Plug-ins

« Previous

Next Chapter »

The purpose of a plug-in, is to extend the functionality of the HTML browser.

HTML Helpers (Plug-ins)

Helper applications are computer programs that extend the standard functionality of a web browser.

Helper applications are also called plug-ins.

Examples of well-known plug-ins are Java applets.

Plug-ins can be added to web pages with the <object> tag or the <embed> tag.

Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

To display video and audio: Use the <video> and <audio> tags.

The <object> Element

The <object> element is supported by all browsers.

The <object> element defines an embedded object within an HTML document.

It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages.

Example

```
<object width="400" height="50" data="bookmark.swf"></object>
```

Try it Yourself »

The <object> element can also be used to include HTML in HTML:

Example

```
<object width="100%" height="500px" data="snippet.html"></object>
```

Try it Yourself »

Or images if you like:

Example

```
<object data="audi.jpeg"></object>
```

Try it Yourself »

The <embed> Element

The <embed> element is supported in all major browsers.

The <embed> element also defines an embedded object within an HTML document.

Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5. The element will validate in an HTML5 page, but not in an HTML 4 page.

Example

```
<embed width="400" height="50" src="bookmark.swf">
```

Try it Yourself »

Note that the <embed> element does not have a closing tag. It can not contain alternative text.

The <embed> element can also be used to include HTML in HTML:

Example

```
<embed width="100%" height="500px" src="snippet.html">
```

Try it Yourself »

Or images if you like:

Example

```
<embed src="audi.jpeg">
```

Try it Yourself »

« Previous

Next Chapter »

HTML YouTube Videos

« Previous

Next Chapter »

The easiest way to play videos in HTML, is to use YouTube.

Struggling with Video Formats?

Different versions of different browsers support different video formats.

Earlier in this tutorial, you have seen that you might have to convert your videos to different video formats to make them play in all browsers.

Converting videos to different format can be difficult and time consuming.

An easier solution might be to let YouTube play the videos in your web page.

YouTube Video Id

YouTube will display an id (like XGSy3_Czz8k), when you save (or play) a video.

You can use this id, and refer to your video in HTML.

Playing a YouTube Video in HTML

To play your video on a web page, do the following:

- Upload the video to YouTube
- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and hight attributes to specify the dimension of the player
- Add any other parameters to the URL

Example - Using iFrame (the recommended method)

```
<iframe width="420" height="315"
src="http://www.youtube.com/embed/XGSy3_Czz8k?autoplay=1">
</iframe>
```

Try it Yourself »

YouTube Parameters

autohide

Value 0: The player controls are always visible.

Value 1: The player controls hides automatically when the video plays.

Value 2 (default): If the player has 16:9 or 4:3 ratio, same as 1, otherwise same as 0.

autoplay

Value 0 (default): The video will not play automatically when the player loads.

Value 1: The video will play automatically when the player loads.

controls

Value 0: Player controls does not display. The video loads immediately.

Value 1 (default): Player controls display. The video loads immediately.

Value 2: Player controls display, but the video does not load before the user initiates playback.

loop

Value 0 (default): The video will play only once.

Value 1: The video will loop (forever).

playlist

A comma separated list of videos to play (in addition to the original URL).

YouTube <object> Embeds

YouTube <object> embeds were deprecated from January 2015.

You should migrate your applications to use <iframe> embeds.

Using <object> (deprecated)

```
<object width="420" height="315"
data="http://www.youtube.com/embed/XGSy3_Czz8k">
</object>
```

Try it Yourself »

Using <embed> (deprecated)

```
<embed width="420" height="315"
src="http://www.youtube.com/embed/XGSy3_Czz8k">
```

Try it Yourself »

« Previous

Next Chapter »

HTML5 Geolocation

« Previous

Next Chapter »

HTML Geolocation is used to locate a user's position.

Try It

Locate the User's Position

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise user privacy, the position is not available unless the user approves it.

Browser Support

The numbers in the table specify the first browser version that fully supports Geolocation.

API					
Geolocation	5.0	9.0	3.5	5.0	16.0

Note: Geolocation is much more accurate for devices with GPS, like iPhone.

Using HTML Geolocation

Use the getCurrentPosition() method to get the user's position.

The example below is a simple Geolocation example returning the latitude and longitude of the user's position:

Example

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Try it yourself »

Example explained:

- Check if Geolocation is supported
- If supported, run the getCurrentPosition() method. If not, display a message to the user
- If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)
- The showPosition() function gets the displays the Latitude and Longitude

The example above is a very basic Geolocation script, with no error handling.

Handling Errors and Rejections

The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

Example

```
function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            x.innerHTML = "User denied the request for Geolocation."
            break;
        case error.POSITION_UNAVAILABLE:
            x.innerHTML = "Location information is unavailable."
            break;
        case error.TIMEOUT:
            x.innerHTML = "The request to get user location timed out."
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML = "An unknown error occurred."
            break;
    }
}
```

Try it yourself »

Error Codes:

- Permission denied - The user did not allow Geolocation
- Position unavailable - It is not possible to get the current location
- Timeout - The operation timed out

Displaying the Result in a Map

To display the result in a map, you need access to a map service that can use latitude and longitude, like Google Maps:

Example

```
function showPosition(position) {
    var latlon = position.coords.latitude + "," + position.coords.longitude;

    var img_url = "http://maps.googleapis.com/maps/api/staticmap?center=
    "+latlon+"&zoom=14&size=400x300&sensor=false";

    document.getElementById("mapholder").innerHTML = "<img src='"+img_url+"'>";
}
```

Try it yourself »

In the example above we use the returned latitude and longitude data to show the location in a Google map (using a static image).

Google Map Script
How to use a script to show an interactive map with a marker, zoom and drag options.

Location-specific Information

This page demonstrated how to show a user's position on a map. However, Geolocation is also very useful for location-specific information.

Examples:

- Up-to-date local information
- Showing Points-of-interest near the user
- Turn-by-turn navigation (GPS)

The getCurrentPosition() Method - Return Data

The getCurrentPosition() method returns an object if it is successful. The latitude, longitude and accuracy properties are always returned. The other properties below are returned if available.

Property	Description

coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

Geolocation object - Other interesting Methods

watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).

clearWatch() - Stops the watchPosition() method.

The example below shows the watchPosition() method. You need an accurate GPS device to test this (like iPhone):

Example

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.watchPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Try it yourself »

« Previous

Next Chapter »

HTML5 Drag and Drop

« Previous

Next Chapter »

Drag and drop is a part of the HTML5 standard.

Drag the W3Schools image into the rectangle.

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard, and any element can be draggable.

Browser Support

The numbers in the table specify the first browser version that fully supports Drag and Drop.

API

Drag and Drop	4.0	9.0	3.5	6.0	12.0

HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

Try it yourself »

It might seem complicated, but lets go through all the different parts of a drag and drop event.

Make an Element Draggable

First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

What to Drag - ondragstart and setData()

Then, specify what should happen when the element is dragged.

In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}
```

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

Where to Drop - ondragover

The ondragover event specifies where the dragged data can be dropped.

By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

This is done by calling the event.preventDefault() method for the ondragover event:

```
event.preventDefault()
```

Do the Drop - ondrop

When the dragged data is dropped, a drop event occurs.

In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
```

Code explained:

- Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)
- Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method
- The dragged data is the id of the dragged element ("drag1")
- Append the dragged element into the drop element

More Examples

Drag image back and forth
How to drag (and drop) an image back and forth between two elements.

« Previous

Next Chapter »

HTML5 Local Storage

« Previous

Next Chapter »

HTML local storage, better than cookies.

What is HTML Local Storage?

With local storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Local storage is per domain. All pages, from one domain, can store and access the same data.

Browser Support

The numbers in the table specify the first browser version that fully supports Local Storage.

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

HTML Local Storage Objects

HTML local storage provides two objects for storing data on the client:

- window.localStorage - stores data with no expiration date
- window.sessionStorage - stores data for one session (data is lost when the tab is closed)

Before using local storage, check browser support for localStorage and sessionStorage:

```
if(typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Example

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

Try it Yourself »

Example explained:

- Create a localStorage name/value pair with name="lastname" and value="Smith"
- Retrieve the value of "lastname" and insert it into the element with id="result"

The example above could also be written like this:

```
// Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
```

The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

Note: Name/value pairs are always stored as strings. Remember to convert them to another format when needed!

The following example counts the number of times a user has clicked a button. In this code the value string is converted to a number to be able to increase the counter:

Example

```
if (localStorage.clickcount) {
    localStorage.clickcount = Number(localStorage.clickcount) + 1;
} else {
    localStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You have clicked the button " +
localStorage.clickcount + " time(s).";
```

Try it Yourself »

The sessionStorage Object

The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the browser window.

The following example counts the number of times a user has clicked a button, in the current session:

Example

```
if (sessionStorage.clickcount) {
    sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
    sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
```

Try it Yourself »

« Previous

Next Chapter »

HTML5 Application Cache

« Previous

Next Chapter »

With application cache it is easy to make an offline version of a web application, by creating a cache manifest file.

What is Application Cache?

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

Application cache gives an application three advantages:

1. Offline browsing - users can use the application when they're offline
2. Speed - cached resources load faster
3. Reduced server load - the browser will only download updated/changed resources from the server

---

Browser Support

The numbers in the table specify the first browser version that fully supports Application Cache.

API

Application Cache	4.0	10.0	3.5	4.0	11.5

HTML Cache Manifest Example

The example below shows an HTML document with a cache manifest (for offline browsing):

Example

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">

<body>
The content of the document......
</body>

</html>
```

Try it Yourself »

Cache Manifest Basics

To enable application cache, include the manifest attribute in the document's <html> tag:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Every page with the manifest attribute specified will be cached when the user visits it. If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).

The recommended file extension for manifest files is: ".appcache"

A manifest file needs to be served with the **correct media type**, which is "text/cache-manifest". Must be configured on the web server.

The Manifest File

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

CACHE MANIFEST

The first line, CACHE MANIFEST, is required:

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

The manifest file above lists three resources: a CSS file, a GIF image, and a JavaScript file. When the manifest file is loaded, the browser will download the three files from the root directory of the web site. Then, whenever the user is not connected to the internet, the resources will still be available.

NETWORK

The NETWORK section below specifies that the file "login.asp" should never be cached, and will not be available offline:

```
NETWORK:
login.asp
```

An asterisk can be used to indicate that all other resources/files require an internet connection:

```
NETWORK:
*
```

FALLBACK

The FALLBACK section below specifies that "offline.html" will be served in place of all files in the /html/ catalog, in case an internet connection cannot be established:

```
FALLBACK:
/html/ /offline.html
```

Note: The first URI is the resource, the second is the fallback.

Updating the Cache

Once an application is cached, it remains cached until one of the following happens:

- The user clears the browser's cache
- The manifest file is modified (see tip below)

- The application cache is programmatically updated

Example - Complete Cache Manifest File

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html
```

Tip: Lines starting with a "#" are comment lines, but can also serve another purpose. An application's cache is only updated when its manifest file changes. If you edit an image or change a JavaScript function, those changes will not be re-cached. Updating the date and version in a comment line is one way to make the browser re-cache your files.

Notes on Application Cache

Be careful with what you cache.

Once a file is cached, the browser will continue to show the cached version, even if you change the file on the server. To ensure the browser updates the cache, you need to change the manifest file.

Note: Browsers may have different size limits for cached data (some browsers have a 5MB limit per site).

« Previous

Next Chapter »

HTML5 Web Workers

« Previous

Next Chapter »

A web worker is a JavaScript running in the background, without affecting the performance of the page.

What is a Web Worker?

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Browser Support

The numbers in the table specify the first browser version that fully support Web Workers.

API					
Web Workers	4.0	10.0	3.5	4.0	11.5

HTML Web Workers Example

The example below creates a simple web worker that count numbers in the background:

Example

Count numbers:

Start Worker Stop Worker

Try it yourself »

Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it:

```
if(typeof(Worker) !== "undefined") {
    // Yes! Web worker support!
    // Some code.....
} else {
    // Sorry! No Web Worker support..
}
```

Create a Web Worker File

Now, let's create our web worker in an external JavaScript.

Here, we create a script that counts. The script is stored in the "demo_workers.js" file:

```
var i = 0;

function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}

timedCount();
```

The important part of the code above is the **postMessage()** method - which is used to post a message back to the HTML page.

Note: Normally web workers are not used for such simple scripts, but for more CPU intensive tasks.

Create a Web Worker Object

Now that we have the web worker file, we need to call it from an HTML page.

The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo_workers.js":

```
if(typeof(w) == "undefined") {
    w = new Worker("demo_workers.js");
}
```

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

```
w.onmessage = function(event){
    document.getElementById("result").innerHTML = event.data;
};
```

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

Terminate a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.

To terminate a web worker, and free browser/computer resources, use the terminate() method:

```
w.terminate();
```

Reuse the Web Worker

If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

```
w = undefined;
```

Full Web Worker Example Code

We have already seen the Worker code in the .js file. Below is the code for the HTML page:

Example

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
```

```
<br><br>

<script>
var w;

function startWorker() {
    if(typeof(Worker) !== "undefined") {
        if(typeof(w) == "undefined") {
            w = new Worker("demo_workers.js");
        }
        w.onmessage = function(event) {
            document.getElementById("result").innerHTML = event.data;
        };
    } else {
        document.getElementById("result").innerHTML = "Sorry! No Web Worker
support.";
    }
}

function stopWorker() {
    w.terminate();
    w = undefined;
}
</script>

</body>
</html>
```

Try it yourself »

Web Workers and the DOM

Since web workers are in external files, they do not have access to the following JavaScript objects:

- The window object
- The document object
- The parent object

« Previous

Next Chapter »

HTML5 Server-Sent Events

« Previous

Next Chapter »

Server-Sent Events allow a web page to get updates from a server.

Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Browser Support

The numbers in the table specify the first browser version that fully support server-sent events.

API					
SSE	6.0	Not supported	6.0	5.0	11.5

Receive Server-Sent Event Notifications

The EventSource object is used to receive server-sent event notifications:

Example

```
var source = new EventSource("demo_sse.php");
source.onmessage = function(event) {
    document.getElementById("result").innerHTML += event.data + "<br>";
};
```

Try it yourself »

Example explained:

- Create a new EventSource object, and specify the URL of the page sending the updates (in this example "demo_sse.php")
- Each time an update is received, the onmessage event occurs
- When an onmessage event occurs, put the received data into the element with id="result"

Check Server-Sent Events Support

In the tryit example above there were some extra lines of code to check browser support for server-sent events:

```
if(typeof(EventSource) !== "undefined") {
    // Yes! Server-sent events support!
    // Some code.....
} else {
    // Sorry! No server-sent events support..
}
```

Server-Side Code Example

For the example above to work, you need a server capable of sending data updates (like PHP or ASP).

The server-side event stream syntax is simple. Set the "Content-Type" header to "text/event-stream". Now you can start sending event streams.

Code in PHP (demo_sse.php):

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

Code in ASP (VB) (demo_sse.asp):

```
<%
Response.ContentType = "text/event-stream"
```

```
Response.Expires = -1
Response.Write("data: The server time is: " & now())
Response.Flush()
%>
```

Code explained:

- Set the "Content-Type" header to "text/event-stream"
- Specify that the page should not cache
- Output the data to send (**Always** start with "data: ")
- Flush the output data back to the web page

The EventSource Object

In the examples above we used the onmessage event to get messages. But other events are also available:

Events	Description
onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs

« Previous

Next Chapter »