HTML Tutorial

HTML

HTML is the standard markup language for Web pages.

With HTML you can create your own Website.

HTML is easy to learn - You will enjoy it!

Easy Learning with HTML "Try it Yourself"

With our "Try it Yourself" editor, you can edit the HTML code and view the result:

Example

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

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

</body>
</html>
```

Click on the "Try it Yourself" button to see how it works.

HTML Examples

In this HTML tutorial, you will find more than 200 examples. With our online "Try it Yourself" editor, you can edit and test each example yourself!

Go to HTML Examples!

HTML Exercises

This HTML tutorial also contains nearly 100 HTML exercises.

HTML Quiz Test

Test your HTML skills with our HTML Quiz!

HTML References

At W3Schools you will find complete references about HTML elements, attributes, events, color names, entities, character-sets, URL encoding, language codes, HTTP messages, browser support, and more:

HTML Elements
Browser Support
Attributes
Global Attributes
Event Attributes
Color Names
Canvas
Audio/Video DOM
Character Sets
URL Encoding
Language Codes
Country Codes
HTTP Messages
Px to Em Converter

HTML Introduction

HTML is the standard markup language for creating Web pages.

What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

A Simple HTML Document

Example

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

What is an HTML Element?

An HTML element is defined by a start tag, some content, and an end tag:

```
<tagname>Content goes here...</tagname>
```

The HTML **element** is everything from the start tag to the end tag:

```
<h1>My First Heading</h1>

<p>My first paragraph.</p>
```

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

Note: Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

Web Browsers

The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them correctly.

A browser does not display the HTML tags, but uses them to determine how to display the document:

HTML Page Structure

Below is a visualization of an HTML page structure:

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Note: The content inside the <body> section (the white area above) will be displayed in a browser. The content inside the <title> element will be shown in the browser's title bar or in the page's tab.

HTML History

Since the early days of the World Wide Web, there have been many versions of HTML:

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017	W3C Recommendation: HTML5.2

This tutorial follows the latest HTML5 standard.

HTML Editors

A simple text editor is all you need to learn HTML.

Learn HTML Using Notepad or TextEdit

Web pages can be created and modified by using professional HTML editors.

However, for learning HTML we recommend a simple text editor like Notepad (PC) or TextEdit (Mac).

We believe in that using a simple text editor is a good way to learn HTML.

Follow the steps below to create your first web page with Notepad or TextEdit.

Step 1: Open Notepad (PC)

Windows 8 or later:

Open the **Start Screen** (the window symbol at the bottom left on your screen). Type **Notepad**.

Windows 7 or earlier:

Open **Start** > **Programs > Accessories > Notepad**

Step 1: Open TextEdit (Mac)

Open **Finder > Applications > TextEdit**

Also change some preferences to get the application to save files correctly. In **Preferences > Format >** choose **"Plain Text"**

Then under "Open and Save", check the box that says "Display HTML files as HTML code instead of formatted text".

Then open a new document to place the code.

Step 2: Write Some HTML

Write or copy the following HTML code into Notepad:

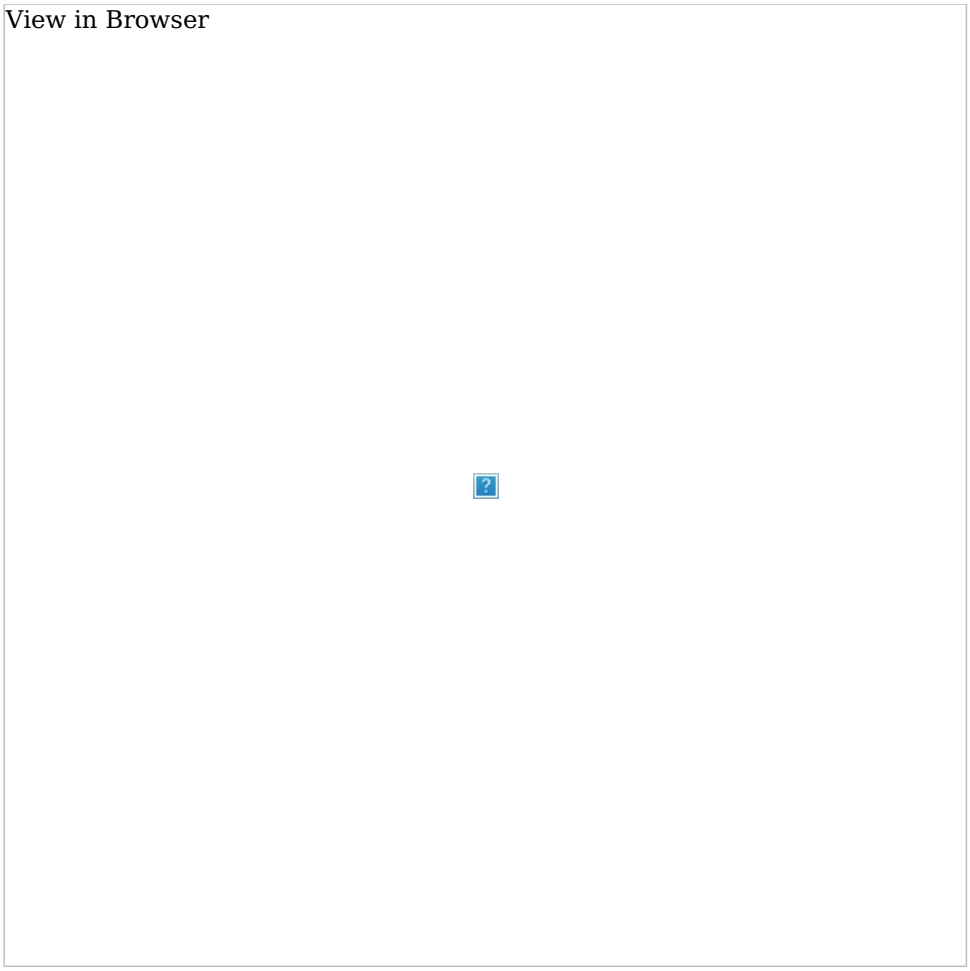
```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Step 3: Save the HTML Page

Save the file on your computer. Select **File > Save as** in the Notepad menu.

Name the file **"index.htm"** and set the encoding to **UTF-8** (which is the preferred encoding for HTML files).

View in Browser



Tip: You can use either .htm or .html as file extension. There is no difference, it is up to you.

Step 4: View the HTML Page in Your Browser

Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").

The result will look much like this:

W3Schools Online Editor - "Try it Yourself"

With our free online editor, you can edit the HTML code and view the result in your browser.

It is the perfect tool when you want to **test** code fast. It also has color coding and the ability to save and share code with others:

Example

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

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

</body>
</html>
```

Click on the "Try it Yourself" button to see how it works.

HTML Basic Examples

In this chapter we will show some basic HTML examples.

Don't worry if we use tags you have not learned about yet.

HTML Documents

All HTML documents must start with a document type declaration: `<!DOCTYPE html>`.

The HTML document itself begins with `<html>` and ends with `</html>`.

The visible part of the HTML document is between `<body>` and `</body>`.

Example

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

The <!DOCTYPE> Declaration

The `<!DOCTYPE>` declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The `<!DOCTYPE>` declaration is not case sensitive.

The `<!DOCTYPE>` declaration for HTML5 is:

```
<!DOCTYPE html>
```

HTML Headings

HTML headings are defined with the `<h1>` to `<h6>` tags.

`<h1>` defines the most important heading. `<h6>` defines the least important heading:Â

Example

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

HTML Paragraphs

HTML paragraphs are defined with the `<p>` tag:

Example

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

HTML Links

HTML links are defined with the `<a>` tag:

Example

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

The link's destination is specified in the `href` attribute.Â

Attributes are used to provide additional information about HTML elements.

You will learn more about attributes in a later chapter.

HTML Images

HTML images are defined with the `<img>` tag.

The source file (`src`), alternative text (`alt`), `width`, and `height` are provided as attributes:

Example

```
<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
```

How to View HTML Source?

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

View HTML Source Code:

Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in Edge), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element:

Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

HTML Elements

An HTML element is defined by a start tag, some content, and an end tag.

HTML Elements

The HTML **element** is everything from the start tag to the end tag:

```
<tagname>Content goes here...</tagname>
```

Examples of some HTML elements:

```
<h1>My First Heading</h1>

<p>My first paragraph.</p>
```

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

Note: Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

Nested HTML Elements

HTML elements can be nested (this means that elements can contain other elements).

All HTML documents consist of nested HTML elements.

The following example contains four HTML elements (<html>, <body>, <h1> and):

Example

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

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

The <html> element is the root element and it defines the whole HTML document.

It has a start tag <html> and an end tag </html>.

Then, inside the <html> element there is a <body> element:

```
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
```

The <body> element defines the document's body.

It has a start tag <body> and an end tag </body>.

Then, inside the <body> element there are two other elements: <h1> and :

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

The <h1> element defines a heading.

It has a start tag <h1> and an end tag </h1>:

```
<h1>My First Heading</h1>
```

The element defines a paragraph.

It has a start tag and an end tag :

```
<p>My first paragraph.</p>
```

Never Skip the End Tag

Some HTML elements will display correctly, even if you forget the end tag:

Example

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

However, never rely on this! Unexpected results and errors may occur if you forget the end tag!

Empty HTML Elements

HTML elements with no content are called empty elements.

The <br> tag defines a line break, and is an empty element without a closing tag:

Example

This is a <br> paragraph with a line break.

HTML is Not Case Sensitive

HTML tags are not case sensitive: <P> means the same as .

The HTML standard does not require lowercase tags, but W3C **recommends** lowercase in HTML, and **demands** lowercase for stricter document types like XHTML.

At W3Schools we always use lowercase tag names.

HTML Tag Reference

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

Tag	Description
<html>	Defines the root of an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Attributes

HTML attributes provide additional information about HTML elements.

HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The href Attribute

The <a> tag defines a hyperlink. The `href` attribute specifies the URL of the page the link goes to:

Example

Visit W3Schools

You will learn more about links in our HTML Links chapter.

The src Attribute

The <img> tag is used to embed an image in an HTML page. The `src` attribute specifies the path to the image to be displayed:

Example

<img src="img_girl.jpg">

There are two ways to specify the URL in the `src` attribute:

1. Absolute URL - Links to an external image that is hosted on another website. Example: src="https://www.w3schools.com/images/img_girl.jpg".

Notes: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

2. Relative URL - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page. Example: src="img_girl.jpg". If the URL begins

with a slash, it will be relative to the domain. Example: src="/images/img_girl.jpg".

Tip: It is almost always best to use relative URLs. They will not break if you change domain.

The width and height Attributes

The `<img>` tag should also contain the `width` and `height` attributes, which specifies the width and height of the image (in pixels):

Example

```
<img src="img_girl.jpg" width="500" height="600">
```

The alt Attribute

The required `alt` attribute for the `<img>` tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to slow connection, or an error in the `src` attribute, or if the user uses a screen reader.

Example

```
<img src="img_girl.jpg" alt="Girl with a jacket">
```

Example

See what happens if we try to display an image that does not exist:

```
<img src="img_typo.jpg" alt="Girl with a jacket">
```

You will learn more about images in our HTML Images chapter.

The style Attribute

The `style` attribute is used to add styles to an element, such as color, font, size, and more.

Example

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

You will learn more about styles in our HTML Styles chapter.

The lang Attribute

You should always include the `lang` attribute inside the `<html>` tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

The following example specifies English as the language:

```
<!DOCTYPE html>
<html lang="en">
<body>
...
</body>
</html>
```

Country codes can also be added to the language code in the `lang` attribute. So, the first two characters define the language of the HTML page, and the last two characters define the country.

The following example specifies English as the language and United States as the country:

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

You can see all the language codes in our HTML Language Code Reference.

The title Attribute

The `title` attribute defines some extra information about an element.

The value of the title attribute will be displayed as a tooltip when you mouse over the element:

Example

```
<p title="I'm a tooltip">This is a paragraph.</p>
```

We Suggest: Always Use Lowercase Attributes

The HTML standard does not require lowercase attribute names.

The title attribute (and all other attributes) can be written with uppercase or lowercase like **title** or **TITLE**.

However, W3C **recommends** lowercase attributes in HTML, and **demands** lowercase attributes for stricter document types like XHTML.

At W3Schools we always use lowercase attribute names.

We Suggest: Always Quote Attribute Values

The HTML standard does not require quotes around attribute values.

However, W3C **recommends** quotes in HTML, and **demands** quotes for stricter document types like XHTML.

Good:

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

Bad:

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

Sometimes you have to use quotes. This example will not display the title attribute correctly, because it contains a space:

Example

```
<p title=About W3Schools>
```

Â At W3Schools we always use quotes around attribute values.

Single or Double Quotes?

Double quotes around attribute values are the most common in HTML, but single quotes can also be used.

In some situations, when the attribute value itself contains double quotes, it is necessary to use single quotes:

```
<p title='John "ShotGun" Nelson'>
```

Or vice versa:

```
<p title="John 'ShotGun' Nelson">
```

Chapter Summary

- All HTML elements can have **attributes**
- The `href` attribute of `<a>` specifies the URL of the page the link goes to
- The `src` attribute of `<img>` specifies the path to the image to be displayed
- The `width` and `height` attributes of `<img>` provide size information for images
- The `alt` attribute of `<img>` provides an alternate text for an image
- The `style` attribute is used to add styles to an element, such as color, font, size, and more
- The `lang` attribute of the `<html>` tag declares the language of the Web page
- The `title` attribute defines some extra information about an element

---

HTML Exercises

HTML Attribute Reference

A complete list of all attributes for each HTML element, is listed in our: HTML Attribute Reference.

HTML Headings

HTML headings are titles or subtitles that you want to display on a webpage.

Example

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

HTML Headings

HTML headings are defined with the `<h1>` to `<h6>` tags.

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

Example

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

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

Headings Are Important

Search engines use the headings to index the structure and content of your web pages.

Users often skim a page by its headings. It is important to use headings to show the document structure.

`<h1>` headings should be used for main headings, followed by `<h2>` headings, then the less important `<h3>`, and so on.

Note: Use HTML headings for headings only. Don't use headings to make text **BIG** or **bold**.

Bigger Headings

Each HTML heading has a default size. However, you can specify the size for any heading with the `style` attribute, using the CSS `font-size` property:

Example

```
<h1 style="font-size:60px;">Heading 1</h1>
```

HTML Exercises

HTML Tag Reference

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

Tag	Description
<html>	Defines the root of an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Paragraphs

A paragraph always starts on a new line, and is usually a block of text.

HTML Paragraphs

The HTML <p> element defines a paragraph.

A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

Example

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

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 display by adding extra spaces or extra lines in your HTML code.

The browser will automatically remove any extra spaces and lines when the page is displayed:

Example

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

<p>
This paragraph
containsÂ Â Â Â Â Â Â Â Â  a lot of spaces
in the sourceÂ Â Â Â Â Â Â Â Â  code,
but theÂ Â Â Â Â Â Â Â  browser
ignores it.
</p>
```

HTML Horizontal Rules

The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

The <hr> element is used to separate content (or define a change) in an HTML page:

Example

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

```
<p>This is some text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>
```

The <hr> tag is an empty tag, which means that it has no end tag.

HTML Line Breaks

The HTML <br> 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 paragraph<br>with line breaks.</p>
```

The <br> tag is an empty tag, which means that it has no end tag.

The Poem Problem

This poem will display on a single line:

Example

```
<p>
Â  My Bonnie lies over the ocean.

Â  My Bonnie lies over the sea.

Â  My Bonnie lies over the ocean.

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

Solution - 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>
Â  My Bonnie lies over the ocean.

Â  My Bonnie lies over the sea.

Â  My Bonnie lies over the ocean.

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

HTML Exercises

HTML Tag Reference

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

Tag	Description
<p>	Defines a paragraph
<hr>	Defines a thematic change in the content
 	Inserts a single line break
<pre>	Defines pre-formatted text

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Styles

The HTML `style` attribute is used to add styles to an element, such as color, font, size, and more.

Example

I am Red

I am Blue

I am Big

The HTML Style Attribute

Setting the style of an HTML element, can be done with the `style` attribute.

The HTML `style` attribute has the following syntax:

<*tagname* 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.

Background Color

The CSS `background-color` property defines the background color for an HTML element.

Example

Set the background color for a page to powderblue:

```
<body style="background-color:powderblue;">

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

</body>
```

Example

Set background color for two different elements:

```
<body>

<h1 style="background-color:powderblue;">This is a heading</h1>
<p style="background-color:tomato;">This is a paragraph.</p>

</body>
```

Text Color

The CSS `color` property defines the text color for an HTML element:

Example

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

Fonts

The CSS `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>
```

Text Size

The CSS `font-size` property defines the text size 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>
```

Text Alignment

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

Example

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

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

HTML Exercises

HTML Text Formatting

HTML contains several elements for defining text with a special meaning.

Example

This text is bold

This text is italic

This is _{subscript} and ^{superscript}

HTML Formatting Elements

Formatting elements were designed to display special types of text:

- `<b>` - Bold text
- `<strong>` - Important text
- `<i>` - Italic text
- `<em>` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `<del>` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

HTML and Elements

The HTML element defines bold text, without any extra importance.

Example

This text is bold

The HTML element defines text with strong importance. The content inside is typically displayed in bold.

Example

This text is important!

HTML and Elements

The HTML element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.

Tip: The tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

Example

This text is italic

The HTML element defines emphasized text. The content inside is typically displayed in italic.

Tip: A screen reader will pronounce the words in with an emphasis, using verbal stress.

Example

This text is emphasized

HTML <small> Element

The HTML <small> element defines smaller text:

Example

<small>This is some smaller text.</small>

HTML <mark> Element

The HTML <mark> element defines text that should be marked or highlighted:

Example

Do not forget to buy <mark>milk</mark> today.

HTML <del> Element

The HTML <del> element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text:

Example

My favorite color is ~~blue~~ red.

HTML <ins> Element

The HTML <ins> element defines a text that has been inserted into a document. Browsers will usually underline inserted text:

Example

My favorite color is ~~blue~~ <ins>red</ins>.

HTML <sub> Element

The HTML <sub> element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H_2O:

Example

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

HTML <sup> Element

The HTML <sup> element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW[1]:

Example

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

HTML Exercises

HTML Text Formatting Elements

Tag	Description
<b>	Defines bold text
	Defines emphasized textÂ
<i>	Defines a part of text in an alternate voice or mood
<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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Quotation and Citation Elements

In this chapter we will go through the <blockquote>,<q>, <abbr>, <address>, <cite>, and <bdo> HTML elements.

Example

Here is a quote from WWF's website:

> For nearly 60 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by more than one million members in the United States and close to five million globally.

HTML <blockquote> for Quotations

The HTML <blockquote> element defines a section that is quoted from another source.

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>

HTML <q> for Short Quotations

The HTML `<q>` tag defines a short quotation.

Browsers normally insert quotation marks around the quotation.

Example

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

HTML <abbr> for Abbreviations

The HTML `<abbr>` tag defines an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".

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

Tip: Use the global title attribute to show the description for the abbreviation/acronym when you mouse over the element.Â

Example

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

HTML <address> for Contact Information

The HTML `<address>` tag defines the contact information for the author/owner of a document or an article.

The contact information can be an email address, URL, physical address, phone number, social media handle, etc.

The text in the `<address>` element usually renders in *italic,* and browsers will always add a line break before and after the `<address>` element.

Example

<address>
Written by John Doe.
Visit us at:
Example.com
Box 564, Disneyland
USA
</address>

HTML <cite> for Work Title

The HTML `<cite>` tag defines the title of a creative work (e.g. a book, a poem, a song, a movie, a painting, a sculpture, etc.).

Note: A person's name is not the title of a work.

The text in the `<cite>` element usually renders in *italic*.

Example

<cite>The Scream</cite> by Edvard Munch. Painted in 1893.

HTML <bdo> for Bi-Directional Override

BDO stands for Bi-Directional Override.

The HTML `<bdo>` tag is used to override the current text direction:

Example

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

HTML Exercises

HTML Quotation and Citation 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
<cite>	Defines the title of a work
<q>	Defines a short inline quotation

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Comments

HTML comments are not displayed in the browser, but they can help document your HTML source code.

HTML Comment Tags

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

<!-- Write your comments here -->

Notice that there is an exclamation point (!) in the start tag, but not in the end tag.

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

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

Example

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

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

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

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 image at the moment
<img border="0" src="pic_trulli.jpg" alt="Trulli">
-->
```

HTML Exercises

HTML Colors

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

Color Names

In HTML, a color can be specified by using a color name:

Try it Yourself Â»

HTML supports 140 standard color names.

Background Color

You can set the background color for HTML elements:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Text Color

You can set the color of text:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="color:Tomato;">Hello World</h1>
```

```
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Border Color

You can set the color of borders:

<h1 style="border:2px solid Tomato;">Hello World</h1>

<h1 style="border:2px solid DodgerBlue;">Hello World</h1>

<h1 style="border:2px solid Violet;">Hello World</h1>

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

The following three elements have their background color set with RGB, HEX, and HSL values:

```
rgb(255, 99, 71)
```

```
#ff6347
```

```
hsl(9, 100%, 64%)
```

The following two elements have their background color set with RGBA and HSLA values, which adds an Alpha channel to the color (here we have 50% transparency):

```
rgba(255, 99, 71, 0.5)
```

```
hsla(9, 100%, 64%, 0.5)
```

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

Learn more about Color Values

You will learn more about RGB, HEX and HSL in the next chapters.

HTML Colors

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

Color Names

In HTML, a color can be specified by using a color name:

Tomato

Orange

DodgerBlue

MediumSeaGreen

Gray

SlateBlue

Violet

LightGray

Try it Yourself Â»

HTML supports 140 standard color names.

Background Color

You can set the background color for HTML elements:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Text Color

You can set the color of text:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

The following three elements have their background color set with RGB, HEX, and HSL values:

```
rgb(255, 99, 71)
```

```
#ff6347
```

```
hsl(9, 100%, 64%)
```

The following two elements have their background color set with RGBA and HSLA values, which adds an Alpha channel to the color (here we have 50% transparency):

```
rgba(255, 99, 71, 0.5)
```

```
hsla(9, 100%, 64%, 0.5)
```

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

Learn more about Color Values

You will learn more about RGB, HEX and HSL in the next chapters.

HTML RGB and RGBA Colors

An RGB color value represents RED, GREEN, and BLUE light sources.

An RGBA color value is an extension of RGB with an Alpha channel (opacity).

RGB Color Values

In HTML, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.

This means that there are 256 x 256 x 256 = 16777216 possible colors!

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255), and the other two (green and blue) are set to 0.

Another example, rgb(0, 255, 0) is displayed as green, because green is set to its highest value (255), and the other two (red and blue) are set to 0.

To display black, set all color parameters to 0, like this: rgb(0, 0, 0).

To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

Experiment by mixing the RGB values below:

Â

RED

255

GREEN

0

BLUE

0

Example

rgb(255, 0, 0)

rgb(0, 0, 255)

rgb(60, 179, 113)

rgb(238, 130, 238)

rgb(255, 165, 0)

rgb(106, 90, 205)

Shades of Gray

Shades of gray are often defined using equal values for all three parameters:

Example

```
rgb(60, 60, 60)
rgb(100, 100, 100)
rgb(140, 140, 140)
rgb(180, 180, 180)
rgb(200, 200, 200)
rgb(240, 240, 240)
```

RGBA Color Values

RGBA color values are an extension of RGB color values with an Alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

rgba(red, green, blue, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Experiment by mixing the RGBA values below:

Â

RED

255

GREEN

0

BLUE

0

ALPHA

0

Example

rgba(255, 99, 71, 0)

rgba(255, 99, 71, 0.2)

rgba(255, 99, 71, 0.4)

rgba(255, 99, 71, 0.6)

rgba(255, 99, 71, 0.8)

rgba(255, 99, 71, 1)

HTML HEX Colors

A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

HEX Color Values

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff), and the other two (green and blue) are set to 00.

Another example, #00ff00 is displayed as green, because green is set to its highest value (ff), and the other two (red and blue) are set to 00.

To display black, set all color parameters to 00, like this: #000000.

To display white, set all color parameters to ff, like this: #ffffff.

Experiment by mixing the HEX values below:

Â

RED

ff

GREEN

0

BLUE

0

Example

#ff0000

#0000ff

#3cb371

#ee82ee

#ffa500

#6a5acd

Shades of Gray

Shades of gray are often defined using equal values for all three parameters:

Example

#404040

#686868

#a0a0a0

#bebebe

#dcdcdc

#f8f8f8

HTML HSL and HSLA Colors

HSL stands for hue, saturation, and lightness.

HSLA color values are an extension of HSL with an Alpha channel (opacity).

HSL Color Values

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(*hue, saturation, lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage value, 0% is black, and 100% is white.

Experiment by mixing the HSL values below:

Â

HUE

0

SATURATION

100%

LIGHTNESS

50%

Example

hsl(0, 100%, 50%)

hsl(240, 100%, 50%)

hsl(147, 50%, 47%)

hsl(300, 76%, 72%)

hsl(39, 100%, 50%)

hsl(248, 53%, 58%)

Saturation

Saturation can be described as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

0% is completely gray, you can no longer see the color.

Example

hsl(0, 100%, 50%)

hsl(0, 80%, 50%)

hsl(0, 60%, 50%)

hsl(0, 40%, 50%)

hsl(0, 20%, 50%)

hsl(0, 0%, 50%)

Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

Example

hsl(0, 100%, 0%)

hsl(0, 100%, 25%)

hsl(0, 100%, 50%)

hsl(0, 100%, 75%)

hsl(0, 100%, 90%)

hsl(0, 100%, 100%)

Shades of Gray

Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

Example

```
hsl(0, 0%, 20%)
```

```
hsl(0, 0%, 30%)
```

```
hsl(0, 0%, 40%)
```

```
hsl(0, 0%, 60%)
```

```
hsl(0, 0%, 70%)
```

```
hsl(0, 0%, 90%)
```

HSLA Color Values

HSLA color values are an extension of HSL color values with an Alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue, saturation, lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Experiment by mixing the HSLA values below:

Â

HUE

0

SATURATION

100%

LIGHTNESS

50%

ALPHA

0.5

Example

```
hsla(9, 100%, 64%, 0)
```

```
hsla(9, 100%, 64%, 0.2)
```

hsla(9, 100%, 64%, 0.4)

hsla(9, 100%, 64%, 0.6)

hsla(9, 100%, 64%, 0.8)

hsla(9, 100%, 64%, 1)

HTML Styles - CSS

CSS stands for Cascading Style Sheets.

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

What is CSS?

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

Tip: The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

Using CSS

CSS can be added to HTML documents in 3 ways:

- **Inline** - by using the `style` attribute inside HTML elements
- **Internal** - by using a `<style>` element in the `<head>` section
- **External** - by using a `<link>` element to link to an external CSS file

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

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the `style` attribute of an HTML element.

The following example sets the text color of the `<h1>` element to blue, and the text color of the `<p>` element to red:

Example

```
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element.

The following example sets the text color of ALL the `<h1>` elements (on that page) to blue, and the text color of ALL the `<p>` elements to red. In addition, the page will be displayed with a "powderblue" background color:Â

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1Â Â  {color: blue;}
pÂ Â Â  {color: red;}
</style>
</head>
<body>

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

</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the `<head>` section of each HTML page:

Example

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

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

</body>
</html>
```

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is what the "styles.css" file looks like:

"styles.css":

```
body {
Â Â background-color: powderblue;
}
h1 {
Â  color: blue;
}
p {
```

Â Â color: red;
}

Tip: With an external style sheet, you can change the look of an entire web site, by changing one file!

CSS Colors, Fonts and Sizes

Here, we will demonstrate some commonly used CSS properties. You will learn more about them later.

The CSS `color` property defines the text color to be used.

The CSS `font-family` property defines the font to be used.

The CSS `font-size` property defines the text size to be used.

Example

Use of CSS color, font-family and font-size properties:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
Â Â color: blue;
Â Â font-family: verdana;
Â Â font-size: 300%;
}
pÂ {
Â  color: red;
Â  font-family: courier;
Â  font-size: 160%;
}
</style>
</head>
<body>

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

</body>
</html>
```

CSS Border

The CSS `border` property defines a border around an HTML element.

Tip: You can define a border for nearly all HTML elements.

Example

Use of CSS border property:Â

```
p {
Â Â border: 2px solid powderblue;
}
```

CSS Padding

The CSS `padding` property defines a padding (space) between the text and the border.

Example

Use of CSS border and padding properties:

```
p {
Â  border: 2px solid powderblue;
Â  padding: 30px;
}
```

CSS Margin

The CSS `margin` property defines a margin (space) outside the border.

Example

Use of CSS border and margin properties:

```
p {
Â  border: 2px solid powderblue;
Â Â margin: 50px;
}
```

Link to External CSS

External style sheets can be referenced with a full URL or with a path relative to the current web page.

Example

This example uses a full URL to link to a style sheet:

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">
```

Example

This example links to a style sheet located in the html folder on the current web site:Â

```
<link rel="stylesheet" href="/html/styles.css">
```

Example

This example links to a style sheet located in the same folder as the current page:

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

You can read more about file paths in the chapter HTML File Paths.

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 borders
- Use the CSS `padding` property for space inside the border
- Use the CSS `margin` property for space outside the border

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

HTML Exercises

HTML Style Tags

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Links

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.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. A link can be an image or any other HTML element!

HTML Links - Syntax

The HTML `<a>` tag defines a hyperlink. It has the following syntax:

link text

The most important attribute of the `<a>` element is the `href` attribute, which indicates the link's destination.

The *link text* is the part that will be visible to the reader.

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

Example

This example shows how to create a link to W3Schools.com:

Visit W3Schools.com!

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

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

Tip: Links can of course be styled with CSS, to get another look!

HTML Links - The target Attribute

By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

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

The `target` attribute can have one of the following values:

- `_self` - Default. Opens the document in the same window/tab as it was clicked
- `_blank` - Opens the document in a new window or tab
- `_parent` - Opens the document in the parent frame
- `_top` - Opens the document in the full body of the window

Example

Use target="_blank" to open the linked document in a new browser window or tab:

Visit W3Schools!

Absolute URLs vs. Relative URLs

Both examples above are using an **absolute URL** (a full web address) in the `href` attribute.

A local link (a link to a page within the same website) is specified with a **relative URL** (without the "https://www" part):

Example

```
<h2>Absolute URLs</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
<p><a href="https://www.google.com/">Google</a></p>

<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

HTML Links - Use an Image as a Link

To use an image as a link, just put the `<img>` tag inside the `<a>` tag:

Example

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

Link to an Email Address

Use `mailto:` inside the `href` attribute to create a link that opens the user's email program (to let them send a new email):

Example

```
<a href="mailto:someone@example.com">Send email</a>
```

Button as a Link

To use an HTML button as a link, you have to add some JavaScript code.

JavaScript allows you to specify what happens at certain events, such as a click of a button:

Example

```
<button onclick="document.location='default.asp'">HTML Tutorial</button>
```

Tip: Learn more about JavaScript in our JavaScript Tutorial.

Link Titles

The `title` attribute specifies extra information about an element. The information is most often shown as a tooltip text when the mouse moves over the element.

Example

```
<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML section">Visit our HTML Tutorial</a>
```

More on Absolute URLs and Relative URLs

Example

Use a full URL to link to a web page:Â

```
<a href="https://www.w3schools.com/html/default.asp">HTML tutorial</a>
```

Example

Link to a page located in the html folder on the current web site:Â

```
<a href="/html/default.asp">HTML tutorial</a>
```

Example

Link to a page located in the same folder as the current page:Â

```
<a href="default.asp">HTML tutorial</a>
```

You can read more about file paths in the chapter HTML File Paths.

Chapter Summary

- Use the `<a>` element to define a link
- Use the `href` attribute to define the link address
- Use the `target` attribute to define where to open the linked document
- Use the `<img>` element (inside `<a>`) to use an image as a link
- Use the `mailto:` scheme inside the `href` attribute to create a link that opens the user's email program

HTML Link Tags

Tag	Description
<a>	Defines a hyperlink

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Links

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.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. A link can be an image or any other HTML element!

HTML Links - Syntax

The HTML `<a>` tag defines a hyperlink. It has the following syntax:

link text

The most important attribute of the `<a>` element is the `href` attribute, which indicates the link's destination.

The *link text* is the part that will be visible to the reader.

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

Example

This example shows how to create a link to W3Schools.com:

Visit W3Schools.com!

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

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

Tip: Links can of course be styled with CSS, to get another look!

HTML Links - The target Attribute

By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

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

The `target` attribute can have one of the following values:

- `_self` - Default. Opens the document in the same window/tab as it was clicked
- `_blank` - Opens the document in a new window or tab
- `_parent` - Opens the document in the parent frame
- `_top` - Opens the document in the full body of the window

Example

Use target="_blank" to open the linked document in a new browser window or tab:

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

Absolute URLs vs. Relative URLs

Both examples above are using an **absolute URL** (a full web address) in the `href` attribute.

A local link (a link to a page within the same website) is specified with a **relative URL** (without the "https://www" part):

Example

```
<h2>Absolute URLs</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
<p><a href="https://www.google.com/">Google</a></p>

<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

HTML Links - Use an Image as a Link

To use an image as a link, just put the `<img>` tag inside the `<a>` tag:

Example

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

Link to an Email Address

Use `mailto:` inside the `href` attribute to create a link that opens the user's email program (to let them send a new email):

Example

```
<a href="mailto:someone@example.com">Send email</a>
```

Button as a Link

To use an HTML button as a link, you have to add some JavaScript code.

JavaScript allows you to specify what happens at certain events, such as a click of a button:

Example

```
<button onclick="document.location='default.asp'">HTML Tutorial</button>
```

Tip: Learn more about JavaScript in our JavaScript Tutorial.

Link Titles

The `title` attribute specifies extra information about an element. The information is most often shown as a tooltip text when the mouse moves over the element.

Example

```
<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML section">Visit our HTML Tutorial</a>
```

More on Absolute URLs and Relative URLs

Example

Use a full URL to link to a web page:Â

```
<a href="https://www.w3schools.com/html/default.asp">HTML tutorial</a>
```

Example

Link to a page located in the html folder on the current web site:Â

HTML tutorial

Example

Link to a page located in the same folder as the current page:Â

HTML tutorial

You can read more about file paths in the chapter HTML File Paths.

Chapter Summary

- Use the `<a>` element to define a link
- Use the `href` attribute to define the link address
- Use the `target` attribute to define where to open the linked document
- Use the `<img>` element (inside `<a>`) to use an image as a link
- Use the `mailto:` scheme inside the `href` attribute to create a link that opens the user's email program

---

HTML Link Tags

Tag	Description
<a>	Defines a hyperlink

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Links - Different Colors

An HTML link is displayed in a different color depending on whether it has been visited, is unvisited, or is active.

HTML Link Colors

By default, a link will appear like 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 link state colors, by using CSS:

Example

Here, an unvisited link will be green with no underline. A visited link will be pink with no underline. An active link will be yellow and underlined. In addition, when mousing over a link (a:hover) it will become red and underlined:

```
<style>
a:link {
Â Â color: green;
Â Â background-color: transparent;
Â Â text-decoration: none;
}

a:visited {
Â  color: pink;
Â  background-color: transparent;
Â  text-decoration: none;
}

a:hover {
Â  color: red;
Â  background-color: transparent;
Â  text-decoration: underline;
}

a:active {
```

Â color: yellow;
Â background-color: transparent;
Â text-decoration: underline;
}
</style>

Link Buttons

A link can also be styled as a button, by using CSS:

This is a link

Example

<style>
a:link, a:visited {
Â background-color: #f44336;
Â color: white;
Â padding: 15px 25px;
Â text-align: center;
Â text-decoration: none;
Â display: inline-block;
}

a:hover, a:active {
Â background-color: red;
}
</style>

To learn more about CSS, go to our CSS Tutorial.

HTML Link Tags

Tag	Description
<a>	Defines a hyperlink

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Links - Create Bookmarks

HTML links can be used to create bookmarks, so that readers can jump to specific parts of a web page.

Create a Bookmark in HTML

Bookmarks can be useful if a web page is very long.

To create a bookmark - first create the bookmark, then add a link to it.

When the link is clicked, the page will scroll down or up to the location with the bookmark.

Example

First, use the `id` attribute to create a bookmark:

<h2 id="C4">Chapter 4</h2>

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

Example

Jump to Chapter 4

You can also add a link to a bookmark on another page:

Jump to Chapter 4

Chapter Summary

- Use the `id` attribute (id="*value*") to define bookmarks in a page
- Use the `href` attribute (href="#*value*") to link to the bookmark

---

HTML Exercises

HTML Link Tags

Tag	Description
<a>	Defines a hyperlink

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Images

Images can improve the design and the appearance of a web page.

Example

<img src="pic_trulli.jpg" alt="Italian Trulli">

Example

<img src="img_girl.jpg" alt="Girl in a jacket">

Example

<img src="img_chania.jpg" alt="Flowers in Chania">

HTML Images Syntax

The HTML `<img>` tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The `<img>` tag creates a holding space for the referenced image.

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

The `<img>` tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

Syntax

<img src="*url*" alt="*alternatetext*">

The src Attribute

The required `src` attribute specifies the path (URL) to the image.

Note: When a web page loads; it is the browser, at that moment, that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise your visitors will get a broken link icon. The broken link icon and the `alt` text are shown if the browser cannot find the image.

Example

<img src="img_chania.jpg" alt="Flowers in Chania">

The alt Attribute

The required `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

The value of the `alt` attribute should describe the image:

Example

<img src="img_chania.jpg" alt="Flowers in Chania">

If a browser cannot find an image, it will display the value of the `alt` attribute:

Example

<img src="wrongname.gif" alt="Flowers in Chania">

Tip: A screen reader is a software program that reads the HTML code, and allows the user to "listen" to the content. Screen readers are useful for people who are visually impaired or learning disabled.

Image Size - Width and Height

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

Example

<img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">

Alternatively, you can use the `width` and `height` attributes:

Example

<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">

The `width` and `height` attributes always define the width and height of the image in pixels.

Note: Always specify the width and height of an image. If width and height are not specified, the web page might flicker while the image loads.

Width and Height, or Style?

The `width`, `height`, and `style` attributes are all valid in HTML.

However, we suggest using the `style` attribute. It prevents styles sheets from changing the size of images:

Example

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

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

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

</body>
</html>
```

Images in Another Folder

If you have your images in a sub-folder, you must include the folder name in the `src` attribute:

Example

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

Images on Another Server/Website

Some web sites point to an image on another server.

To point to an image on another server, you must specify an absolute (full) URL in the `src` attribute:

Example

<img src="https://www.w3schools.com/images/w3schools_green.jpg" alt="W3Schools.com">

Notes on external images: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

Animated Images

HTML allows animated GIFs:

Example

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

Image as a Link

To use an image as a link, put the `<img>` tag inside the `<a>` tag:

Example

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

Image Floating

Use the CSS `float` property to let the image float to the right or to the left of a text:

Example

```
<p><img src="smiley.gif" alt="Smiley face" style="float:right;width:42px;height:42px;">
The image will float to the right of the text.</p>

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

Tip: To learn more about CSS Float, read our CSS Float Tutorial.

Common Image Formats

Here are the most common image file types, which are supported in all browsers (Chrome, Edge, Firefox, Safari, Opera):

Abbreviation	File Format	File Extension
APNG	Animated Portable Network Graphics	.apng
GIF	Graphics Interchange Format	.gif
ICO	Microsoft Icon	.ico, .cur
JPEG	Joint Photographic Expert Group image	.jpg, .jpeg, .jfif, .pjpeg, .pjp
PNG	Portable Network Graphics	.png
SVG	Scalable Vector Graphics	.svg

Chapter Summary

- Use the HTML `<img>` element to define an image
- Use the HTML `src` attribute to define the URL of the image
- Use the HTML `alt` attribute to define an alternate text for an image, if it cannot be displayed
- Use the HTML `width` and `height` attributes or the CSS `width` and `height` properties to define the size of the image
- Use the CSS `float` property to let the image float to the left or to the right

Note: Loading large images takes time, and can slow down your web page. Use images carefully.

HTML Exercises

HTML Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map
<picture>	Defines a container for multiple image resources

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Images

Images can improve the design and the appearance of a web page.

Example

<img src="pic_trulli.jpg" alt="Italian Trulli">

Example

<img src="img_girl.jpg" alt="Girl in a jacket">

Example

<img src="img_chania.jpg" alt="Flowers in Chania">

HTML Images Syntax

The HTML `<img>` tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The `<img>` tag creates a holding space for the referenced image.

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

The `<img>` tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

Syntax

<img src="*url*" alt="*alternatetext*">

The src Attribute

The required `src` attribute specifies the path (URL) to the image.

Note: When a web page loads; it is the browser, at that moment, that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise your visitors will get a broken link icon. The broken link icon and the `alt` text are shown if the browser cannot find the image.

Example

<img src="img_chania.jpg" alt="Flowers in Chania">

The alt Attribute

The required `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

The value of the `alt` attribute should describe the image:

Example

<img src="img_chania.jpg" alt="Flowers in Chania">

If a browser cannot find an image, it will display the value of the `alt` attribute:

Example

<img src="wrongname.gif" alt="Flowers in Chania">

Tip: A screen reader is a software program that reads the HTML code, and allows the user to "listen" to the content. Screen readers are useful for people who are visually impaired or learning disabled.

Image Size - Width and Height

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

Example

<img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">

Alternatively, you can use the `width` and `height` attributes:

Example

<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">

The `width` and `height` attributes always define the width and height of the image in pixels.

Note: Always specify the width and height of an image. If width and height are not specified, the web page might flicker while the image loads.

Width and Height, or Style?

The `width`, `height`, and `style` attributes are all valid in HTML.

However, we suggest using the `style` attribute. It prevents styles sheets from changing the size of images:

Example

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

```
<head>
<style>
img {
Â Â width: 100%;
}
</style>
</head>
<body>

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

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

</body>
</html>
```

Images in Another Folder

If you have your images in a sub-folder, you must include the folder name in the `src` attribute:

Example

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

Images on Another Server/Website

Some web sites point to an image on another server.

To point to an image on another server, you must specify an absolute (full) URL in the `src` attribute:

Example

```
<img src="https://www.w3schools.com/images/w3schools_green.jpg" alt="W3Schools.com">
```

Notes on external images: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

Animated Images

HTML allows animated GIFs:

Example

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

Image as a Link

To use an image as a link, put the `<img>` tag inside the `<a>` tag:

Example

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

Image Floating

Use the CSS `float` property to let the image float to the right or to the left of a text:

Example

```
<p><img src="smiley.gif" alt="Smiley face" style="float:right;width:42px;height:42px;">
The image will float to the right of the text.</p>

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

Tip: To learn more about CSS Float, read our CSS Float Tutorial.

Common Image Formats

Here are the most common image file types, which are supported in all browsers (Chrome, Edge, Firefox, Safari, Opera):

Abbreviation	File Format	File Extension
APNG	Animated Portable Network Graphics	.apng
GIF	Graphics Interchange Format	.gif
ICO	Microsoft Icon	.ico, .cur
JPEG	Joint Photographic Expert Group image	.jpg, .jpeg, .jfif, .pjpeg, .pjp
PNG	Portable Network Graphics	.png
SVG	Scalable Vector Graphics	.svg

Chapter Summary

- Use the HTML `<img>` element to define an image
- Use the HTML `src` attribute to define the URL of the image
- Use the HTML `alt` attribute to define an alternate text for an image, if it cannot be displayed
- Use the HTML `width` and `height` attributes or the CSS `width` and `height` properties to define the size of the image
- Use the CSS `float` property to let the image float to the left or to the right

Note: Loading large images takes time, and can slow down your web page. Use images carefully.

HTML Exercises

HTML Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map
<picture>	Defines a container for multiple image resources

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Image Maps

With HTML image maps, you can create clickable areas on an image.

Image Maps

The HTML `<map>` tag defines an image map. An image map is an image with clickable areas. The areas are defined with one or more `<area>` tags.

Try to click on the computer, phone, or the cup of coffee in the image below:

Example

Here is the HTML source code for the image map above:

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

<map name="workmap">
Â  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
Â  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
Â  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```

How Does it Work?

The idea behind an image map is that you should be able to perform different actions depending on where in the image you click.

To create an image map you need an image, and some HTML code that describes the clickable areas.

The Image

The image is inserted using the `<img>` tag. The only difference from other images is that you must add a `usemap` attribute:

<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

The `usemap` value starts with a hash tag `#` followed by the name of the image map, and is used to create a relationship between the image and the image map.

Tip: You can use any image as an image map!

Create Image Map

Then, add a `<map>` element.

The `<map>` element is used to create an image map, and is linked to the image by using the required `name` attribute:

<map name="workmap">

The `name` attribute must have the same value as the `<img>`'s `usemap` attribute .

The Areas

Then, add the clickable areas.

A clickable area is defined using an `<area>` element.

Shape

You must define the shape of the clickable area, and you can choose one of these values:

- `rect` - defines a rectangular region
- `circle` - defines a circular region
- `poly` - defines a polygonal region
- `default` - defines the entire region

You must also define some coordinates to be able to place the clickable area onto the image.Â

Shape="rect"

The coordinates for `shape="rect"` come in pairs, one for the x-axis and one for the y-axis.

So, the coordinates `34,44` is located 34 pixels from the left margin and 44 pixels from the top:

The coordinates `270,350` is located 270 pixels from the left margin and 350 pixels from the top:

Now we have enough data to create a clickable rectangular area:

Example

<area shape="rect" coords="34, 44, 270, 350" href="computer.htm">

This is the area that becomes clickable and will send the user to the page "computer.htm":

Shape="circle"

To add a circle area, first locate the coordinates of the center of the circle:

337,300

Then specify the radius of the circle:

44 pixels

Now you have enough data to create a clickable circular area:

Example

<area shape="circle" coords="337, 300, 44" href="coffee.htm">

This is the area that becomes clickable and will send the user to the page "coffee.htm":

Shape="poly"

The `shape="poly"` contains several coordinate points, which creates a shape formed with straight lines (a polygon).

This can be used to create any shape.

Like maybe a croissant shape!

How can we make the croissant in the image below become a clickable link?

We have to find the x and y coordinates for all edges of the croissant:

The coordinates come in pairs, one for the x-axis and one for the y-axis:

Example

<area shape="poly"
coords="140,121,181,116,204,160,204,222,191,270,140,329,85,355,58,352,37,322,40,259,103,161,128,147"
href="croissant.htm">

This is the area that becomes clickable and will send the user to the page "croissant.htm":

Image Map and JavaScript

A clickable area can also trigger a JavaScript function.

Add a `click` event to the `<area>` element to execute a JavaScript function:

Example

Here, we use the onclick attribute to execute a JavaScript function when the area is clicked:

```
<map name="workmap">
Â  <area shape="circle" coords="337,300,44" href="coffee.htm" onclick="myFunction()">
</map>

<script>
function myFunction() {
Â  alert("You clicked the coffee cup!");
}
</script>
```

Chapter Summary

- Use the HTML `<map>` element to define an image map

- Use the HTML `<area>` element to define the clickable areas in the image map
- Use the HTML `usemap` attribute of the `<img>` element to point to an image map

---

HTML Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map
<picture>	Defines a container for multiple image resources

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Background Images

A background image can be specified for almost any HTML element.

Background Image on a HTML element

To add a background image on an HTML element, use the HTML `style` attribute and the CSS `background-image` property:

Example

Add a background image on a HTML element:

You can also specify the background image in the `<style>` element, in the `<head>` section:

Example

Specify the background image in the `<style>` element:

```
<style>
div {
Â  background-image: url('img_girl.jpg');
}
</style>
```

Background Image on a Page

If you want the entire page to have a background image, you must specify the background image on the `<body>` element:

Example

Add a background image for the entire page:

```
<style>
body {
Â  background-image: url('img_girl.jpg');
}
</style>
```

Background Repeat

If the background image is smaller than the element, the image will repeat itself, horizontally and vertically, until it reaches the end of the element:

Example

```
<style>
body {
Â  background-image: url('example_img_girl.jpg');
}
</style>
```

To avoid the background image from repeating itself, set the `background-repeat` property to `no-repeat`.

Example

```
<style>
body {
Â  background-image: url('example_img_girl.jpg');
Â  background-repeat: no-repeat;
}
</style>
```

Background Cover

If you want the background image to cover the entire element, you can set the `background-size` property to `cover`.

Also, to make sure the entire element is always covered, set the `background-attachment` property to `fixed`:

This way, the background image will cover the entire element, with no stretching (the image will keep its original proportions):

Example

```
<style>
body {
Â  background-image: url('img_girl.jpg');
Â  background-repeat: no-repeat;
Â  background-attachment: fixed;
Â  background-size: cover;
}
</style>
```

Background Stretch

If you want the background image to stretch to fit the entire element, you can set the `background-size` property to `100% 100%`:

Try resizing the browser window, and you will see that the image will stretch, but always cover the entire element.

Example

```
<style>
body {
Â  background-image: url('img_girl.jpg');
Â  background-repeat: no-repeat;
Â  background-attachment: fixed;
Â  background-size: 100% 100%;
}
</style>
```

Learn More CSS

From the examples above you have learned that background images can be styled by using the CSS background properties.

To learn more about CSS background properties, study our CSS Background Tutorial.

HTML <picture> Element

The HTML `<picture>` element allows you to display different pictures for different devices or screen sizes.

The HTML <picture> Element

The HTML `<picture>` element gives web developers more flexibility in specifying image resources.

The `<picture>` element contains one or more `<source>` elements, each referring to different images through the `srcset` attribute. This way the browser can choose the image that best fits the current view and/or device.

Each `<source>` element has a `media` attribute that defines when the image is the most suitable.

Example

Show different images for different screen sizes:

```
<picture>
Â  <source media="(min-width: 650px)" srcset="img_food.jpg">
Â  <source media="(min-width: 465px)" srcset="img_car.jpg">
Â  <img src="img_girl.jpg">
</picture>
```

Note: Always specify an `<img>` element as the last child element of the `<picture>` element. The `<img>` element is used by browsers that do not support the `<picture>` element, or if none of the `<source>` tags match.

When to use the Picture Element

There are two main purposes for the `<picture>` element:

1. Bandwidth

If you have a small screen or device, it is not necessary to load a large image file. The browser will use the first `<source>` element with matching attribute values, and ignore any of the following elements.

2. Format Support

Some browsers or devices may not support all image formats. By using the `<picture>` element, you can add images of all formats, and the browser will use the first format it recognizes, and ignore any of the following elements.

Example

The browser will use the first image format it recognizes:

```
<picture>
Â  <source srcset="img_avatar.png">
```

Â <source srcset="img_girl.jpg">
Â <img src="img_beatles.gif" alt="Beatles" style="width:auto;">
</picture>

Note: The browser will use the first `<source>` element with matching attribute values, and ignore any following `<source>` elements.

HTML Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map
<picture>	Defines a container for multiple image resources

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Tables

HTML tables allow web developers to arrange data into rows and columns.

Example

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Define an HTML Table

The `<table>` tag defines an HTML table.

Each table row is defined with a `<tr>` tag. Each table header is defined with a `<th>` tag. Each table data/cell is defined with a `<td>` tag.

By default, the text in `<th>` elements are bold and centered.

By default, the text in `<td>` elements are regular and left-aligned.

Example

A simple HTML table:

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

Note: The `<td>` elements are the data containers of the table.
They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

HTML Table - Add a Border

To add a border to a table, use the CSS `border` property:

Example

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

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

HTML Table - Collapsed Borders

To let the borders collapse into one border, add the CSS `border-collapse` property:

Example

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

HTML Table - Add 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

```
th, td {
Â  padding: 15px;
}
```

HTML Table - Left-align Headings

By default, table headings are bold and centered.

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

Example

```
th {
Â Â text-align: left;
}
```

HTML Table - Add 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 {
Â  border-spacing: 5px;
}
```

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

HTML Table - Cell that Spans Many Columns

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

Example

```
<table style="width:100%">
Â  <tr>
Â Â Â  <th>Name</th>
Â Â Â  <th colspan="2">Telephone</th>
Â  </tr>
Â  <tr>
Â Â Â  <td>Bill Gates</td>
Â Â Â  <td>55577854</td>
Â Â Â  <td>55577855</td>
Â  </tr>
</table>
```

HTML Table - Cell that Spans Many Rows

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

Example

```
<table style="width:100%">
Â  <tr>
Â Â Â  <th>Name:</th>
Â Â Â  <td>Bill Gates</td>
Â  </tr>
Â  <tr>
Â Â Â  <th rowspan="2">Telephone:</th>
Â Â Â  <td>55577854</td>
Â  </tr>
Â  <tr>
Â Â Â  <td>55577855</td>
Â  </tr>
</table>
```

HTML Table - Add a Caption

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

Example

```
<table style="width:100%">
Â  <caption>Monthly savings</caption>
Â  <tr>
Â Â Â  <th>Month</th>
Â Â Â  <th>Savings</th>
Â  </tr>
Â  <tr>
Â Â Â  <td>January</td>
Â Â Â  <td>$100</td>
Â  </tr>
Â  <tr>
Â Â Â  <td>February</td>
Â Â Â  <td>$50</td>
Â  </tr>
</table>
```

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

A Special Style for One Table

To define a special style for one particular table, add an `id` attribute to the table:

Example

```
<table id="t01">
Â  <tr>
```

Â Â Â <th>Firstname</th>
Â Â Â <th>Lastname</th>
Â Â Â <th>Age</th>
Â </tr>
Â <tr>
Â Â Â <td>Eve</td>
Â Â Â <td>Jackson</td>
Â Â Â <td>94</td>
Â </tr>
</table>

Now you can define a special style for this table:

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

And add more styles:

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

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

HTML Exercises

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Lists

HTML lists allow web developers to group a set of related items in lists.

Example

An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

Unordered HTML List

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) by default:

Example

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

Ordered HTML List

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

The list items will be marked with numbers by default:

Example

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

HTML Description Lists

HTML also supports description lists.

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

The `<dl>` tag defines the description list, the `<dt>` tag defines the term (name), and the `<dd>` tag describes each term:

Example

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

HTML List Tags

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Lists

HTML lists allow web developers to group a set of related items in lists.

Example

An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

Unordered HTML List

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) by default:

Example

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

Ordered HTML List

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

The list items will be marked with numbers by default:

Example

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

HTML Description Lists

HTML also supports description lists.

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

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

Example

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

HTML List Tags

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Unordered Lists

The HTML <ul> tag defines an unordered (bulleted) list.

Unordered HTML List

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) by default:

Example

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

Unordered HTML List - Choose List Item Marker

The CSS `list-style-type` property is used to define the style of the list item marker. It can have one of the following values:

Value	Description
disc	Sets the list item marker to a bullet (default)
circle	Sets the list item marker to a circle
square	Sets the list item marker to a square
none	The list items will not be marked

Example - Disc

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

Example - Circle

```
<ul style="list-style-type:circle;">
```

Â Coffee
Â Tea
Â Milk

Example - Square

<ul style="list-style-type:square;">
Â Coffee
Â Tea
Â Milk

Example - None

<ul style="list-style-type:none;">
Â Coffee
Â Tea
Â Milk

Nested HTML Lists

Lists can be nested (list inside list):

Example

Â Coffee
Â Tea
Â Â
Â Â Â Â Black tea
Â Â Â Â Â Green tea
Â Â Â
Â
Â Milk

Note: A list item (`<li>`) can contain a new list, and other HTML elements, like images and links, etc.

Horizontal List with CSS

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

One popular way is to style a list horizontally, to create a navigation menu:

Example

<!DOCTYPE html>
<html>
<head>
<style>
ul {
Â list-style-type: none;
Â margin: 0;
Â padding: 0;
Â Â overflow: hidden;
Â Â background-color: #333333;
}

li {
Â Â float: left;
}

li a {
Â Â display: block;
Â Â color: white;
Â Â text-align: center;
Â Â padding: 16px;
Â text-decoration: none;
}

```
li a:hover {
Â Â background-color: #111111;
}
</style>
</head>
<body>

<ul>
Â  <li><a href="#home">Home</a></li>
Â  <li><a href="#news">News</a></li>
Â  <li><a href="#contact">Contact</a></li>
Â  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

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

Chapter Summary

- Use the HTML `<ul>` element to define an unordered list
- Use the CSS `list-style-type` property to define the list item marker
- Use the HTML `<li>` element to define a list item
- Lists can be nested
- List items can contain other HTML elements
- Use the CSS property `float:left` to display a list horizontally

HTML List Tags

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Ordered Lists

The HTML `<ol>` tag defines an ordered list. An ordered list can be numerical or alphabetical.

Ordered HTML List

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

The list items will be marked with numbers by default:

Example

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

Ordered HTML List - The Type Attribute

The `type` attribute of the `<ol>` tag, defines the type of the list item 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">
Â  <li>Coffee</li>
Â  <li>Tea</li>
Â  <li>Milk</li>
</ol>
```

Uppercase Letters:

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

Lowercase Letters:

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

Uppercase Roman Numbers:

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

Lowercase Roman Numbers:

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

Control List Counting

By default, an ordered list will start counting from 1. If you want to start counting from a specified number, you can use the `start` attribute:

Example

```
<ol start="50">
Â  <li>Coffee</li>
Â  <li>Tea</li>
Â  <li>Milk</li>
</ol>
```

Nested HTML Lists

Lists can be nested (list inside list):

Example

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

```
Â Â Â  Â  <li>Black tea</li>
Â Â Â Â Â  <li>Green tea</li>
Â Â Â  </ol>
Â  </li>
Â  <li>Milk</li>
</ol>
```

Note: A list item (`<li>`) can contain a new list, and other HTML elements, like images and links, etc.

Chapter Summary

- 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
- Lists can be nested
- List items can contain other HTML elements

HTML List Tags

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Other Lists

HTML also supports description lists.

HTML Description Lists

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

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

Example

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

Chapter Summary

- 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 describe the term in a description list

---

HTML Exercises

HTML List Tags

Tag	Description

 Defines an unordered list
 Defines an ordered list
 Defines a list item
<dl> Defines a description list
<dt> Defines a term in a description list
<dd> Describes the term in a description list

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Block and Inline Elements

Every HTML element has a default display value, depending on what type of element it is.

There are two display values: block and inline.

Block-level Elements

A block-level element always starts on a new line.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

A block level element has a top and a bottom margin, whereas an inline element does not.

The element is a block-level element.

Example

Hello World

Here are the block-level elements in HTML:

<address>
<article>
<aside>
<blockquote>
<canvas>
<dd>

<dl>
<dt>
<fieldset>
<figcaption>
<figure>
<footer>
<form>
<h1>-<h6>
<header>
<hr>

<main>
<nav>
<noscript>

<p>
<pre>
<section>
<table>
<tfoot>

<video>

Inline Elements

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

This is a element inside a paragraph.

Example

Hello World

Here are the inline elements in HTML:

<a>
<abbr>
<acronym>
<b>
<bdo>
<big>

<button>
<cite>
<code>
<dfn>

<i>
<img>
<input>
<kbd>
<label>
<map>
<object>
<output>
<q>
<samp>
<script>
<select>
<small>

<sub>
<sup>
<textarea>
<time>
<tt>
<var>

Note: An inline element cannot contain a block-level element!

The Element

The `<div>` element is often used as a container for other HTML elements.

The `<div>` element has no required attributes, but `style`, `class` and `id` are common.

When used together with CSS, the `<div>` element can be used to style blocks of content:

Example

```
<div style="background-color:black;color:white;padding:20px;">
Â  <h2>London</h2>
Â  <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>
```

The Element

The `<span>` element is an inline container used to mark up a part of a text, or a part of a document.

The `<span>` element has no required attributes, but `style`, `class` and `id` are common.

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

Example

```
<p>My mother has <span style="color:blue;font-weight:bold">blue</span> eyes and my father has <span style="color:darkolivegreen;font-weight:bold">dark green</span> eyes.</p>
```

Chapter Summary

- There are two display values: block and inline
- A block-level element always starts on a new line and takes up the full width available
- An inline element does not start on a new line and it only takes up as much width as necessary
- The `<div>` element is a block-level and is often used as a container for other HTML elements
- The `<span>` element is an inline container used to mark up a part of a text, or a part of a document

---

HTML Tags

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML class Attribute

The HTML `class` attribute is used to specify a class for an HTML element.

Multiple HTML elements can share the same class.

Using The class Attribute

The `class` attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.

In the following example we have three `<div>` elements with a `class` attribute with the value of "city". All of the three `<div>` elements will be styled equally according to the `.city` style definition in the head section:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
Â Â background-color: tomato;
Â  color: white;
Â  border: 2px solid black;
Â  margin: 20px;
Â  padding: 20px;
}
</style>
</head>
```

```
<body>

<div class="city">
Â  <h2>London</h2>
Â  <p>London is the capital of England.</p>
</div>

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

<div class="city">
Â  <h2>Tokyo</h2>
Â  <p>Tokyo is the capital of Japan.</p>
</div>

</body>
</html>
```

In the following example we have two `<span>` elements with a `class` attribute with the value of "note". Both `<span>` elements will be styled equally according to the `.note` style definition in the head section:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
Â  font-size: 120%;
Â Â color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>

</body>
</html>
```

Tip: The `class` attribute can be used on **any** HTML element.

Note: The class name is case sensitive!

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

The Syntax For Class

To create a class; write a period (.) character, followed by a class name. Then, define the CSS properties within curly braces {}:

Example

Create a class named "city":

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
Â  background-color: tomato;
Â  color: white;
Â  padding: 10px;
}
</style>
</head>
<body>

<h2 class="city">London</h2>
<p>London is the capital of England.</p>
```

```
<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>

</body>
</html>
```

Multiple Classes

HTML elements can belong to more than one class.

To define multiple classes, separate the class names with a space, e.g. . The element will be styled according to all the classes specified.

In the following example, the first `<h2>` element belongs to both the `city` class and also to the `main` class, and will get the CSS styles from both of the classes:Â

Example

```
<h2 class="city main">London</h2>
<h2 class="city">Paris</h2>
<h2 class="city">Tokyo</h2>
```

Different Elements Can Share Same Class

Different HTML elements can point to the same class name.

In the following example, both `<h2>` and `<p>` points to the "city" class and will share the same style:

Example

```
<h2 class="city">Paris</h2>
<p class="city">Paris is the capital of France</p>
```

Use of The class Attribute in JavaScript

The class name can also be used by JavaScript to perform certain tasks for specific elements.

JavaScript can access elements with a specific class name with the `getElementsByClassName()` method:

Example

Click on a button to hide all elements with the class name "city":

```
<script>
function myFunction() {
Â  var x = document.getElementsByClassName("city");
Â  for (var i = 0; i < x.length; i++) {
Â Â Â  x[i].style.display = "none";
Â  }
}
</script>
```

Don't worry if you don't understand the code in the example above.

You will learn more about JavaScript in our HTML JavaScript chapter, or you can study our JavaScript Tutorial.

Chapter Summary

- The HTML `class` attribute specifies one or more class names for an element
- Classes are used by CSS and JavaScript to select and access specific elements
- The `class` attribute can be used on any HTML element
- The class name is case sensitive
- Different HTML elements can point to the same class name
- JavaScript can access elements with a specific class name with the `getElementsByClassName()` method

---

HTML Exercises

HTML id Attribute

The HTML `id` attribute is used to specify a unique id for an HTML element.

You cannot have more than one element with the same id in an HTML document.

Using The id Attribute

The `id` attribute specifies a unique id for an HTML element. The value of the `id` attribute must be unique within the HTML document.

The `id` attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.

In the following example we have an `<h1>` element that points to the id name "myHeader". This `<h1>` element will be styled according to the `#myHeader` style definition in the head section:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
Â  background-color: lightblue;
Â  color: black;
Â  padding: 40px;
Â  text-align: center;
}
</style>
</head>
<body>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

Note: The id name is case sensitive!

Note: The id name must contain at least one character, cannot start with a number, and must not contain whitespaces (spaces, tabs, etc.).

Difference Between Class and ID

A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page:

Example

```
<style>
/* Style the element with the id "myHeader" */
#myHeader {
Â  background-color: lightblue;
Â  color: black;
Â Â padding: 40px;
Â Â text-align: center;
}

/* Style all elements with the class name "city" */
.city {
Â  background-color: tomato;
```

```
Â Â color: white;
Â Â padding: 10px;
}
</style>

<!-- An element with a unique id -->
<h1 id="myHeader">My Cities</h1>

<!-- Multiple elements with same class -->
<h2 class="city">London</h2>
<p>London is the capital of England.</p>

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

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
```

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

HTML Bookmarks with ID and Links

HTML bookmarks are used to allow readers to jump to specific parts of a webpage.

Bookmarks can be useful if your page is very long.

To use a bookmark, you must first create it, and then add a link to it.

Then, when the link is clicked, the page will scroll to the location with the bookmark.

Example

First, create a bookmark with the `id` attribute:

```
<h2 id="C4">Chapter 4</h2>
```

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

Example

```
<a href="#C4">Jump to Chapter 4</a>
```

Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

Using The id Attribute in JavaScript

The `id` attribute can also be used by JavaScript to perform some tasks for that specific element.

JavaScript can access an element with a specific id with the `getElementById()` method:

Example

Use the `id` attribute to manipulate text with JavaScript:

```
<script>
function displayResult() {
Â Â document.getElementById("myHeader").innerHTML = "Have a nice day!";
}
</script>
```

Tip: Study JavaScript in the HTML JavaScript chapter, or in our JavaScript Tutorial.

Chapter Summary

- The `id` attribute is used to specify a unique id for an HTML element
- The value of the `id` attribute must be unique within the HTML document
- The `id` attribute is used by CSS and JavaScript to style/select a specific element
- The value of the `id` attribute is case sensitive
- The `id` attribute is also used to create HTML bookmarks

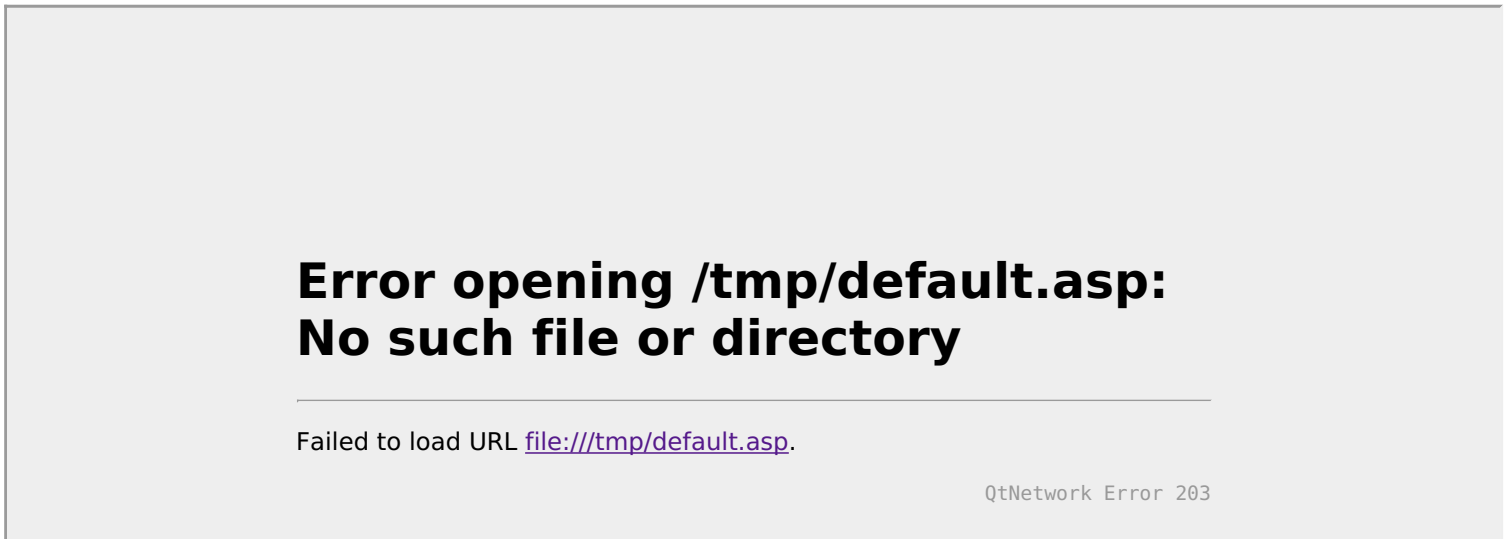
- JavaScript can access an element with a specific id with the `getElementById()` method

---

HTML Exercises

HTML Iframes

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



HTML Iframe Syntax

The HTML `<iframe>` tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

Syntax

<iframe src="*url*" title="*description*"></iframe>

Tip: It is a good practice to always include a `title` attribute for the `<iframe>`. This is used by screen readers to read out what the content of the iframe is.

Iframe - Set Height and Width

Use the `height` and `width` attributes to specify the size of the iframe.

The height and width are specified in pixels by default:

Example

<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>

Or you can add the `style` attribute and use the CSS `height` and `width` properties:

Example

<iframe src="demo_iframe.htm" style="height:200px;width:300px;" title="Iframe Example"></iframe>

Iframe - Remove the Border

By default, an iframe has a 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;" title="Iframe Example"></iframe>
```

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

Example

```
<iframe src="demo_iframe.htm" style="border:2px solid red;" title="Iframe Example"></iframe>
```

Iframe - 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" title="Iframe Example"></iframe>

<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
```

Chapter Summary

- The HTML `<iframe>` tag specifies an inline frame
- The `src` attribute defines the URL of the page to embed
- Always include a `title` attribute (for screen readers)
- The `height` and `width` attributes specifies the size of the iframe
- Use `border:none;` to remove the border around the iframe

HTML Exercises

HTML iframe Tag

Tag	Description
<iframe>	Defines an inline frame

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML JavaScript

JavaScript makes HTML pages more dynamic and interactive.

Example

My First JavaScript

Click me to display Date and Time

px	em	percent
5px	0.3125em	31.25%
6px	0.3750em	37.50%
7px	0.4375em	43.75%
8px	0.5000em	50.00%
9px	0.5625em	56.25%
10px	0.6250em	62.50%
11px	0.6875em	68.75%
12px	0.7500em	75.00%
13px	0.8125em	81.25%
14px	0.8750em	87.50%
15px	0.9375em	93.75%

16px	1.0000em	100.00%
17px	1.0625em	106.25%
18px	1.1250em	112.50%
19px	1.1875em	118.75%
20px	1.2500em	125.00%
21px	1.3125em	131.25%
22px	1.3750em	137.50%
23px	1.4375em	143.75%
24px	1.5000em	150.00%
25px	1.5625em	156.25%

The HTML <script> Tag

The HTML `<script>` tag is used to define a client-side script (JavaScript).

The `<script>` element either contains script 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.

To select an HTML element, JavaScript most often uses the `document.getElementById()` method.

This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

Example

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

Tip: You can learn much more about JavaScript in our JavaScript Tutorial.

A Taste of JavaScript

Here are some examples of what JavaScript can do:

Example

JavaScript can change content:

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

Example

JavaScript can change styles:

```
document.getElementById("demo").style.fontSize = "25px";
document.getElementById("demo").style.color = "red";
document.getElementById("demo").style.backgroundColor = "yellow";
```

Example

JavaScript can change attributes:

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

The HTML <noscript> Tag

The HTML `<noscript>` tag defines an alternate content to be displayed to users that have disabled scripts in their browser or have a browser that doesn't support scripts:

Example

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

HTML Exercises

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML File Paths

A file path describes the location of a file in a web site's folder structure.

File Path Examples

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current folder
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder

---

HTML File Paths

A file path describes the location of a file in a web site's folder structure.

File paths are used when linking to external files, like:

- Web pages
- Images
- Style sheets
- JavaScripts

Absolute File Paths

An absolute file path is the full URL to a file:

Example

<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">

The <img> tag is explained in the chapter: HTML Images.

Relative File Paths

A relative file path points to a file relative to the current page.

In the following example, the file path points to a file in the images folder located at the root of the current web:

Example

<img src="/images/picture.jpg" alt="Mountain">

In the following example, the file path points to a file in the images folder located in the current folder:

Example

<img src="images/picture.jpg" alt="Mountain">

In the following example, the file path points to a file in the images folder located in the folder one level up from the

current folder:

Example

```
<img src="../images/picture.jpg" alt="Mountain">
```

Best Practice

It is best practice to use relative file paths (if possible).

When using relative file paths, your web pages will not be bound to your current base URL. All links will work on your own computer (localhost) as well as on your current public domain and your future public domains.

HTML - The Head Element

The HTML `<head>` element is a container for the following elements: `<title>`, `<style>`, `<meta>`, `<link>`, `<script>`, and `<base>`.

The HTML <head> Element

The `<head>` element is a container for metadata (data about data) and is placed between the `<html>` tag and the `<body>` tag.

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

Metadata typically define the document title, character set, styles, scripts, and other meta information.

The HTML <title> Element

The `<title>` element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The `<title>` element is required in HTML documents!

The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

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

So, try to make the title as accurate and meaningful as possible!

A simple HTML document:

Example

```
<!DOCTYPE html>
<html>
<head>
Â  <title>A Meaningful Page Title</title>
</head>
<body>

The content of the document.....

</body>
</html>
```

The HTML <style> Element

The `<style>` element is used to define style information for a single HTML page:

Example

```
<style>
Â  body {background-color: powderblue;}
Â  h1 {color: red;}
Â  p {color: blue;}
</style>
```

The HTML <link> Element

The `<link>` element defines the relationship between the current document and an external resource.

The `<link>` tag is most often used to link to external style sheets:

Example

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

Tip: To learn all about CSS, visit our CSS Tutorial.

The HTML <meta> Element

The `<meta>` element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.

The metadata will not be displayed on the page, but are used by browsers (how to display content or reload page), by search engines (keywords), and other web services.

Examples

Define the character set used:

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

Define keywords for search engines:

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

Define a description of your web page:

```
<meta name="description" content="Free Web tutorials">
```

Define the author of a page:

```
<meta name="author" content="John Doe">
```

Refresh document every 30 seconds:

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

Setting the viewport to make your website look good on all devices:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Example of `<meta>` tags:

Example

```
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML, CSS, JavaScript">
<meta name="author" content="John Doe">
```

Setting The Viewport

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following `<meta>` element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This gives the browser instructions on how to control the page's dimensions and scaling.

The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Tip: If you are browsing this page with a phone or a tablet, you can click on the two links below to see the difference.

Without the viewport meta tag

With the viewport meta tag

The HTML <script> Element

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

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

Example

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

Tip: To learn all about JavaScript, visit our JavaScript Tutorial.

The HTML <base> Element

The `<base>` element specifies the base URL and/or target for all relative URLs in a page.

The `<base>` tag must have either an href or a target attribute present, or both.

There can only be one single `<base>` element in a document!

Example

Specify a default URL and a default target for all links on a page:

```
<head>
<base href="https://www.w3schools.com/" target="_blank">
</head>

<body>
<img src="images/stickman.gif" width="24" height="39" alt="Stickman">
<a href="tags/tag_base.asp">HTML base Tag</a>
</body>
```

Chapter Summary

- The `<head>` element is a container for metadata (data about data)
- The `<head>` element is placed between the `<html>` tag and the `<body>` tag
- The `<title>` element is required and it defines the title of the document
- The `<style>` element is used to define style information for a single document
- The `<link>` tag is most often used to link to external style sheets
- The `<meta>` element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings
- The `<script>` element is used to define client-side JavaScripts
- The `<base>` element specifies the base URL and/or target for all relative URLs in a page

---

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Layout Elements and Techniques

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

Example

Cities

- London
- Paris
- Tokyo

London

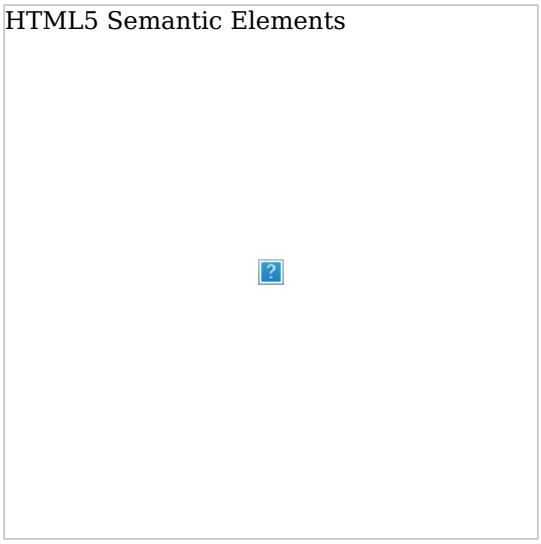
London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

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.

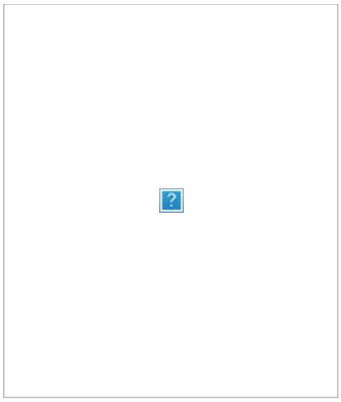
Footer

HTML Layout Elements

HTML has several semantic elements that define the different parts of a web page:



`<header>` - Defines a header for a document or a section
`<nav>` - Defines a set of navigation links
`<section>` - Defines a section in a document
`<article>` - Defines an independent, self-contained content
`<aside>` - Defines content aside from the content (like a sidebar)
`<footer>` - Defines a footer for a document or a section
`<details>` - Defines additional details that the user can open and close on demand
`<summary>` - Defines a heading for the `<details>` element



- `<header>` - Defines a header for a document or a section
- `<nav>` - Defines a set of navigation links
- `<section>` - Defines a section in a document
- `<article>` - Defines an independent, self-contained content
- `<aside>` - Defines content aside from the content (like a sidebar)
- `<footer>` - Defines a footer for a document or a section
- `<details>` - Defines additional details that the user can open and close on demand
- `<summary>` - Defines a heading for the `<details>` element

You can read more about semantic elements in our HTML Semantics chapter.

HTML Layout Techniques

There are four different techniques to create multicolumn layouts. Each technique has its pros and cons:

- CSS framework
- CSS float property
- CSS flexbox
- CSS grid

CSS Frameworks

If you want to create your layout fast, you can use a CSS framework, like W3.CSS or Bootstrap.

CSS Float Layout

It is common to do entire web layouts using the CSS `float` property. Float is easy to learn - you just need to remember how the `float` and `clear` properties work. **Disadvantages:** Floating elements are tied to the document flow, which may harm the flexibility. Learn more about float in our CSS Float and Clear chapter.

Example

Cities

- 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.

Footer

CSS Flexbox Layout

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

Learn more about flexbox in our CSS Flexbox chapter.

Example

Cities

- 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.

Footer

CSS Grid Layout

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Learn more about CSS grids in our CSS Grid View chapter.

HTML Responsive Web Design

Responsive web design is about creating web pages that look good on all devices!

A responsive web design will automatically adjust for different screen sizes and viewports.

Responsive Web Design

What is Responsive Web Design?

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

Try it Yourself Â»

Setting The Viewport

To create a responsive website, add the following `<meta>` tag to all your web pages:

Example

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Without the viewport meta tag:

With the viewport meta tag:

Tip: If you are browsing this page on a phone or a tablet, you can click on the two links above to see the difference.

Responsive Images

Responsive images are images that scale nicely to fit any browser size.

Using the width Property

If the CSS `width` property is set to 100%, the image will be responsive and scale up and down:

Example

<img src="img_girl.jpg" **style="width:100%;">**

Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in

many cases, will be to use the `max-width` property instead.

Using the max-width Property

If the `max-width` property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

Example

<img src="img_girl.jpg" style="**max-width:100%;**height:auto;">

Show Different Images Depending on Browser Width

The HTML `<picture>` element allows you to define different images for different browser window sizes.

Resize the browser window to see how the image below change depending on the width:

Example

```
<picture>
Â  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
Â  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
Â  <source srcset="flowers.jpg">
Â  <img src="img_smallflower.jpg" alt="Flowers">
</picture>
```

Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width".

That way the text size will follow the size of the browser window:

Example

<h1 style="**font-size:10vw**">Hello World</h1>

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

Media Queries

In addition to resize text and images, it is also common to use media queries in responsive web pages.

With media queries you can define completely different styles for different browser sizes.

Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stacked vertically on small screens:

Left Menu

Main Content

Right Content

Example

```
<style>
.left, .right {
Â  float: left;
Â  width: 20%; /* The width is 20%, by default */
}

.main {
Â  float: left;
Â  width: 60%; /* The width is 60%, by default */
```

```
}

/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
Â  .left, .main, .right {
Â Â Â  width: 100%; /* The width is 100%, when the viewport is 800px or smaller */
Â  }
}
</style>
```

Tip: To learn more about Media Queries and Responsive Web Design, read our RWD Tutorial.

Responsive Web Page - Full Example

A responsive web page should look good on large desktop screens and on small mobile phones.

Try it Yourself Â»

Responsive Web Design - Frameworks

All popular CSS Frameworks offer responsive design.

They are free, and easy to use.

W3.CSS

W3.CSS is a modern CSS framework with support for desktop, tablet, and mobile design by default.

W3.CSS is smaller and faster than similar CSS frameworks.

W3.CSS is designed to be a high quality alternative to Bootstrap.

W3.CSS is designed to be independent of jQuery or any other JavaScript library.

W3.CSS Demo

Resize the page to see the responsiveness!

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.

Paris

Paris is the capital of France.

The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.

Tokyo

Tokyo is the capital of Japan.

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

Example

```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<body>

<div class="w3-container w3-green">
Â  <h1>W3Schools Demo</h1>
Â  <p>Resize this responsive page!</p>
</div>

<div class="w3-row-padding">
Â  <div class="w3-third">
Â Â Â  <h2>London</h2>
Â Â Â  <p>London is the capital city of England.</p>
Â Â Â  <p>It is the most populous city in the United Kingdom,
Â Â Â  with a metropolitan area of over 13 million inhabitants.</p>
Â  </div>

Â  <div class="w3-third">
Â Â Â  <h2>Paris</h2>
Â Â Â  <p>Paris is the capital of France.</p>
Â Â Â  <p>The Paris area is one of the largest population centers in Europe,
Â Â Â  with more than 12 million inhabitants.</p>
Â  </div>

Â  <div class="w3-third">
Â Â Â  <h2>Tokyo</h2>
Â Â Â  <p>Tokyo is the capital of Japan.</p>
Â Â Â  <p>It is the center of the Greater Tokyo Area,
Â Â Â  and the most populous metropolitan area in the world.</p>
Â  </div>
</div>

</body>
</html>
```

To learn more about W3.CSS, read our W3.CSS Tutorial.

Bootstrap

Another popular CSS framework is Bootstrap. Bootstrap uses HTML, CSS and jQuery to make responsive web pages.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Bootstrap Example</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
Â  <div class="jumbotron">
Â Â Â  <h1>My First Bootstrap Page</h1>
Â  </div>
Â  <div class="row">
Â Â Â  <div class="col-sm-4">
Â Â Â Â Â  ...
Â Â Â  </div>
Â Â Â  <div class="col-sm-4">
Â Â Â Â Â  ...
Â Â Â  </div>
Â Â Â  <div class="col-sm-4">
Â Â Â  ...
Â Â Â  </div>
Â  </div>
</div>

</body>
</html>
```

To learn more about Bootstrap, go to our Bootstrap Tutorial.

HTML Computer Code Elements

HTML contains several elements for defining user input and computer code.

Example

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

HTML <kbd> For Keyboard Input

The HTML `<kbd>` element is used to define keyboard input. The content inside is displayed in the browser's default monospace font.

Example

Define some text as keyboard input in a document:

```
<p>Save the document by pressing <kbd>Ctrl + S</kbd></p>
```

Result:

Save the document by pressing `Ctrl + S`

HTML <samp> For Program Output

The HTML `<samp>` element is used to define sample output from a computer program. The content inside is displayed in the browser's default monospace font.

Example

Define some text as sample output from a computer program in a document:

```
<p>Message from my computer:</p>
<p><samp>File not found.<br>Press F1 to continue</samp></p>
```

Result:

Message from my computer:

```
File not found.
Press F1 to continue
```

HTML <code> For Computer Code

The HTML <code> elementÂ is used to define a piece of computer code. The content inside is displayed in the browser's default monospace font.

Example

Define some text as computer code in a document:

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

Result:

```
x = 5; y = 6; z = x + y;
```

Notice that the <code> element does not preserve extra whitespace and line-breaks.

To fix this, you can put the <code> element inside a <pre> element:

Example

```
<pre>
<code>
x = 5;
y = 6;
z = x + y;
</code>
</pre>
```

Result:

```
x = 5;
y = 6;
z = x + y;
```

HTML <var> For Variables

The HTML <var> elementÂ is used to define a variable in programming or in a mathematical expression. The content inside is typically displayed in italic.

Example

Define some text as variables in a document:

```
<p>The area of a triangle is: 1/2 x <var>b</var> x <var>h</var>, where <var>b</var> is the base, and <var>h</var> is the vertical height.</p>
```

Result:

The area of a triangle is: 1/2 x *b* x *h*, where *b* is the base, and *h* is the vertical height.

Chapter Summary

- The <kbd> element defines keyboard input
- The <samp> element defines sample output from a computer program
- The <code> elementÂ defines a piece of computer code
- The <var> elementÂ defines a variable in programming or in a mathematical expression
- The <pre> element defines preformatted text

HTML Exercises

HTML Computer Code Elements

Tag	Description
<code>	Defines programming code
<kbd>	Defines keyboard inputÂ
<samp>	Defines computer output
<var>	Defines a variable
<pre>	Defines preformatted text

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Semantic Elements

Semantic elements = 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: `<div>` and `<span>` - Tells nothing about its content.

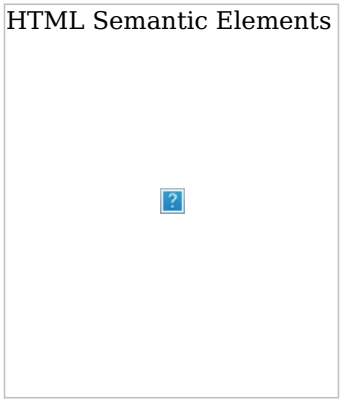
Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

Semantic Elements in HTML

Many web sites contain HTML code like: to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page: Â

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML <section> Element

The `<section>` element defines a section in a document.

According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."

A web page could normally be split into sections for introduction, content, and contact information.

Example

Two sections in a document:

```
<section>
<h1>WWF</h1>
<p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>

<section>
```

```
<h1>WWF's Panda symbol</h1>
<p>The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the London Zoo in the same year of the establishment of WWF.</p>
</section>
```

HTML <article> Element

The `<article>` element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to distribute 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

Three articles with independent, self-contained content:

```
<article>
<h2>Google Chrome</h2>
<p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p>
</article>

<article>
<h2>Mozilla Firefox</h2>
<p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.</p>
</article>

<article>
<h2>Microsoft Edge</h2>
<p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.</p>
</article>
```

Example 2

Use CSS to style the <article> element:

```
<html>
<head>
<style>
.all-browsers {
Â  margin: 0;
Â  padding: 5px;
Â  background-color: lightgray;
}

.all-browsers > h1, .browser {
Â  margin: 10px;
Â  padding: 5px;
}

.browser {
Â  background: white;
}

.browser > h2, p {
Â  margin: 4px;
Â  font-size: 90%;
}
</style>
</head>
<body>

<article class="all-browsers">
Â  <h1>Most Popular Browsers</h1>
Â  <article class="browser">
Â Â Â  <h2>Google Chrome</h2>
```

Â Â Â Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!
Â </article>
Â <article class="browser">
Â Â Â <h2>Mozilla Firefox</h2>
Â Â Â Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.
Â </article>
Â <article class="browser">
Â Â Â <h2>Microsoft Edge</h2>
Â Â Â Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.
Â </article>
</article>

</body>
</html>

Nesting <article> in <section> or Vice Versa?

The `<article>` element specifies independent, self-contained content.

The `<section>` element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, you will find HTML pages with `<section>` elements containing `<article>` elements, and `<article>` elements containing `<section>` elements.

HTML <header> Element

The `<header>` element represents a container for introductory content or a set of navigational links.

A `<header>` element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

Note: You can have several `<header>` elements in one HTML document. However, `<header>` cannot be placed within a `<footer>`, `<address>` or another `<header>` element.

Example

A header for an <article>:Â

<article>
Â <header>
Â Â Â <h1>What Does WWF Do?</h1>
Â Â Â WWF's mission:
Â </header>
Â WWF's mission is to stop the degradation of our planet's natural environment,
Â and build a future in which humans live in harmony with nature.
</article>

HTML <footer> Element

The `<footer>` element defines a footer for a document or section.

A `<footer>` element typically contains:

- authorship information
- copyright information
- contact information
- sitemap
- back to top links
- related documents

You can have several `<footer>` elements in one document.

Example

A footer section in a document:

```
<footer>
Â  <p>Author: Hege Refsnes</p>
Â  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

HTML <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

Example

A set of navigation links:

```
<nav>
Â  <a href="/html/">HTML</a> |
Â  <a href="/css/">CSS</a> |
Â  <a href="/js/">JavaScript</a> |
Â  <a href="/jquery/">jQuery</a>
</nav>
```

HTML <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The <aside> content should be indirectly related to the surrounding content.

Example

Display some content aside from the content it is placed in:

```
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>

<aside>
<h4>Epcot Center</h4>
<p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>
```

Example 2

Use CSS to style the <aside> element:

```
<html>
<head>
<style>
aside {
Â  width: 30%;
Â  padding-left: 15px;
Â  margin-left: 15px;
Â  float: right;
Â  font-style: italic;
Â  background-color: lightgray;
}
</style>
</head>
<body>

<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>

<aside>
<p>The Epcot center is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>
```

My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!
My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!

</body>
</html>

HTML <figure> and <figcaption> Elements

The `<figure>` tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

The `<figcaption>` tag defines a caption for a `<figure>` element. The `<figcaption>` element can be placed as the first or as the last child of a `<figure>` element.

The `<img>` element defines the actual image/illustration.Â

Example

<figure>
Â <img src="pic_trulli.jpg" alt="Trulli">
Â <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>

Why Semantic Elements?

According to the W3C: "A semantic Web allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML

Below is a list of some of the semantic elements in HTML.

Tag	Description
<article>	Defines independent, self-contained content
<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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Style Guide and Coding Conventions

A consistent, clean, and tidy HTML code makes it easier for others to read and understand your code.

Here are some guidelines and tips for creating good HTML code.

Always Declare Document Type

Always declare the document type as the first line in your document.

The correct document type for HTML is:

<!DOCTYPE html>

Use Lowercase Element Names

HTML allows mixing uppercase and lowercase letters in element names.

However, we recommend using lowercase element names, because:

- Mixing uppercase and lowercase names looks bad
- Developers normally use lowercase names
- Lowercase looks cleaner
- Lowercase is easier to write

Good:

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

Bad:

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

Close All HTML Elements

In HTML, you do not have to close all elements (for example the <p> element).

However, we strongly recommend closing all HTML elements, like this:

Good:

```
<section>
Â  <p>This is a paragraph.</p>
Â  <p>This is a paragraph.</p>
</section>
```

Bad:

```
<section>
Â  <p>This is a paragraph.
Â  <p>This is a paragraph.
</section>
```

Use Lowercase Attribute Names

HTML allows mixing uppercase and lowercase letters in attribute names.

However, we recommend using lowercase attribute names, because:

- Mixing uppercase and lowercase names looks bad
- Developers normally use lowercase names
- Lowercase look cleaner
- Lowercase are easier to write

Good:

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

Bad:

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

Always Quote Attribute Values

HTML allows attribute values without quotes.

However, we recommend quoting attribute values, because:

- Developers normally quote attribute values
- Quoted values are easier to read
- You MUST use quotes if the value contains spaces

Good:

<table class="striped">

Bad:

<table class=striped>

Very bad:

This will not work, because the value contains spaces:

<table class=table striped>

Always Specify alt, width, and height for Images

Always specify the `alt` attribute for images. This attribute is important if the image for some reason cannot be displayed.

Also, always define the `width` and `height` of images. This reduces flickering, because the browser can reserve space for the image before loading.

Good:

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

Bad:

<img src="html5.gif">

Spaces and Equal Signs

HTML allows spaces around equal signs. But space-less is easier to read and groups entities better together.

Good:

<link rel="stylesheet" href="styles.css">

Bad:

<link rel = "stylesheet" href = "styles.css">

Avoid Long Code Lines

When using an HTML editor, it is NOT convenient to scroll right and left to read the HTML code.

Try to avoid too long code lines.

Blank Lines and Indentation

Do not add blank lines, spaces, or indentations without a reason.

For readability, add blank lines to separate large or logical code blocks.

For readability, add two spaces of indentation. Do not use the tab key.

Good:

<body>

<h1>Famous Cities</h1>

<h2>Tokyo</h2>
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.

</body>

Bad:

<body>

Â <h1>Famous Cities</h1>

Â <h2>Tokyo</h2>

Â
Â Â Â Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
Â Â Â and the most populous metropolitan area in the world.
Â Â Â It is the seat of the Japanese government and the Imperial Palace,
Â Â Â and the home of the Japanese Imperial Family.
Â

</body>

Good Table Example:

<table>
Â <tr>
Â Â Â <th>Name</th>
Â Â Â <th>Description</th>
Â </tr>
Â <tr>
Â Â Â <td>A</td>
Â Â Â <td>Description of A</td>
Â </tr>
Â <tr>
Â Â Â <td>B</td>
Â Â Â <td>Description of B</td>
Â </tr>
</table>

Good List Example:

Â London
Â Paris
Â Tokyo

Never Skip the <title> Element

The `<title>` element is required in HTML.

The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

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

So, try to make the title as accurate and meaningful as possible:Â

<title>HTML Style Guide and Coding Conventions</title>

Omitting <html> and <body>?

An HTML page will validate without the `<html>` and `<body>` tags:

Example

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

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

However, we strongly recommend to always add the `<html>` and `<body>` tags!

Omitting `<body>` can produce errors in older browsers.

Omitting `<html>` and `<body>` can also crash DOM and XML software.

Omitting <head>?

The HTML <head> tag can also be omitted.

Browsers will add all elements before `<body>`, to a default `<head>` element.

Example

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

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

</body>
</html>
```

However, we recommend using the `<head>` tag.

Close Empty HTML Elements?

In HTML, it is optional to close empty elements.

Allowed:

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

Also Allowed:

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

If you expect XML/XHTML software to access your page, keep the closing slash (/), because it is required in XML and XHTML.

Add the lang Attribute

You should always include the `lang` attribute inside the `<html>` tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

Example

```
<!DOCTYPE html>
<html lang="en-us">
<head>
Â  <title>Page Title</title>
</head>
<body>

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

</body>
</html>
```

Meta Data

To ensure proper interpretation and correct search engine indexing, both the language and the character encoding `<meta charset="charset">` should be defined as early as possible in an HTML document:

```
<!DOCTYPE html>
<html lang="en-us">
<head>
Â  <meta charset="UTF-8">
Â  <title>Page Title</title>
</head>
```

Setting The Viewport

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following `<meta>` element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This gives the browser instructions on how to control the page's dimensions and scaling.

The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Tip: If you are browsing this page with a phone or a tablet, you can click on the two links below to see the difference.

Without the viewport meta tag

With the viewport meta tag

HTML Comments

Short comments should be written on one line, like this:

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

Comments that spans more than one line, should be written like this:

```
<!--
Â  This is a long comment example. This is a long comment example.
Â  This is a long comment example. This is a long comment example.
-->
```

Long comments are easier to observe if they are indented with two spaces.

Using Style Sheets

Use simple syntax for linking to style sheets (the `type` attribute is not necessary):

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

Short CSS rules can be written compressed, like this:

```
p.intro {font-family:Verdana;font-size:16em;}
```

Long CSS rules should be written over multiple lines:

```
body {
Â  background-color: lightgrey;
Â  font-family: "Arial Black", Helvetica, sans-serif;
Â  font-size: 16em;
Â  color: black;
}
```

- Place the opening bracket on the same line as the selector
- Use one space before the opening bracket
- Use two spaces of indentation

- 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

---

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

Using "untidy" HTML code can result in JavaScript errors.

These two JavaScript statements will produce different results:

Example

getElementById("Demo").innerHTML = "Hello";

getElementById("demo").innerHTML = "Hello";

Visit the JavaScript Style Guide.

Use Lower Case File Names

Some 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".

If you use a mix of uppercase and lowercase, you have to be aware of this.

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 lowercase file names!

File Extensions

HTML files should have a **.html** extension (**.htm** is allowed).

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

Differences Between .htm and .html?

There is no difference between the .htm and .html file extensions!

Both will be treated as HTML by any web browser and web server.

Default Filenames

When a URL does not specify a filename at the end (like "https://www.w3schools.com/"), the server just adds a default filename, such as "index.html", "index.htm", "default.html", or "default.htm".

If your server is configured only with "index.html" as the default filename, your file must be named "index.html", and not "default.html".

However, servers can be configured with more than one default filename; usually you can set up as many default filenames as you want.

HTML Entities

Reserved characters in HTML must be replaced with character 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 **<**

Advantage of using an entity name: An entity name is easy to remember.
Disadvantage of using an entity name: Browsers may not support all entity names, but the support for entity numbers is good.

Non-breaking Space

A commonly used entity in HTML is the non-breaking space: ** **

A non-breaking space is a space that will not break into a new line.

Two words separated by a non-breaking space will stick together (not break into a new line). This is handy when breaking the words might be disruptive.

Examples:

- Â§ 10
- 10 km/h
- 10 PM

Another common use of the non-breaking space is to prevent browsers from truncating 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.

Tip: The non-breaking hyphen (‑) is used to define a hyphen character (â€‘) that does not break into a new line.

Some Useful HTML Character Entities

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
"	double quotation mark	"	"
'	single quotation mark (apostrophe)	'	'
Â¢	cent	¢	¢
Â£	pound	£	£
Â¥	yen	¥	¥
â‚¬	euro	€	€
Â©	copyright	©	©
Â®	registered trademark	®	®

Note: Entity names are case sensitive.

Combining Diacritical Marks

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

Some diacritical marks, like grave (Â Ì€) and acute (Â Ì) 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
Â Ì€	a	à	aÌ€
Â Ì	a	á	aÌ
Ì‚	a	â	aÌ‚
Â Ìƒ	a	ã	aÌƒ
Â Ì€	O	Ò	OÌ€
Â Ì	O	Ó	OÌ
Ì‚	O	Ô	OÌ‚
Â Ìƒ	O	Õ	OÌƒ

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

HTML Symbols

Symbols that are not present on your keyboard can also be added by using entities.

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 such symbols to an HTML page, you can use the entity name or the entity number (a decimal or a hexadecimal reference) for the symbol.

Example

Display the euro sign, â‚¬, with an entity name, a decimal, and a hexadecimal value:

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

Will display as:

I will display â‚¬
I will display â‚¬
I will display â‚¬

Some Mathematical Symbols Supported by HTML

Char	Number	Entity	Description
â^€	&#8704;	&forall;	FOR ALL
â^‚	&#8706;	&part;	PARTIAL DIFFERENTIAL
â^ƒ	&#8707;	&exist;	THERE EXISTS
â^…	&#8709;	&empty;	EMPTY SETS
â^‡	&#8711;	&nabla;	NABLA
â^ˆ	&#8712;	&isin;	ELEMENT OF
â^‰	&#8713;	&notin;	NOT AN ELEMENT OF
â^‹	&#8715;	&ni;	CONTAINS AS MEMBER
â^	&#8719;	&prod;	N-ARY PRODUCT
â^‘	&#8721;	&sum;	N-ARY SUMMATION

Full Math Reference

Some Greek Letters Supported by HTML

Char	Number	Entity	Description
Î‘	Α	Α	GREEK CAPITAL LETTER ALPHA
Î’	Β	Β	GREEK CAPITAL LETTER BETA
Î“	Γ	Γ	GREEK CAPITAL LETTER GAMMA
Î”	Δ	Δ	GREEK CAPITAL LETTER DELTA
Î•	Ε	Ε	GREEK CAPITAL LETTER EPSILON
Î–	Ζ	Ζ	GREEK CAPITAL LETTER ZETA

Full Greek Reference

Some Other Entities Supported by HTML

Char	Number	Entity	Description
Â©	©	©	COPYRIGHT SIGN
Â®	®	®	REGISTERED SIGN
â‚¬	€	€	EURO SIGN
â„¢	™	™	TRADEMARK
â†	←	←	LEFTWARDS ARROW
â†‘	↑	↑	UPWARDS ARROW
â†’	→	→	RIGHTWARDS ARROW
â†“	↓	↓	DOWNWARDS ARROW
â™	♠	♠	BLACK SPADE SUIT
â™£	♣	♣	BLACK CLUB SUIT
â™¥	♥	♥	BLACK HEART SUIT
â™¦	♦	♦	BLACK DIAMOND SUIT

Full Currency Reference

Full Arrows Reference

Full Symbols Reference

Using Emojis in HTML

Emojis are characters from the UTF-8 character set: ðŸ˜„ ðŸ˜ ðŸ’—

What are Emojis?

Emojis look like images, or icons, but they are not.

They are letters (characters) from the UTF-8 (Unicode) character set.

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

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:

<meta charset="UTF-8">

If not specified, UTF-8 is the default character set in HTML.

UTF-8 Characters

Many UTF-8 characters cannot be typed on a keyboard, but they can always be displayed using numbers (called entity

numbers):

- A is 65
- B is 66
- C is 67

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<p>I will display A B C</p>
<p>I will display &#65; &#66; &#67;</p>

</body>
</html>
```

Example Explained

The `<meta charset="UTF-8">` element defines the character set.

The characters A, B, and C, are displayed by the numbers 65, 66, and 67.

To let the browser understand that you are displaying a character, you must start the entity number with &# and end it with ; (semicolon).

Emoji Characters

Emojis are also characters from the UTF-8 alphabet:

- ðŸ˜„ is 128516
- ðŸ˜Ž is 128525
- ðŸ’— is 128151

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<h1>My First Emoji</h1>

<p>&#128512;</p>

</body>
</html>
```

Since Emojis are characters, they can be copied, displayed, and sized just like any other character in HTML.

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<h1>Sized Emojis</h1>

<p style="font-size:48px">
&#128512; &#128516; &#128525; &#128151;
</p>

</body>
</html>
```

Some Emoji Symbols in UTF-8

Emoji	Value
ðŸ—»	🗻
ðŸ—¼	🗼
ðŸ—½	🗽
ðŸ—¾	🗾
ðŸ—¿	🗿
ðŸ˜€	😀
ðŸ˜	😁
ðŸ˜‚	😂
ðŸ˜ƒ	😃
ðŸ˜„	😄
ðŸ˜…	😅

For a full list, please go to our HTML Emoji Reference.

HTML Encoding (Character Sets)

To display an HTML page correctly, a web browser must know which character set to use.

From ASCII to UTF-8

ASCII was the first character encoding standard. ASCII defined 128 different characters that could be used on the internet: numbers (0-9), English letters (A-Z), and some special characters like ! $ + - ( ) @ < > .

ISO-8859-1 was the default character set for HTML 4. This character set supported 256 different character codes. HTML 4 also supported UTF-8.

ANSI (Windows-1252) was the original Windows character set. ANSI is identical to ISO-8859-1, except that ANSI has 32 extra characters.

The HTML5 specification encourages web developers to use the UTF-8 character set, which covers almost all of the characters and symbols in the world!

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:

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

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	Â	Â	Â	Â
128	Â	â‚¬	Â	Â	euro sign
129	Â	Â	Â	Â	NOT USED
130	Â	â€š	Â	Â	single low-9 quotation mark
131	Â	Æ’	Â	Â	Latin small letter f with hook
132	Â	â€ž	Â	Â	double low-9 quotation mark
133	Â	â€¦	Â	Â	horizontal ellipsis
134	Â	â€	Â	Â	dagger
135	Â	â€¡	Â	Â	double dagger
136	Â	Ë†	Â	Â	modifier letter circumflex accent
137	Â	â€°	Â	Â	per mille sign
138	Â	Å	Â	Â	Latin capital letter S with caron
139	Â	â€¹	Â	Â	single left-pointing angle quotation mark
140	Â	Å’	Â	Â	Latin capital ligature OE
141	Â	Â	Â	Â	NOT USED
142	Â	Å½	Â	Â	Latin capital letter Z with caron
143	Â	Â	Â	Â	NOT USED
144	Â	Â	Â	Â	NOT USED
145	Â	â€˜	Â	Â	left single quotation mark
146	Â	â€™	Â	Â	right single quotation mark
147	Â	â€œ	Â	Â	left double quotation mark
148	Â	â€	Â	Â	right double quotation mark
149	Â	â€¢	Â	Â	bullet
150	Â	â€“	Â	Â	en dash
151	Â	â€”	Â	Â	em dash
152	Â	Ëœ	Â	Â	small tilde
153	Â	â„¢	Â	Â	trade mark sign
154	Â	Å¡	Â	Â	Latin small letter s with caron
155	Â	â€º	Â	Â	single right-pointing angle quotation mark
156	Â	Å“	Â	Â	Latin small ligature oe
157	Â	Â	Â	Â	NOT USED
158	Â	Å¾	Â	Â	Latin small letter z with caron
159	Â	Å¸	Â	Â	Latin capital letter Y with diaeresis
160	Â	Â	Â	Â	no-break space
161	Â	Â¡	Â¡	Â¡	inverted exclamation mark
162	Â	Â¢	Â¢	Â¢	cent sign
163	Â	Â£	Â£	Â£	pound sign
164	Â	Â¤	Â¤	Â¤	currency sign
165	Â	Â¥	Â¥	Â¥	yen sign
166	Â	Â¦	Â¦	Â¦	broken bar
167	Â	Â§	Â§	Â§	section sign

168	Â	Â¨	Â¨	Â¨	diaeresis
169	Â	Â©	Â©	Â©	copyright sign
170	Â	Âª	Âª	Âª	feminine ordinal indicator
171	Â	Â«	Â«	Â«	left-pointing double angle quotation mark
172	Â	Â¬	Â¬	Â¬	not sign
173	Â	Â	Â	Â	soft hyphen
174	Â	Â®	Â®	Â®	registered sign
175	Â	Â¯	Â¯	Â¯	macron
176	Â	Â°	Â°	Â°	degree sign
177	Â	Â±	Â±	Â±	plus-minus sign
178	Â	Â²	Â²	Â²	superscript two
179	Â	Â³	Â³	Â³	superscript three
180	Â	Â´	Â´	Â´	acute accent
181	Â	Âµ	Âµ	Âµ	micro sign
182	Â	Â¶	Â¶	Â¶	pilcrow sign
183	Â	Â·	Â·	Â·	middle dot
184	Â	Â¸	Â¸	Â¸	cedilla
185	Â	Â¹	Â¹	Â¹	superscript one
186	Â	Âº	Âº	Âº	masculine ordinal indicator
187	Â	Â»	Â»	Â»	right-pointing double angle quotation mark
188	Â	Â¼	Â¼	Â¼	vulgar fraction one quarter
189	Â	Â½	Â½	Â½	vulgar fraction one half
190	Â	Â¾	Â¾	Â¾	vulgar fraction three quarters
191	Â	Â¿	Â¿	Â¿	inverted question mark
192	Â	Ã€	Ã€	Ã€	Latin capital letter A with grave
193	Â	Ã	Ã	Ã	Latin capital letter A with acute
194	Â	Ã‚	Ã‚	Ã‚	Latin capital letter A with circumflex
195	Â	Ãƒ	Ãƒ	Ãƒ	Latin capital letter A with tilde
196	Â	Ã„	Ã„	Ã„	Latin capital letter A with diaeresis
197	Â	Ã…	Ã…	Ã…	Latin capital letter A with ring above
198	Â	Ã†	Ã†	Ã†	Latin capital letter AE
199	Â	Ã‡	Ã‡	Ã‡	Latin capital letter C with cedilla
200	Â	Ãˆ	Ãˆ	Ãˆ	Latin capital letter E with grave
201	Â	Ã‰	Ã‰	Ã‰	Latin capital letter E with acute
202	Â	ÃŠ	ÃŠ	ÃŠ	Latin capital letter E with circumflex
203	Â	Ã‹	Ã‹	Ã‹	Latin capital letter E with diaeresis
204	Â	ÃŒ	ÃŒ	ÃŒ	Latin capital letter I with grave
205	Â	Ã	Ã	Ã	Latin capital letter I with acute
206	Â	ÃŽ	ÃŽ	ÃŽ	Latin capital letter I with circumflex
207	Â	Ã	Ã	Ã	Latin capital letter I with diaeresis
208	Â	Ã	Ã	Ã	Latin capital letter Eth
209	Â	Ã‘	Ã‘	Ã‘	Latin capital letter N with tilde
210	Â	Ã’	Ã’	Ã’	Latin capital letter O with grave
211	Â	Ã“	Ã“	Ã“	Latin capital letter O with acute
212	Â	Ã”	Ã”	Ã”	Latin capital letter O with circumflex
213	Â	Ã•	Ã•	Ã•	Latin capital letter O with tilde
214	Â	Ã–	Ã–	Ã–	Latin capital letter O with diaeresis
215	Â	Ã—	Ã—	Ã—	multiplication sign
216	Â	Ã˜	Ã˜	Ã˜	Latin capital letter O with stroke
217	Â	Ã™	Ã™	Ã™	Latin capital letter U with grave
218	Â	Ãš	Ãš	Ãš	Latin capital letter U with acute
219	Â	Ã›	Ã›	Ã›	Latin capital letter U with circumflex
220	Â	Ãœ	Ãœ	Ãœ	Latin capital letter U with diaeresis
221	Â	Ã	Ã	Ã	Latin capital letter Y with acute
222	Â	Ãž	Ãž	Ãž	Latin capital letter Thorn
223	Â	ÃŸ	ÃŸ	ÃŸ	Latin small letter sharp s
224	Â	Ã	Ã	Ã	Latin small letter a with grave
225	Â	Ã¡	Ã¡	Ã¡	Latin small letter a with acute
226	Â	Ã¢	Ã¢	Ã¢	Latin small letter a with circumflex
227	Â	Ã£	Ã£	Ã£	Latin small letter a with tilde
228	Â	Ã¤	Ã¤	Ã¤	Latin small letter a with diaeresis
229	Â	Ã¥	Ã¥	Ã¥	Latin small letter a with ring above

230	Â	Ã¦	Ã¦	Ã¦	Latin small letter ae
231	Â	Ã§	Ã§	Ã§	Latin small letter c with cedilla
232	Â	Ã¨	Ã¨	Ã¨	Latin small letter e with grave
233	Â	Ã©	Ã©	Ã©	Latin small letter e with acute
234	Â	Ãª	Ãª	Ãª	Latin small letter e with circumflex
235	Â	Ã«	Ã«	Ã«	Latin small letter e with diaeresis
236	Â	Ã¬	Ã¬	Ã¬	Latin small letter i with grave
237	Â	Ã	Ã	Ã	Latin small letter i with acute
238	Â	Ã®	Ã®	Ã®	Latin small letter i with circumflex
239	Â	Ã¯	Ã¯	Ã¯	Latin small letter i with diaeresis
240	Â	Ã°	Ã°	Ã°	Latin small letter eth
241	Â	Ã±	Ã±	Ã±	Latin small letter n with tilde
242	Â	Ã²	Ã²	Ã²	Latin small letter o with grave
243	Â	Ã³	Ã³	Ã³	Latin small letter o with acute
244	Â	Ã´	Ã´	Ã´	Latin small letter o with circumflex
245	Â	Ãµ	Ãµ	Ãµ	Latin small letter o with tilde
246	Â	Ã¶	Ã¶	Ã¶	Latin small letter o with diaeresis
247	Â	Ã·	Ã·	Ã·	division sign
248	Â	Ã¸	Ã¸	Ã¸	Latin small letter o with stroke
249	Â	Ã¹	Ã¹	Ã¹	Latin small letter u with grave
250	Â	Ãº	Ãº	Ãº	Latin small letter u with acute
251	Â	Ã»	Ã»	Ã»	Latin small letter with circumflex
252	Â	Ã¼	Ã¼	Ã¼	Latin small letter u with diaeresis
253	Â	Ã½	Ã½	Ã½	Latin small letter y with acute
254	Â	Ã¾	Ã¾	Ã¾	Latin small letter thorn
255	Â	Ã¿	Ã¿	Ã¿	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

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

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

ISO-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.Â

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.

HTML Uniform Resource Locators

A URL is another word for a web address.

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

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

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

scheme://prefix.domain:port/path/filename

Explanation:

- **scheme** - defines the **type** of Internet service (most common is **http or https**)
- **prefix** - defines a domain **prefix** (default for http is **www**)
- **domain** - defines the Internet **domain name** (like 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	Â	A file on your computer

URL Encoding

URLs can only be sent over the Internet using the ASCII character-set. If a URL contains characters outside the ASCII set, the URL has to be converted.

URL encoding converts non-ASCII 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 GÃƒÂ¼nter 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

Ã¢Â‚Â¬	%80	%E2%82%AC
Ã‚Â£	%A3	%C2%A3
Ã‚Â©	%A9	%C2%A9
Ã‚Â®	%AE	%C2%AE
ÃƒÂ€	%C0	%C3%80
ÃƒÂ	%C1	%C3%81
ÃƒÂ‚	%C2	%C3%82
ÃƒÂƒ	%C3	%C3%83
ÃƒÂ„	%C4	%C3%84
ÃƒÂ…	%C5	%C3%85

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

HTML Versus XHTML

XHTML is a stricter, more XML-based version of HTML.

What is XHTML?

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is a stricter, more XML-based version of HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Why XHTML?

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

XHTML was developed to make HTML more extensible and flexible to work with other data formats (such as XML). In addition, browsers ignore errors in HTML pages, and try to display the website even if it has some errors in the markup. So XHTML comes with a much stricter error handling.

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

The Most Important Differences from HTML

- <!DOCTYPE> is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**
- Elements must always be **properly nested**
- Elements must always be **closed**
- Elements must always be in **lowercase**
- Attribute names must always be in **lowercase**
- Attribute values must always be **quoted**
- Attribute minimization is **forbidden**

XHTML - <!DOCTYPE ....> Is Mandatory

An XHTML document must have an XHTML <!DOCTYPE> declaration.

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.

Example

Here is an XHTML document with a minimum of required tags:Â

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

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

Â *some content here...*

</body>
</html>

XHTML Elements Must be Properly Nested

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

Correct:

Some text

Wrong:

Some text

XHTML Elements Must Always be Closed

In XHTML, elements must always be closed, like this:

Correct:

This is a paragraph
This is another paragraph

Wrong:

This is a paragraph
This is another paragraph

XHTML Empty Elements Must Always be Closed

In XHTML, empty elements must always be closed, like this:

Correct:

A break:
A horizontal rule: <hr />
An image:

Wrong:

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

XHTML Elements Must be in Lowercase

In XHTML, element names must always be in lowercase, like this:

Correct:

<body>
This is a paragraph
</body>

Wrong:

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

XHTML Attribute Names Must be in Lowercase

In XHTML, attribute names must always be in lowercase, like this:

Correct:

Visit our HTML tutorial

Wrong:

<a HREF="https://www.w3schools.com/html/">Visit our HTML tutorial

---

XHTML Attribute Values Must be Quoted

In XHTML, attribute values must always be quoted, like this:

Correct:

Visit our HTML tutorial

Wrong:

Visit our HTML tutorial

XHTML Attribute Minimization is Forbidden

In XHTML, attribute minimization is forbidden:

Correct:

<input type="checkbox" name="vehicle" value="car" checked="checked" />
<input type="text" name="lastname" disabled="disabled" />

Wrong:

<input type="checkbox" name="vehicle" value="car" checked />
<input type="text" name="lastname" disabled />

Validate HTML With The W3C Validator

Put your web address in the box below:

https://www.w3schools.com/html/html_validate.html

Validate the page

HTML Forms

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

Example

First name:
John
Last name:
Doe

Submit

The <form> Element

The HTML `<form>` element is used to create an HTML form for user input:

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

The `<form>` element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

All the different form elements are covered in this chapter: HTML Form Elements.

The <input> Element

The HTML `<input>` element is the most used form element.

An `<input>` element can be displayed in many ways, depending on the `type` attribute.

Here are some examples:

Type	Description
<input type="text">	Displays a single-line text input field
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices)
<input type="submit">	Displays a submit button (for submitting the form)
<input type="button">	Displays a clickable button

All the different input types are covered in this chapter: HTML Input Types.

Text Fields

The `<input type="text">` defines a single-line input field for text input.

Example

A form with input fields for text:

```
<form>
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname"><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname">
</form>
```

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

First name:

Last name:

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

The <label> Element

Notice the use of the `<label>` element in the example above.

The `<label>` tag defines a label for many form elements.

The `<label>` element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

The `<label>` element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the `<label>` element, it toggles the radio button/checkbox.

The `for` attribute of the `<label>` tag should be equal to the `id` attribute of the `<input>` element to bind them together.

Radio Buttons

The `<input type="radio">` defines a radio button.

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

Example

A form with radio buttons:

```
<form>
Â  <input type="radio" id="male" name="gender" value="male">
Â  <label for="male">Male</label><br>
Â  <input type="radio" id="female" name="gender" value="female">
Â  <label for="female">Female</label><br>
Â  <input type="radio" id="other" name="gender" value="other">
Â  <label for="other">Other</label>
</form>
```

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

◯ Male
◯ Female
◯ Other

Checkboxes

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

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

Example

A form with checkboxes:

```
<form>
Â  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
Â  <label for="vehicle1"> I have a bike</label><br>
Â  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
Â  <label for="vehicle2"> I have a car</label><br>
Â  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
Â  <label for="vehicle3"> I have a boat</label>
</form>
```

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

☐ I have a bike
☐ I have a car
☐ I have a boat

The Submit Button

The `<input type="submit">` defines a button for submitting the form data to a form-handler.

The form-handler is typically a file on the server with a script for processing input data.

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

Example

A form with a submit button:

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname" value="John"><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname" value="Doe"><br><br>
Â  <input type="submit" value="Submit">
</form>
```

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

First name:

John
Last name:
Doe

Submit

The Name Attribute for <input>

Notice that each input field must have a `name` attribute to be submitted.

If the `name` attribute is omitted, the value of the input field will not be sent at all.

Example

This example will not submit the value of the "First name" input field:Â

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" value="John"><br><br>
Â  <input type="submit" value="Submit">
</form>
```

HTML Exercises

HTML Form Attributes

This chapter describes the different attributes for the HTML `<form>` element.

The Action Attribute

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

Usually, the form data is sent to a file on the server when the user clicks on the submit button.

In the example below, the form data is sent to a file called "action_page.php". This file contains a server-side script that handles the form data:

Example

On submit, send form data to "action_page.php":

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname" value="John"><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname" value="Doe"><br><br>
Â  <input type="submit" value="Submit">
</form>
```

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

The Target Attribute

The `target` attribute specifies where to display the response that is received after submitting the form.

The `target` attribute can have one of the following values:

Value	Description
_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window

framename The response is displayed in a named iframe

The default value is `_self` which means that the response will open in the current window.

Example

Here, the submitted result will open in a new browser tab:

<form action="/action_page.php" target="_blank">

The Method Attribute

The `method` attribute specifies the HTTP method to be used when submitting the form data.

The form-data can be sent as URL variables (with `method="get"`) or as HTTP post transaction (with `method="post"`).

The default HTTP method when submitting form data is GET.Â

Example

This example uses the GET method when submitting the form data:

<form action="/action_page.php" method="get">

Example

This example uses the POST method when submitting the form data:

<form action="/action_page.php" method="post">

Notes on GET:

- Appends the form data to the URL, in name/value pairs
- NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
- The length of a URL is limited (2048 characters)
- Useful for form submissions where a user wants to bookmark the result
- GET is good for non-secure data, like query strings in Google

Notes on POST:

- Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

Tip: Always use POST if the form data contains sensitive or personal information!

The Autocomplete Attribute

The `autocomplete` attribute specifies whether a form should have autocomplete on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

Example

A form with autocomplete on:

<form action="/action_page.php" autocomplete="on">

The Novalidate Attribute

The `novalidate` attribute is a boolean attribute.

When present, it specifies that the form-data (input) should not be validated when submitted.

Example

A form with a novalidate attribute:

<form action="/action_page.php" novalidate>

HTML Exercises

List of All <form> Attributes

Attribute	Description
accept-charset	Specifies the character encodings used for form submission
action	Specifies where to send the form-data when a form is submitted
autocomplete	Specifies whether a form should have autocomplete on or off
enctype	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
method	Specifies the HTTP method to use when sending form-data
name	Specifies the name of the form
novalidate	Specifies that the form should not be validated when submitted
rel	Specifies the relationship between a linked resource and the current document
target	Specifies where to display the response that is received after submitting the form

HTML Form Elements

This chapter describes all the different HTML form elements.

The HTML <form> Elements

The HTML `<form>` element can contain one or more of the following form elements:

- `<input>`
- `<label>`
- `<select>`
- `<textarea>`
- `<button>`
- `<fieldset>`
- `<legend>`
- `<datalist>`
- `<output>`
- `<option>`
- `<optgroup>`

The <input> Element

One of the most used form element is the `<input>` element.

The `<input>` element can be displayed in several ways, depending on the `type` attribute.

Example

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
```

All the different values of the `type` attribute are covered in the next chapter: HTML Input Types.

The <label> Element

The `<label>` element defines a label for several form elements.

The `<label>` element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

The `<label>` element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the `<label>` element, it toggles the radio button/checkbox.

The `for` attribute of the `<label>` tag should be equal to the `id` attribute of the `<input>` element to bind them together.

The <select> Element

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

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars">
Â  <option value="volvo">Volvo</option>
Â  <option value="saab">Saab</option>
Â  <option value="fiat">Fiat</option>
Â  <option value="audi">Audi</option>
</select>
```

The `<option>` elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

To define a pre-selected option, add the `selected` attribute to the option:

Example

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

Visible Values:

Use the `size` attribute to specify the number of visible values:

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="3">
Â  <option value="volvo">Volvo</option>
Â  <option value="saab">Saab</option>
Â  <option value="fiat">Fiat</option>
Â  <option value="audi">Audi</option>
</select>
```

Allow Multiple Selections:

Use the `multiple` attribute to allow the user to select more than one value:

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="4" multiple>
Â  <option value="volvo">Volvo</option>
Â  <option value="saab">Saab</option>
Â  <option value="fiat">Fiat</option>
Â  <option value="audi">Audi</option>
</select>
```

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>
```

The `rows` attribute specifies the visible number of lines in a text area.

The `cols` attribute specifies the visible width of a text area.

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

The cat was playing in the garden.

You can also define the size of the text area by using CSS:

Example

```
<textarea name="message" style="width:200px; height:600px;">
The cat was playing in the garden.
</textarea>
```

The <button> Element

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

Example

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

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

Click Me!

Note: Always specify the `type` attribute for the button element. Different browsers may use different default types for the button element.

The <fieldset> and <legend> Elements

The `<fieldset>` element is used to group related data in a form.

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

Example

```
<form action="/action_page.php">
Â  <fieldset>
Â Â Â  <legend>Personalia:</legend>
Â Â Â  <label for="fname">First name:</label><br>
Â Â Â  <input type="text" id="fname" name="fname" value="John"><br>
Â Â Â  <label for="lname">Last name:</label><br>
Â Â Â  <input type="text" id="lname" name="lname" value="Doe"><br><br>
Â Â Â  <input type="submit" value="Submit">
Â  </fieldset>
</form>
```

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

Personalia:

First name:
John

Last name:
Doe

Submit

The <datalist> Element

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

Users will see a drop-down list of the 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

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

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

HTML Exercises

HTML Form Elements

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
<output>	Defines the result of a calculation

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Input Types

This chapter describes the different types for the HTML `<input>` element.

HTML Input Types

Here are the different input types you can use in HTML:

- `<input type="button">`
- `<input type="checkbox">`
- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="hidden">`
- `<input type="image">`
- `<input type="month">`
- `<input type="number">`
- `<input type="password">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="reset">`
- `<input type="search">`
- `<input type="submit">`
- `<input type="tel">`
- `<input type="text">`
- `<input type="time">`
- `<input type="url">`
- `<input type="week">`

Tip: The default value of the `type` attribute is "text".

Input Type Text

`<input type="text">` defines a **single-line text input field**:

Example

<form>
Â <label for="fname">First name:</label>
Â <input type="text" id="fname" name="fname">
Â <label for="lname">Last name:</label>
Â <input type="text" id="lname" name="lname">
</form>

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>
Â <label for="username">Username:</label>
Â <input type="text" id="username" name="username">
Â <label for="pwd">Password:</label>
Â <input type="password" id="pwd" name="pwd">
</form>

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

Username:

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 data 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">
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname" value="John"><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname" value="Doe"><br><br>
Â  <input type="submit" value="Submit">
</form>
```

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

First name:
John
Last name:
Doe

Submit

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

Example

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname" value="John"><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname" value="Doe"><br><br>
Â  <input type="submit">
</form>
```

Input Type Reset

`<input type="reset">` defines a **reset button** that will reset all form values to their default values:

Example

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname" value="John"><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname" value="Doe"><br><br>
Â  <input type="submit" value="Submit">
Â  <input type="reset">
</form>
```

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

First name:
John
Last name:
Doe

Submit Reset

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

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>
Â  <input type="radio" id="male" name="gender" value="male">
Â  <label for="male">Male</label><br>
Â  <input type="radio" id="female" name="gender" value="female">
Â  <label for="female">Female</label><br>
Â  <input type="radio" id="other" name="gender" value="other">
Â  <label for="other">Other</label>
</form>
```

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

◯ Male
◯ Female
◯ Other

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>
Â  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
Â  <label for="vehicle1"> I have a bike</label><br>
Â  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
Â  <label for="vehicle2"> I have a car</label><br>
Â  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
Â  <label for="vehicle3"> I have a boat</label>
</form>
```

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

☐ I have a bike
☐ I have a car
☐ I have a boat

Input Type Button

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

Example

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

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

Click Me!

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>
Â  <label for="favcolor">Select your favorite color:</label>
Â  <input type="color" id="favcolor" name="favcolor">
</form>
```

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>
Â  <label for="birthday">Birthday:</label>
Â  <input type="date" id="birthday" name="birthday">
</form>
```

You can also use the `min` and `max` attributes to add restrictions to dates:

Example

```
<form>
Â  <label for="datemax">Enter a date before 1980-01-01:</label>
Â  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
Â  <label for="datemin">Enter a date after 2000-01-01:</label>
Â  <input type="date" id="datemin" name="datemin" min="2000-01-02">
</form>
```

Input Type Datetime-local

The `<input type="datetime-local">` specifies a date and time input field, with no time zone.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
Â  <label for="birthdaytime">Birthday (date and time):</label>
Â  <input type="datetime-local" id="birthdaytime" name="birthdaytime">
</form>
```

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 add ".com" to the keyboard to match email input.

Example

```
<form>
Â  <label for="email">Enter your email:</label>
Â  <input type="email" id="email" name="email">
</form>
```

Input Type File

The `<input type="file">` defines a file-select field and a "Browse" button for file uploads.

Example

```
<form>
Â  <label for="myfile">Select a file:</label>
Â  <input type="file" id="myfile" name="myfile">
</form>
```

Input Type Hidden

The `<input type="hidden">` defines a hidden input field (not visible to a user).

A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.

A hidden field often stores what database record that needs to be updated when the form is submitted.

Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

Example

```
<form>
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <input type="hidden" id="custId" name="custId" value="3487">
Â  <input type="submit" value="Submit">
</form>
```

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>
Â  <label for="bdaymonth">Birthday (month and year):</label>
Â  <input type="month" id="bdaymonth" name="bdaymonth">
</form>
```

Input Type Number

The `<input type="number">` defines a **numeric** input field.

You can also set restrictions on what numbers are accepted.

The following example displays a numeric input field, where you can enter a value from 1 to 5:

Example

```
<form>
Â  <label for="quantity">Quantity (between 1 and 5):</label>
Â  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

Input Restrictions

Here is a list of some common input restrictions:

Attribute	Description
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
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.

The following example displays a numeric input field, where you can enter a value from 0 to 100, in steps of 10. The default value is 30:

Example

```
<form>
Â  <label for="quantity">Quantity:</label>
Â  <input type="number" id="quantity" name="quantity" min="0" max="100" step="10" value="30">
</form>
```

Input Type Range

The `<input type="range">` defines a control for entering a number whose exact value is not important (like a slider control).

Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the `min`, `max`, and `step` attributes:

Example

```
<form>
Â  <label for="vol">Volume (between 0 and 50):</label>
Â  <input type="range" id="vol" name="vol" min="0" max="50">
</form>
```

Input Type Search

The `<input type="search">` is used for search fields (a search field behaves like a regular text field).

Example

```
<form>
Â  <label for="gsearch">Search Google:</label>
Â  <input type="search" id="gsearch" name="gsearch">
</form>
```

Input Type Tel

The `<input type="tel">` is used for input fields that should contain a telephone number.

Example

```
<form>
Â  <label for="phone">Enter your phone number:</label>
Â  <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

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>
Â  <label for="appt">Select a time:</label>
Â  <input type="time" id="appt" name="appt">
</form>
```

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>
Â  <label for="homepage">Add your homepage:</label>
Â  <input type="url" id="homepage" name="homepage">
</form>
```

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>
```

Â <label for="week">Select a week:</label>
Â <input type="week" id="week" name="week">
</form>

HTML Exercises

HTML Input Type Attribute

Tag	Description
<input type="">	Specifies the input type to display

HTML Input Attributes

This chapter describes the different attributes for the HTML `<input>` element.

The value Attribute

The input `value` attribute specifies an initial value for an input field:

Example

Input fields with initial (default) values:

<form>
Â <label for="fname">First name:</label>
Â <input type="text" id="fname" name="fname" value="John">
Â <label for="lname">Last name:</label>
Â <input type="text" id="lname" name="lname" value="Doe">
</form>

The readonly Attribute

The input `readonly` attribute specifies that an input field is read-only.

A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).

The value of a read-only input field will be sent when submitting the form!

Example

A read-only input field:

<form>
Â <label for="fname">First name:</label>
Â <input type="text" id="fname" name="fname" value="John" readonly>
Â <label for="lname">Last name:</label>
Â <input type="text" id="lname" name="lname" value="Doe">
</form>

The disabled Attribute

The input `disabled` attribute specifies that an input field should be disabled.

A disabled input field is unusable and un-clickable.

The value of a disabled input field will not be sent when submitting the form!

Example

A disabled input field:

<form>
Â <label for="fname">First name:</label>
Â <input type="text" id="fname" name="fname" value="John" disabled>

Â <label for="lname">Last name:</label>
Â <input type="text" id="lname" name="lname" value="Doe">
</form>

The size Attribute

The input `size` attribute specifies the visible width, in characters, of an input field.

The default value for `size` is 20.

Note: The `size` attribute works with the following input types: text, search, tel, url, email, and password.

Example

Set a width for an input field:

<form>
Â <label for="fname">First name:</label>
Â <input type="text" id="fname" name="fname" size="50">
Â <label for="pin">PIN:</label>
Â <input type="text" id="pin" name="pin" size="4">
</form>

The maxlength Attribute

The input `maxlength` attribute specifies the maximum number of characters allowed in an input field.

Note: When a `maxlength` is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.

Example

Set a maximum length for an input field:

<form>
Â <label for="fname">First name:</label>
Â <input type="text" id="fname" name="fname" size="50">
Â <label for="pin">PIN:</label>
Â <input type="text" id="pin" name="pin" maxlength="4" size="4">
</form>

The min and max Attributes

The input `min` and `max` attributes specify the minimum and maximum values for an input field.

The `min` and `max` attributes work with the following input types: number, range, date, datetime-local, month, time and week.

Tip: Use the max and min attributes together to create a range of legal values.

Example

Set a max date, a min date, and a range of legal values:

<form>
Â <label for="datemax">Enter a date before 1980-01-01:</label>
Â <input type="date" id="datemax" name="datemax" max="1979-12-31">

Â <label for="datemin">Enter a date after 2000-01-01:</label>
Â <input type="date" id="datemin" name="datemin" min="2000-01-02">

Â <label for="quantity">Quantity (between 1 and 5):</label>
Â <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>

The multiple Attribute

The input `multiple` attribute specifies that the user is allowed to enter more than one value in an input field.

The `multiple` attribute works with the following input types: email, and file.

Example

A file upload field that accepts multiple values:

```
<form>
Â  <label for="files">Select files:</label>
Â  <input type="file" id="files" name="files" multiple>
</form>
```

The pattern Attribute

The input `pattern` attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.

The `pattern` attribute works with the following input types: text, date, 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):

```
<form>
Â  <label for="country_code">Country code:</label>
Â  <input type="text" id="country_code" name="country_code"
Â  pattern="[A-Za-z]{3}" title="Three letter country code">
</form>
```

The placeholder Attribute

The input `placeholder` attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format).

The short 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:

```
<form>
Â  <label for="phone">Enter a phone number:</label>
Â  <input type="tel" id="phone" name="phone"
Â  placeholder="123-45-678"
Â  pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

The required Attribute

The input `required` attribute 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:

```
<form>
Â  <label for="username">Username:</label>
Â  <input type="text" id="username" name="username" required>
</form>
```

The step Attribute

The input `step` attribute specifies the legal number intervals for an input field.

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

Tip: This 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-local, month, time and week.

Example

An input field with a specified legal number intervals:

```
<form>
Â  <label for="points">Points:</label>
Â  <input type="number" id="points" name="points" step="3">
</form>
```

Note: Input restrictions are not foolproof, and JavaScript provides many ways to add illegal input. To safely restrict input, it must also be checked by the receiver (the server)!

The autofocus Attribute

The input `autofocus` attribute specifies that an input field should automatically get focus when the page loads.

Example

Let the "First name" input field automatically get focus when the page loads:

```
<form>
Â  <label for="fname">First name:</label><br>
Â  <input type="text" id="fname" name="fname" autofocus><br>
Â  <label for="lname">Last name:</label><br>
Â  <input type="text" id="lname" name="lname">
</form>
```

The height and width Attributes

The input `height` and `width` attributes specify the height and width of an `<input type="image">` element.

Tip: Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. Without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).

Example

Define an image as the submit button, with height and width attributes:

```
<form>
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <label for="lname">Last name:</label>
Â  <input type="text" id="lname" name="lname"><br><br>
Â  <input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```

The list Attribute

The input `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>:

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

The autocomplete Attribute

The input `autocomplete` attribute specifies whether a form or an input field should have autocomplete on or off.

Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.

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">
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <label for="lname">Last name:</label>
Â  <input type="text" id="lname" name="lname"><br><br>
Â  <label for="email">Email:</label>
Â  <input type="email" id="email" name="email" autocomplete="off"><br><br>
Â  <input type="submit" value="Submit">
</form>
```

Tip: In some browsers you may need to activate an autocomplete function for this to work (Look under "Preferences" in the browser's menu).

HTML Exercises

HTML Form and Input Elements

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Input form* Attributes

This chapter describes the different `form*` attributes for the HTML `<input>` element.

The form Attribute

The input `form` attribute specifies the form the `<input>` element belongs to.

The value of this attribute must be equal to the id attribute of the <form> element it belongs to.

Example

An input field located outside of the HTML form (but still a part of the form):

```
<form action="/action_page.php" id="form1">
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <input type="submit" value="Submit">
</form>

<label for="lname">Last name:</label>
<input type="text" id="lname" name="lname" form="form1">
```

The formaction Attribute

The input `formaction` attribute specifies the URL of the file that will process the input when the form is submitted.

Note: This attribute overrides the `action` attribute of the `<form>` element.

The `formaction` attribute works with the following input types: submit and image.

Example

An HTML form with two submit buttons, with different actions:

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <label for="lname">Last name:</label>
Â  <input type="text" id="lname" name="lname"><br><br>
Â  <input type="submit" value="Submit">
Â  <input type="submit" formaction="/action_page2.php" value="Submit as Admin">
</form>
```

The formenctype Attribute

The input `formenctype` attribute specifies how the form-data should be encoded when submitted (only for forms with method="post").

Note: This attribute overrides the enctype attribute of the `<form>` element.

The `formenctype` attribute works with the following input types: submit and image.

Example

A form with two submit buttons. The first sends the form-data with default encoding, the second sends the form-data encoded as "multipart/form-data":

```
<form action="/action_page_binary.asp" method="post">
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <input type="submit" value="Submit">
Â  <input type="submit" formenctype="multipart/form-data"
Â  value="Submit as Multipart/form-data">
</form>
```

The formmethod Attribute

The input `formmethod` attribute defines the HTTP method for sending form-data to the action URL.

Note: This attribute overrides the method attribute of the `<form>` element.

The `formmethod` attribute works with the following input types: submit and image.

The form-data can be sent as URL variables (method="get") or as an HTTP post transaction (method="post").

Notes on the "get" method:

- This method appends the form-data to the URL in name/value pairs
- This method is useful for form submissions where a user want to bookmark the result
- There is a limit to how much data you can place in a URL (varies between browsers), therefore, you cannot be sure that all of the form-data will be correctly transferred
- Never use the "get" method to pass sensitive information! (password or other sensitive information will be visible in the browser's address bar)

Notes on the "post" method:

- This method sends the form-data as an HTTP post transaction
- Form submissions with the "post" method cannot be bookmarked
- The "post" method is more robust and secure than "get", and "post" does not have size limitations

Example

A form with two submit buttons. The first sends the form-data with method="get". The second sends the form-data with method="post":

```
<form action="/action_page.php" method="get">
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <label for="lname">Last name:</label>
Â  <input type="text" id="lname" name="lname"><br><br>
```

Â <input type="submit" value="Submit using GET">
Â <input type="submit" formmethod="post" value="Submit using POST">
</form>

The formtarget Attribute

The input `formtarget` attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

Note: This attribute overrides the target attribute of the `<form>` element.

The `formtarget` attribute works with the following input types: submit and image.

Example

A form with two submit buttons, with different target windows:

```
<form action="/action_page.php">
Â  <label for="fname">First name:</label>
Â  <input type="text" id="fname" name="fname"><br><br>
Â  <label for="lname">Last name:</label>
Â  <input type="text" id="lname" name="lname"><br><br>
Â  <input type="submit" value="Submit">
Â  <input type="submit" formtarget="_blank" value="Submit to a new window/tab">
</form>
```

The formnovalidate Attribute

The input `formnovalidate` attribute specifies that an <input> element should not be validated when submitted.

Note: This attribute overrides the novalidate attribute of the `<form>` element.

The `formnovalidate` attribute works with the following input types: submit.

Example

A form with two submit buttons (with and without validation):

```
<form action="/action_page.php">
Â  <label for="email">Enter your email:</label>
Â  <input type="email" id="email" name="email"><br><br>
Â  <input type="submit" value="Submit">
Â  <input type="submit" formnovalidate="formnovalidate"
Â  value="Submit without validation">
</form>
```

The novalidate Attribute

The `novalidate` attribute is a `<form>` attribute.

When present, novalidate specifies that all of the form-data should not be validated when submitted.

Example

Specify that no form-data should be validated on submit:

```
<form action="/action_page.php" novalidate>
Â  <label for="email">Enter your email:</label>
Â  <input type="email" id="email" name="email"><br><br>
Â  <input type="submit" value="Submit">
</form>
```

HTML Form and Input Elements

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

For a complete list of all available HTML tags, visit our HTML Tag Reference.

HTML Canvas Graphics

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 JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript 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.

Here is an example of a basic, empty canvas:

Example

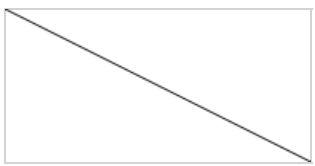
```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

Add a JavaScript

After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

Here are some examples:

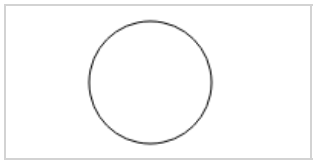
Draw a Line



Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
</script>
```

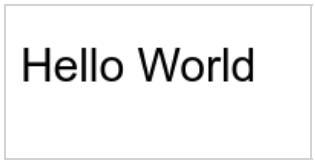
Draw a Circle



Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
</script>
```

Draw a Text



Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</script>
```

Stroke Text

Example

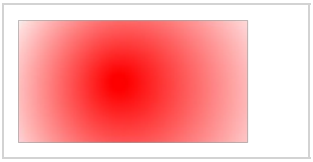
```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
</script>
```

Draw Linear Gradient

Example

```
<script>
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);
</script>
```

Draw Circular Gradient



Example

```
<script>
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);
</script>
```

Draw Image

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img, 10, 10);
</script>
```

HTML Canvas Tutorial

To learn more about <canvas>, please read our HTML Canvas Tutorial.

HTML SVG Graphics

SVG defines vector-based graphics in XML format.

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 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.

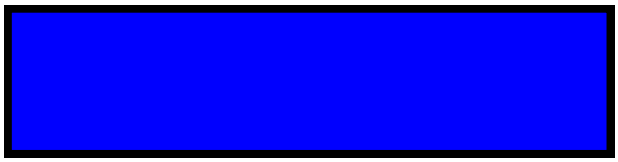
Element					
<svg>	4.0	9.0	3.0	3.2	10.1

SVG Circle

Example

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

<svg width="100" height="100">
Â  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

SVG Rectangle



Example

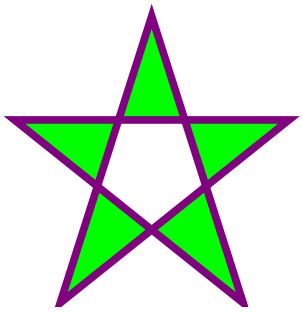
```
<svg width="400" height="100">
Â  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
</svg>
```

SVG Rounded Rectangle

Example

```
<svg width="400" height="180">
Â  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
Â  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>
```

SVG Star



Example

```
<svg width="300" height="200">
Â  <polygon points="100,10 40,198 190,78 10,78 160,198"
Â  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

SVG Logo

Example

```
<svg height="130" width="500">
Â  <defs>
Â Â Â  <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
Â Â Â Â Â  <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
Â Â Â Â Â  <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
Â Â Â  </linearGradient>
Â  </defs>
Â  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
Â  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86">SVG</text>
Â  Sorry, your browser does not support inline SVG.
</svg>
```

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

---

SVG Tutorial

To learn more about SVG, please read our SVG Tutorial.

HTML Multimedia

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, like images, music, sound, videos, records, films, animations, and more.

Web pages often contain multimedia elements of different types and 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, fonts, images, and multimedia!

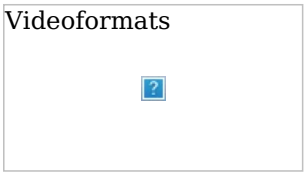
Multimedia Formats

Multimedia elements (like audio or video) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension.

Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Common Video Formats



There are many video formats out there.

The MP4, WebM, and Ogg formats are supported by HTML.

The MP4 format is recommended by YouTube.

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Not supported anymore in HTML.
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.
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. 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 HTML.
WebM	.webm	WebM. Developed by Mozilla, Opera, Adobe, and Google. Supported by HTML.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Commonly used in video cameras and TV hardware. Supported by all browsers andÂ recommended by YouTube.Â

Note: Only MP4, WebM, and Ogg video are supported by the HTML standard.

Common Audio Formats

MP3 is the best 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. 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 HTML.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
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. Supported by all browsers.

Note: Only MP3, WAV, and Ogg audio are supported by the HTML standard.

HTML Video

The HTML `<video>` element is used to show a video on a web page.

Example

Courtesy of Big Buck Bunny:

The HTML <video> Element

To show a video in HTML, use the `<video>` element:

Example

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

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 page might flicker while the video loads.

The `<source>` element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the `<video>` and `</video>` tags will only be displayed in browsers that do not support the `<video>` element.

HTML <video> Autoplay

To start a video automatically, use the `autoplay` attribute:

Example

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

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add `muted` after `autoplay` to let your video start playing automatically (but muted):

Example

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

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

HTML Video Formats

There are three supported video formats: MP4, WebM, and Ogg. The browser support for the different formats is:

Browser	MP4	WebM	Ogg
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	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

The HTML DOM defines 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

Video courtesy of Big Buck Bunny.

For a full DOM reference, go to our HTML Audio/Video DOM Reference.

HTML 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

HTML Audio

The HTML `<audio>` element is used to play an audio file on a web page.

The HTML <audio> Element

To play an audio file in HTML, use the `<audio>` element:

Example

```
<audio controls>
Â  <source src="horse.ogg" type="audio/ogg">
Â  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML Audio - How It Works

The `controls` attribute adds audio controls, like play, pause, and volume.

The `<source>` element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the `<audio>` and `</audio>` tags will only be displayed in browsers that do not support the `<audio>` element.

HTML <audio> Autoplay

To start an audio file automatically, use the `autoplay` attribute:

Example

```
<audio controls autoplay>
Â  <source src="horse.ogg" type="audio/ogg">
Â  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add `muted` after `autoplay` to let your audio file start playing automatically (but muted):

Example

```
<audio controls autoplay muted>
Â  <source src="horse.ogg" type="audio/ogg">
Â  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

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

HTML Audio Formats

There are three supported audio formats: MP3, WAV, and OGG. The browser support for the different formats is:Â

Browser	MP3	WAV	OGG
Edge/IE	YES	YES*	YES*
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

*From Edge 79

HTML Audio - Media Types

File Format	Media Type
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

HTML Audio - Methods, Properties, and Events

The HTML DOM defines methods, properties, and events for the `<audio>` element.

This allows you to load, play, and pause audios, as well as set 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 HTML Audio/Video DOM Reference.

HTML Audio Tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and <audio>

HTML Plug-ins

Plug-ins are computer programs that extend the standard functionality of the browser.

Plug-ins

Plug-ins were designed to be used for many different purposes:

- To run Java applets
- To run Microsoft ActiveX controls
- To display Flash movies
- To display maps
- To scan for viruses
- To verify a bank id

Warning !

Most browsers no longer support Java Applets and Plug-ins.

ActiveX controls are no longer supported in any browsers.

The support for Shockwave Flash has also been turned off in modern browsers.

The <object> Element

The `<object>` element is supported by all browsers.

The `<object>` element defines an embedded object within an HTML document.

It was designed to embed plug-ins (like Java applets, PDF readers, and Flash Players) in web pages, but can also be used to include HTML in HTML:

Example

```
<object width="100%" height="500px" data="snippet.html"></object>
```

Or images if you like:

Example

```
<object data="audi.jpeg"></object>
```

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.

Example

```
<embed src="audi.jpeg">
```

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">
```

HTML YouTube Videos

The easiest way to play videos in HTML, is to use YouTube.

Struggling with Video Formats?

Converting videos to different formats can be difficult and time-consuming.

An easier solution is to let YouTube play the videos in your web page.

YouTube Video Id

YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.

You can use this id, and refer to your video in the HTML code.

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 `height` attributes to specify the dimension of the player
- Add any other parameters to the URL (see below)

Example

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```

YouTube Autoplay + Mute

You can let your video start playing automatically when a user visits the page, by adding `autoplay=1` to the YouTube URL. However, automatically starting a video is annoying for your visitors!

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add `mute=1` after `autoplay=1` to let your video start playing automatically (but muted).

YouTube - Autoplay + Muted

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>
```

YouTube Playlist

A comma separated list of videos to play (in addition to the original URL).

YouTube Loop

Add `loop=1` to let your video loop forever.

Value 0 (default): The video will play only once.

Value 1: The video will loop (forever).

YouTube - Loop

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">
</iframe>
```

YouTube Controls

Add `controls=0` to not display controls in the video player.

Value 0: Player controls does not display.

Value 1 (default): Player controls display.

YouTube - Controls

<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>

HTML Geolocation API

The HTML Geolocation API is used to locate a user's position.

Locate the User's Position

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it.

Try It

Note: Geolocation is most accurate for devices with GPS, like smartphones.

Browser Support

The numbers in the table specify the first browser version that fully supports Geolocation.

API					
Geolocation	5.0 - 49.0 (http) 50.0 (https)	9.0	3.5	5.0	16.0

Note: As of Chrome 50, the Geolocation API will only work on secure contexts such as HTTPS. If your site is hosted on an non-secure origin (such as HTTP) the requests to get the users location will no longer function.

Using HTML Geolocation

The `getCurrentPosition()` method is used to return the user's position.

The example below returns the latitude and longitude of the user's position:

Example

```
<script>
var x = document.getElementById("demo");
function getLocation() {
Â Â if (navigator.geolocation) {
Â Â Â  navigator.geolocation.getCurrentPosition(showPosition);
Â Â } else {
Â Â Â  x.innerHTML = "Geolocation is not supported by this browser.";
Â Â }
}

function showPosition(position) {
Â  x.innerHTML = "Latitude: " + position.coords.latitude +
Â Â "<br>Longitude: " + position.coords.longitude;
}
</script>
```

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 outputs 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) {
Â Â switch(error.code) {
Â Â Â  case error.PERMISSION_DENIED:
Â Â Â Â Â  x.innerHTML = "User denied the request for Geolocation."
Â Â Â Â Â  break;
Â Â Â Â case error.POSITION_UNAVAILABLE:
Â Â Â Â Â  x.innerHTML = "Location information is unavailable."
Â Â Â Â Â  break;
Â Â Â Â case error.TIMEOUT:
Â Â Â Â Â  x.innerHTML = "The request to get user location timed out."
Â Â Â Â Â  break;
Â Â Â  case error.UNKNOWN_ERROR:
Â Â Â Â Â Â x.innerHTML = "An unknown error occurred."
Â Â Â Â Â Â break;
Â Â }
}
```

Displaying the Result in a Map

To display the result in a map, you need access to a map service, like Google Maps.

In the example below, the returned latitude and longitude is used to show the location in a Google Map (using a static image):

Example

```
function showPosition(position) {
Â Â var latlon = position.coords.latitude + "," + position.coords.longitude;

Â Â var img_url = "https://maps.googleapis.com/maps/api/staticmap?center=
Â Â "+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR_KEY";

Â Â document.getElementById("mapholder").innerHTML = "<img src='"+img_url+"'>";
}
```

Location-specific Information

This page has demonstrated how to show a user's position on a map.

Geolocation is also very useful for location-specific information, like:

- 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 on success. The latitude, longitude and accuracy properties are always returned. The other properties are returned if available:

Property	Returns
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

Geolocation Object - Other interesting Methods

The Geolocation object also has 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 smartphone):

Example

```
<script>
var x = document.getElementById("demo");
function getLocation() {
Â  if (navigator.geolocation) {
Â Â Â  navigator.geolocation.watchPosition(showPosition);
Â Â } else {
Â Â Â  x.innerHTML = "Geolocation is not supported by this browser.";
Â Â }
}
function showPosition(position) {
Â  x.innerHTML = "Latitude: " + position.coords.latitude +
Â Â "<br>Longitude: " + position.coords.longitude;
}
</script>
```

HTML Drag and Drop API

In HTML, any element can be dragged and dropped.

Example

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.

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) {
Â Â ev.preventDefault();
}

function drag(ev) {
Â  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
Â  ev.preventDefault();
```

Â Â var data = ev.dataTransfer.getData("text");
Â Â ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>
</html>

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) {
Â 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) {
Â ev.preventDefault();
Â var data = ev.dataTransfer.getData("text");
Â 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

Example

How to drag (and drop) an image back and forth between two elements:

HTML Web Storage API

HTML web storage; better than cookies.

What is HTML Web Storage?

With web 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. Web 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.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

Browser Support

The numbers in the table specify the first browser version that fully supports Web Storage.

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

HTML Web Storage Objects

HTML web 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 browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

```
if (typeof(Storage) !== "undefined") {
Â  // Code for localStorage/sessionStorage.
} else {
Â  // 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");
```

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) {
Â  localStorage.clickcount = Number(localStorage.clickcount) + 1;
} else {
Â  localStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You have clicked the button " +
localStorage.clickcount + " time(s).";
```

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 specific browser tab.

The following example counts the number of times a user has clicked a button, in the current session:

Example

```
if (sessionStorage.clickcount) {
Â  sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
Â  sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
```

HTML Web Workers API

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

Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it:

```
if (typeof(Worker) !== "undefined") {
Â Â // Yes! Web worker support!
Â  // Some code.....
} else {
Â  // 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() {
Â  i = i + 1;
Â  postMessage(i);
Â Â 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") {
Â  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){
Â Â 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>

<script>
var w;

function startWorker() {
Â  if (typeof(Worker) !== "undefined") {
Â Â Â  if (typeof(w) == "undefined") {
Â Â Â Â Â  w = new Worker("demo_workers.js");
Â Â Â  }
Â Â Â Â w.onmessage = function(event) {
Â Â Â Â Â  document.getElementById("result").innerHTML = event.data;
Â Â Â Â };
Â Â } else {
Â Â Â  document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";
Â Â }
}

function stopWorker() {
Â Â w.terminate();
Â Â w = undefined;
}
</script>

</body>
</html>
```

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

HTML SSE API

Server-Sent Events (SSE) 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	79.0	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) {
Â Â document.getElementById("result").innerHTML += event.data + "<br>";
};
```

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") {
Â  // Yes! Server-sent events support!
Â  // Some code.....
} else {
Â  // 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

HTML Examples

HTML Basic

HTML document HTML headings HTML paragraphs HTML links HTML images HTML buttons HTML lists

Examples explained

HTML Attributes

The title attribute The href attribute The width and height attributes The alt attribute Attribute without quotes Attribute without quotes does not work

Examples explained

HTML Headings

HTML headings HTML horizontal rules HTML head

Examples explained

HTML Paragraphs

HTML paragraphs More HTML paragraphs The use of line breaks in HTML Poem problems (some problems with HTML formatting) How to control the line breaks and spaces with the <pre> tag

Examples explained

HTML Styles

HTML styles HTML background color HTML text color HTML text font HTML text size HTML text alignment

Examples explained

HTML Text Formatting

Bold formatting using the <b> element Strong formatting using the element Italic formatting using the <i> element Emphasized formatting using the element Small formatting using the <small> element Marked formatting using the <mark> element Marked deleted using the <del> element Marked inserted using the <ins> element Marked deleted and inserted using <del> and <ins> Subscript formatting using the <sub> element Superscript formatting using the <sup> element

Examples explained

HTML Quotations and Citations

Formatting short quotations with the <q> element. Formatting quoted sections with the <blockquote> element. Formatting document author/owner information with the <address> element Formatting abbreviations and acronyms the <abbr> element Formatting work title with the <cite> element Formatting text direction with the <bdo> element

Examples explained

HTML Comments

Hidden comments Conditional comments Comments for debugging

Examples explained

HTML CSS

HTML with inline CSS HTML with internal CSS HTML with external CSS HTML with CSS fonts HTML with CSS using the id attribute HTML with CSS using the class attribute HTML and CSS borders HTML and CSS padding HTML and CSS margin HTML and CSS full demo

Examples explained

HTML Links

Linking, using an absolute URL Linking, using a relative URL Changing the color of links Removing the underline from links Changing the target of a link An image as a link Creating a bookmark link A link that breaks out of a frame A mailto link A mailto link with subject

Examples explained

HTML Images

An image An image height and width using attributes An image height and width using CSS An image height and width using both An image in another folder An image with a broken link An image on another server Using an image as a link A moving image An image map with clickable regions A floating image

Examples explained

HTML Tables

Basic HTML tables A table with borders A table with collapsed borders A table with cell padding A table with headings A table with left-aligned headings Horizontal/Vertical table headings A table with a caption Table cells that span more than one column Table cells that span more than one row A table with cell spacing A table with HTML tags inside Tables with different style using id I Tables with different style using id II Tables with different style using class I Tables with different style using class II

Examples explained

HTML Lists

An unordered list (default) An unordered list with disc bullets An unordered list with circle bullets An unordered list with square bullets An unordered list without bullets An ordered list (default) An ordered list with numbers An ordered list with letters An ordered list with lowercase letters An ordered list with roman numbers An ordered list with lowercase roman numbers A description list A nested list I A nested list II A horizontal list A horizontal list menu

Examples explained

HTML Block and inline elements

The element The element Styling a element Styling a element

Examples explained

HTML Classes

Style all elements with a specified class name Access elements with a specified class name, with JavaScript Multiple classes Same class, different tag

Examples explained

HTML Id

Style an element with a specific id Difference between class and id Access an element with a specific id, with JavaScript

Examples explained

HTML Layout

Layout using float Layout using flexbox Layout using flexbox 2 Layout using flexbox 3

Examples explained

HTML IFrame

Inline frame (a frame inside an HTML page)

Examples explained

HTML head Elements

A valid HTML document with no <html> <body, and <head> A valid HTML document with no <head> element The <title> element defines the document title The <style> element contains style information The <link> element defines a relationship to an external resource The <meta> element defines special meta information The <script> element defines client-side JavaScripts The <base> element defines the base URL for all URLs

Examples explained

HTML Scripts

Insert a script Use of the <noscript> tag

Examples explained

HTML Computercode Elements

Keyboard input formatting using the <kbd> element Computer output formatting using the <samp> element Programming code formatting using the <code> element Programming code formatting preserving whitespace and line-breaks Variable formatting using the <var> element

Examples explained

HTML Forms

Form with text input Form with radio button input Form with text fields and a submit button Form with a text fields without a name attribute Grouping Form Data

Examples explained

HTML Form Elements

A simple drop-down list A drop-down list with a pre-selected value A textarea (a multi-line text input field) An input button Using the <datalist> Element Using the <output> Element

Examples explained

HTML Input Types

Input type text Input type password Input type radio Input type checkbox Input type button Input type number - with restrictions Input type number - with steps Input type date - with date picker Input type date - with restrictions Input type color - with color picker Input type range Input type month Input type week Input type time Input type datetime Input type datetime-local Input type email Input type search Input type tel Input type url

Examples explained

HTML Input Attributes

The autocomplete attribute The novalidate attribute The autofocus_attribute The form attribute The formaction attribute The formenctype attribute The formmethod attribute The formnovalidate attribute The formtarget attribute The height and width attributes The list attribute The min and max attributes The multiple attribute The pattern attribute The placeholder attribute The required attribute The step attribute

Examples explained

HTML Canvas Graphics

Draw on the canvas with JavaScript Draw a line with lineTo() Draw a circle with arc() Draw a text with fillText() Draw a text with strokeText() Draw a linear gradient Draw a circular gradient Draw an image with drawImage()

Examples explained

HTML SVG Graphics

SVG Circle SVG Rectangle SVG Rounded Rectangle SVG Star SVG Logo

Examples explained

HTML Media

Play Bunny Play bear video with controls Play bear video with autoplay Play Horse sound with controls

Examples explained

HTML Geolocation

Get geolocation coordinates Handle geolocation errors Get geolocation and watch the position

Examples explained

HTML Local Storage

Store a name permanently Store a counter permanently Store a counter for one session

Examples explained

HTML Media

Play a video file Play an audio file in HTML Play a YouTube video in HTML

Examples explained

More HTML Examples

HTML drag and drop HTML web workers HTML server sent events

HTML Quiz

You can test your HTML skills with W3Schools' Quiz.

The Test

The test contains 40 questions and there is no time limit.Â

The test is not official, it's just a nice way to see how much you know, or don't know, about HTML.

Count Your Score

You will get 1 point for each correct answer. At the end of the Quiz, your total score will be displayed. Maximum score is 40 points.

Start the Quiz

Good luck!

If you don't know HTML, we suggest that you read our HTML Tutorial from scratch.

HTML Exercises

You can test your HTML skills with W3Schools' Exercises.

Exercises

We have gathered a variety of HTML exercises (with answers) for each HTML Chapter.

Try to solve an exercise by editing some code. Get a "hint" if you're stuck, or show the answer to see what you've done wrong.

Count Your Score

You will get 1 point for each correct answer. Your score and total score will always be displayed.

Start HTML Exercises

Good luck!

If you don't know HTML, we suggest that you read our HTML Tutorial from scratch.

W3Schools HTML Certificate

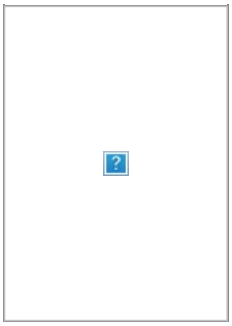
Â Â W3Schools offers an Online Certification Program.

The perfect solution for busy professionals who need to balance work, family, and career building.

More than 25 000 certificates already issued!

Document Your Skills

Knowledge is power, especially in the current job market. Documentation of your skills enables you to advance your career, or help you to start a new one.



Get a Certificate

Getting a certificate proves your commitment to upgrade your skills, gives you the credibility needed for more responsibilities, larger projects, and a higher salary.

Get Your Certificate Â»

How Does It Work?

- Study for free at W3Schools.com
- Study at your own speed
- Test your skills with W3Schools online quizzes
- Apply for your certificate by paying an exam fee
- Take your exam online, at any time, and from any location

You Have Learned HTML, Now What?

HTML Summary

This tutorial has taught you how to use HTML to create your own web site.

HTML is the universal markup language for the Web. HTML lets you format text, add graphics, create links, input forms, frames and tables, etc., and save it all in a text file that any browser can read and display.

For more information on HTML, please take a look at our HTML examples and our HTML reference.

You can also test your HTML skills with HTML Exercises and HTML Quizzes.

Now You Know HTML, What's Next?

Learn CSS

CSS lets you style your HTML pages.

CSS gives you total control of the layout, without messing up the document content.

To learn more about CSS, please visit our CSS tutorial.

Learn JavaScript

JavaScript makes your website more dynamic. A dynamic website can react to events and allow user interaction.

JavaScript is the most popular scripting language on the internet and it works with all major browsers.

If you want to learn more about JavaScript, please visit our JavaScript tutorial.

Publishing Your Website

To make your website available to the world, you must publish it.

For this, you have two options:

- Use an Internet Service Provider
- Host your own website

Using an Internet Service Provider

An Internet service provider (ISP) is a company that provides services for accessing and using the Internet.

Internet services typically provided by ISPs include Internet access, Internet transit, domain name registration, web hosting, Usenet service, and colocation.

Using an Internet Service Provider (ISP) is the most common option.

Advantages:

- **Connection Speed** - ISPs have very fast connections to the internet.
- **Powerful Hardware** - ISPs have powerful web servers that can be shared by several clients. You can also expect an effective load balancing and necessary backup servers
- **Security and Stability** - ISPs are specialists on web hosting. Expect more than 99% up time, the latest software patches, and the best virus protection

Things to Consider:

- **24-hour support** - The ISP should offer 24-hours support. Toll-free phone could also be vital
- **Daily Backup** - The ISP must run a daily backup routine
- **Traffic Volume** - Check the ISP's traffic volume restrictions (do not end up paying a fortune for unexpected high traffic)
- **Bandwidth or Content Restrictions** - Check the ISP's bandwidth and content restrictions (Is it possible to publish pictures, video, or sound?)
- **E-mail Capabilities** - Make sure the ISP supports the e-mail capabilities you need
- **Database Access** - Make sure the ISP supports the database access you need

---

Hosting Your Own Website

Hosting your own website, on your own server, is also an option.

Things to Consider:

- **Hardware Expenses** - To run a "real" web site, you must buy powerful server hardware (a low cost PC will not do the job). You will also need a permanent (24/7) high-speed connection
- **Software Expenses** - Server-licenses are often higher than client-licenses. Server-licenses also might have limits on number of users
- **Labor Expenses** - Don't expect low labor expenses. You have to install your own hardware and software. You also have to deal with bugs and viruses, and keep your server constantly running

---

HTML Accessibility

HTML Accessibility

Always write HTML code with accessibility in mind!

Provide the user a good way to navigate and interact with your site. Make your HTML code as **semantic** as possible.

Semantic HTML

Semantic HTML means using correct HTML elements for their correct purpose as much as possible. Semantic elements are elements with a meaning; if you need a button, use the `<button>` element (and not a `<div>` element).

Semantic

```
<button>Click Me</button>
```

Non-semantic

```
<div>Click Me</div>
```

Semantic HTML gives context to screen readers, which read the contents of a page out loud.

With the button example in mind:

- buttons have more suitable styling by default
- a screen reader identifies it as a button
- focusable
- clickable

A button is also accessible for people relying on keyboard-only navigation; it can be clickable with both mouse and keys, and it can be tabbed between (using the tab key on the keyboard).

Examples of **non-semantic** elements: `<div>` and `<span>` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

Headings Are Important

Headings are defined with the `<h1>` to `<h6>` tags:

Example

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Search engines use the 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 and the relationships between different sections.

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

Note: Use HTML headings for headings only. Don't use headings to make text **BIG** or **bold**.

Alternative Text

The `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the `src` attribute, or if the user uses a screen reader).

The value of the `alt` attribute should describe the image:

Example

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

If a browser cannot find an image, it will display the value of the `alt` attribute:

Example

```
<img src="wrongname.gif" alt="Flowers in Chania">
```

Declare the Language

You should always include the `lang` attribute inside the `<html>` tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

The following example specifies English as the language:

```
<!DOCTYPE html>
<html lang="en">
<body>

...

</body>
</html>
```

Use Clear Language

Always use a clear language, that is easy to understand. Also try to avoid characters that cannot be read clearly by a screen reader. For example:

- Keep sentences as short as possible
- Avoid dashes. Instead of writing 1-3, write 1 to 3
- Avoid abbreviations. Instead of writing Feb, write February
- Avoid slang words

Create Good Link Text

A link text should explain clearly what information the reader will get by clicking on that link.

Examples of good and bad links:

Good

ind out more about the HTML language

ead more about how to eat healthy

uy tickets to Mars here

Bad

lick here

ead more..

uy tickets to Mars here

The title Attribute

The `title` attribute specifies extra information about an element.

The information is most often shown as a tooltip text when the mouse moves over the element.

The `title` attribute can be used on any HTML element (it will validate on any HTML element. However, it is not necessarily useful).

Example

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

<p title="Free Web tutorials">W3Schools.com</p>
```

HTML Element Reference

HTML Tags Ordered Alphabetically

Tag	Description
<!--...-->	Defines a comment
<!DOCTYPE>Â	Defines the document type
<a>	Defines a hyperlink
<abbr>	Defines an abbreviation or an acronym
<acronym>	Not supported in HTML5. Use <abbr> instead. Defines an acronym
<address>	Defines contact information for the author/owner of a document
<applet>	Not supported in HTML5. Use <embed> or <object> instead. Defines an embedded applet
<area>	Defines an area inside an image map
<article>	Defines an article
<aside>	Defines content aside from the page content
<audio>	Defines embedded sound content
<b>	Defines bold text
<base>	Specifies the base URL/target for all relative URLs in a document
<basefont>	Not supported in HTML5. Use CSS instead. Specifies a default color, size, and font for all text in a document
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<bdo>	Overrides the current text direction
<big>	Not supported in HTML5. Use CSS instead. Defines big text
<blockquote>	Defines a section that is quoted from another source
<body>	Defines the document's body
 	Defines a single line break
<button>	Defines a clickable button
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<caption>	Defines a table caption
	Not supported in HTML5. Use CSS instead. Defines centered text
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<col>	Specifies column properties for each column within a <colgroup> elementÂ
<colgroup>	Specifies a group of one or more columns in a table for formatting
<data>	Adds a machine-readable translation of a given content
<datalist>	Specifies a list of pre-defined options for input controls
<dd>	Defines a description/value of a term in a description list
<del>	Defines text that has been deleted from a document
<details>	Defines additional details that the user can view or hide
<dfn>	Specifies a term that is going to be defined within the content
<dialog>	Defines a dialog box or window
<dir>	Not supported in HTML5. Use instead. Defines a directory list
	Defines a section in a document

<dl>	Defines a description list
<dt>	Defines a term/name in a description list
	Defines emphasized textÂ
<embed>	Defines a container for an external application
<fieldset>	Groups related elements in a form
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content
<font>	Not supported in HTML5. Use CSS instead. Defines font, color, and size for text
<footer>	Defines a footer for a document or section
<form>	Defines an HTML form for user input
<frame>	Not supported in HTML5. Defines a window (a frame) in a frameset
<frameset>	Not supported in HTML5. Defines a set of frames
<h1> to <h6>	Defines HTML headings
<head>	Contains metadata/information for the document
<header>	Defines a header for a document or section
<hr>	Defines a thematic change in the content
<html>	Defines the root of an HTML document
<i>	Defines a part of text in an alternate voice or mood
<iframe>	Defines an inline frame
<img>	Defines an image
<input>	Defines an input control
<ins>	Defines a text that has been inserted into a document
<kbd>	Defines keyboard input
<label>	Defines a labelÂ for an <input> element
<legend>	Defines a caption for a <fieldset> element
	Defines a list item
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<main>	Specifies the main content of a document
<map>	Defines an image map
<mark>	Defines marked/highlighted text
<meta>	Defines metadata about an HTML document
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links
<noframes>	Not supported in HTML5. Defines an alternate content for users that do not support frames
<noscript>	Defines an alternate content for users that do not support client-side scripts
<object>	Defines a container for an external application
	Defines an ordered list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<output>	Defines the result of a calculation
<p>	Defines a paragraph
<param>	Defines a parameter for an object
<picture>	Defines a container for multiple image resources
<pre>	Defines preformatted text
<progress>	Represents the progress of a task
<q>	Defines a short quotation
<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)
<s>	Defines text that is no longer correct
<samp>	Defines sample output from a computer program
<script>	Defines a client-side script
<section>	Defines a section in a document
<select>	Defines a drop-down list
<small>	Defines smaller text
<source>	Defines multiple media resources for media elements (<video> and <audio>)
	Defines a section in a document
	Not supported in HTML5. Use <del> or <s> instead.

<strike>	Defines strikethrough text
	Defines important text
<style>	Defines style information for a document
<sub>	Defines subscripted text
<summary>	Defines a visible heading for a <details> element
<sup>	Defines superscripted text
<svg>	Defines a container for SVG graphics
<table>	Defines a table
<tbody>	Groups the body content in a table
<td>	Defines a cell in a table
<template>	Defines a container for content that should be hidden when the page loads
<textarea>	Defines a multiline input control (text area)
<tfoot>	Groups the footer content in a table
<th>	Defines a header cell in a table
<thead>	Groups the header content in a table
<time>	Defines a specific time (or datetime)
<title>	Defines a title for the document
<tr>	Defines a row in a table
<track>	Defines text tracks for media elements (<video> and <audio>)
<tt>	Not supported in HTML5. Use CSS instead. Defines teletype text
<u>	Defines some text that is unarticulated and styled differently from normal text
	Defines an unordered list
<var>	Defines a variable
<video>	Defines embedded video content
<wbr>	Defines a possible line-break

HTML Attribute Reference

HTML Attribute Reference

The table below lists all HTML attributes and what elements they can be used within:

Attribute	Belongs to	Description
accept	<input>	Specifies the types of files that the server accepts (only for type="file")
accept-charset	<form>	Specifies the character encodings that are to be used for the form submission
accesskey	Global Attributes	Specifies a shortcut key to activate/focus an element
action	<form>	Specifies where to send the form-data when a form is submitted
align	Not supported in HTML 5.	Specifies the alignment according to surrounding elements. Use CSS instead
alt	<area>, <img>, <input>	Specifies an alternate text when the original element fails to display
async	<script>	Specifies that the script is executed asynchronously (only for external scripts)
autocomplete	<form>, <input>	Specifies whether the <form> or the <input> element should have autocomplete enabled
autofocus	<button>, <input>, <select>, <textarea>	Specifies that the element should automatically get focus when the page loads
autoplay	<audio>, <video>	Specifies that the audio/video will start playing as soon as it is ready
bgcolor	Not supported in HTML 5.	Specifies the background color of an element. Use CSS instead
border	Not supported in HTML 5.	Specifies the width of the border of an element. Use CSS instead
charset	<meta>, <script>	Specifies the character encoding

checked	<input>	Specifies that an <input> element should be pre-selected when the page loads (for type="checkbox" or type="radio")
cite	<blockquote>, <del>, <ins>, <q>	Specifies a URL which explains the quote/deleted/inserted text
class	Global Attributes	Specifies one or more classnames for an element (refers to a class in a style sheet)
color	Not supported in HTML 5.	Specifies the text color of an element. Use CSS instead
cols	<textarea>	Specifies the visible width of a text area
colspan	<td>, <th>	Specifies the number of columns a table cell should span
content	<meta>	Gives the value associated with the http-equiv or name attribute
contenteditable	Global Attributes	Specifies whether the content of an element is editable or not
controls	<audio>, <video>	Specifies that audio/video controls should be displayed (such as a play/pause button etc)
coords	<area>	Specifies the coordinates of the area
data	<object>	Specifies the URL of the resource to be used by the object
data-*	Global Attributes	Used to store custom data private to the page or application
datetime	<del>, <ins>, <time>	Specifies the date and time
default	<track>	Specifies that the track is to be enabled if the user's preferences do not indicate that another track would be more appropriate
defer	<script>	Specifies that the script is executed when the page has finished parsing (only for external scripts)
dir	Global Attributes	Specifies the text direction for the content in an element
dirname	<input>, <textarea>	Specifies that the text direction will be submitted
disabled	<button>, <fieldset>, <input>, <optgroup>, <option>, <select>, <textarea>	Specifies that the specified element/group of elements should be disabled
download	<a>, <area>	Specifies that the target will be downloaded when a user clicks on the hyperlink
draggable	Global Attributes	Specifies whether an element is draggable or not
enctype	<form>	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
for	<label>, <output>	Specifies which form element(s) a label/calculation is bound to
form	<button>, <fieldset>, <input>, <label>, <meter>, <object>, <output>, <select>, <textarea>	Specifies the name of the form the element belongs to
formaction	<button>, <input>	Specifies where to send the form-data when a form is submitted. Only for type="submit"
headers	<td>, <th>	Specifies one or more headers cells a cell is related to
height	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Specifies the height of the element
hidden	Global Attributes	Specifies that an element is not yet, or is no longer, relevant
high	<meter>	Specifies the range that is considered to be a high value
href	<a>, <area>, <base>, <link>	Specifies the URL of the page the link goes to
hreflang	<a>, <area>, <link>	Specifies the language of the linked document
http-equiv	<meta>	Provides an HTTP header for the information/value of the content attribute
id	Global Attributes	Specifies a unique id for an element
ismap	<img>	Specifies an image as a server-side image map

kind	<track>	Specifies the kind of text track
label	<track>, <option>, <optgroup>	Specifies the title of the text track
lang	Global Attributes	Specifies the language of the element's content
list	<input>	Refers to a <datalist> element that contains pre-defined options for an <input> element
loop	<audio>, <video>	Specifies that the audio/video will start over again, every time it is finished
low	<meter>	Specifies the range that is considered to be a low value
max	<input>, <meter>, <progress>	Specifies the maximum value
maxlength	<input>, <textarea>	Specifies the maximum number of characters allowed in an element
media	<a>, <area>, <link>, <source>, <style>	Specifies what media/device the linked document is optimized for
method	<form>	Specifies the HTTP method to use when sending form-data
min	<input>, <meter>	Specifies a minimum value
multiple	<input>, <select>	Specifies that a user can enter more than one value
muted	<video>, <audio>	Specifies that the audio output of the video should be muted
name	<button>, <fieldset>, <form>, <iframe>, <input>, <map>, <meta>, <object>, <output>, <param>, <select>, <textarea>	Specifies the name of the element
novalidate	<form>	Specifies that the form should not be validated when submitted
onabort	<audio>, <embed>, <img>, <object>, <video>	Script to be run on abort
onafterprint	<body>	Script to be run after the document is printed
onbeforeprint	<body>	Script to be run before the document is printed
onbeforeunload	<body>	Script to be run when the document is about to be unloaded
onblur	All visible elements.	Script to be run when the element loses focus
oncanplay	<audio>, <embed>, <object>, <video>	Script to be run when a file is ready to start playing (when it has buffered enough to begin)
oncanplaythrough	<audio>, <video>	Script to be run when a file can be played all the way to the end without pausing for buffering
onchange	All visible elements.	Script to be run when the value of the element is changed
onclick	All visible elements.	Script to be run when the element is being clicked
oncontextmenu	All visible elements.	Script to be run when a context menu is triggered
oncopy	All visible elements.	Script to be run when the content of the element is being copied
oncuechange	<track>	Script to be run when the cue changes in a <track> element
oncut	All visible elements.	Script to be run when the content of the element is being cut
ondblclick	All visible elements.	Script to be run when the element is being double-clicked
ondrag	All visible elements.	Script to be run when the element is being dragged
ondragend	All visible elements.	Script to be run at the end of a drag operation
ondragenter	All visible elements.	Script to be run when an element has been dragged to a valid drop target
ondragleave	All visible elements.	Script to be run when an element leaves a valid drop target
ondragover	All visible elements.	Script to be run when an element is being dragged over a valid drop target
		Script to be run at the start of a drag

ondragstart	All visible elements.	operation
ondrop	All visible elements.	Script to be run when dragged element is being dropped
ondurationchange	<audio>, <video>	Script to be run when the length of the media changes
onemptied	<audio>, <video>	Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)
onended	<audio>, <video>	Script to be run when the media has reach the end (a useful event for messages like "thanks for listening")
onerror	<audio>, <body>, <embed>, <img>, <object>, <script>, <style>, <video>	Script to be run when an error occurs
onfocus	All visible elements.	Script to be run when the element gets focus
onhashchange	<body>	Script to be run when there has been changes to the anchor part of the a URL
oninput	All visible elements.	Script to be run when the element gets user input
oninvalid	All visible elements.	Script to be run when the element is invalid
onkeydown	All visible elements.	Script to be run when a user is pressing a key
onkeypress	All visible elements.	Script to be run when a user presses a key
onkeyup	All visible elements.	Script to be run when a user releases a key
onload	<body>, <iframe>, <img>, <input>, <link>, <script>, <style>	Script to be run when the element is finished loading
onloadeddata	<audio>, <video>	Script to be run when media data is loaded
onloadedmetadata	<audio>, <video>	Script to be run when meta data (like dimensions and duration) are loaded
onloadstart	<audio>, <video>	Script to be run just as the file begins to load before anything is actually loaded
onmousedown	All visible elements.	Script to be run when a mouse button is pressed down on an element
onmousemove	All visible elements.	Script to be run as long as theÂ mouse pointer is moving over an element
onmouseout	All visible elements.	Script to be run when a mouse pointer moves out of an element
onmouseover	All visible elements.	Script to be run when a mouse pointer moves over an element
onmouseup	All visible elements.	Script to be run when a mouse button is released over an element
onmousewheel	All visible elements.	Script to be run when a mouse wheel is being scrolled over an element
onoffline	<body>	Script to be run when the browser starts to work offline
ononline	<body>	Script to be run when the browser starts to work online
onpagehide	<body>	Script to be run when a user navigates away from a page
onpageshow	<body>	Script to be run when a user navigates to a page
onpaste	All visible elements.	Script to be run when the user pastes some content in an element
onpause	<audio>, <video>	Script to be run when the media is paused either by the user or programmatically
onplay	<audio>, <video>	Script to be run when the media has started playing
onplaying	<audio>, <video>	Script to be run when the media has started playing
onpopstate	<body>	Script to be run when the window's history changes.
onprogress	<audio>, <video>	Script to be run when the browser is in the process of getting the media data
onratechange	<audio>, <video>	Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode).
onreset	<form>	Script to be run when a reset button in a form is clicked.

onresize	<body>	Script to be run when the browser window is being resized.
onscroll	All visible elements.	Script to be run when an element's scrollbar is being scrolled
onsearch	<input>	Script to be run when the user writes something in a search field (for <input="search">)
onseeked	<audio>, <video>	Script to be run when the seeking attribute is set to false indicating that seeking has ended
onseeking	<audio>, <video>	Script to be run when the seeking attribute is set to true indicating that seeking is active
onselect	All visible elements.	Script to be run when the element gets selected
onstalled	<audio>, <video>	Script to be run when the browser is unable to fetch the media data for whatever reason
onstorage	<body>	Script to be run when a Web Storage area is updated
onsubmit	<form>	Script to be run when a form is submitted
onsuspend	<audio>, <video>	Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason
ontimeupdate	<audio>, <video>	Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media)
ontoggle	<details>	Script to be run when the user opens or closes the <details> element
onunload	<body>	Script to be run when a page has unloaded (or the browser window has been closed)
onvolumechange	<audio>, <video>	Script to be run each time the volume of a video/audio has been changed
onwaiting	<audio>, <video>	Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data)
onwheel	All visible elements.	Script to be run when the mouse wheel rolls up or down over an element
open	<details>	Specifies that the details should be visible (open) to the user
optimum	<meter>	Specifies what value is the optimal value for the gauge
pattern	<input>	Specifies a regular expression that an <input> element's value is checked against
placeholder	<input>, <textarea>	Specifies a short hint that describes the expected value of the element
poster	<video>	Specifies an image to be shown while the video is downloading, or until the user hits the play button
preload	<audio>, <video>	Specifies if and how the author thinks the audio/video should be loaded when the page loads
readonly	<input>, <textarea>	Specifies that the element is read-only
rel	<a>, <area>, <form>, <link>	Specifies the relationship between the current document and the linked document
required	<input>, <select>, <textarea>	Specifies that the element must be filled out before submitting the form
reversed		Specifies that the list order should be descending (9,8,7...)
rows	<textarea>	Specifies the visible number of lines in a text area
rowspan	<td>, <th>	Specifies the number of rows a table cell should span
sandbox	<iframe>	Enables an extra set of restrictions for the content in an <iframe>
scope	<th>	Specifies whether a header cell is a header for a column, row, or group of columns or rows
selected	<option>	Specifies that an option should be pre-selected when the page loads

shape	<area>	Specifies the shape of the area
size	<input>, <select>	Specifies the width, in characters (for <input>) or specifies the number of visible options (for <select>)
sizes	<img>, <link>, <source>	Specifies the size of the linked resource
span	<col>, <colgroup>	Specifies the number of columns to span
spellcheck	Global Attributes	Specifies whether the element is to have its spelling and grammar checked or not
src	<audio>, <embed>, <iframe>, <img>, <input>, <script>, <source>, <track>, <video>	Specifies the URL of the media file
srcdoc	<iframe>	Specifies the HTML content of the page to show in the <iframe>
srclang	<track>	Specifies the language of the track text data (required if kind="subtitles")
srcset	<img>, <source>	Specifies the URL of the image to use in different situations
start		Specifies the start value of an ordered list
step	<input>	Specifies the legal number intervals for an input field
style	Global Attributes	Specifies an inline CSS style for an element
tabindex	Global Attributes	Specifies the tabbing order of an element
target	<a>, <area>, <base>, <form>	Specifies the target for where to open the linked document or where to submit the form
title	Global Attributes	Specifies extra information about an element
translate	Global Attributes	Specifies whether the content of an element should be translated or not
type	<a>, <button>, <embed>, <input>, <link>, <menu>, <object>, <script>, <source>, <style>	Specifies the type of element
usemap	<img>, <object>	Specifies an image as a client-side image map
value	<button>, <input>, , <option>, <meter>, <progress>, <param>	Specifies the value of the element
width	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Specifies the width of the element
wrap	<textarea>	Specifies how the text in a text area is to be wrapped when submitted in a form

HTML Global Attributes

HTML Global Attributes

The global attributes are attributes that can be used with all HTML elements.

Attribute	Description
accesskey	Specifies a shortcut key to activate/focus an element
class	Specifies one or more classnames for an element (refers to a class in a style sheet)
contenteditable	Specifies whether the content of an element is editable or not
data-*	Used to store custom data private to the page or application
dir	Specifies the text direction for the content in an element
draggable	Specifies whether an element is draggable or not
hidden	Specifies that an element is not yet, or is no longer, relevant
id	Specifies a unique id for an element
lang	Specifies the language of the element's content
spellcheck	Specifies whether the element is to have its spelling and grammar checked or not
style	Specifies an inline CSS style for an element
tabindex	Specifies the tabbing order of an element
title	Specifies extra information about an element
translate	Specifies whether the content of an element should be translated or not

HTML Reference - Browser Support

HTML Reference With Browser Support

The table below lists all HTML elements and their attributes, along with browser support:

Â

<a>	Yes	Yes	Yes	Yes	Yes
download	14.0	18.0	20.0	10.1	15.0
href	Yes	Yes	Yes	Yes	Yes
hreflang	Yes	Yes	Yes	Yes	Yes
media	Yes	Yes	Yes	Yes	Yes
ping	Yes	No	Yes	No	Yes
referrerpolicy	51.0	79.0	50.0	11.1	38.0
rel	Yes	Yes	Yes	Yes	Yes
target	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes

Â

<abbr>	Yes	Yes	Yes	Yes	Yes

Â

<address>	Yes	Yes	Yes	Yes	Yes

Â

<area>	Yes	Yes	Yes	Yes	Yes
alt	Yes	Yes	Yes	Yes	Yes
coords	Yes	Yes	Yes	Yes	Yes
download	Yes	Yes	Yes	Yes	Yes
href	Yes	Yes	Yes	Yes	Yes
hreflang	Yes	Yes	Yes	Yes	Yes
media	Yes	Yes	Yes	Yes	Yes
referrerpolicy	51.0	79.0	50.0	11.1	38.0
rel	Yes	Yes	Yes	Yes	Yes
shape	Yes	Yes	Yes	Yes	Yes
target	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes

Â

<article>	6.0	9.0	4.0	5.0	11.1

Â

<aside>	6.0	9.0	4.0	5.0	11.1

Â

<audio>	4.0	9.0	3.5	4.0	11.5
autoplay	4.0	9.0	3.5	4.0	11.5
controls	4.0	9.0	3.5	4.0	11.5
loop	4.0	9.0	3.5	4.0	11.5
muted	4.0	10.0	11.0	7.1	11.5
preload	4.0	9.0	4.0	4.0	11.5
src	4.0	9.0	3.5	4.0	11.5

Â

<b>	Yes	Yes	Yes	Yes	Yes

Â

<base>	Yes	Yes	Yes	Yes	Yes
href	Yes	Yes	Yes	Yes	Yes
target	Yes	Yes	Yes	Yes	Yes

Â

<bdi>	16.0	79.0	10.0	No	15.0

Â

<bdo>	Yes	Yes	Yes	Yes	Yes
dir	Yes	Yes	Yes	Yes	Yes

Â

<blockquote>	Yes	Yes	Yes	Yes	Yes
cite	Yes	Yes	Yes	Yes	Yes

Â

<body>	Yes	Yes	Yes	Yes	Yes

Â

** **	Yes	Yes	Yes	Yes	Yes

Â

<button>	Yes	Yes	Yes	Yes	Yes
autofocus	5.0	10.0	4.0	5.0	9.6
disabled	Yes	Yes	Yes	Yes	Yes
form	10.0	16.0	4.0	5.1	9.5
formaction	9.0	10.0	4.0	5.1	15.0
formenctype	9.0	10.0	4.0	5.1	11.5
formmethod	9.0	10.0	4.0	5.1	15.0
formnovalidate	6.0	11.0	4.0	Yes	Yes
formtarget	9.0	10.0	4.0	5.1	10.6
name	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes
value	Yes	Yes	Yes	Yes	Yes

Â

<canvas>	4.0	9.0	2.0	3.1	9.0
height	4.0	9.0	2.0	3.1	9.0
width	4.0	9.0	2.0	3.1	9.0

Â

<caption>	Yes	Yes	Yes	Yes	Yes

Â

<cite>	Yes	Yes	Yes	Yes	Yes

Â

<code>	Yes	Yes	Yes	Yes	Yes

Â

<col>	Yes	Yes	Yes	Yes	Yes
span	Yes	Yes	Yes	Yes	Yes

Â

<colgroup>	Yes	Yes	Yes	Yes	Yes
span	Yes	Yes	Yes	Yes	Yes

Â

<data>	62.0	13.0	22.0	No	49.0
value	62.0	13.0	22.0	No	49.0

Â

<datalist>	20.0	10.0	4.0	12.1	9.5

Â

<dd>	Yes	Yes	Yes	Yes	Yes

Â

<del>	Yes	Yes	Yes	Yes	Yes
cite	Yes	Yes	Yes	Yes	Yes
datetime	Yes	Yes	Yes	Yes	Yes

Â

<details>	12.0Â	79.0	49.0	6.0	15.0
open	12.0	79.0	49.0	6.0	15.0

Â

<dfn>	Yes	Yes	Yes	Yes	Yes

Â

<dialog>	37.0	79.0	53.0*	No	24.0
open	37.0	79.0	53.0*	No	24.0

* Not supported by default, but can be enabled in about:config (set dom.dialog_element.enabled to true).

Â

** **	Yes	Yes	Yes	Yes	Yes

Â

<dl>	Yes	Yes	Yes	Yes	Yes

Â

<dt>	Yes	Yes	Yes	Yes	Yes

Â

****	Yes	Yes	Yes	Yes	Yes

Â

<embed>	Yes	Yes	Yes	Yes	Yes
height	Yes	Yes	Yes	Yes	Yes
src	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes
width	Yes	Yes	Yes	Yes	Yes

Â

<fieldset>	Yes	Yes	Yes	Yes	Yes
disabled	Yes	Yes	Yes	6.0	Yes
form	Yes	Yes	Yes	Yes	Yes
name	Yes	11.0	Yes	Yes	Yes

Â

<figcaption>	8.0	9.0	4.0	5.1	11.0

Â

<figure>	8.0	9.0	4.0	5.1	11.0

Â

<footer>	5.0	9.0	4.0	5.0	11.1

Â

<form>	Yes	Yes	Yes	Yes	Yes
accept-charset	Yes	Yes	Yes	Yes	Yes
action	Yes	Yes	Yes	Yes	Yes

autocomplete	Yes	Yes	4.0	5.2	15.0
enctype	Yes	Yes	Yes	Yes	Yes
method	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
novalidate	Yes	10.0	4.0	10.1	15.0
rel	Yes	Yes	Yes	Yes	Yes
target	Yes	Yes	Yes	Yes	Yes

Â

<h1> - <h6>	Yes	Yes	Yes	Yes	Yes

Â

<head>	Yes	Yes	Yes	Yes	Yes

Â

<header>	5.0Â	9.0	4.0	5.0	11.1

Â

<hr>	Yes	Yes	Yes	Yes	Yes

Â

<html>	Yes	Yes	Yes	Yes	Yes
xmlns	Yes	Yes	Yes	Yes	Yes

Â

<i>	Yes	Yes	Yes	Yes	Yes

Â

<iframe>	Yes	Yes	Yes	Yes	Yes
allow	60.0	79.0	74.0	11.1	47.0
allowfullscreen	27.0	11.0 -ms-	18.0	7.0	Yes
allowpaymentrequest	No	No	No	No	No
height	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
referrerpolicy	51.0	79.0	50.0	11.1	38.0
sandbox	4.0	10.0	17.0	5.0	15.0
src	Yes	Yes	Yes	Yes	Yes
srcdoc	20.0	79.0	25.0	6.0	15.0
width	Yes	Yes	Yes	Yes	Yes

Â

<img>	Yes	Yes	Yes	Yes	Yes
alt	Yes	Yes	Yes	Yes	Yes
crossorigin	Yes	Yes	Yes	Yes	Yes
height	Yes	Yes	Yes	Yes	Yes
ismap	Yes	Yes	Yes	Yes	Yes
loading	77.0	79.0	75.0	No	64.0
longdesc	Yes	Yes	Yes	Yes	Yes
referrerpolicy	51.0	79.0	50.0	11.1	38.0
sizes	Yes	Yes	Yes	Yes	Yes
src	Yes	Yes	Yes	Yes	Yes
srcset	34.0	?	38.0	8.0	21.0
usemap	Yes	Yes	Yes	Yes	Yes
width	Yes	Yes	Yes	Yes	Yes

Â

<input>	Yes	Yes	Yes	Yes	Yes
accept	26.0	10.0	37.0	11.1	15.0
alt	Yes	Yes	Yes	Yes	Yes
autocomplete	17.0	6.0	2.0	5.1	10.0
autofocus	5.0	11.0	4.0	5.0	9.6

checked	Yes	Yes	Yes	Yes	Yes
dirname	Yes	79.0	No	Yes	Yes
disabled	Yes	Yes	Yes	Yes	Yes
form	Yes	Yes	Yes	5.1	10.6
formaction	Yes	10.0	Yes	5.1	10.6
formenctype	Yes	10.0	Yes	5.1	10.6
formmethod	Yes	10.0	Yes	5.1	10.6
formnovalidate	Yes	10.0	Yes	10.1	10.6
formtarget	Yes	10.0	Yes	5.1	10.6
height	Yes	Yes	16.0	Yes	Yes
list	20.0	10.0	4.0	No	9.6
max	5.0	10.0	16.0	5.1	10.6
maxlength	4.0	10.0	4.0	5.1	15.0
min	5.0	10.0	16.0	5.1	10.6
minlength	40.0	17.0	51.0	10.1	27.0
multiple	6.0	10.0	3.6	5.0	11.0
name	Yes	Yes	Yes	Yes	Yes
pattern	5.0	10.0	4.0	10.1	9.6
placeholder	10.0	10.0	4.0	5.0	11.0
readonly	Yes	Yes	Yes	Yes	Yes
required	5.0	10.0	4.0	10.1	9.6
size	Yes	Yes	Yes	Yes	Yes
src	Yes	Yes	Yes	Yes	Yes
step	6.0	10.0	16.0	5.0	10.6
type	Yes	Yes	Yes	Yes	Yes
value	Yes	Yes	Yes	Yes	Yes
width	Yes	Yes	16.0	Yes	Yes

Â

<ins>	Yes	Yes	Yes	Yes	Yes
cite	Yes	Yes	Yes	Yes	Yes
datetime	Yes	Yes	Yes	Yes	Yes

Â

<kbd>	Yes	Yes	Yes	Yes	Yes

Â

<label>	Yes	Yes	Yes	Yes	Yes
for	Yes	Yes	Yes	Yes	Yes
form	Yes	Yes	Yes	Yes	Yes

Â

<legend>	Yes	Yes	Yes	Yes	Yes

Â

** **	Yes	Yes	Yes	Yes	Yes
value	Yes	Yes	Yes	Yes	Yes

Â

<link>	Yes	Yes	Yes	Yes	Yes
crossorigin	25.0	79.0	18.0	?	15.0
href	Yes	Yes	Yes	Yes	Yes
hreflang	Yes	Yes	Yes	Yes	Yes
media	Yes	Yes	Yes	Yes	Yes
referrerpolicy	51.0	79.0	50.0	11.1	38.0
rel	Yes	Yes	Yes	Yes	Yes
sizes	No	No	No	No	No
title	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes

Â

<main>	26.0	12.0	21.0	7.0	16.0

Â

<map>	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes

Â

<mark>	6.0	9.0	4.0	5.0	11.1

Â

<meta>	Yes	Yes	Yes	Yes	Yes
charset	Yes	Yes	Yes	Yes	Yes
content	Yes	Yes	Yes	Yes	Yes
http_equiv	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes

Â

<meter>	8.0	13.0	16.0	6.0	11.5
form	No	No	No	No	No
high	8.0	13.0	16.0	6.0	11.5
low	8.0	13.0	16.0	6.0	11.5
max	8.0	13.0	16.0	6.0	11.5
min	8.0	13.0	16.0	6.0	11.5
optimum	8.0	13.0	16.0	6.0	11.5
value	8.0	13.0	16.0	6.0	11.5

Â

<nav>	5.0	9.0	4.0	5.0	11.1

Â

<noscript>	Yes	Yes	Yes	Yes	Yes

Â

<object>	Yes	Yes	Yes	Yes	Yes
data	Yes	Yes	Yes	Yes	Yes
form	No	No	No	No	No
height	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes
usemap	No	No	Yes	No	No
width	Yes	Yes	Yes	Yes	Yes

Â

** **	Yes	Yes	Yes	Yes	Yes
reversed	18.0	79.0	18.0	6.0	12.1
start	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes

Â

<optgroup>	Yes	Yes	Yes	Yes	Yes
disabled	Yes	8.0	Yes	Yes	Yes
label	Yes	Yes	Yes	Yes	Yes

Â

<option>	Yes	Yes	Yes	Yes	Yes
disabled	Yes	8.0	Yes	Yes	Yes
label	Yes	8.0	No	Yes	Yes
selected	Yes	Yes	Yes	Yes	Yes
value	Yes	Yes	Yes	Yes	Yes

Â

<output>	10.0	13.0	4.0	5.1	11.0
for	10.0	13.0	4.0	7.0	11.5
form	No	No	No	No	No
name	10.0	13.0	4.0	7.0	11.5

Â

<p>	Yes	Yes	Yes	Yes	Yes

Â

<param>	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
value	Yes	Yes	Yes	Yes	Yes

Â

<picture>	38.0	13.0	38.0	9.1	25.0

Â

<pre>	Yes	Yes	Yes	Yes	Yes

Â

<progress>	8.0	10.0	16.0	6.0	11.0
max	8.0	10.0	16.0	6.0	11.0
value	8.0	10.0	16.0	6.0	11.0

Â

<q>	Yes	Yes	Yes	Yes	Yes
cite	Yes	Yes	Yes	Yes	Yes

Â

<rp>	5.0	5.5	38.0	5.0	15.0

Â

<rt>	5.0	5.5	38.0	5.0	15.0

Â

<ruby>	5.0	5.5	38.0	5.0	15.0

Â

<s>	Yes	Yes	Yes	Yes	Yes

Â

<samp>	Yes	Yes	Yes	Yes	Yes

Â

<script>	Yes	Yes	Yes	Yes	Yes
async	8.0	10.0	3.6	5.1	15.0
crossorigin	30.0	18.0	13.0	13.0	12.1
defer	8.0	10.0	3.5	5.0	15.0
integrity	45.0	17.0	43.0	13.0	66.0
nomodule	61.0	16.0	60.0	11.0	48.0
referrerpolicy	70.0	79.0	65.0	No	Yes
src	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes

Â

<section>	5.0	9.0	4.0	5.0	11.5

Â

<select>	Yes	Yes	Yes	Yes	Yes
autofocus	Yes	10.0	No	Yes	Yes

disabled	Yes	9.0	Yes	Yes	Yes
form	Yes	Yes	Yes	Yes	Yes
multiple	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
required	Yes	10.0	4.0	Yes	Yes
size	Yes	Yes	Yes	Yes	Yes

Â

<slot>	53.0	79.0	63.0	10.0	40.0

Â

<small>	Yes	Yes	Yes	Yes	Yes

Â

<source>	4.0	9.0	3.5	4.0	10.5
media	38.0	9.0	15.0	9.1	25.0
sizes	38.0	13.0	3.8	9.1	25.0
src	4.0	9.0	3.5	4.0	10.5
srcset	38.0	13.0	38.0	9.1	25.0
type	4.0	9.0	3.5	4.0	10.5

Â

****	Yes	Yes	Yes	Yes	Yes

Â

****	Yes	Yes	Yes	Yes	Yes

Â

<style>	Yes	Yes	Yes	Yes	Yes
media	Yes	Yes	Yes	Yes	Yes
type	Yes	Yes	Yes	Yes	Yes

Â

<sub>	Yes	Yes	Yes	Yes	Yes

Â

<summary>	12.0	79.0	49.0	6.0	15.0

Â

<sup>	Yes	Yes	Yes	Yes	Yes

Â

<svg>	4.0	9.0	3.0	3.2	10.1

Â

<table>	Yes	Yes	Yes	Yes	Yes

Â

<tbody>	Yes	Yes	Yes	Yes	Yes

Â

<td>	Yes	Yes	Yes	Yes	Yes
colspan	Yes	Yes	Yes	Yes	Yes
headers	Yes	Yes	Yes	Yes	Yes
rowspan	Yes	Yes	Yes	Yes	Yes

Â

<template>	26.0	13.0	22.0	8.0	15.0

Â

<textarea>	Yes	Yes	Yes	Yes	Yes
autocomplete	No	No	59.0	13.0	No
autofocus	Yes	10.0	4.0	Yes	Yes
cols	Yes	Yes	Yes	Yes	Yes
dirname	Yes	79.0	No	Yes	Yes
disabled	Yes	Yes	Yes	Yes	Yes
form	Yes	11.0	Yes	Yes	Yes
maxlength	Yes	10.0	4.0	Yes	Yes
minlength	Yes	Yes	Yes	Yes	Yes
name	Yes	Yes	Yes	Yes	Yes
placeholder	Yes	10.0	4.0	5.0	11.5
readonly	Yes	Yes	Yes	Yes	Yes
required	Yes	10.0	4.0	Yes	Yes
rows	Yes	Yes	Yes	Yes	Yes
spellcheck	Yes	11.0	Yes	Yes	Yes
wrap	Yes	Yes	Yes	Yes	Yes

Â

<tfoot>	Yes	Yes	Yes	Yes	Yes

Â

<th>	Yes	Yes	Yes	Yes	Yes
abbr	Yes	Yes	Yes	Yes	Yes
colspan	Yes	Yes	Yes	Yes	Yes
headers	Yes	Yes	Yes	Yes	Yes
rowspan	Yes	Yes	Yes	Yes	Yes
scope	Yes	Yes	Yes	Yes	Yes

Â

<thead>	Yes	Yes	Yes	Yes	Yes

Â

<time>	62.0	18.0	22.0	7.0	49.0
datetime	62.0	18.0	22.0	7.0	49.0

Â

<title>	Yes	Yes	Yes	Yes	Yes

Â

<tr>	Yes	Yes	Yes	Yes	Yes

Â

<track>	23.0	10.0	31.0	6.0	12.1
default	23.0	10.0	31.0	6.0	12.1
kind	23.0	10.0	31.0	6.0	12.1
label	23.0	10.0	31.0	6.0	12.1
src	23.0	10.0	31.0	6.0	12.1
srclang	23.0	10.0	31.0	6.0	12.1

Â

<u>	Yes	Yes	Yes	Yes	Yes

Â

** **	Yes	Yes	Yes	Yes	Yes

Â

<var>	Yes	Yes	Yes	Yes	Yes

Â

<video>	4.0	9.0	3.5	3.1	11.5
autoplay	4.0	9.0	3.5	3.1	11.5

controls	4.0	9.0	3.5	3.1	11.5
height	4.0	9.0	3.5	3.1	11.5
loop	4.0	9.0	11.0	3.1	11.5
muted	30.0	10.0	11.0	5.0	Yes
poster	4.0	9.0	3.6	3.1	10.5
preload	4.0	9.0	4.0	3.1	10.5
src	4.0	9.0	3.5	3.1	11.5
width	4.0	9.0	3.5	3.1	11.5

Â

<wbr>	Yes	Yes	Yes	Yes	Yes

HTML Event Attributes

Global Event Attributes

HTML has the ability to let events trigger actions in a browser, like starting a JavaScript when a user clicks on an element.

To learn more about programming events, please visit our JavaScript tutorial.

Below are the global event attributes that can be added to HTML elements to define event actions.

Window Event Attributes

Events triggered for the window object (applies to the <body> tag):

Attribute	Value	Description
onafterprint	*script*	Script to be run after the document is printed
onbeforeprint	*script*	Script to be run before the document is printed
onbeforeunload	*script*	Script to be run when the document is about to be unloaded
onerror	*script*	Script to be run when an error occurs
onhashchange	*script*	Script to be run when there has been changes to the anchor part of the a URL
onload	*script*	Fires after the page is finished loading
onmessage	*script*	Script to be run when the message is triggered
onoffline	*script*	Script to be run when the browser starts to work offline
ononline	*script*	Script to be run when the browser starts to work online
onpagehide	*script*	Script to be run when a user navigates away from a page
onpageshow	*script*	Script to be run when a user navigates to a page
onpopstate	*script*	Script to be run when the window's history changes
onresize	*script*	Fires when the browser window is resized
onstorage	*script*	Script to be run when a Web Storage area is updated
onunload	*script*	Fires once a page has unloaded (or the browser window has been closed)

Form Events

Events triggered by actions inside a HTML form (applies to almost all HTML elements, but is most used in form elements):

Attribute	Value	Description
onblur	*script*	Fires the moment that the element loses focus
onchange	*script*	Fires the moment when the value of the element is changed
oncontextmenu	*script*	Script to be run when a context menu is triggered
onfocus	*script*	Fires the moment when the element gets focus
oninput	*script*	Script to be run when an element gets user input
oninvalid	*script*	Script to be run when an element is invalid
onreset	*script*	Fires when the Reset button in a form is clicked
onsearch	*script*	Fires when the user writes something in a search field (for <input="search">)
onselect	*script*	Fires after some text has been selected in an element

onsubmit	*script*	Fires when a form is submitted

Keyboard Events

Attribute	Value	Description
onkeydown	*script*	Fires when a user is pressing a key
onkeypress	*script*	Fires when a user presses a key
onkeyup	*script*	Fires when a user releases a key

Mouse Events

Attribute	Value	Description
onclick	*script*	Fires on a mouse click on the element
ondblclick	*script*	Fires on a mouse double-click on the element
onmousedown	*script*	Fires when a mouse button is pressed down on an element
onmousemove	*script*	Fires when the mouse pointer is moving while it is over an element
onmouseout	*script*	Fires when the mouse pointer moves out of an element
onmouseover	*script*	Fires when the mouse pointer moves over an element
onmouseup	*script*	Fires when a mouse button is released over an element
onmousewheel	*script*	Deprecated. Use the onwheel attribute instead
onwheel	*script*	Fires when the mouse wheel rolls up or down over an element

Drag Events

Attribute	Value	Description
ondrag	*script*	Script to be run when an element is dragged
ondragend	*script*	Script to be run at the end of a drag operation
ondragenter	*script*	Script to be run when an element has been dragged to a valid drop target
ondragleave	*script*	Script to be run when an element leaves a valid drop target
ondragover	*script*	Script to be run when an element is being dragged over a valid drop target
ondragstart	*script*	Script to be run at the start of a drag operation
ondrop	*script*	Script to be run when dragged element is being dropped
onscroll	*script*	Script to be run when an element's scrollbar is being scrolled

Clipboard Events

Attribute	Value	Description
oncopy	*script*	Fires when the user copies the content of an element
oncut	*script*	Fires when the user cuts the content of an element
onpaste	*script*	Fires when the user pastes some content in an element

Media Events

Events triggered by medias like videos, images and audio (applies to all HTML elements, but is most common in media elements, like <audio>, <embed>, <img>, <object>, and <video>).

Tip: Look at our HTML Audio and Video DOM Reference for more information.

Attribute	Value	Description
onabort	*script*	Script to be run on abort
oncanplay	*script*	Script to be run when a file is ready to start playing (when it has buffered enough to begin)
oncanplaythrough	*script*	Script to be run when a file can be played all the way to the end without pausing for buffering
oncuechange	*script*	Script to be run when the cue changes in a <track> element
ondurationchange	*script*	Script to be run when the length of the media changes
onemptied	*script*	Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)
onended	*script*	Script to be run when the media has reach the end (a useful event for messages like "thanks for listening")
onerror	*script*	Script to be run when an error occurs when the file is being loaded

onloadeddata	*script*	Script to be run when media data is loaded
onloadedmetadata	*script*	Script to be run when meta data (like dimensions and duration) are loaded
onloadstart	*script*	Script to be run just as the file begins to load before anything is actually loaded
onpause	*script*	Script to be run when the media is paused either by the user or programmatically
onplay	*script*	Script to be run when the media is ready to start playing
onplaying	*script*	Script to be run when the media actually has started playing
onprogress	*script*	Script to be run when the browser is in the process of getting the media data
onratechange	*script*	Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode)
onseeked	*script*	Script to be run when the seeking attribute is set to false indicating that seeking has ended
onseeking	*script*	Script to be run when the seeking attribute is set to true indicating that seeking is active
onstalled	*script*	Script to be run when the browser is unable to fetch the media data for whatever reason
onsuspend	*script*	Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason
ontimeupdate	*script*	Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media)
onvolumechange	*script*	Script to be run each time the volume is changed which (includes setting the volume to "mute")
onwaiting	*script*	Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data)

Misc Events

Attribute	Value	Description
ontoggle	*script*	Fires when the user opens or closes the <details> element

HTML Color Names

Color Names Supported by All Browsers

All modern browsers support the following 140 color names (click on a color name, or a hex value, to view the color as the background-color along with different text colors):

For a full overview of HTML colors, visit our colors tutorial.

AliceBlue
#F0F8FF

AntiqueWhite
#FAEBD7

Aqua
#00FFFF

Aquamarine
#7FFFD4

Azure
#F0FFFF

Beige
#F5F5DC

Bisque
#FFE4C4

Black
#000000

BlanchedAlmond
#FFEBCD

Blue
#0000FF

BlueViolet
#8A2BE2

Brown
#A52A2A

BurlyWood
#DEB887

CadetBlue
#5F9EA0

Chartreuse
#7FFF00

Chocolate
#D2691E

Coral
#FF7F50

CornflowerBlue
#6495ED

Cornsilk
#FFF8DC

Crimson
#DC143C

Cyan
#00FFFF

DarkBlue
#00008B

DarkCyan
#008B8B

DarkGoldenRod
#B8860B

DarkGray
#A9A9A9

DarkGrey
#A9A9A9

DarkGreen
#006400

DarkKhaki
#BDB76B

DarkMagenta
#8B008B

DarkOliveGreen
#556B2F

DarkOrange
#FF8C00

DarkOrchid
#9932CC

DarkRed
#8B0000

DarkSalmon
#E9967A

DarkSeaGreen
#8FBC8F

DarkSlateBlue
#483D8B

DarkSlateGray
#2F4F4F

DarkSlateGrey
#2F4F4F

DarkTurquoise
#00CED1

DarkViolet
#9400D3

DeepPink
#FF1493

DeepSkyBlue
#00BFFF

DimGray
#696969

DimGrey
#696969

DodgerBlue
#1E90FF

FireBrick
#B22222

FloralWhite
#FFFAF0

ForestGreen
#228B22

Fuchsia
#FF00FF

Gainsboro
#DCDCDC

GhostWhite
#F8F8FF

Gold
#FFD700

GoldenRod
#DAA520

Gray
#808080

Grey
#808080

Green
#008000

GreenYellow
#ADFF2F

HoneyDew
#F0FFF0

HotPink
#FF69B4

IndianRed
#CD5C5C

Indigo
#4B0082

Ivory
#FFFFF0

Khaki
#F0E68C

Lavender
#E6E6FA

LavenderBlush
#FFF0F5

LawnGreen
#7CFC00

LemonChiffon
#FFFACD

LightBlue
#ADD8E6

LightCoral
#F08080

LightCyan
#E0FFFF

LightGoldenRodYellow
#FAFAD2

LightGray
#D3D3D3

LightGrey
#D3D3D3

LightGreen
#90EE90

LightPink
#FFB6C1

LightSalmon
#FFA07A

LightSeaGreen
#20B2AA

LightSkyBlue
#87CEFA

LightSlateGray
#778899

LightSlateGrey
#778899

LightSteelBlue
#B0C4DE

LightYellow
#FFFFE0

Lime
#00FF00

LimeGreen
#32CD32

Linen
#FAF0E6

Magenta
#FF00FF

Maroon
#800000

MediumAquaMarine
#66CDAA

MediumBlue
#0000CD

MediumOrchid
#BA55D3

MediumPurple
#9370DB

MediumSeaGreen
#3CB371

MediumSlateBlue
#7B68EE

MediumSpringGreen
#00FA9A

MediumTurquoise
#48D1CC

MediumVioletRed
#C71585

MidnightBlue
#191970

MintCream
#F5FFFA

MistyRose
#FFE4E1

Moccasin
#FFE4B5

NavajoWhite
#FFDEAD

Navy
#000080

OldLace
#FDF5E6

Olive
#808000

OliveDrab
#6B8E23

Orange
#FFA500

OrangeRed
#FF4500

Orchid
#DA70D6

PaleGoldenRod
#EEE8AA

PaleGreen
#98FB98

PaleTurquoise
#AFEEEE

PaleVioletRed
#DB7093

PapayaWhip
#FFEFD5

PeachPuff
#FFDAB9

Peru
#CD853F

Pink
#FFC0CB

Plum
#DDA0DD

PowderBlue
#B0E0E6

Purple
#800080

RebeccaPurple
#663399

Red
#FF0000

RosyBrown
#BC8F8F

RoyalBlue
#4169E1

SaddleBrown
#8B4513

Salmon
#FA8072

SandyBrown
#F4A460

SeaGreen
#2E8B57

SeaShell
#FFF5EE

Sienna
#A0522D

Silver
#C0C0C0

SkyBlue
#87CEEB

SlateBlue
#6A5ACD

SlateGray
#708090

SlateGrey
#708090

Snow
#FFFAFA

SpringGreen
#00FF7F

SteelBlue
#4682B4

Tan
#D2B48C

Teal
#008080

Thistle
#D8BFD8

Tomato
#FF6347

Turquoise
#40E0D0

Violet
#EE82EE

Wheat
#F5DEB3

White
#FFFFFF

WhiteSmoke
#F5F5F5

Yellow
#FFFF00

YellowGreen
#9ACD32

HTML Canvas Reference

HTML Canvas Reference

The HTML <canvas> tag is used to draw graphics, on the fly, via scripting (usually JavaScript).

To learn more about <canvas>, please read our HTML Canvas tutorial.

Colors, Styles, and Shadows

Property	Description
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes
shadowColor	Sets or returns the color to use for shadows
shadowBlur	Sets or returns the blur level for shadows
shadowOffsetX	Sets or returns the horizontal distance of the shadow from the shape
shadowOffsetY	Sets or returns the vertical distance of the shadow from the shape

Method	Description
createLinearGradient()	Creates a linear gradient (to use on canvas content)
createPattern()	Repeats a specified element in the specified direction
createRadialGradient()	Creates a radial/circular gradient (to use on canvas content)
addColorStop()	Specifies the colors and stop positions in a gradient object

Line Styles

Property	Description
lineCap	Sets or returns the style of the end caps for a line
lineJoin	Sets or returns the type of corner created, when two lines meet
lineWidth	Sets or returns the current line width
miterLimit	Sets or returns the maximum miter length

Rectangles

Method	Description
rect()	Creates a rectangle
fillRect()	Draws a "filled" rectangle
strokeRect()	Draws a rectangle (no fill)
clearRect()	Clears the specified pixels within a given rectangle

Paths

Method	Description
fill()	Fills the current drawing (path)
stroke()	Actually draws the path you have defined
beginPath()	Begins a path, or resets the current path
moveTo()	Moves the path to the specified point in the canvas, without creating a line
closePath()	Creates a path from the current point back to the starting point
lineTo()	Adds a new point and creates a line to that point from the last specified point in the canvas
clip()	Clips a region of any shape and size from the original canvas
quadraticCurveTo()	Creates a quadratic BÃƒÂ©zier curve
bezierCurveTo()	Creates a cubic BÃƒÂ©zier curve
arc()	Creates an arc/curve (used to create circles, or parts of circles)
arcTo()	Creates an arc/curve between two tangents
isPointInPath()	Returns true if the specified point is in the current path, otherwise false

Transformations

Method	Description
scale()	Scales the current drawing bigger or smaller
rotate()	Rotates the current drawing
translate()	Remaps the (0,0) position on the canvas
transform()	Replaces the current transformation matrix for the drawing
setTransform()	Resets the current transform to the identity matrix. Then runs transform()

Text

Property	Description
font	Sets or returns the current font properties for text content
textAlign	Sets or returns the current alignment for text content
textBaseline	Sets or returns the current text baseline used when drawing text

Method	Description
fillText()	Draws "filled" text on the canvas
strokeText()	Draws text on the canvas (no fill)
measureText()	Returns an object that contains the width of the specified text

Image Drawing

Method	Description
drawImage()	Draws an image, canvas, or video onto the canvas

Pixel Manipulation

Property	Description
width	Returns the width of an ImageData object
height	Returns the height of an ImageData object
data	Returns an object that contains image data of a specified ImageData object

Method	Description
createImageData()	Creates a new, blank ImageData object
getImageData()	Returns an ImageData object that copies the pixel data for the specified rectangle on a canvas
putImageData()	Puts the image data (from a specified ImageData object) back onto the canvas

Compositing

Property	Description
globalAlpha	Sets or returns the current alpha or transparency value of the drawing
globalCompositeOperation	Sets or returns how a new image is drawn onto an existing image

Other

Method	Description
save()	Saves the state of the current context
restore()	Returns previously saved path state and attributes
createEvent()	Â
getContext()	Â
toDataURL()	Â

HTML Audio/Video DOM Reference

HTML Audio and Video DOM Reference

The HTML5 DOM has methods, properties, and events for the `<audio>` and `<video>` elements.

HTML Audio/Video Methods

Method	Description
addTextTrack()	Adds a new text track to the audio/video
canPlayType()	Checks if the browser can play the specified audio/video type
load()	Re-loads the audio/video element
play()	Starts playing the audio/video
pause()	Pauses the currently playing audio/video

HTML Audio/Video Properties

Property	Description
audioTracks	Returns an AudioTrackList object representing available audio tracks
autoplay	Sets or returns whether the audio/video should start playing as soon as it is loaded
buffered	Returns a TimeRanges object representing the buffered parts of the audio/video
controller	Returns the MediaController object representing the current media controller of the audio/video
controls	Sets or returns whether the audio/video should display controls (like play/pause etc.)
crossOrigin	Sets or returns the CORS settings of the audio/video
currentSrc	Returns the URL of the current audio/video
currentTime	Sets or returns the current playback position in the audio/video (in seconds)
defaultMuted	Sets or returns whether the audio/video should be muted by default
defaultPlaybackRate	Sets or returns the default speed of the audio/video playback
duration	Returns the length of the current audio/video (in seconds)
ended	Returns whether the playback of the audio/video has ended or not
error	Returns a MediaError object representing the error state of the audio/video
loop	Sets or returns whether the audio/video should start over again when finished
mediaGroup	Sets or returns the group the audio/video belongs to (used to link multiple audio/video elements)
muted	Sets or returns whether the audio/video is muted or not
networkState	Returns the current network state of the audio/video
paused	Returns whether the audio/video is paused or not
playbackRate	Sets or returns the speed of the audio/video playback
played	Returns a TimeRanges object representing the played parts of the audio/video
preload	Sets or returns whether the audio/video should be loaded when the page loads
readyState	Returns the current ready state of the audio/video
seekable	Returns a TimeRanges object representing the seekable parts of the audio/video
seeking	Returns whether the user is currently seeking in the audio/video
src	Sets or returns the current source of the audio/video element
startDate	Returns aÂ Date object representing the current time offset
textTracks	Returns a TextTrackList object representing the available text tracks
videoTracks	Returns a VideoTrackList object representing the available video tracks
volume	Sets or returns the volume of the audio/video

HTML Audio/Video Events

Event	Description
abort	Fires when the loading of an audio/video is aborted
canplay	Fires when the browser can start playing the audio/video
canplaythrough	Fires when the browser can play through the audio/video without stopping for buffering
durationchange	Fires when the duration of the audio/video is changed
emptied	Fires when the current playlist is empty
ended	Fires when the current playlist is ended
error	Fires when an error occurred during the loading of an audio/video
loadeddata	Fires when the browser has loaded the current frame of the audio/video
loadedmetadata	Fires when the browser has loaded meta data for the audio/video
loadstart	Fires when the browser starts looking for the audio/video
pause	Fires when the audio/video has been paused
play	Fires when the audio/video has been started or is no longer paused
playing	Fires when the audio/video is playing after having been paused or stopped for buffering
progress	Fires when the browser is downloading the audio/video
ratechange	Fires when the playing speed of the audio/video is changed
seeked	Fires when the user is finished moving/skipping to a new position in the audio/video
seeking	Fires when the user starts moving/skipping to a new position in the audio/video
stalled	Fires when the browser is trying to get media data, but data is not available
suspend	Fires when the browser is intentionally not getting media data
timeupdate	Fires when the current playback position has changed
volumechange	Fires when the volume has been changed
waiting	Fires when the video stops because it needs to buffer the next frame

HTML <!DOCTYPE>

The HTML Document Type

All HTML documents must start with a <!DOCTYPE> declaration.

The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect.

In HTML5, the <!DOCTYPE> declaration is simple:

<!DOCTYPE html>

In older documents (HTML 4 or XHTML), the declaration is more complicated because the declaration must refer to a DTD (Document Type Definition).

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

You can read more about document types in the <!DOCTYPE> reference.

Valid HTML Elements in Different DOCTYPES

Tag	HTML 5	HTML 4	XHTML
<a>	Yes	Yes	Yes
<abbr>	Yes	Yes	Yes
<acronym>	No	Yes	Yes
<address>	Yes	Yes	Yes
<applet>	No	Yes	No
<area>	Yes	Yes	No
<article>	Yes	No	No
<aside>	Yes	No	No
<audio>	Yes	No	No
<b>	Yes	Yes	Yes
<base>	Yes	Yes	Yes
<basefont>	No	Yes	No
<bdi>	Yes	No	No
<bdo>	Yes	Yes	No
<big>	No	Yes	Yes
<blockquote>	Yes	Yes	Yes
<body>	Yes	Yes	Yes
 	Yes	Yes	Yes
<button>	Yes	Yes	Yes
<canvas>	Yes	No	No
<caption>	Yes	Yes	Yes
	No	Yes	No
<cite>	Yes	Yes	Yes
<code>	Yes	Yes	Yes
<col>	Yes	Yes	No
<colgroup>	Yes	Yes	No
<datalist>	Yes	No	No
<dd>	Yes	Yes	Yes
<del>	Yes	Yes	No
<details>	Yes	No	No
<dfn>	Yes	Yes	Yes
<dialog>	Yes	No	No
<dir>	No	Yes	No
	Yes	Yes	Yes
<dl>	Yes	Yes	Yes
<dt>	Yes	Yes	Yes
	Yes	Yes	Yes
<embed>	Yes	No	No

<fieldset>	Yes	Yes	Yes
<figcaption>	Yes	No	No
<figure>	Yes	No	No
<font>	No	Yes	No
<footer>	Yes	No	No
<form>	Yes	Yes	Yes
<frame>	No	No	No
<frameset>	No	Yes	No
<h1> to <h6>	Yes	Yes	Yes
<head>	Yes	Yes	Yes
<header>	Yes	No	No
<hr>	Yes	Yes	Yes
<html>	Yes	Yes	Yes
<i>	Yes	Yes	Yes
<iframe>	Yes	Yes	No
<img>	Yes	Yes	Yes
<input>	Yes	Yes	Yes
<ins>	Yes	Yes	No
<kbd>	Yes	Yes	Yes
<label>	Yes	Yes	Yes
<legend>	Yes	Yes	Yes
	Yes	Yes	Yes
<link>	Yes	Yes	Yes
<main>	Yes	No	No
<map>	Yes	Yes	No
<mark>	Yes	No	No
<meta>	Yes	Yes	Yes
<meter>	Yes	No	No
<nav>	Yes	No	No
<noframes>	No	Yes	No
<noscript>	Yes	Yes	Yes
<object>	Yes	Yes	Yes
	Yes	Yes	Yes
<optgroup>	Yes	Yes	Yes
<option>	Yes	Yes	Yes
<output>	Yes	No	No
<p>	Yes	Yes	Yes
<param>	Yes	Yes	Yes
<pre>	Yes	Yes	Yes
<progress>	Yes	No	No
<q>	Yes	Yes	Yes
<rp>	Yes	No	No
<rt>	Yes	No	No
<ruby>	Yes	No	No
<s>	Yes	Yes	No
<samp>	Yes	Yes	Yes
<script>	Yes	Yes	Yes
<section>	Yes	No	No
<select>	Yes	Yes	Yes
<small>	Yes	Yes	Yes
<source>	Yes	No	No
	Yes	Yes	Yes
<strike>	No	Yes	No
	Yes	Yes	Yes
<style>	Yes	Yes	Yes
<sub>	Yes	Yes	Yes
<summary>	Yes	No	No
<sup>	Yes	Yes	Yes
<table>	Yes	Yes	Yes
<tbody>	Yes	Yes	No
<td>	Yes	Yes	Yes
<textarea>	Yes	Yes	Yes

<tfoot>	Yes	Yes	No
<th>	Yes	Yes	Yes
<thead>	Yes	Yes	No
<time>	Yes	No	No
<title>	Yes	Yes	Yes
<tr>	Yes	Yes	Yes
<track>	Yes	No	No
<tt>	No	Yes	Yes
<u>	Yes	Yes	No
	Yes	Yes	Yes
<var>	Yes	Yes	Yes
<video>	Yes	No	No
<wbr>	Yes	No	No

HTML Character Sets

Common HTML Character Sets

The default character set in HTML5 is UTF-8.

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

Number	ASCII	ANSI	8859-1	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	Â	Â	Â	Â
128	Â	â,¬	Â	Â	euro sign
129	Â	Â	Â	Â	NOT USED

130	Â	â€š	Â	Â	single low-9 quotation mark
131	Â	Æ’	Â	Â	Latin small letter f with hook
132	Â	â€ž	Â	Â	double low-9 quotation mark
133	Â	â€¦	Â	Â	horizontal ellipsis
134	Â	â€	Â	Â	dagger
135	Â	â€¡	Â	Â	double dagger
136	Â	Ë†	Â	Â	modifier letter circumflex accent
137	Â	â€°	Â	Â	per mille sign
138	Â	Å	Â	Â	Latin capital letter S with caron
139	Â	â€¹	Â	Â	single left-pointing angle quotation mark
140	Â	Å’	Â	Â	Latin capital ligature OE
141	Â	Â	Â	Â	NOT USED
142	Â	Å½	Â	Â	Latin capital letter Z with caron
143	Â	Â	Â	Â	NOT USED
144	Â	Â	Â	Â	NOT USED
145	Â	â€˜	Â	Â	left single quotation mark
146	Â	â€™	Â	Â	right single quotation mark
147	Â	â€œ	Â	Â	left double quotation mark
148	Â	â€	Â	Â	right double quotation mark
149	Â	â€¢	Â	Â	bullet
150	Â	â€“	Â	Â	en dash
151	Â	â€”	Â	Â	em dash
152	Â	Ëœ	Â	Â	small tilde
153	Â	â„¢	Â	Â	trade mark sign
154	Â	Å¡	Â	Â	Latin small letter s with caron
155	Â	â€º	Â	Â	single right-pointing angle quotation mark
156	Â	Å“	Â	Â	Latin small ligature oe
157	Â	Â	Â	Â	NOT USED
158	Â	Å¾	Â	Â	Latin small letter z with caron
159	Â	Å¸	Â	Â	Latin capital letter Y with diaeresis
160	Â	Â	Â	Â	no-break space
161	Â	Â¡	Â¡	Â¡	inverted exclamation mark
162	Â	Â¢	Â¢	Â¢	cent sign
163	Â	Â£	Â£	Â£	pound sign
164	Â	Â¤	Â¤	Â¤	currency sign
165	Â	Â¥	Â¥	Â¥	yen sign
166	Â	Â¦	Â¦	Â¦	broken bar
167	Â	Â§	Â§	Â§	section sign
168	Â	Â¨	Â¨	Â¨	diaeresis
169	Â	Â©	Â©	Â©	copyright sign
170	Â	Âª	Âª	Âª	feminine ordinal indicator
171	Â	Â«	Â«	Â«	left-pointing double angle quotation mark
172	Â	Â¬	Â¬	Â¬	not sign
173	Â	Â	Â	Â	soft hyphen
174	Â	Â®	Â®	Â®	registered sign
175	Â	Â¯	Â¯	Â¯	macron
176	Â	Â°	Â°	Â°	degree sign
177	Â	Â±	Â±	Â±	plus-minus sign
178	Â	Â²	Â²	Â²	superscript two
179	Â	Â³	Â³	Â³	superscript three
180	Â	Â´	Â´	Â´	acute accent
181	Â	Âµ	Âµ	Âµ	micro sign
182	Â	Â¶	Â¶	Â¶	pilcrow sign
183	Â	Â·	Â·	Â·	middle dot
184	Â	Â¸	Â¸	Â¸	cedilla
185	Â	Â¹	Â¹	Â¹	superscript one
186	Â	Âº	Âº	Âº	masculine ordinal indicator
187	Â	Â»	Â»	Â»	right-pointing double angle quotation mark
188	Â	Â¼	Â¼	Â¼	vulgar fraction one quarter
189	Â	Â½	Â½	Â½	vulgar fraction one half
190	Â	Â¾	Â¾	Â¾	vulgar fraction three quarters
191	Â	Â¿	Â¿	Â¿	inverted question mark

192	Â	Ã€	Ã€	Ã€	Latin capital letter A with grave
193	Â	Ã	Ã	Ã	Latin capital letter A with acute
194	Â	Ã,	Ã,	Ã,	Latin capital letter A with circumflex
195	Â	Ãƒ	Ãƒ	Ãƒ	Latin capital letter A with tilde
196	Â	Ã„	Ã„	Ã„	Latin capital letter A with diaeresis
197	Â	Ã…	Ã…	Ã…	Latin capital letter A with ring above
198	Â	Ã†	Ã†	Ã†	Latin capital letter AE
199	Â	Ã‡	Ã‡	Ã‡	Latin capital letter C with cedilla
200	Â	Ãˆ	Ãˆ	Ãˆ	Latin capital letter E with grave
201	Â	Ã‰	Ã‰	Ã‰	Latin capital letter E with acute
202	Â	ÃŠ	ÃŠ	ÃŠ	Latin capital letter E with circumflex
203	Â	Ã‹	Ã‹	Ã‹	Latin capital letter E with diaeresis
204	Â	ÃŒ	ÃŒ	ÃŒ	Latin capital letter I with grave
205	Â	Ã	Ã	Ã	Latin capital letter I with acute
206	Â	ÃŽ	ÃŽ	ÃŽ	Latin capital letter I with circumflex
207	Â	Ã	Ã	Ã	Latin capital letter I with diaeresis
208	Â	Ã	Ã	Ã	Latin capital letter Eth
209	Â	Ã‘	Ã‘	Ã‘	Latin capital letter N with tilde
210	Â	Ã’	Ã’	Ã’	Latin capital letter O with grave
211	Â	Ã“	Ã“	Ã“	Latin capital letter O with acute
212	Â	Ã”	Ã”	Ã”	Latin capital letter O with circumflex
213	Â	Ã•	Ã•	Ã•	Latin capital letter O with tilde
214	Â	Ã–	Ã–	Ã–	Latin capital letter O with diaeresis
215	Â	Ã—	Ã—	Ã—	multiplication sign
216	Â	Ã˜	Ã˜	Ã˜	Latin capital letter O with stroke
217	Â	Ã™	Ã™	Ã™	Latin capital letter U with grave
218	Â	Ãš	Ãš	Ãš	Latin capital letter U with acute
219	Â	Ã›	Ã›	Ã›	Latin capital letter U with circumflex
220	Â	Ãœ	Ãœ	Ãœ	Latin capital letter U with diaeresis
221	Â	Ã	Ã	Ã	Latin capital letter Y with acute
222	Â	Ãž	Ãž	Ãž	Latin capital letter Thorn
223	Â	ÃŸ	ÃŸ	ÃŸ	Latin small letter sharp s
224	Â	Ã	Ã	Ã	Latin small letter a with grave
225	Â	Ã¡	Ã¡	Ã¡	Latin small letter a with acute
226	Â	Ã¢	Ã¢	Ã¢	Latin small letter a with circumflex
227	Â	Ã£	Ã£	Ã£	Latin small letter a with tilde
228	Â	Ã¤	Ã¤	Ã¤	Latin small letter a with diaeresis
229	Â	Ã¥	Ã¥	Ã¥	Latin small letter a with ring above
230	Â	Ã¦	Ã¦	Ã¦	Latin small letter ae
231	Â	Ã§	Ã§	Ã§	Latin small letter c with cedilla
232	Â	Ã¨	Ã¨	Ã¨	Latin small letter e with grave
233	Â	Ã©	Ã©	Ã©	Latin small letter e with acute
234	Â	Ãª	Ãª	Ãª	Latin small letter e with circumflex
235	Â	Ã«	Ã«	Ã«	Latin small letter e with diaeresis
236	Â	Ã¬	Ã¬	Ã¬	Latin small letter i with grave
237	Â	Ã	Ã	Ã	Latin small letter i with acute
238	Â	Ã®	Ã®	Ã®	Latin small letter i with circumflex
239	Â	Ã¯	Ã¯	Ã¯	Latin small letter i with diaeresis
240	Â	Ã°	Ã°	Ã°	Latin small letter eth
241	Â	Ã±	Ã±	Ã±	Latin small letter n with tilde
242	Â	Ã²	Ã²	Ã²	Latin small letter o with grave
243	Â	Ã³	Ã³	Ã³	Latin small letter o with acute
244	Â	Ã´	Ã´	Ã´	Latin small letter o with circumflex
245	Â	Ãµ	Ãµ	Ãµ	Latin small letter o with tilde
246	Â	Ã¶	Ã¶	Ã¶	Latin small letter o with diaeresis
247	Â	Ã·	Ã·	Ã·	division sign
248	Â	Ã¸	Ã¸	Ã¸	Latin small letter o with stroke
249	Â	Ã¹	Ã¹	Ã¹	Latin small letter u with grave
250	Â	Ãº	Ãº	Ãº	Latin small letter u with acute
251	Â	Ã»	Ã»	Ã»	Latin small letter with circumflex
252	Â	Ã¼	Ã¼	Ã¼	Latin small letter u with diaeresis
253	Â	Ã½	Ã½	Ã½	Latin small letter y with acute

254	Â	Ã¾	Ã¾	Ã¾	Latin small letter thorn
255	Â	Ã¿	Ã¿	Ã¿	Latin small letter y with diaeresis

HTML URL Encoding Reference

URL - Uniform Resource Locator

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

The URL is the address of a web page, like: **https://www.w3schools.com.**

URL Encoding (Percent Encoding)

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

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 a valid ASCII format.

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

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

Try It Yourself

If you click the "Submit" button below, the browser will URL encode the input before it is sent to the server. A page at the server will display the received input.

Hello GÃƒÂ¼nter Submit

Try some other input and click Submit again.

URL Encoding Functions

In JavaScript, PHP, and ASP there are functions that can be used to URL encode a string.

PHP has the rawurlencode() function, and ASP has the Server.URLEncode() function.

In JavaScript you can use the **encodeURIComponent()** function.

Click the "URL Encode" button to see how the JavaScript function encodes the text.

Hello GÃƒÂ¼nter URL Encode

Note: The JavaScript function encodes space as %20.

ASCII Encoding Reference

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
space	%20	%20
!	%21	%21
"	%22	%22
#	%23	%23
$	%24	%24
%	%25	%25
&	%26	%26
'	%27	%27

(	%28	%28
)	%29	%29
*	%2A	%2A
+	%2B	%2B
,	%2C	%2C
-	%2D	%2D
.	%2E	%2E
/	%2F	%2F
0	%30	%30
1	%31	%31
2	%32	%32
3	%33	%33
4	%34	%34
5	%35	%35
6	%36	%36
7	%37	%37
8	%38	%38
9	%39	%39
:	%3A	%3A
;	%3B	%3B
<	%3C	%3C
=	%3D	%3D
>	%3E	%3E
?	%3F	%3F
@	%40	%40
A	%41	%41
B	%42	%42
C	%43	%43
D	%44	%44
E	%45	%45
F	%46	%46
G	%47	%47
H	%48	%48
I	%49	%49
J	%4A	%4A
K	%4B	%4B
L	%4C	%4C
M	%4D	%4D
N	%4E	%4E
O	%4F	%4F
P	%50	%50
Q	%51	%51
R	%52	%52
S	%53	%53
T	%54	%54
U	%55	%55
V	%56	%56
W	%57	%57
X	%58	%58
Y	%59	%59
Z	%5A	%5A
[	%5B	%5B
\	%5C	%5C
]	%5D	%5D
^	%5E	%5E
_	%5F	%5F
`	%60	%60
a	%61	%61
b	%62	%62
c	%63	%63
d	%64	%64
e	%65	%65

f	%66	%66
g	%67	%67
h	%68	%68
i	%69	%69
j	%6A	%6A
k	%6B	%6B
l	%6C	%6C
m	%6D	%6D
n	%6E	%6E
o	%6F	%6F
p	%70	%70
q	%71	%71
r	%72	%72
s	%73	%73
t	%74	%74
u	%75	%75
v	%76	%76
w	%77	%77
x	%78	%78
y	%79	%79
z	%7A	%7A
{	%7B	%7B
|	%7C	%7C
}	%7D	%7D
~	%7E	%7E
Â	%7F	%7F
`	%80	%E2%82%AC
Ã‚Â	%81	%81
Ã¢Â€Âš	%82	%E2%80%9A
Ã†Â’	%83	%C6%92
Ã¢Â€Âž	%84	%E2%80%9E
Ã¢Â€Â¦	%85	%E2%80%A6
Ã¢Â€Â	%86	%E2%80%A0
Ã¢Â€Â¡	%87	%E2%80%A1
Ã‹Â†	%88	%CB%86
Ã¢Â€Â°	%89	%E2%80%B0
Ã…Â	%8A	%C5%A0
Ã¢Â€Â¹	%8B	%E2%80%B9
Ã…Â’	%8C	%C5%92
Ã‚Â	%8D	%C5%8D
Ã…Â½	%8E	%C5%BD
Ã‚Â	%8F	%8F
Ã‚Â	%90	%C2%90
Ã¢Â€Â˜	%91	%E2%80%98
Ã¢Â€Â™	%92	%E2%80%99
Ã¢Â€Âœ	%93	%E2%80%9C
Ã¢Â€Â	%94	%E2%80%9D
Ã¢Â€Â¢	%95	%E2%80%A2
Ã¢Â€Â“	%96	%E2%80%93
Ã¢Â€Â”	%97	%E2%80%94
Ã‹Âœ	%98	%CB%9C
Ã¢Â„Â¢	%99	%E2%84
Ã…Â¡	%9A	%C5%A1
Ã¢Â€Âº	%9B	%E2%80
Ã…Â“	%9C	%C5%93
Ã‚Â	%9D	%9D
Ã…Â¾	%9E	%C5%BE
Ã…Â¸	%9F	%C5%B8
Â	%A0	%C2%A0
Ã‚Â¡	%A1	%C2%A1
Ã‚Â¢	%A2	%C2%A2
Ã‚Â£	%A3	%C2%A3

Ã‚Â¤	%A4	%C2%A4
Ã‚Â¥	%A5	%C2%A5
Ã‚Â¦	%A6	%C2%A6
Ã‚Â§	%A7	%C2%A7
Ã‚Â¨	%A8	%C2%A8
Ã‚Â©	%A9	%C2%A9
Ã‚Âª	%AA	%C2%AA
Ã‚Â«	%AB	%C2%AB
Ã‚Â¬	%AC	%C2%AC
Ã‚Â	%AD	%C2%AD
Ã‚Â®	%AE	%C2%AE
Ã‚Â¯	%AF	%C2%AF
Ã‚Â°	%B0	%C2%B0
Ã‚Â±	%B1	%C2%B1
Ã‚Â²	%B2	%C2%B2
Ã‚Â³	%B3	%C2%B3
Ã‚Â´	%B4	%C2%B4
Ã‚Âµ	%B5	%C2%B5
Ã‚Â¶	%B6	%C2%B6
Ã‚Â·	%B7	%C2%B7
Ã‚Â¸	%B8	%C2%B8
Ã‚Â¹	%B9	%C2%B9
Ã‚Âº	%BA	%C2%BA
Ã‚Â»	%BB	%C2%BB
Ã‚Â¼	%BC	%C2%BC
Ã‚Â½	%BD	%C2%BD
Ã‚Â¾	%BE	%C2%BE
Ã‚Â¿	%BF	%C2%BF
ÃƒÂ€	%C0	%C3%80
ÃƒÂ	%C1	%C3%81
ÃƒÂ‚	%C2	%C3%82
ÃƒÂƒ	%C3	%C3%83
ÃƒÂ„	%C4	%C3%84
ÃƒÂ…	%C5	%C3%85
ÃƒÂ†	%C6	%C3%86
ÃƒÂ‡	%C7	%C3%87
ÃƒÂˆ	%C8	%C3%88
ÃƒÂ‰	%C9	%C3%89
ÃƒÂŠ	%CA	%C3%8A
ÃƒÂ‹	%CB	%C3%8B
ÃƒÂŒ	%CC	%C3%8C
ÃƒÂ	%CD	%C3%8D
ÃƒÂŽ	%CE	%C3%8E
ÃƒÂ	%CF	%C3%8F
ÃƒÂ	%D0	%C3%90
ÃƒÂ‘	%D1	%C3%91
ÃƒÂ’	%D2	%C3%92
ÃƒÂ“	%D3	%C3%93
ÃƒÂ”	%D4	%C3%94
ÃƒÂ•	%D5	%C3%95
ÃƒÂ–	%D6	%C3%96
ÃƒÂ—	%D7	%C3%97
ÃƒÂ˜	%D8	%C3%98
ÃƒÂ™	%D9	%C3%99
ÃƒÂš	%DA	%C3%9A
ÃƒÂ›	%DB	%C3%9B
ÃƒÂœ	%DC	%C3%9C
ÃƒÂ	%DD	%C3%9D
ÃƒÂž	%DE	%C3%9E
ÃƒÂŸ	%DF	%C3%9F
ÃƒÂ	%E0	%C3%A0
ÃƒÂ¡	%E1	%C3%A1

ÃƒÂ¢	%E2	%C3%A2
ÃƒÂ£	%E3	%C3%A3
ÃƒÂ¤	%E4	%C3%A4
ÃƒÂ¥	%E5	%C3%A5
ÃƒÂ¦	%E6	%C3%A6
ÃƒÂ§	%E7	%C3%A7
ÃƒÂ¨	%E8	%C3%A8
ÃƒÂ©	%E9	%C3%A9
ÃƒÂª	%EA	%C3%AA
ÃƒÂ«	%EB	%C3%AB
ÃƒÂ¬	%EC	%C3%AC
ÃƒÂ	%ED	%C3%AD
ÃƒÂ®	%EE	%C3%AE
ÃƒÂ¯	%EF	%C3%AF
ÃƒÂ°	%F0	%C3%B0
ÃƒÂ±	%F1	%C3%B1
ÃƒÂ²	%F2	%C3%B2
ÃƒÂ³	%F3	%C3%B3
ÃƒÂ´	%F4	%C3%B4
ÃƒÂµ	%F5	%C3%B5
ÃƒÂ¶	%F6	%C3%B6
ÃƒÂ·	%F7	%C3%B7
ÃƒÂ¸	%F8	%C3%B8
ÃƒÂ¹	%F9	%C3%B9
ÃƒÂº	%FA	%C3%BA
ÃƒÂ»	%FB	%C3%BB
ÃƒÂ¼	%FC	%C3%BC
ÃƒÂ½	%FD	%C3%BD
ÃƒÂ¾	%FE	%C3%BE
ÃƒÂ¿	%FF	%C3%BF

URL Encoding Reference

The ASCII control characters **%00-%1F** were originally designed to control hardware devices.

Control characters have nothing to do inside a URL.

ASCII Character	Description	URL-encoding
NUL	null character	%00
SOH	start of header	%01
STX	start of text	%02
ETX	end of text	%03
EOT	end of transmission	%04
ENQ	enquiry	%05
ACK	acknowledge	%06
BEL	bell (ring)	%07
BS	backspace	%08
HT	horizontal tab	%09
LF	line feed	%0A
VT	vertical tab	%0B
FF	form feed	%0C
CR	carriage return	%0D
SO	shift out	%0E
SI	shift in	%0F
DLE	data link escape	%10
DC1	device control 1	%11
DC2	device control 2	%12
DC3	device control 3	%13
DC4	device control 4	%14
NAK	negative acknowledge	%15
SYN	synchronize	%16
ETB	end transmission block	%17

CAN	cancel	%18
EM	end of medium	%19
SUB	substitute	%1A
ESC	escape	%1B
FS	file separator	%1C
GS	group separator	%1D
RS	record separator	%1E
US	unit separator	%1F

HTML Language Code Reference

ISO Language Codes

You should always include the `lang` attribute inside the `<html>` tag, to declare the language of the Web page. This is meant to assist search engines and browsers:

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

In XHTML, the language is declared inside the `<html>` tag as follows:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
...
</html>
```

ISO 639-1 Language Codes

ISO 639-1 defines abbreviations for languages:

See also: Reference for Country Codes.

Language	ISO Code
Abkhazian	ab
Afar	aa
Afrikaans	af
Akan	ak
Albanian	sq
Amharic	am
Arabic	ar
Aragonese	an
Armenian	hy
Assamese	as
Avaric	av
Avestan	ae
Aymara	ay
Azerbaijani	az
Bambara	bm
Bashkir	ba
Basque	eu
Belarusian	be
Bengali (Bangla)	bn
Bihari	bh
Bislama	bi
Bosnian	bs
Breton	br
Bulgarian	bg
Burmese	my
Catalan	ca
Chamorro	ch
Chechen	ce
Chichewa, Chewa, Nyanja	ny

Chinese	zh
Chinese (Simplified)	zh-Hans
Chinese (Traditional)	zh-Hant
Chuvash	cv
Cornish	kw
Corsican	co
Cree	cr
Croatian	hr
Czech	cs
Danish	da
Divehi, Dhivehi, Maldivian	dv
Dutch	nl
Dzongkha	dz
English	en
Esperanto	eo
Estonian	et
Ewe	ee
Faroese	fo
Fijian	fj
Finnish	fi
French	fr
Fula, Fulah, Pulaar, Pular	ff
Galician	gl
Gaelic (Scottish)	gd
Gaelic (Manx)	gv
Georgian	ka
German	de
Greek	el
Greenlandic	kl
Guarani	gn
Gujarati	gu
Haitian Creole	ht
Hausa	ha
Hebrew	he
Herero	hz
Hindi	hi
Hiri Motu	ho
Hungarian	hu
Icelandic	is
Ido	io
Igbo	ig
Indonesian	id, in
Interlingua	ia
Interlingue	ie
Inuktitut	iu
Inupiak	ik
Irish	ga
Italian	it
Japanese	ja
Javanese	jv
Kalaallisut, Greenlandic	kl
Kannada	kn
Kanuri	kr
Kashmiri	ks
Kazakh	kk
Khmer	km
Kikuyu	ki
Kinyarwanda (Rwanda)	rw
Kirundi	rn
Kyrgyz	ky
Komi	kv
Kongo	kg

Korean	ko
Kurdish	ku
Kwanyama	kj
Lao	lo
Latin	la
Latvian (Lettish)	lv
Limburgish (Limburger)	li
Lingala	ln
Lithuanian	lt
Luga-Katanga	lu
Luganda, Ganda	lg
Luxembourgish	lb
Manx	gv
Macedonian	mk
Malagasy	mg
Malay	ms
Malayalam	ml
Maltese	mt
Maori	mi
Marathi	mr
Marshallese	mh
Moldavian	mo
Mongolian	mn
Nauru	na
Navajo	nv
Ndonga	ng
Northern Ndebele	nd
Nepali	ne
Norwegian	no
Norwegian bokmÃƒÂ¥l	nb
Norwegian nynorsk	nn
Nuosu	ii
Occitan	oc
Ojibwe	oj
Old Church Slavonic, Old Bulgarian	cu
Oriya	or
Oromo (Afaan Oromo)	om
Ossetian	os
PÃ„Âli	pi
Pashto, Pushto	ps
Persian (Farsi)	fa
Polish	pl
Portuguese	pt
Punjabi (Eastern)	pa
Quechua	qu
Romansh	rm
Romanian	ro
Russian	ru
Sami	se
Samoan	sm
Sango	sg
Sanskrit	sa
Serbian	sr
Serbo-Croatian	sh
Sesotho	st
Setswana	tn
Shona	sn
Sichuan Yi	ii
Sindhi	sd
Sinhalese	si
Siswati	ss
Slovak	sk

Slovenian	sl
Somali	so
Southern Ndebele	nr
Spanish	es
Sundanese	su
Swahili (Kiswahili)	sw
Swati	ss
Swedish	sv
Tagalog	tl
Tahitian	ty
Tajik	tg
Tamil	ta
Tatar	tt
Telugu	te
Thai	th
Tibetan	bo
Tigrinya	ti
Tonga	to
Tsonga	ts
Turkish	tr
Turkmen	tk
Twi	tw
Uyghur	ug
Ukrainian	uk
Urdu	ur
Uzbek	uz
Venda	ve
Vietnamese	vi
VolapÃƒÂ¼k	vo
Wallon	wa
Welsh	cy
Wolof	wo
Western Frisian	fy
Xhosa	xh
Yiddish	yi, ji
Yoruba	yo
Zhuang, Chuang	za
Zulu	zu

HTTP Status Messages

HTML Error Messages

When a browser requests a service from a web server, an error might occur, and the server might return an error code like "404 Not Found".

It is common to name these errors HTML error messages.

But these messages are something called HTTP status messages. In fact, the server always returns a message for every request. The most common message is 200 OK.

Below is a list of HTTP status messages that might be returned:

1xx: Information

Message:	Description:
100 Continue	The server has received the request headers, and the client should proceed to send the request body
101 Switching Protocols	The requester has asked the server to switch protocols
	Used in the resumable requests proposal to resume aborted PUT or

103 Checkpoint	POST requests

2xx: Successful

Message:	Description:
200 OK	The request is OK (this is the standard response for successful HTTP requests)
201 Created	The request has been fulfilled, and a new resource is createdÂ
202 Accepted	The request has been accepted for processing, but the processing has not been completed
203 Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204 No Content	The request has been successfully processed, but is not returning any content
205 Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view
206 Partial Content	The server is delivering only part of the resource due to a range header sent by the client

3xx: Redirection

Message:	Description:
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses Â
301 Moved Permanently	The requested page has moved to a new URLÂ
302 Found	The requested page has moved temporarily to a new URLÂ
303 See Other	The requested page can be found under a different URL
304 Not Modified	Indicates the requested page has not been modified since last requested
306 Switch Proxy	*No longer used*
307 Temporary Redirect	The requested page has moved temporarily to a new URL
308 Resume Incomplete	Used in the resumable requests proposal to resume aborted PUT or POST requests

---

4xx: Client Error

Message:	Description:
400Â Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	*Reserved for future use*
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client
407 Proxy Authentication Required	The client must first authenticate itself with the proxy
408 Request Timeout	The server timed out waiting for the request
409 Conflict	The request could not be completed because of a conflict in the request
410 Gone	The requested page is no longer available
411 Length Required	The "Content-Length" is not defined. The server will not accept the request without itÂ
412 Precondition Failed	The precondition given in the request evaluated to false by the server
413 Request Entity Too Large	The server will not accept the request, because the request entity is too large
414 Request-URI Too Long	The server will not accept the request, because the URL is too long. Occurs when you convert a POST request to a GET request with a long query informationÂ
415 Unsupported Media Type	The server will not accept the request, because the media type is not

	supportedÂ
416Â Requested Range Not Satisfiable	The client has asked for a portion of the file, but the server cannot supply that portion
417 Expectation Failed	The server cannot meet the requirements of the Expect request-header field

5xx: Server Error

Message:	Description:
500Â Internal Server Error	A generic error message, given when no more specific message is suitable
501 Not Implemented	The server either does not recognize the request method, or it lacks the ability to fulfill the request
502 Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503 Service Unavailable	The server is currently unavailable (overloaded or down)
504 Gateway Timeout	The server was acting as a gateway or proxy and did not receive a timely response from the upstream server
505 HTTP Version Not Supported	The server does not support the HTTP protocol version used in the request
511 Network Authentication Required	The client needs to authenticate to gain network access

HTTP Request Methods

What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

HTTP Methods

- **GET**
- **POST**
- **PUT**
- **HEAD**
- **DELETE**
- **PATCH**
- **OPTIONS**

The two most common HTTP methods are: GET and POST.

The GET Method

GET is used to request data from a specified resource.

GET is one of the most common HTTP methods.

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo_form.php?name1=value1&name2=value2

Some other notes on GET requests:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests are only used to request data (not modify)

The POST Method

POST is used to send data to a server to create/update a resource.

The data sent to the server with POST is stored in the request body of the HTTP request:

POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2

POST is one of the most common HTTP methods.

Some other notes on POST requests:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

The PUT Method

PUT is used to send data to a server to create/update a resource.

The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

The HEAD Method

HEAD is almost identical to GET, but without the response body.

In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users.

HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

The DELETE Method

The DELETE method deletes the specified resource.

The OPTIONS Method

The OPTIONS method describes the communication options for the target resource.

Compare GET vs. POST

The following table compares the two HTTP methods: GET and POST.

Â	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed

Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Pixels to Ems Conversion

Pixel to Em Converter

The tool below allows you to work out the em sizes from pixels (or vice versa).

- Set a default pixel size for body (usually 16px)
- Then, convert a pixel value to em, based on the default pixel size
- Or, convert an em value to pixels, based on the default pixel size

Set a default pixel size:

16 px

Convert PX to EM:

px

Convert EM to PX:

em

Convert

Result:

Body Font Size

In the table below, select a body font size in pixels (px) to display a complete "px to em and percent" conversion table.

Tip: The default font size is usually 16px.

What is the difference between PX, EM and Percent?

Pixel is a static measurement, while percent and EM are relative measurements. The size of an EM or percent depends on its parent. If the text size of body is 16 pixels, then 150% or 1.5 EM will be 24 pixels (1.5 * 16). Look at CSS Units for more measurement units.

Keyboard Shortcuts

Keyboard Shortcuts For Windows and Mac

Keyboard shortcuts are often used in modern operating systems and computer software programs.

Learning and using keyboard shortcuts can save you a lot of time.

Basic Shortcuts

Description	Windows	Mac OS
Edit menu	Alt + E	Ctrl + F2 + F
File menu	Alt + F	Ctrl + F2 + E
View menu	Alt + V	Ctrl + F2 + V
Select all text	Ctrl + A	Cmd + A
Copy text	Ctrl + C	Cmd + C
Find text	Ctrl + F	Cmd + F
Find and replace text	Ctrl + H	Cmd + F
New Document	Ctrl + N	Cmd + N
Open a file	Ctrl + O	Cmd + O
Print options	Ctrl + P	Cmd + P
Save file	Ctrl + S	Cmd + S
Paste text	Ctrl + V	Cmd + V
Cut text	Ctrl + X	Cmd + X
Redo text	Ctrl + Y	Shift + Cmd + Z
Undo text	Ctrl + Z	Cmd + Z

Text Editing

Description	Windows	Mac OS
Cursor Movement		
Go to the right or to the beginning of next line break	Right Arrow	Right Arrow
Go to the left or to the end of previous line break	Left Arrow	Left Arrow
Go up one row	Up Arrow	Up Arrow
Go down one row	Down Arrow	Down Arrow
Go to the beginning of the current line	Home	Cmd + Left Arrow
Go to the end of the current line	End	Cmd + Right Arrow
Go to the beginning of the document	Ctrl + Home	Cmd + Up Arrow
Go to the end of the document	Ctrl + End	Cmd + Down Arrow
Move up one frame	Page Up	Fn + Up Arrow
Move down one frame	Page Down	Fn + Down Arrow
Go to beginning of previous word	Ctrl + Left Arrow	Option + Left Arrow
Go to beginning of next word	Ctrl + Right Arrow	Option + Right Arrow
Go to beginning of line break	Ctrl + Up Arrow	Cmd + Left Arrow
Go to end of line break	Ctrl + Down Arrow	Cmd + Right Arrow
Â	Â	Â
Text Selection		
Select characters to the left	Shift + Left Arrow	Shift + Left Arrow
Select characters to the right	Shift + Right Arrow	Shift + Right Arrow
Select lines upwards	Shift + Up Arrow	Shift + Up Arrow
Select lines downwards	Shift + Down Arrow	Shift + Down Arrow
Select words to the left	Shift + Ctrl + Left	Shift + Opt + Left
Select words to the right	Shift + Ctrl + Right	Shift + Opt + Right
Select paragraphs to the left	Shift + Ctrl + Up	Shift + Opt + Up
Select paragraphs to the right	Shift + Ctrl + Down	Shift + Opt + Down
Select text between the cursor and the beginning of the current line	Shift + Home	Cmd + Shift + Left Arrow
Select text between the cursor and the end of the current line	Shift + End	Cmd + Shift + Right Arrow
Select text between the cursor and the beginning of the document	Shift + Ctrl + Home	Cmd + Shift + Up Arrow or Cmd + Shift + Fn + Left Arrow
Select text between the cursor and the end of the document	Shift + Ctrl + End	Cmd + Shift + Down Arrow or Cmd + Shift + Fn + Right Arrow
Select one frame at a time of text above the cursor	Shift + Page Up	Shift + Fn + Up Arrow
Select one frame at a time of text below the cursor	Shift + Page Down	Shift + Fn + Down Arrow
Select all text	Ctrl + A	Cmd + A
Find text	Ctrl + F	Cmd + F
Â	Â	Â
Text Formatting		

Make selected text bold	Ctrl + B	Cmd + B
Make selected text italic	Ctrl + I	Cmd + I
Underline selected text	Ctrl + U	Cmd + U
Make selected text superscript	Ctrl + Shift + =	Cmd + Shift + =
Make selected text subscript	Ctrl + =	Cmd + =
Â	Â	Â
Text Editing		
Delete characters to the left	Backspace	Backspace
Delete characters to the right	Delete	Fn + Backspace
Delete words to the right	Ctrl + Del	Cmd + Backspace
Delete words to the left	Ctrl + Backspace	Cmd + Fn + Backspace
Indent	Tab	Tab
Outdent	Shift + Tab	Shift + Tab
Copy text	Ctrl + C	Cmd + C
Find and replace text	Ctrl + H	Cmd + F
Paste text	Ctrl + V	Cmd + V
Cut text	Ctrl + X	Cmd + X
Redo text	Ctrl + Y	Shift + Cmd + Z
Undo text	Ctrl + Z	Cmd + Z

Web Browsers

Description	Windows	Mac OS
Navigation		
Scroll down a frame	Space or Page Down	Space or Fn + Down Arrow
Scroll up a frame	Shift + Space or Page Up	Shift + Space or Fn + Up Arrow
Go to bottom of the page	End	Cmd + Down Arrow
Go to top of the page	Home	Cmd + Up Arrow
Go back	Alt + Left Arrow or Backspace	Cmd + Left Arrow
Go forward	Alt + Right Arrow or Shift + Backspace	Cmd + Right Arrow
Refresh a webpage	F5	Cmd + R
Refresh a webpage (no cache)	Ctrl + F5	Cmd + Shift + R
Stop	Esc	Esc
Toggle full-screen	F11	Cmd + Shift + F
Zoom in	Ctrl + +	Cmd + +
Zoom out	Ctrl + -	Cmd + -
Zoom 100% (default)	Ctrl + 0	Cmd + 0
Open homepage	Alt + Home	Option + Home or Option + Fn + Left Arrow
Find text	Ctrl + F	Cmd + F
Â	Â	Â
Tab / Window Management		
Open a new tab	Ctrl + T	Cmd + T
Close current tab	Ctrl + W	Cmd + W
Close all tabs	Ctrl + Shift + W	Cmd + Q
Close all tabs except the current tab	Ctrl + Alt + F4	Cmd + Opt + W
Go to next tab	Ctrl + Tab	Control + Tab or Cmd + Shift + Right Arrow
Go to previous tab	Ctrl + Shift + Tab	Shift + Control + Tab or Cmd + Shift + Left Arrow
Go to a specific tab number	Ctrl + 1-8	Cmd + 1-8
Go to the last tab	Ctrl + 9	Cmd + 9
Reopen the last closed tab	Ctrl + Shift + T	Cmd + Shift + T
Open a new window	Ctrl + N	Cmd + N
Close current window	Alt + F4	Cmd + W
Go to next window	Alt + Tab	Cmd + Tab
Go to previous window	Alt + Shift + Tab	Cmd + Shift + Tab
Reopen the last closed window	Ctrl + Shift + N	Â
Open links in a new tab in the background	Ctrl + Click	Cmd + Click

Open links in a new tab in the foreground	Ctrl + Shift + Click	Cmd + Shift + Click
Print current webpage	Ctrl + P	Cmd + P
Save current webpage	Ctrl + S	Cmd + S
Â	Â	Â
Address Bar		
Cycle between toolbar, search bar, and page elements	Tab	Tab
Go to browser's address bar	Ctrl + L or Alt + D	Cmd + L
Focus and select the browser's search bar	Ctrl + E	Cmd + E / Cmd + K
Open the address bar location in a new tab	Alt + Enter	Opt + Enter
Display a list of previously typed addresses	F4	Â
Add "www." to the beginning and ".com" to the end of the text typed in the address bar (e.g., type "w3schools" and press Ctrl + Enter to open "www.w3schools.com")	Ctrl + Enter	Cmd + Enter or Control + Enter
Â	Â	Â
Bookmarks		
Open the bookmarks menu	Ctrl + B	Cmd + B
Add bookmark for current page	Ctrl + D	Cmd + Opt + B or Cmd + Shift + B
Open browsing history	Ctrl + H	Cmd + Shift + H or Cmd + Y
Open download history	Ctrl + J	Cmd + J or Cmd + Shift + J

Screenshots

Description	Windows	Mac OS
Save screenshot of the whole screen as file	Â	Cmd + Shift + 3
Copy screenshot of the whole screen to the clipboard	PrtScr (Print Screen) or Ctrl + PrtScr	Cmd + Ctrl + Shift + 3
Save screenshot of window as file	Â	Cmd + Shift + 4, then Space
Copy screenshot of window to the clipboard	Alt + PrtScr	Cmd + Ctrl + Shift + 4, then Space
Copy screenshot of wanted area to the clipboard		Cmd + Ctrl + Shift + 4
Save screenshot of wanted area as file	Â	Cmd + Shift + 4

Note: Due to different keyboard setups, some shortcuts may not be compatible for all users.