HTML Introduction

Example Explained

- The `<!DOCTYPE html>` declaration defines this document to be HTML5
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the document
- The `<title>` element specifies a title for the document
- The `<body>` element contains the visible page content
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

HTML Tags

HTML tags are element names surrounded by angle brackets:

<tagname>content goes here...</tagname>

- HTML tags normally come **in pairs** like `<p>` and `</p>`
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

Tip: The start tag is also called the **opening tag**, and the end tag the **closing tag**.

Web Browsers

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.
The 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: Only the content inside the <body> section (the white area above) is displayed in a browser.

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

HTML Headings

HTML headings are defined with the `<h1>` to `<h6>` tags.
`<h1>` defines the most important heading. `<h6>` defines the least important heading:

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. Attributes are used to provide additional information about HTML elements.
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">
```

HTML buttons are defined with the `<button>` tag:

Example

```
<button>Click me</button>
```

HTML Lists

HTML lists are defined with the `<ul>` (unordered/bullet list) or the `<ol>` (ordered/numbered list) tag, followed by `<li>` tags (list items):

Example

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

HTML elements with no content are called empty elements. Empty elements do not have an end tag, such as the
 element (which indicates a line break).

HTML Attributes

.All HTML elements can have **attributes**

Attributes provide **additional information** about an element

Attributes are always specified in **the start tag**

Attributes usually come in name/value pairs like: **name="value".**

The alt Attribute

The `alt` attribute specifies an alternative text to be used, when an image cannot be displayed.

The value of the attribute can be read by screen readers. This way, someone "listening" to the webpage, e.g. a blind person, can "hear" the element.

Example

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

The width and height Attributes

Images in HTML have a set of size attributes, which specifies the width and height of the image:

Example

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

<h2>Size Attributes</h2>
<p>Images in HTML have a set of size attributes, which specifies the width and height of the image:</p>

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

```
</body>
</html>
```

The title Attribute

Here, a `title` attribute is added to the `<p>` element. The value of the title attribute will be displayed as a tooltip when you mouse over the paragraph:

Example

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

Chapter Summary

- **All HTML elements can have attributes**
- **The `title` attribute provides additional "tool-tip" information**
- **The `href` attribute provides address information for linksThe `width` and `height` attributes provide size information for images**
- **The `alt` attribute provides text for screen readers**
- **At W3Schools we always use lowercase attribute names**
- **At W3Schools we always quote attribute values with double quotes.**
-

Below is an alphabetical list of some attributes often used in HTML:

Attribute	Description
alt	Specifies an alternative text for an image, when the image cannot be displayed
disabled	Specifies that an input element should be disabled
href	Specifies the URL (web address) for a link
id	Specifies a unique id for an element
src	Specifies the URL (web address) for an image
style	Specifies an inline CSS style for an element
title	Specifies extra information about an element (displayed as a tool tip)

Headings Are Important

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.

<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 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

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

<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>

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

</body>
</html>
```

Output

This is heading 1

This is some text.

This is heading 2

This is some other text.

This is heading 2

This is some other text.
These two lines is a effect of <hr> tag

The HTML <head> Element

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

The <head> element is a container for metadata. HTML metadata is data about the HTML document. Metadata is not displayed.

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

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>My First HTML</title>
  <meta charset="UTF-8">
</head>
<body>...
```

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 IE), 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 Tag Reference

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

HTML Display

You cannot be sure how HTML will be displayed.

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

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

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

HTML Global Attributes

= Attribute added in HTML5.

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
dropzone	Specifies whether the dragged data is copied, moved, or linked, when dropped
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

Don't Forget the End Tag

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

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

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

Note: Dropping the end tag can produce unexpected results or errors.

HTML Line Breaks

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

**Use
 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>
```

Output

This is
a paragraph
with line breaksutput

The HTML <pre> Element

The HTML `<pre>` element defines preformatted text.

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

Example

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

  My Bonnie lies over the sea.

  My Bonnie lies over the ocean.

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

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

HTML Styles

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.

HTML Background Color

The `background-color` property defines the background color for an HTML element.
This example sets the background color for a page to powderblue:

Example

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

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

</body>
```

HTML Text Color

The `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>
```

HTML Fonts

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

Example

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

HTML Text Size

The font-size property defines the text size for an HTML element:

Example
<h1 style="font-size:300%;">This is a heading</h1>
This is a paragraph.

HTML Text Alignment

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

Example

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

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

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

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

Formatting elements were designed to display special types of text:

1. `<b>` - Bold text
2. `<strong>` - Important text
3. `<i>` - Italic text
4. `<em>` - Emphasized text
5. `<mark>` - Marked text
6. `<small>` - Small text
7. `<del>` - Deleted text
8. `<ins>` - Inserted text
9. `<sub>` - Subscript text
10. `<sup>` - Superscript text

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

Example

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

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

Example

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

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

Example

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

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

Example
<h2>HTML <mark>Marked</mark> Formatting</h2>
The HTML <small> element defines smaller text:
Example
<h2>HTML <small>Small</small> Formatting</h2>
The HTML <del> element defines deleted (removed) text.

Example
My favorite color is ~~blue~~ red.
The HTML <ins> element defines inserted (added) text.

Example
My favorite <ins>color</ins> is red.
The HTML <sub> element defines subscripted text.
Example
This is _{subscripted} text.
The HTML <sup> element defines superscripted text.

Example
This is ^{superscripted} text.

HTML Text Formatting Elements

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

HTML Quotation and Citation Elements

HTML <q> for Short Quotations

The HTML <q> element defines a short quotation.

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

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

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 <abbr> for Abbreviations

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

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

Example

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

HTML <address> for Contact Information

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

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

Example

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

HTML <cite> for Work Title

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

Browsers usually display <cite> elements in italic.

Example

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

HTML <bdo> for Bi-Directional Override

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

The <bdo> element is used to override the current text direction:

Example

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

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

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 opening tag, but not in the closing 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:

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

This is a paragraph.

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

HTML Colors

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

Color Names
In HTML, a color can be specified by using a color name:
Tomato
Orange
DodgerBlue
MediumSeaGreen
Gray
SlateBlue
Violet
LightGray

HTML supports 140 standard color names.

Background Color

You can set the background color for HTML elements:

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:

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:

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:

Same as color name "Tomato":

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

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>
```

RGB Value

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 between 0 and 255.

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

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

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

Experiment by mixing the RGB values below:
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)

HEX Value

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 others are set to the lowest value (00).

Example
#ff0000
#0000ff
#3cb371
#ee82ee
#ffa500
#6a5acd

Shades of gray are often defined using equal values for all the 3 light sources:

Example
#000000
#3c3c3c
#787878
#b4b4b4
#f0f0f0
#ffffff

HSL Value

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, 0% is black, 50% is neither light or dark, 100% is white

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 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%, 0%)

hsl(0, 0%, 24%)
hsl(0, 0%, 47%)
hsl(0, 0%, 71%)
hsl(0, 0%, 94%)
hsl(0, 0%, 100%)

RGBA Value

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):

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)

HSLA Value

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):

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)

Styling HTML with CSS

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

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

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

CSS can be added to HTML elements in 3 ways:

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

The most common way to add CSS, is to keep the styles in separate CSS files. However, here we will use inline and internal styling, 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.

This example sets the text color of the <h1> element to blue:

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

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:

Example
```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {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.

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

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

Example

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

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

</body>
</html>
```

An 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 how the "styles.css" looks:

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

CSS Fonts

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

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

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

</body>
</html>
```

CSS Border

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

Example

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

CSS Padding

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

Example

```
p {
   border: 1px solid powderblue;
   padding: 30px;
}
```

CSS Margin

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

Example

```
p {
   border: 1px solid powderblue;
   margin: 50px;
}
```

The id Attribute

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

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

then define a style for the element with the specific id:

Example

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

Note: The id of an element should be unique within a page, so the id selector is used to select one unique element!

The class Attribute

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

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

then define a style for the elements with the specific class:

Example

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

External References

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

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

Example

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

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

Example

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

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

Example

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

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

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

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. It can be an image or any other HTML element.

HTML Links - Syntax

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

link text
Example
Visit our HTML tutorial
The href attribute specifies the destination address
(https://www.w3schools.com/html/) of the link.

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

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

Note: Without a forward slash at the end of subfolder addresses, you might generate two requests to the server. Many servers will automatically add a forward slash to the end of the address, and then create a new request.

Local Links

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

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

Example
HTML Images
<!DOCTYPE html>
<html>
<body>

<h2>Local Links</h2>

```
<p><a href="html_images.asp">HTML Images</a> is a link to a page on this website.</p>

<p><a href="https://www.w3.org/">W3C</a> is a link to a website on the World Wide Web.</p>

</body>
</html>
```

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 default colors, by using CSS:

Example

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

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

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

a:active {
   color: yellow;
   background-color: transparent;
   text-decoration: underline;
}
</style>
```

HTML Links - The target Attribute

The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

_blank - Opens the linked document in a new window or tab
_self - Opens the linked document in the same window/tab as it was clicked

(this is default)
_parent - Opens the linked document in the parent frame
_top - Opens the linked document in the full body of the window
framename - Opens the linked document in a named frame
This example will open the linked document in a new browser window/tab:

Example
Visit W3Schools!
Tip: If your webpage is locked in a frame, you can use target="_top" to break out of the frame:

Example
HTML5 tutorial!

HTML Links - Image as Link

It is common to use images as links:

Example

  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0;">

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

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
Visit our HTML Tutorial

HTML Links - Create a Bookmark

HTML bookmarks are used to allow readers to jump to specific parts of a Web page.

Bookmarks can be useful if your webpage is very long.

To make a bookmark, you must first create the bookmark, and then add a link to it.

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:
Jump to Chapter 4
Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:
Example
Jump to Chapter 4

External Paths

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

This example uses a full URL to link to a web page:

Example
HTML tutorial
This example links to a page located in the html folder on the current web site:

Example
HTML tutorial
This example links to a page located in the same folder as the current page:

Example
HTML tutorial

Chapter Summary

1.Use the <a> element to define a link
2.Use the href attribute to define the link address
3.Use the target attribute to define where to open the linked document
4.Use the <img> element (inside <a>) to use an image as a link
5.Use the id attribute (id="value") to define bookmarks in a page
6.Use the href attribute (href="#value") to link to the bookmark.

HTML Images Syntax

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

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

The src attribute specifies the URL (web address) of the image:

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

The alt Attribute

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

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 defines 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 page might flicker while the image loads.

Width and Height, or Style?

The width, height, and style attributes are valid in HTML5.

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

Example
<!DOCTYPE html>
<html>
<head>
<style>
img {

/*This stylesheet sets the width of all images to 100%: */

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

However, it is common to store images in a sub-folder. You must then 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

Some web sites store their images on image servers.

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

Example

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

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

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

Image Floating

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

Example

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

<h2>Floating Images</h2>
<p><strong>Float the image to the right:</strong></p>

<p>
<img src="smiley.gif" alt="Smiley face"
style="float:right;width:42px;height:42px;">
A paragraph with a floating image. A paragraph with a floating image. A paragraph with a floating image.
</p>

<p><strong>Float the image to the left:</strong></p>
<p>
<img src="smiley.gif" alt="Smiley face"
style="float:left;width:42px;height:42px;">
A paragraph with a floating image. A paragraph with a floating image. A paragraph with a floating image.
</p>

</body>
</html>
```

Image Maps

The <map> tag defines an image-map. An image-map is an image with clickable areas.

In the image below, click on the computer, the phone, or the cup of coffee:

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

<h2>Image Maps</h2>
<p>Click on the computer, the phone, or the cup of coffee to go to a new page and read more about the topic:</p>

<img src="workplace.jpg" alt="Workplace" usemap="#workmap" width="400" height="379">

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

</body>
</html>
```

Note: The Problem is to find out the coordinates.

The name attribute of the <map> tag is associated with the <img>'s usemap attribute and creates a relationship between the image and the map.

The <map> element contains a number of <area> tags, that define the clickable areas in the image-map.

Background Image

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

Example
To add a background image on a web page, specify the background-image property on the BODY element:

```
<body style="background-image:url('clouds.jpg')">

<h2>Background Image</h2>

</body>
```

By default the background image will repeat itself if it is smaller than the element where it is specified, in this case the BODY element.

Example
To add a background image on a paragraph, specify the background-image property on the P element:

```
<body>

<p style="background-image:url('clouds.jpg')">
...
</p>
</body>
```

The <picture> Element

HTML5 introduced the <picture> element to add more flexibility when specifying image resources.

The <picture> element contains a number of <source> elements, each referring to different image sources. This way the browser can choose the image that best fits the current view and/or device.

Each <source> element have attributes describing when their image is the most suitable.

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

Example
Show one picture if the browser window (viewport) is a minimum of 650 pixels, and another image if not, but larger than 465 pixels.

```
<picture>
  <source media="(min-width: 650px)" srcset="img_pink_flowers.jpg">
  <source media="(min-width: 465px)" srcset="img_white_flower.jpg">
  <img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</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 matched.

HTML Screen Readers

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

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 to define the size of the image
Use the CSS width and height properties to define the size of the image (alternatively)
Use the CSS float property to let the image float
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 <img>'s element usemap attribute to point to an image-map
Use the HTML <picture> element to show different images for different devices
Note: Loading images takes time. Large images can slow down your page.

Use images carefully.

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

HTML Tables

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

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  font-family: arial, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

td, th {
  border: 1px solid #dddddd;
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {
  background-color: #dddddd;
}
</style>
</head>
<body>

<h2>HTML Table</h2>
```

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

</body>
</html>
```

Defining an HTML Table

An HTML table is defined with the <table> tag.

Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

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 - Adding a Border

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

A border is set using the CSS border property:

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

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

HTML Table - Collapsed Borders

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

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

HTML Table - Adding 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 {
  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 {
  text-align: left;
}
```

HTML Table - Adding Border Spacing

Border spacing specifies the space between the cells.

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

Example

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

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

HTML Table - Cells that Span Many Columns

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

Example

Name	**Telephone**	
Bill Gates	55577854	55577855

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

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

As :

```
<th rowspan="2">Telephone:</th>
```

Name:	Bill Gates
Telephone:	55577854
	55577855

HTML Table - Adding a Caption

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

Example

Monthly saving

Month	**Savings**
January	$100
February	$50

```
<table style="width:100%">
  <caption>Monthly savings</caption>
```

```
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

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

A Special Style for One Table

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

Example

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

Now you can define a special style for this table:

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

And add more styles:

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

```
  background-color: black;
}
```

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

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

HTML Lists

Unordered HTML List - Choose List Item Marker

The CSS list-style-type property is used to define the style of the list item marker:

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
<!DOCTYPE html>
<html>
<body>

<h2>Unordered List with Circle Bullets</h2>

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

</body>
</html>

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:

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:

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

Example
<!DOCTYPE html>
<html>
<body>

<h2>Ordered List with Lowercase Letters</h2>

<ol type="a">
  Coffee

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

</body>
</html>
```

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

A Description List

Coffee
- black hot drink
Milk
- white cold drink

```
<!DOCTYPE html>
<html>
<body>
<h2>A Description List</h2>
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
</body>
</html>
```

Nested HTML Lists

List can be nested (lists inside lists):

Example

A Nested List

List can be nested (lists inside lists):

- Coffee
- Tea
    - Black tea
    - Green tea
- Milk

```
<!DOCTYPE html>
<html>
<body>
<h2>A Nested List</h2>
<p>List can be nested (lists inside lists):</p>
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
</body>
</html>
```

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

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

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

Navigation Menu

In this example, we use CSS to style the list horizontally, to create a navigation menu:

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

```
li {
    float: left;
}

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

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

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

</body>
</html>
```

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 `<ol>` element to define an ordered list
- Use the HTML `type` attribute to define the numbering type
- Use the HTML `<li>` element to define a list item
- Use the HTML `<dl>` element to define a description list
- Use the HTML `<dt>` element to define the description term
- Use the HTML `<dd>` element to describe the term in a description list
- Lists can be nested inside lists
- List items can contain other HTML elements
- Use the CSS property `float:left` or `display:inline` to display a list horizontally
-

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

HTML Block and Inline Elements

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

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

The element is a block-level element.
Example
<!DOCTYPE html>
<html>
<body>

Hello
World

The DIV element is a block element, and will start on a new line.

</body>
</html>

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>
<output> <pre><section><table><tfoot>
<video>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Example
Hello
World

Inline elements in HTML:

<a><abbr><acronym><b><bdo><big>
<button><cite><code><dfn><i><img><input><kbd><label><map><object><q><samp><script><select><small><sub><sup><textarea><time><tt><var>

The Element

The element is often used as a container for other HTML elements.

The element has no required attributes, but style, class and id are common.

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

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.

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

<div style="background-color:red;color:black;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
  <p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>
</div>

</body>
</html>
```

The Element

The element is often used as a container for some text.
The element has no required attributes, but style, class and id are common.
When used together with CSS, the element can be used to style parts of the text:

Example
My *Important* Heading

```
<!DOCTYPE html>
<html>
<body>
<h1>My <span style="color:red">Important</span> Heading</h1>
</body>
</html>
```

HTML Grouping Tags

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

HTML The class Attribute

Using The class Attribute

The class attribute specifies one or more class names for an HTML element.

The class name can be used by CSS and JavaScript to perform certain tasks for elements with the specified class name.

In CSS, to select elements with a specific class, write a period (.) character, followed by the name of the class:

Example
Use CSS to style all elements with the class name "city":

```
<style>
.city {
   background-color: tomato;
   color: white;
   padding: 10px;
}
</style>
<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>
```

Result:

London

London is the capital of England.

Paris

Paris is the capital of France.

Tokyo

Tokyo is the capital of Japan.

Tip: The class attribute can be used on any HTML element.
Note: The class name is case sensitive!

Using The class Attribute in JavaScript

JavaScript can access elements with a specified class name by using the getElementsByClassName() method:

Example
When a user clicks on a button, hide all elements with the class name "city":

```
<!DOCTYPE html>
<html>
<body>
<h2>Using The class Attribute in JavaScript</h2>
<p>Click the button, to hide all elements with the class name "city", with JavaScript:</p>
<button onclick="myFunction()">Hide elements</button>
```

```
<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>

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

</body>
</html>
```

Multiple Classes

HTML elements can have more than one class name, each class name must be separated by a space.

Example

Style elements with the class name "city", also style elements with the class name "main":

```
<!DOCTYPE html>
<html>
<style>
.city {
    background-color: tomato;
    color: white;
    padding: 10px;
}
.main {
    text-align: center;
}
</style>
<body>
<h2>Multiple Classes</h2>
<p>All three headers have the class name "city". In addition, London also have the class name "main", which center-aligns the text.</p>
<h2 class="city main">London</h2>
<h2 class="city">Paris</h2>
<h2 class="city">Tokyo</h2>
</body>
</html>
```

In the example above, the first `<h2>` element belongs to both the "city" class and the "main" class.

Same Class, Different Tag

Different tags, like <h2> and , can have the same class name and thereby share the same style:

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

HTML The id Attribute

Using The id Attribute

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

The id value can be used by CSS and JavaScript to perform certain tasks for a unique element with the specified id value.

In CSS, to select an element with a specific id, write a hash (#) character, followed by the id of the element:

Example
Use CSS to style an element with the id "myHeader":

```
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>

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

Tip: *The id attribute can be used on any HTML element.*
Note: *The id value is case-sensitive.*
The id value must contain at least one character, and must not contain whitespace (spaces, tabs, etc.).

Difference Between Class and ID

Important :Id is selected by #id and class is selected by .id

An HTML element can only have one unique id that belongs to that single element, while a class name can be used by multiple elements:

Example

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

/* Style all elements with the class name "city" */
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>

<!-- A unique element -->
<h1 id="myHeader">My Cities</h1>

<!-- Multiple similar elements -->
<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>
```

Note : we should id for unique identification but if we wish can use id as well more than once.

Using The id Attribute in JavaScript

JavaScript can access an element with a specified id by using the getElementById() method:

Example

Use the id attribute to manipulate text with JavaScript:

```
<script>
function displayResult() {
```

```
  document.getElementById("myHeader").innerHTML = "Have a nice day!";
}
</script>
```

Bookmarks with ID and Links

HTML bookmarks are used to allow readers to jump to specific parts of a Web page.

Bookmarks can be useful if your webpage is very long.
To make a bookmark, you must first create the bookmark, and then add a link to it.
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:

```
<a href="#C4">Jump to Chapter 4</a>
```

Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:
Example

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

HTML Iframes

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

Iframe Syntax

An HTML iframe is defined with the <iframe> tag:

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

The src attribute specifies the URL (web address) of the inline frame page.

Iframe - Set Height and Width

Use the height and width attributes to specify the size of the iframe.

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

Example

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

<h2>HTML Iframes</h2>
<p>You can use the height and width attributes to specify the size of the iframe:</p>

<iframe src="demo_iframe.htm" height="200" width="300"></iframe>

</body>
</html>
```

Or you can use CSS to set the height and width of the iframe:

Example

```
<iframe src="demo_iframe.htm"
style="height:200px;width:300px;"></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;"></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;"></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

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

```
<body>

<h2>Iframe - Target for a Link</h2>

<iframe height="300px" width="100%" src="demo_iframe.htm"
name="iframe_a"></iframe>

<p><a href="https://www.w3schools.com"
target="iframe_a">W3Schools.com</a></p>
<p>When the target of a link matches the name of an iframe, the link will
open in the iframe.</p>
</body>
</html>
```

HTML JavaScript

JavaScript makes HTML pages more dynamic and interactive.
Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>
</body>
</html>
```

Result:
My First JavaScript
Click me to display Date and Time.
Sat Sep 01 2018 17:22:44 GMT+0530 (India Standard Time)

The HTML <script> Tag

The <script> tag is used to define a client-side script (JavaScript).
The <script> element either contains scripting statements, or it points to

an external script file through the src attribute.
Common uses for JavaScript are image manipulation, form validation, and

dynamic changes of content.

To select an HTML element, JavaScript very often uses the document.getElementById() method.

This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

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

<h2>Use JavaScript to Change Text</h2>
<p>This example writes "Hello JavaScript!" into an HTML element with id="demo":</p>
<p id="demo"></p>

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

</body>
</html>
```

A Taste of JavaScript

Here are some examples of what JavaScript can do:

JavaScript can change HTML content

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

Example :

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

<h1>My First JavaScript</h1>
<p>JavaScript can change the content of an HTML element:</p>
<button type="button" onclick="myFunction()">Click Me!</button>
<p id="demo">This is a demonstration.</p>
<script>
function myFunction() {
   document.getElementById("demo").innerHTML = "Hello JavaScript!";
}
</script>
```

```
</body>
</html>
```

JavaScript can change HTML styles

```
<p id="demo">JavaScript can change the style of an HTML element.</p>

<script>
function myFunction() {
   document.getElementById("demo").style.fontSize = "25px";
   document.getElementById("demo").style.color = "red";
   document.getElementById("demo").style.backgroundColor = "yellow";
}
</script>

<button type="button" onclick="myFunction()">Click Me!</button>
```

JavaScript can change HTML attributes

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

(see example of bulb u hv note in very first)

The HTML <noscript> Tag

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

Example:

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscript>
<p>A browser without support for JavaScript will show the text written inside the noscript element.</p>
```

```
</body>
</html>
```

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

HTML File Paths

Path	Description
<img src="picture.jpg">	picture.jpg is located in the same folder as the current page
<img src="images/picture.jpg">	picture.jpg is located in the images folder in the current folder
<img src="/images/picture.jpg">	picture.jpg is located in the images folder at the root of the current web
<img src="../picture.jpg">	picture.jpg 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 an internet file:

Example
<img src="https://www.w3schools.com/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.

Relative File Paths

1. the file path points to a file in the images folder located at the root of the current web:
2. the file path points to a file in the images folder located in the current folder:
3. the file path points to a file in the images folder located in the folder one level above the current folder:

HTML head Elements

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, links, scripts, and other meta information.

The following tags describe metadata: <title>, <style>, <meta>, <link>, <script>, and <base>.

The HTML <title> Element

The <title> element defines the title of the document, and is required in all HTML/XHTML documents.

The <title> element:

1.defines a title in the browser tab
2.provides a title for the page when it is added to favorites
3.displays a title for the page in search engine results
A simple HTML document:

Example

```
<!DOCTYPE html>
<html>

<head>
  <title>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>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

The HTML <link> Element

The <link> element is used to link to external style sheets:

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

The HTML <meta> Element

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

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

Define the character set used:
```
<meta charset="UTF-8">
```

Define a description of your web page:
```
<meta name="description" content="Free Web tutorials">
```

Define keywords for search engines:
```
<meta name="keywords" content="HTML, CSS, XML, JavaScript">
```

Define the author of a page:
```
<meta name="author" content="John Doe">
```

Refresh document every 30 seconds:
<meta http-equiv="refresh" content="30">

Example of <meta> tags:
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML,CSS,XML,JavaScript">
<meta name="author" content="John Doe">

Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

The viewport is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen.

You should include the following <meta> viewport element in all your web pages:

<meta name="viewport" content="width=device-width,
initial-scale=1.0">
A <meta> viewport element 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: You can see the difference if you open this page with a phone or a tablet. Check the two links:
Without the viewport meta tag-*https://www.w3schools.com/html/example_withoutviewport.htm*
With the viewport meta tag-*https://www.w3schools.com/html/example_withviewport.htm*

The HTML <script> Element

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

This JavaScript writes "Hello JavaScript!" into an HTML element with id="demo":

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

The HTML <base> Element

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

Example
<base href="https://www.w3schools.com/images/" target="_blank">

Omitting <html>, <head> and <body>?

According to the HTML5 standard; the <html>, the <body>, and the <head> tag can be omitted.

The following code will validate as HTML5:
<!DOCTYPE html>
<title>loveabhishek</title>

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

Note:
W3Schools does not recommend omitting the `<html>` and `<body>` tags. Omitting these tags can crash DOM or XML software and produce errors in older browsers (IE9).
However, omitting the `<head>` tag has been a common practice for quite some time now.

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

HTML Layouts

HTML Layout Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Template</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

body {
  font-family: Arial, Helvetica, sans-serif;
}

/* Style the header */
header {
  background-color: #666;
  padding: 30px;
  text-align: center;
  font-size: 35px;
  color: white;
}

/* Create two columns/boxes that floats next to each other */
nav {
  float: left;
  width: 30%;
  height: 300px; /* only for demonstration, should be removed */
  background: #ccc;
  padding: 20px;
}

/* Style the list inside the menu */
nav ul {
  list-style-type: none;
  padding: 0;
```

```
}

article {
 float: left;

   padding: 20px;
   width: 70%;
   background-color: #f1f1f1;
   height: 300px; /* only for demonstration, should be removed */
}


/* Clear floats after the columns */
section:after {
   content: "";
   display: table;
   clear: both;
}

/* Style the footer */
footer {
   background-color: #777;
   padding: 10px;
   text-align: center;
   color: white;
}

/* Responsive layout - makes the two columns/boxes stack on top of each other
instead of next to each other, on small screens */
@media (max-width: 600px) {
   nav, article {
      width: 100%;
      height: auto;
   }
}
</style>
</head>
<body>

<h2>CSS Layout Float</h2>

<p>In this example, we have created a header, two columns/boxes  and a

footer. On smaller screens, the columns will stack on top of each other.</p>
<p>Resize the browser window to see the responsive effect (you will learn more
about this in our next chapter - HTML Responsive.)</p>
```

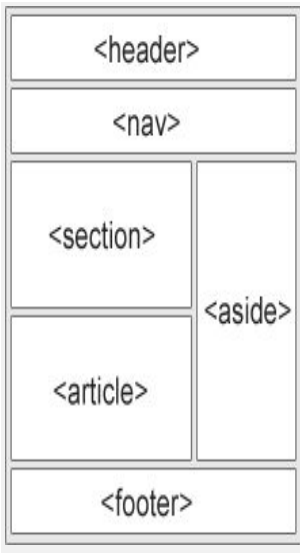
```
<header>
  <h2>Cities</h2>
</header>

<section>
<nav>
   <ul>
     <li><a href="#">London</a></li>
     <li><a href="#">Paris</a></li>
     <li><a href="#">Tokyo</a></li>

    </ul>
  </nav>

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

    </article>
</section>

<footer>
  <p>Footer</p>
</footer>

</body>
</html>
```

HTML Layout Elements

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

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

See example in the web….



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

HTML Layout Techniques

There are four different ways to create multicolumn layouts. Each way has its pros and cons:

- HTML tables (not recommended)
- CSS float property
- CSS flexbox
- CSS framework
-

Which One to Choose?

HTML Tables

The <table> element was not designed to be a layout tool! The purpose of the <table> element is to display tabular data. So, do not use tables for your page layout! They will bring a mess into your code. And imagine how hard it will be to redesign your site after a couple of months.

Tip: Do NOT use tables for your page layout!

CSS Frameworks

If you want to create your layout fast, you can use a framework, like W3.CSS or Bootstrap.

CSS Floats

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.

As in the very first example.

CSS Flexbox

Flexbox is a new layout mode in CSS3.

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices. *Disadvantages:* Does not work in IE10 and earlier.
Example :open this link.
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_flexbox
Almost similar to css floats

HTML 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):
Example

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
* {
  box-sizing: border-box;
}
.menu {
  float: left;
  width: 20%;
}
.menuitem {
  padding: 8px;
  margin-top: 7px;
  border-bottom: 1px solid #f1f1f1;
}
.main {
  float: left;
  width: 60%;
  padding: 0 20px;
  overflow: hidden;
}
.right {
  background-color: lightblue;
```

```
  float: left;
  width: 20%;
  padding: 10px 15px;
  margin-top: 7px;
}

@media only screen and (max-width:800px) {
  /* For tablets: */
  .main {
    width: 80%;
    padding: 0;
  }
  .right {
    width: 100%;
  }
}
@media only screen and (max-width:500px) {
  /* For mobile phones: */
  .menu, .main, .right {
    width: 100%;
  }
}
</style>
</head>
<body style="font-family:Verdana;">

<div style="background-color:#f1f1f1;padding:15px;">
  <h1>Cinque Terre</h1>
  <h3>Resize the browser window</h3>
</div>

<div style="overflow:auto">
  <div class="menu">
    <div class="menuitem">The Walk</div>
    <div class="menuitem">Transport</div>
    <div class="menuitem">History</div>
    <div class="menuitem">Gallery</div>
  </div>

  <div class="main">
    <h2>The Walk</h2>
    <p>The walk from Monterosso to Riomaggiore will take you approximately two hours, give or take an hour depending on the weather conditions and your physical shape.</p>
    <img src="img_5terre.jpg" style="width:100%">
  </div>
```

```
  <div class="right">
    <h2>What?</h2>
    <p>Cinque Terre comprises five villages: Monterosso, Vernazza, Corniglia, Manarola, and Riomaggiore.</p>
    <h2>Where?</h2>
    <p>On the northwest cost of the Italian Riviera, north of the city La Spezia.</p>
    <h2>Price?</h2>
    <p>The Walk is free!</p>
  </div>
</div>


<div
style="background-color:#f1f1f1;text-align:center;padding:10px;margin-top:7px;font-size:12px;"> This web page is a part of a demonstration of fluid web design made by w3schools.com. Resize the browser window to see the content respond to the resizing.</div>

</body>
</html>
```

To see result open the link:https://www.w3schools.com/html/tryit.asp?filename=tryhtml_responsive_page

Note: A web page should look good on any device!

Setting The Viewport(covered earlier)

When making responsive web pages, add the following <meta> element in all your web pages:

Example

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

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>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  <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:

Hello World

Resize the browser window to see how the text size scales.

Example

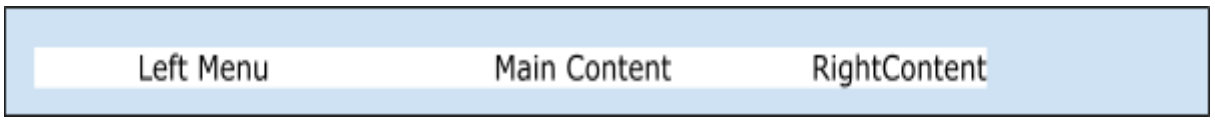
```
<h1 style="font-size:10vw">Hello World</h1>
```

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:Example




```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<style>
* {
  box-sizing:border-box;
}

.left {
  background-color:#2196F3;
  padding:20px;
  float:left;
  width:20%; /* The width is 20%, by default */
}

.main {
  background-color:#f1f1f1;
  padding:20px;
  float:left;
  width:60%; /* The width is 60%, by default */
}

.right {
  background-color:#4CAF50;
  padding:20px;
  float:left;
  width:20%; /* The width is 20%, by default */
}

/* Use a media query to add a break point at 800px: */
@media screen and (max-width:800px) {
```

```
  .left, .main, .right {
    width:100%; /* The width is 100%, when the viewport is 800px or
smaller */
  }
}
</style>
</head>
<body>

<h2>Media Queries</h2>
<p>Resize the browser window.</p>

<p>Make sure you reach the breakpoint at 800px when resizing this
frame.</p>

<div class="left">
  <p>Left Menu</p>
</div>

<div class="main">
  <p>Main Content</p>
</div>

<div class="right">
  <p>Right Content</p>
</div>

</body>
</html>
```

Tip: To learn more about Media Queries and Responsive Web Design, read RWD Tutorial.
https://www.w3schools.com/css/css_rwd_intro.asp

Responsive Web Page - Full Example

A responsive web page should look good on large desktop screens and small mobile phones.

Example

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<style>
* {
  box-sizing: border-box;
}
.menu {
  float:left;
  width:20%;
  text-align:center;
}
.menu a {
  background-color:#e5e5e5;
  padding:8px;
  margin-top:7px;
  display:block;
  width:100%;
  color:black;
}
.main {
  float:left;
  width:60%;
  padding:0 20px;
}
.right {
  background-color:#e5e5e5;
  float:left;
  width:20%;
  padding:15px;
  margin-top:7px;
  text-align:center;
}

@media only screen and (max-width:620px) {
  /* For mobile phones: */
  .menu, .main, .right {
    width:100%;
  }
}
</style>
</head>
<body style="font-family:Verdana;color:#aaaaaa;">

<div style="background-color:#e5e5e5;padding:15px;text-align:center;">
  <h1>Hello World</h1>
</div>

<div style="overflow:auto">
```

```
  <div class="menu">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
    <a href="#">Link 4</a>
  </div>

  <div class="main">
    <h2>Lorum Ipsum</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  </div>

  <div class="right">
    <h2>About</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
  </div>
</div>

<div style="background-color:#e5e5e5;text-align:center;padding:10px;margin-top:7px;">©
copyright w3schools.com</div>

</body>
</html>
```

To result open the link:
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_responsive_media_query3

Responsive Web Design - Frameworks

There are many existing CSS Frameworks that offer Responsive Design.

They are free, and easy to use.

Using W3.CSS

A great way to create a responsive design, is to use a responsive style sheet, like W3.CSS
W3.CSS makes it easy to develop sites that look nice at any size; desktop, laptop, tablet, or phone:
Example

```
<!DOCTYPE html>
<html>
<title>W3.CSS</title>
<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">
  <h1>W3Schools Demo</h1>
  <p>Resize this responsive page!</p>
</div>

<div class="w3-row-padding">
  <div class="w3-third">
    <h2>London</h2>
    <p>capital city of England.</p>
    <p>It is....</p>
  </div>

  <div class="w3-third">
    <h2>Paris</h2>
    <p>capital of France.</p>
    <p>The Paris....</p>
  </div>

  <div class="w3-third">
    <h2>Tokyo</h2>
    <p>capital of Japan.</p>
    <p>It is....</p>
  </div>
</div>

</body>
</html>
```

Bootstrap

Another popular framework is Bootstrap, it 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/3.3.7/css/bootstrap.min.
```

```
css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></sc
ript>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
></script>
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>My First Bootstrap Page</h1>
  </div>
  <div class="row">
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
    ...
    </div>
  </div>
</div>

</body>
</html>
```

Result :https://www.w3schools.com/html/tryit.asp?filename=tryhtml_responsive_bootstrap

HTML Entities

Reserved characters in HTML must be replaced with character entities.

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

HTML Entities

Some characters are reserved in HTML.

If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.
Character entities are used to display reserved characters in HTML.

A character entity looks like this:
&entity_name;
OR
&#entity_number;

To display a less than sign (<) we must write: < or <

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 numbers is good.

A common character entity used 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:

§ 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.

The non-breaking hyphen (‑) lets you use a hyphen character (-) that won't break.

Some Other Useful HTML Character Entities there...
Note: Entity names are case sensitive.

Combining Diacritical Marks
A diacritical mark is a "glyph" added to a letter.

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 an HTML entity name.

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

Example
I will display €
I will display €
I will display €
Will display as:
I will display €
I will display €
I will display €

Some other are also there...

HTML Encoding (Character Sets)

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

What is Character Encoding?

ASCII was the first character encoding standard (also called character set). ASCII defined 128 different alphanumeric characters that could be used on the internet: numbers (0-9), English letters (A-Z), and some special characters like ! $ + - ( ) @ < > .

ANSI (Windows-1252) was the original Windows character set, with support for 256 different character codes.

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

Because ANSI and ISO-8859-1 were so limited, HTML 4 also supported UTF-8.

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

The default character encoding for HTML5 is UTF-8.

The HTML charset Attribute

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

This is specified in the <meta> tag:

For HTML4:
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
For HTML5:
<meta charset="UTF-8">

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

Differences Between Character Sets:find the table???
The following table displays the differences between the character sets described above:

The ASCII Character Set

ASCII uses the values from 0 to 31 (and 127) for control characters.
ASCII uses the values from 32 to 126 for letters, digits, and symbols.
ASCII does not use the values from 128 to 255.

The ANSI Character Set (Windows-1252)

ANSI is identical to ASCII for the values from 0 to 127.
ANSI has a proprietary set of characters for the values from 128 to 159.
ANSI is identical to UTF-8 for the values from 160 to 255.

The ISO-8859-1 Character Set

8859-1 is identical to ASCII for the values from 0 to 127.
8859-1 does not use the values from 128 to 159.
8859-1 is identical to UTF-8 for the values from 160 to 255.

The UTF-8 Character Set

UTF-8 is identical to ASCII for the values from 0 to 127.
UTF-8 does not use the values from 128 to 159.
UTF-8 is identical to both ANSI and 8859-1 for the values from 160 to 255.
UTF-8 continues from the value 256 with more than 10 000 different characters.
For a closer look, study our Complete HTML Character Set Reference.

The @charset CSS Rule

You can use the CSS @charset rule to specify the character encoding used in a style sheet:

Example

Set the encoding of the style sheet to Unicode UTF-8:
@charset "UTF-8";

HTML Uniform Resource Locators

A URL is another word for a web address.

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

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

URL - Uniform Resource Locator

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

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		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.

The server has processed your input and returned this answer.
Processing input is explained in our PHP tutorial.

ASCII Encoding Examples

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

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

Character	From Windows-1252	From UTF-8
€	%80	%E2%82%AC
£	%A3	%C2%A3
©	%A9	%C2%A9
®	%AE	%C2%AE

These are few to count.

HTML and XHTML

XHTML is HTML written as XML.

What Is XHTML?

XHTML stands for EXtensible HyperText Markup Language
XHTML is almost identical to HTML
XHTML is stricter than HTML
XHTML is HTML defined as an XML application
XHTML is supported by all major browsers

Why XHTML?

Many pages on the internet contain "bad" HTML.

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

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

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

Today's market consists of different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices. Smaller devices often lack the resources or power to interpret "bad" markup.
XML is a markup language where documents must be marked up correctly (be "well-formed"). read XML tutorial.
XHTML was developed by combining the strengths of HTML and XML.
XHTML is HTML redesigned as XML.

The Most Important Differences from HTML:

Document Structure
XHTML DOCTYPE is mandatory
The xmlns attribute in <html> is mandatory
<html>, <head>, <title>, and <body> are mandatory
XHTML Elements
XHTML elements must be properly nested
XHTML elements must always be closed
XHTML elements must be in lowercase
XHTML documents must have one root element
XHTML Attributes
Attribute names must be in lower case
Attribute values must be quoted
Attribute minimization is forbidden

<!DOCTYPE ....> Is Mandatory

An XHTML document must have an XHTML DOCTYPE declaration.

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

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

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

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

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

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

<body>
  some content
</body>
</html>
```

XHTML Elements Must Be Properly Nested

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

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

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

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

XHTML Elements Must Always Be Closed

This is wrong:

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

This is correct:

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

This is another paragraph

Empty Elements Must Also Be Closed

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

This is correct:
A break:
A horizontal rule: <hr />
An image:

XHTML Elements Must Be In Lower Case
This is wrong:
<BODY>
<P>This is a paragraph</P>
</BODY>

This is correct:
<body>
This is a paragraph
</body>

XHTML Attribute Names Must Be In Lower Case
This is wrong:
<table WIDTH="100%">

This is correct:
<table width="100%">

Attribute Values Must Be Quoted
This is wrong:
<table width=100%>

This is correct:
<table width="100%">

Attribute Minimization Is Forbidden
Wrong:
<input type="checkbox" name="vehicle" value="car" checked />

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

Wrong:
<input type="text" name="lastname" disabled />

Correct:
<input type="text" name="lastname" disabled="disabled" />

How to Convert from HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page
2. Add an xmlns attribute to the html element of every page
3. Change all element names to lowercase
4. Close all empty elements
5. Change all attribute names to lowercase
6. Quote all attribute values
7.

Validate HTML With The W3C Validator

HTML Forms

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

<h2>HTML Forms</h2>

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

<p>If you click the "Submit" button, the form-data will be sent to a page called
"/action_page.php".</p>
</body></html>
```

The <form> Element

The HTML <form> element defines a form that is used to collect user input:

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

An HTML form contains form elements.

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

The <input> Element

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

The <input> element can be displayed in several ways, depending on the type attribute.

Here are some examples:

Type	*Description*
<input type="text">	*Defines a one-line text input field*
<input type="radio">	*Defines a radio button (for selecting one of many choices)*
<input type="submit">	*Defines a submit button (for submitting the form)*

Example

```
 <!DOCTYPE html>
<html>
<body>
<h2>Text Input</h2>

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

<p>Note that the form itself is not visible.</p>

<p>Also note that the default width of a text input field is 20 characters.</p>
</body>
</html>
```

Radio Button Input

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

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

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

<h2>Radio Buttons</h2>

<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>

</body>
</html>
```
Result

Radio Buttons

- Male
- Female
- Other
-

The Submit Button

<input type="submit"> defines a button for submitting the 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
```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, the form-data will be sent to a page
```

called"/action_page.php".</p>

</body>
</html>

The Action Attribute

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

Normally, the form data is sent to a web page on the server when the user clicks on the submit button.

In the example above, the form data is sent to a page on the server called "/action_page.php". This page contains a server-side script that handles the form data:

<form action="/action_page.php">
If the action attribute is omitted, the action is set to the current page.

The Target Attribute

The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is "_self" which means the form will be submitted in the current window.

To make the form result open in a new browser tab, use the value "_blank":
Example
<form action="/action_page.php" target="_blank">

Other legal values are "_parent", "_top", or a name representing the name of an iframe.

The Method Attribute

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

Example
<form action="/action_page.php" method="get">
or:
<form action="/action_page.php" method="post">

When to Use GET?

The default method when submitting form data is GET.
However, when GET is used, the submitted form data will be visible in the page address field:

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

Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google
-

When to Use POST?

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

Notes on POST:
POST has no size limitations, and can be used to send large amounts of data.
Form submissions with POST cannot be bookmarked.

The Name Attribute

Each input field must have a name attribute to be submitted.

If the name attribute is omitted, the data of that input field will not be sent at all.
This example will only submit the "Last name" input field:

Example

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

Grouping Form Data with <fieldset>

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

Here is the list of `<form>` attributes:

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

HTML Form Elements

```
The <input> Element
The most important form element is the <input> element.

The <input> element can be displayed in several ways,
depending on the type attribute.

Example
<input name="firstname" type="text">
```

If the type attribute is omitted, the input field gets the default type: "text".

The <select> Element

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

Example

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

<h2>The select Element</h2>
<p>The select element defines a drop-down list:</p>
```

```
<form action="/action_page.php">
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>

</body>
</html>
```

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

```
<select name="cars" size="3">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Allow Multiple Selections:

Use the multiple attribute to allow the user to select more than one value:
Example

```
<select name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

The <textarea> Element

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

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

<h2>Textarea</h2>
<p>The textarea element defines a multi-line input field.</p>

<form action="/action_page.php">
  <textarea name="message" rows="10" cols="30">The cat was playing in the
garden.</textarea>
  <br>
  <input type="submit">
</form>

</body>
</html>
```

The rows attribute specifies the visible number of lines in a text area.
The cols attribute specifies the visible width of a text area.

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>
```

Note: *Always specify the type attribute for the button element. Different browsers may use different default types for the button element.*

HTML5 Form Elements

HTML5 added the following form elements:

<datalist>
<output>

Note: *Browsers do not display unknown elements. New elements that are not supported in older browsers will not "destroy" your web page.*

HTML5 <datalist> Element

The <datalist> element specifies a list of pre-defined options for an <input> element.
Users will see a drop-down list of 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">
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

HTML5 <output> Element

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

Example

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

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

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

HTML Input Types

This chapter describes the different input types for the <input> element.

Input Type Text

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

Example

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

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

Input Type Password

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

Example

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

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

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

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">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```
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"> defines a radio button. ___do___

Input Type Checkbox

<input type="checkbox"> defines a checkbox.
Checkboxes let a user select ZERO or MORE options of a limited number of choices.
Example
```
<form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
</form>
```

Input Type Button

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

HTML5 Input Types

HTML5 added several new input types:
Color ,date, datetime-local ,email ,month ,number ,range ,search ,tel ,time ,url ,week.

New input types that are not supported by older web browsers, will behave as <input type="text">.

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.

Opera Safari Chrome Firefox Internet Explorer
Example
```
<form>
  Select your favorite color:
```

```
  <input type="color" name="favcolor">
</form>
```
Note: type="color" is not supported in Internet Explorer 11 and earlier versions or Safari 9.1 and earlier versions.

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.
Opera Safari Chrome Firefox Internet Explorer
Example

```
<form>
  Birthday:
  <input type="date" name="bday">
</form>
```

Note: type="date" is not supported in Safari or Internet Explorer 11 and earlier versions.
You can also use the min and max attributes to add restrictions to dates:
Opera Safari Chrome Firefox Internet Explorer
Example

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```

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.

Opera Safari Chrome Firefox Internet Explorer
Example

```
<form>
  Birthday (date and time):
  <input type="datetime-local" name="bdaytime">
</form>
```

Note: type="datetime-local" is not supported in Firefox, Safari or Internet Explorer 12 and earlier versions.

Input Type Email

The <input type="email"> is used for input fields that should contain an e-mail address.
Depending on browser support, the e-mail address can be automatically validated when submitted.
Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.
Opera Safari Chrome Firefox Internet Explorer
Example

```
<h2>Email Field</h2>
<p>The <strong>input type="email"</strong> is used for input fields that should contain an e-mail address:</p>
```

```
<form action="/action_page.php">
  E-mail:
  <input type="email" name="email">
  <input type="submit">
</form>
```
Note:type="email" is not supported in IE9 and earlier.

Input Type File

The <input type="file"> defines a file-select field and a "Browse" button for file uploads
Example

```
<p>Show a file-select field which allows a file to be chosen for upload:</p>
<form action="/action_page.php">
  Select a file: <input type="file" name="myFile"><br><br>
  <input type="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>
  Birthday (month and year):
  <input type="month" name="bdaymonth">
</form>
```
Note: type="month" is not supported in Firefox, Safari, or Internet Explorer 11 and earlier versions.

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

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

<h2>Number Field</h2>
<p>The <strong>input type="number"</strong> defines a numeric input field.</p>
<p>You can use the min and max attributes to add numeric restrictions in the input field:</p>

<form action="/action_page.php">
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">
  <input type="submit">
</form>

<p><b>Note:</b> type="number" is not supported in IE9 and earlier.</p>
```

```
</body>
</html>
```

Input Restrictions

Here is a list of some common input restrictions (some are new in HTML5):

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

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 action="/action_page.php">
  Quantity:
  <input type="number" name="quantity"  min="0" max="100" step="10" value="30">
  <input type="submit">
</form>

<p><b>Note:</b>type="number" is not supported in IE9 and earlier.
</p>
```

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>
  <input type="range" name="points" min="0" max="10">
</form>
```
Note: type="range" is not supported in Internet Explorer 9 and earlier versions.

Input Type Search

The <input type="search"> is used for search fields (a search field behaves like a regular text field).

Example

```
<form>
  Search Google:
  <input type="search" name="googlesearch">
</form>
```

Input Type Tel

The <input type="tel"> is used for input fields that should contain a telephone number.

Note: The tel type is currently only supported in Safari 8.

Example

```
<form>
  Telephone:
  <input type="tel" name="usrtel">
</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>
  Select a time:
  <input type="time" name="usr_time">
</form>
```

Note: type="time" is not supported in Safari or Internet Explorer 12 and earlier versions.

Input Type Url

The <input type="url"> is used for input fields that should contain a URL address.

Depending on browser support, the url field can be automatically validated when submitted.

Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

Example

```
<form>
  Add your homepage:
  <input type="url" name="homepage">
</form>
```

Note: The type="url" is not supported in IE9 and earlier versions.

Input Type Week

The <input type="week"> allows the user to select a week and year.

Depending on browser support, a date picker can show up in the input field.

Example

```
<form>
```

```
  Select a week:
  <input type="week" name="week_year">
</form>
```

Note: type="week" is not supported in Firefox, Safari or Internet Explorer 11 and earlier versions.

HTML Input Attributes

The value Attribute

The value attribute specifies the initial value for an input field:

Example

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John">
</form>
```

The readonly Attribute

The readonly attribute specifies that the input field is read only (cannot be changed):

Example

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John" readonly>
</form>
```

The disabled Attribute

The disabled attribute specifies that the input field is disabled.

A disabled input field is unusable and un-clickable, and its value will not be sent when submitting the form:

Example

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John" disabled>
</form>
```

The size Attribute

The size attribute specifies the size (in characters) for the input field:

Example

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" value="John" size="40">
</form>
```

The maxlength Attribute

The maxlength attribute specifies the maximum allowed length for the input field:

Example

```
<form action="">
  First name:<br>
  <input type="text" name="firstname" maxlength="10">
</form>
```

With a maxlength attribute, the input field will not accept more than the allowed number of characters.

The maxlength attribute does not provide any feedback. If you want to alert the user, you must write JavaScript code.

Note: Input restrictions are not foolproof, and JavaScript provides many ways to add illegal input. To safely restrict input, it must be checked by the receiver (the server) as well!

HTML5 Attributes

HTML5 added the following attributes for <input>:

Autocomplete ,autofocus ,form , formaction , formenctype , formmethod ,formnovalidate, formtarget ,height and width ,list ,min and max ,multiple ,pattern (regexp) ,placeholder, Required ,step

and the following attributes for <form>:

Autocomplete ,novalidate

The autocomplete Attribute

The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.

When autocomplete is on, the browser automatically completes the input values based on values that the user has entered before.

Tip: It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

Example

An HTML form with autocomplete on (and off for one input field):

```
<form action="/action_page.php" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email" autocomplete="off"><br>
  <input type="submit">
</form>
```

Tip: In some browsers you may need to activate the autocomplete function for this to work.

The novalidate Attribute

The novalidate attribute is a <form> attribute.

When present, novalidate specifies that the form data should not be validated when submitted.

Example

Indicates that the form is not to be validated on submit:

```
<form action="/action_page.php" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

The autofocus Attribute

The autofocus attribute specifies that the 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 action="/action_page.php">
  First name:<input type="text" name="fname" autofocus><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit">
</form>
```

Note: The autofocus attribute of the input tag is not supported in Internet Explorer 9 and earlier versions.

The form Attribute

The form attribute specifies one or more forms an <input> element belongs to.

Tip: To refer to more than one form, use a space-separated list of form ids.
Example
An input field located outside the HTML form (but still a part of the form):

```
<form action="/action_page.php" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>
Last name: <input type="text" name="lname" form="form1">
```

The formaction Attribute

The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.

The formaction attribute overrides the action attribute of the <form> element.

The formaction attribute is used with type="submit" and type="image".
Example
An HTML form with two submit buttons, with different actions:

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="/action_page2.php"
  value="Submit as admin">
</form>
```

The formenctype Attribute

The formenctype attribute specifies how the form data should be encoded when submitted (only for forms with method="post").

The formenctype attribute overrides the enctype attribute of the <form> element.

The formenctype attribute is used with type="submit" and type="image".
Example
Send form-data that is default encoded (the first submit button), and encoded as "multipart/form-data" (the second submit button):

```
<form action="/action_page_binary.asp" method="post">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

Note: The formenctype attribute of the input tag is not supported in Internet Explorer 9 and earlier versions.

The formmethod Attribute

The formmethod attribute defines the HTTP method for sending form-data to the action URL.

The formmethod attribute overrides the method attribute of the <form> element.

The formmethod attribute can be used with type="submit" and type="image".
Example
The second submit button overrides the HTTP method of the form:

```
<form action="/action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post" value="Submit using POST">
</form>
```

The formnovalidate Attribute

The formnovalidate attribute overrides the novalidate attribute of the <form> element.

The formnovalidate attribute can be used with type="submit".
Example
A form with two submit buttons (with and without validation):

```
<form action="/action_page.php">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
```

```
  <input type="submit" formnovalidate value="Submit without validation">
</form>
```

The formtarget Attribute

The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

The formtarget attribute overrides the target attribute of the <form> element.

The formtarget attribute can be used with type="submit" and type="image".
Example
A form with two submit buttons, with different target windows:

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit as normal">
  <input type="submit" formtarget="_blank"
  value="Submit to a new window">
</form>
```

The height and width Attributes

The height and width attributes specify the height and width of an <input type="image"> element.

Always specify the size of images. If the browser does not know the size, the page will flicker while images load.
Example
Define an image as the submit button, with height and width attributes:

```
<input type="image" src="img_submit.gif" alt="Submit" width="48"
height="48">
```

The list Attribute

The list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.
Example
An <input> element with pre-defined values in a <datalist>:

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

The min and max Attributes

The min and max attributes specify the minimum and maximum values for an <input> element.

The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

Example

<input> elements with min and max values:

Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">

The multiple Attribute

The multiple attribute specifies that the user is allowed to enter more than one value in the <input> element.

The multiple attribute works with the following input types: email, and file.

Example

A file upload field that accepts multiple values:

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

The pattern Attribute

The pattern attribute specifies a regular expression that the <input> element's value is checked against.

The pattern attribute works with the following input types: text, search, url, tel, email, and password.

Tip: Use the global title attribute to describe the pattern to help the user.

Tip: Learn more about regular expressions in our JavaScript tutorial.

Example

An input field that can contain only three letters (no numbers or special characters):

Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">

The placeholder Attribute

The placeholder attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).

The hint is displayed in the input field before the user enters a value.

The placeholder attribute works with the following input types: text, search, url, tel, email, and password.
Example
An input field with a placeholder text:

<input type="text" name="fname" placeholder="First name">

The required Attribute
The 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:

Username: <input type="text" name="usrname" required>

The step Attribute
The step attribute specifies the legal number intervals for an <input> element.

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

Tip: The step attribute can be used together with the max and min attributes to create a range of legal values.

The step attribute works with the following input types: number, range, date, datetime-local, month, time and week.
Example
An input field with a specified legal number intervals:

<input type="number" name="points" step="3">

html5
Define Semantic Elements as Block Elements
HTML5 defines eight new semantic elements. All these are block-level elements.

To secure correct behavior in older browsers, you can set the CSS display property for these HTML elements to block:

```
header, section, footer, aside, nav, main, article, figure {
   display: block;
}
```

Add New Elements to HTML
You can also add new elements to an HTML page with a browser trick.

This example adds a new element called <myHero> to an HTML page, and

defines a style for it:

Example
```
<!DOCTYPE html>
<html>
<head>
<script>document.createElement("myHero")</script>
<style>
myHero {
   display: block;
   background-color: #dddddd;
   padding: 50px;
   font-size: 30px;
}
</style>
</head>
<body>

<h1>A Heading</h1>
<myHero>My Hero Element</myHero>

</body>
</html>
```
The JavaScript statement document.createElement("myHero") is needed to create a new element in IE 9, and earlier.

Problem With Internet Explorer 8
You could use the solution described above for all new HTML5 elements.

However, IE8 (and earlier) does not allow styling of unknown elements!

Thankfully, Sjoerd Visscher created the HTML5Shiv! The HTML5Shiv is a JavaScript workaround to enable styling of HTML5 elements in versions of Internet Explorer prior to version 9.

You will require the HTML5shiv to provide compatibility for IE Browsers older than IE 9.

Syntax For HTML5Shiv
The HTML5Shiv is placed within the <head> tag.

The HTML5Shiv is a javascript file that is referenced in a <script> tag.

You should use the HTML5Shiv when you are using the new HTML5 elements such as: <article>, <section>, <aside>, <nav>, <footer>.

You can download the latest version of HTML5shiv from github or reference the CDN version at https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js

Syntax

```
<head>
  <!--[if lt IE 9]>
    <script src="/js/html5shiv.js"></script>
  <![endif]-->
</head>
```

HTML5Shiv Example

If you do not want to download and store the HTML5Shiv on your site, you could reference the version found on the CDN site.

The HTML5Shiv script must be placed in the <head> element, after any stylesheets:

Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!--[if lt IE 9]>
  <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<![endif]-->
</head>
<body>

<section>

<h1>Famous Cities</h1>

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

<article>
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</article>

<article>
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the
most populous metropolitan area in the world.</p>
</article>

</section>

</body>
</html>
```

New Input Types

New Input Types

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

New Input Attributes

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

HTML5 Semantic Elements

Semantics is the study of the meanings of words and phrases in a language.
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: and - Tells nothing about its content.
Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

HTML5 <footer> Element

You may have several `<footer>` elements in one document.

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

<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>

</body>
</html
```

HTML5 <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.

Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

HTML5 <figure> and <figcaption> Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a <figure> element:

Example

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Why Semantic Elements?

With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.

This made it impossible for search engines to identify the correct web page content.

With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.

According to the W3C, a Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML5

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section

<main> Specifies the main content of a document
<mark> Defines marked/highlighted text
<nav> Defines navigation links
<section> Defines a section in a document
<summary> Defines a visible heading for a <details> element
<time> Defines a date/time

Example
Using the <details> element:

```
<details>
  <summary>Copyright 1999-2018.</summary>
  <p> - by Refsnes Data. All Rights Reserved.</p>
  <p>All content and graphics on this web site are the property of the company
Refsnes Data.</p>
</details>
```

Migration from HTML4 to HTML5

This chapter demonstrates how to convert an HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

You can migrate from XHTML to HTML5, using the same recipe.

Typical HTML4	**Typical HTML5**
	<header>
	<nav>
	<section>
	<article>
	<footer>

```
<style>
body {
   font-family: Verdana,sans-serif;
   font-size: 0.9em;
}

div#header, div#footer {
   padding: 10px;
   color: white;
   background-color: black;
}

div#content {
   margin: 5px;
   padding: 10px;
   background-color: lightgrey;
}

div.article {
   margin: 5px;
```

```
    padding: 10px;
    background-color: white;
}

div#menu ul {
    padding: 0;
}

div#menu ul li {
    display: inline;
    margin: 5px;
}
}
</style>
```

Change to HTML5 Doctype

Change the doctype:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

to the HTML5 doctype:

Example

```
<!DOCTYPE html>
```

Change to HTML5 Encoding

Change the encoding information:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

to HTML5 encoding:

Example

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

Add The HTML5Shiv

The new HTML5 semantic elements are supported in all modern browsers. In addition, you can "teach" older browsers how to handle "unknown elements".

However, IE8 and earlier, does not allow styling of unknown elements. So, the HTML5Shiv is a JavaScript workaround to enable styling of HTML5 elements in versions of Internet Explorer prior to version 9.

Add the HTML5Shiv:

Example

```
<!--[if lt IE 9]>
  <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<![endif]-->
```

The Difference Between <article> <section> and

There is a confusing (lack of) difference in the HTML5 standard, between <article> <section> and .
In the HTML5 standard, the <section> element is defined as a block of related elements.
The <article> element is defined as a complete, self-contained block of related elements.
The element is defined as a block of children elements.
How to interpret that?
In the example above, we have used <section> as a container for related <articles>.
But, we could have used <article> as a container for articles as well.

Can be:
<article> in <article>
 in <article>
 in <section> in <article>

HTML5 Style Guide and Coding Conventions

Web developers are often uncertain about the coding style and syntax to use in HTML.
Between 2000 and 2010, many web developers converted from HTML to XHTML.
With XHTML, developers were forced to write valid and "well-formed" code.
HTML5 is a bit more sloppy when it comes to code validation.

Be Smart and Future Proof

A consistent use of style makes it easier for others to understand your HTML.
In the future, programs like XML readers may want to read your HTML.
Using a well-formed-"close to XHTML" syntax can be smart.
Always keep your code tidy, clean and well-formed.

Use Correct Document Type

Always declare the document type as the first line in your document:
<!DOCTYPE html>
If you want consistency with lower case tags, you can use:
<!doctype html>

Use Lower Case Element Names

HTML5 allows mixing uppercase and lowercase letters in element names.

We recommend using lowercase element names because:
Mixing uppercase and lowercase names is bad
Developers normally use lowercase names (as in XHTML)
Lowercase look cleaner
Lowercase are easier to write

Close All HTML Elements

In HTML5, you don't have to close all elements (for example the <p> element).
We recommend closing all HTML elements.

Close Empty HTML Elements
In HTML5, it is optional to close empty elements.

Allowed:
<meta charset="utf-8">
Also Allowed:
<meta charset="utf-8" />

However, the closing slash (/) is REQUIRED in XHTML and XML.
If you expect XML software to access your page, it is a good idea to keep the closing slash!

Use Lower Case Attribute Names
HTML5 allows mixing uppercase and lowercase letters in attribute names.
Bad:

Good:

Quote Attribute Values
HTML5 allows attribute values without quotes.

We recommend quoting attribute values because:
Mixing uppercase and lowercase values is bad
Quoted values are easier to read
You MUST use quotes if the value contains spaces

Image Attributes
Always add the alt attribute to images. This attribute is important when the image for some reason cannot be displayed. Also, always define image width and height. It reduces flickering because the browser can reserve space for the image before loading.
Bad:
<img src="html5.gif">
Good:
<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">

Spaces and Equal Signs
HTML5 allows spaces around equal signs. But space-less is easier to read and groups entities better together.
Bad:
<link rel = "stylesheet" href = "styles.css">
Good:
<link rel="stylesheet" href="styles.css">

Avoid Long Code Lines
When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code.

Try to avoid code lines longer than 80 characters.

Blank Lines and Indentation
Do not add blank lines without a reason.
For readability, add blank lines to separate large or logical code blocks.
For readability, add two spaces of indentation. Do not use the tab key.
Do not use unnecessary blank lines and indentation. It is not necessary to indent every element:

Omitting <html> and <body>?
In HTML5, the <html> tag and the <body> tag can be omitted.
However, we do not recommend omitting the <html> and the <body> tag.

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

<!DOCTYPE html>
<html lang="en-US">
Declaring a language is important for accessibility applications (screen readers) and search engines.

Omitting <html> or <body> can crash DOM and XML software.
Omitting <body> can produce errors in older browsers (IE9).

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

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

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

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

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

</html>

However, we do not recommend omitting the <head> tag.
Omitting tags is unfamiliar to web developers. It needs time to be established as a guideline.

Meta Data
The <title> element is required in HTML5. Make the title as meaningful as possible:

<title>HTML5 Syntax and Coding Style</title>

To ensure proper interpretation and correct search engine indexing, both the language and the character encoding should be defined as early as possible in a document:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Syntax and Coding Style</title>
</head>
```

Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

The viewport is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen.

You should include the following <meta> viewport element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

A <meta> viewport element 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.

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:

```
<!--
  This is a long comment example. This is a long comment example.
  This is a long comment example. This is a long comment example.
-->
```

Long comments are easier to observe if they are indented two spaces.

Style Sheets

Use simple syntax for linking to style sheets (the type attribute is not necessary):

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

Short rules can be written compressed, like this:

```
p.intro {font-family: Verdana; font-size: 16em;}
```

Long rules should be written over multiple lines:

```
body {
  background-color: lightgrey;
  font-family: "Arial Black", Helvetica, sans-serif;
  font-size: 16em;
  color: black;
}
```

Place the opening bracket on the same line as the selector
Use one space before the opening bracket
Use 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
Avoid lines over 80 characters

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" or "london.jpg".

If you use a mix of upper and lower case, you have to be extremely consistent. If you move from a case insensitive to a case sensitive server, even small errors will break your web!
To avoid these problems, always use lowercase file names.

File Extensions

HTML files should have a .html or .htm extension.
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 extensions. Both will be treated as HTML by any web browser or web server.

The differences are cultural:

.htm "smells" of early DOS systems where the system limited the extensions to 3 characters.
.html "smells" of Unix operating systems that did not have this limitation.

Technical Differences

When a URL does not specify a filename (like https://www.w3schools.com/css/), the server returns a default filename. Common default filenames are index.html, index.htm, default.html and default.htm.

If your server is configured only with "index.html" as default filename, your file must be named "index.html", not "index.htm."

However, servers can be configured with more than one default filename, and normally you can set up as many default filenames as needed.

Anyway, the full extension for HTML files is .html, and there's no reason it should not be used.

HTML5 Canvas

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.

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>
```

Draw a Line

Example

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

Draw a Circle

Example

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(90,50,40,0,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```

Draw a Text

Example

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
```

Stroke Text

Example

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
```

Draw Linear Gradient

Example

```
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);
```

Draw Circular Gradient

Example

```
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);
```

Draw Image

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

<p>Image to use:</p>
<img id="scream" src="img_the_scream.jpg" alt="The Scream" width="220"
height="277">

<p>Canvas to fill:</p>
<canvas id="myCanvas" width="250" height="300"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<p><button onclick="myCanvas()">Try it</button></p>

<script>
function myCanvas() {
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    var img = document.getElementById("scream");
    ctx.drawImage(img,10,10);
}
</script>

</body>
</html>
```

HTML5 SVG

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.

SVG Circle

Example

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

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow"
/>
</svg>

</body>
</html>
```

SVG Rectangle

Example

```
<svg width="400" height="100">
  <rect width="400" height="100"
style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
</svg>
```

SVG Rounded Rectangle

Example

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>
```

SVG Star

Example

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

SVG Logo

Example

```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50"
y="86">SVG</text>
  Sorry, your browser does not support inline SVG.
</svg>
```

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

HTML Google Maps

Google Maps allows you to display maps on your web page:
Use ctrl + scroll to zoom the map

A Basic Web Page

To demonstrate how to add a Google Map to a web page, we will use a basic HTML page:
Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Google Map</h1>
<div id="map">My map will go here</div>
</body>
```

<html>

Set the Map Size
Set the size of the map:
Example

Create a Function to Set The Map Properties
This example defines a Google Map centered in London, England:

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

<h1>My First Google Map</h1>

<div id="map" style="width:400px;height:400px;"></div>

<script>
function myMap() {
   var mapOptions = {
      center: new google.maps.LatLng(51.5, -0.12),
      zoom: 10,
      mapTypeId: google.maps.MapTypeId.HYBRID
   }
var map = new google.maps.Map(document.getElementById("map"),
mapOptions);
}
</script>

</body>
</html>
```

Example Explained
The mapOptions variable defines the properties for the map.

The center property specifies where to center the map (using latitude and longitude coordinates).

The zoom property specifies the zoom level for the map (try to experiment with the zoom level).

The mapTypeId property specifies the map type to display. The following map types are supported: ROADMAP, SATELLITE, HYBRID, and TERRAIN.

The line: var map=new google.maps.Map(document.getElementById("map"), mapOptions); creates a new map inside the element with id="map", using the parameters that are passed (mapOptions).

Add the Google Maps API
Finally, show the map on the page!

The functionality of the map is provided by a JavaScript library located at Google. Add a script to refer to the Google Maps API with a callback to the myMap function:

Example

```
<html>
<body>

<h1>My First Google Map</h1>

<div id="map" style="width:400px;height:400px;background:yellow"></div>

<script>
function myMap() {
var mapOptions = {
   center: new google.maps.LatLng(20.5937, 78.9629), // india as centre
   zoom: 10,
   mapTypeId: google.maps.MapTypeId.HYBRID
}
var map = new google.maps.Map(document.getElementById("map"),
mapOptions);
}
</script>

<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBu-916DdpKAjTmJN
IgngS6HL_kDIKU0aU&callback=myMap"></script>   //may not be included
<!--
To use this code on your website, get a free API key from Google.
Read more at: https://www.w3schools.com/graphics/google_maps_basic.asp
-->

</body>
</html>
```

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.

Examples: Images, music, sound, videos, records, films, animations, and more.
Web pages often contain multimedia elements of different types and formats.
In this chapter you will learn about the different multimedia formats.

Common Video Formats

MP4 is the new and upcoming format for internet video.
MP4 is recommended by YouTube.
MP4 is supported by Flash Players.
MP4 is supported by HTML5.

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5 (See MP4).
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers. (See MP4)
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.

WebM	.webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube.

Only MP4, WebM, and Ogg video are supported by the HTML5 standard.

Only MP3, WAV, and Ogg audio are supported by the HTML5 standard.
// audio format development has similar story as of video format…

Playing Videos in HTML

Before HTML5, a video could only be played in a browser with a plug-in (like flash).
The HTML5 <video> element specifies a standard way to embed a video in a web page.

The HTML <video> Element

To show a video in HTML, use the <video> element:

Example

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

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

</body>
</html>
```

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

The autoplay attribute does not work in mobile devices like iPad and iPhone.

HTML Video - Methods, Properties, and Events

HTML5 defines DOM methods, properties, and events for the <video> element. This allows you to load, play, and pause videos, as well as setting duration and volume.

There are also DOM events that can notify you when a video begins to play, is paused, etc.

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

<div style="text-align:center">
  <button onclick="playPause()">Play/Pause</button>
  <button onclick="makeBig()">Big</button>
  <button onclick="makeSmall()">Small</button>
  <button onclick="makeNormal()">Normal</button>
  <br><br>
  <video id="video1" width="420">
    <source src="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogg" type="video/ogg">
    Your browser does not support HTML5 video.
  </video>
</div>

<script>
var myVideo = document.getElementById("video1");

function playPause() {
    if (myVideo.paused)
        myVideo.play();
    else
        myVideo.pause();
}

function makeBig() {
    myVideo.width = 560;
}

function makeSmall() {
    myVideo.width = 320;
}
```

```
function makeNormal() {
   myVideo.width = 420;
}
</script>

<p>Video courtesy of <a href="https://www.bigbuckbunny.org/"
target="_blank">Big Buck Bunny</a>.</p>
</body>
</html>
```

HTML5 Video Tags

Tag	Description
1.<video>	Defines a video or movie
2.<source>	Defines multiple media resources for media elements, such as <video> and <audio>
3.<track>	Defines text tracks in media players

HTML5 Audio

Audio on the Web

Before HTML5, audio files could only be played in a browser with a plug-in (like flash).

The HTML5 <audio> element specifies a standard way to embed audio in a web page.

The HTML <audio> Element

To play an audio file in HTML, use the <audio> element:

Example

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

<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>

</body>
</html>
```

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 - Media Types

File Format	Media Type
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

HTML Audio - Methods, Properties, and Events //**similar as video methods**
HTML5 defines DOM 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.

HTML Plug-ins

The purpose of a plug-in is to extend the functionality of a web browser.

HTML Helpers (Plug-ins)

Helper applications (plug-ins) are computer programs that extend the standard functionality of a web browser.
Examples of well-known plug-ins are Java applets.Plug-ins can be added to web pages with the <object> tag or the <embed> tag.
Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.
To display video and audio: Use the <video> and <audio> tags.

The <object> Element

The <object> element is supported by all browsers.
The <object> element defines an embedded object within an HTML document.
It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages.
Example

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

<object width="400" height="50" data="bookmark.swf"></object>

</body>
</html>
```

The <object> element 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 width="400" height="50" src="bookmark.swf">
```

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">
```

Or images if you like:

Example

```
<embed src="audi.jpeg">
```

HTML YouTube Videos

The easiest way to play videos in HTML, is to use YouTube.

Struggling with Video Formats?

Earlier in this tutorial, you have seen that you might have to convert your videos to different formats to make them play in all browsers.
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 - Using iFrame (recommended)

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

```
<iframe width="420" height="345"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>

</body>
</html>
```

YouTube Autoplay

You can have your video start playing automatically when a user visits that page by adding a simple parameter to your YouTube URL.

Note: Take careful consideration when deciding to autoplay your videos. Automatically starting a video can annoy your visitor and end up causing more harm than good.

Value 0 (default): The video will not play automatically when the player loads.

Value 1: The video will play automatically when the player loads.

YouTube - Autoplay

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1">
</iframe>
```

YouTube Playlist

A comma separated list of videos to play (in addition to the original URL).

YouTube Loop

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

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>
```

YouTube - Using <object> or <embed>

Note: YouTube <object> and <embed> were **deprecated** from January 2015. You should migrate your videos to use <iframe> instead.

Example - Using <object> (deprecated)

```
<object width="420" height="315"
data="https://www.youtube.com/embed/tgbNymZ7vqY">
</object>
```

Example - Using <embed> (deprecated)

```
<embed width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
```

HTML APIs(application program interface)

HTML5 Geolocation

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.

Note: Geolocation is most accurate for devices with GPS, like iPhone.

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:

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

<p>Click the button to get your coordinates.</p>
<button onclick="getLocation()">Try It</button>
<p id="demo"></p>

<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

```
</body>
</html>
```

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) {
   switch(error.code) {
      case error.PERMISSION_DENIED:
         x.innerHTML = "User denied the request for Geolocation."
         break;
      case error.POSITION_UNAVAILABLE:
         x.innerHTML = "Location information is unavailable."
         break;
      case error.TIMEOUT:
         x.innerHTML = "The request to get user location timed out."
         break;
      case error.UNKNOWN_ERROR:
         x.innerHTML = "An unknown error occurred."
         break;
   }
}
```

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) {
   var latlon = position.coords.latitude + "," + position.coords.longitude;

   var img_url = "https://maps.googleapis.com/maps/api/staticmap?center=
   "+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR_:KEY";

   document.getElementById("mapholder").innerHTML = "<img
src='"+img_url+"'>";
}
```

//To use this code on your website, get a free API key from Google.
//Read more at: https://www.w3schools.com/graphics/google_maps_basic.asp

How to show an interactive Google Map with a marker, zoom and drag options.

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

<p id="demo">Click the button to get your position.</p>

<button onclick="getLocation()">Try It</button>

<div id="mapholder"></div>

<script
src="https://maps.google.com/maps/api/js?key=AIzaSyBu-916DdpKAjTmJNIgngS6HL_kDIKU0aU"></script>
<!--
To use this code on your website, get a free API key from Google.
Read more at: https://www.w3schools.com/graphics/google_maps_basic.asp
-->
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition, showError);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}

function showPosition(position) {
    var lat = position.coords.latitude;
    var lon = position.coords.longitude;
    var latlon = new google.maps.LatLng(lat, lon)
    var mapholder = document.getElementById('mapholder')
    mapholder.style.height = '250px';
    mapholder.style.width = '500px';

    var myOptions = {
    center:latlon,zoom:14,
    mapTypeId:google.maps.MapTypeId.ROADMAP,
    mapTypeControl:false,
    navigationControlOptions:{style:google.maps.NavigationControlStyle.SMALL}
    }

    var map = new google.maps.Map(document.getElementById("mapholder"),
myOptions);
    var marker = new google.maps.Marker({position:latlon,map:map,title:"You
are here!"});
```

```
}

function showError(error) {
   switch(error.code) {
      case error.PERMISSION_DENIED:
         x.innerHTML = "User denied the request for Geolocation."
         break;
      case error.POSITION_UNAVAILABLE:
         x.innerHTML = "Location information is unavailable."
         break;
      case error.TIMEOUT:
         x.innerHTML = "The request to get user location timed out."
         break;
      case error.UNKNOWN_ERROR:
         x.innerHTML = "An unknown error occurred."
         break;
   }
}
</script>

</body>
</html>
```

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:

1.Up-to-date local information
2.Showing Points-of-interest near the user
3.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 iPhone):

Example

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.watchPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

HTML5 Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.
In HTML5, drag and drop is part of the standard: Any element can be draggable.

HTML Drag and Drop Example

The example below is a simple drag and drop example:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1 {
   width: 350px;
   height: 70px;
   padding: 10px;
   border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
   ev.preventDefault();
}

function drag(ev) {
   ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
   ev.preventDefault();
   var data = ev.dataTransfer.getData("text");
   ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<p>Drag the W3Schools image into the rectangle:</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

It might seem complicated, but let's go through all the different parts of a drag and drop event.

Make an Element Draggable

First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

What to Drag - ondragstart and setData()

Then, specify what should happen when the element is dragged.

In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
```

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

Where to Drop - ondragover

The ondragover event specifies where the dragged data can be dropped.

By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

This is done by calling the event.preventDefault() method for the ondragover event:

```
event.preventDefault()
```

Do the Drop - ondrop

When the dragged data is dropped, a drop event occurs.

In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

Code explained:

Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)

Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method

The dragged data is the id of the dragged element ("drag1")

Append the dragged element into the drop element

Drag image back and forth

How to drag (and drop) an image back and forth between two elements:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1, #div2 {
```

```
    float: left;
    width: 100px;
    height: 35px;
    margin: 10px;
    padding: 10px;
    border: 1px solid black;
}
</style>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<h2>Drag and Drop</h2>
<p>Drag the image back and forth between the two div elements.</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
  <img src="img_w3slogo.gif" draggable="true" ondragstart="drag(event)"
id="drag1" width="88" height="31">
</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

</body>
</html>
```

HTML5 Web Storage

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.

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") {
   // Code for localStorage/sessionStorage.
} else {
   // Sorry! No Web Storage support..
}
```

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Example

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

<div id="result"></div>

<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
   // Store
   localStorage.setItem("lastname", "Smith");
   // Retrieve
   document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
} else {
   document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
}
</script>

</body>
</html>
```

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

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
   if(typeof(Storage) !== "undefined") {
      if (localStorage.clickcount) {
         localStorage.clickcount = Number(localStorage.clickcount)+1;
      } else {
         localStorage.clickcount = 1;
      }
      document.getElementById("result").innerHTML = "You have clicked the
button " + localStorage.clickcount + " time(s).";
   } else {
      document.getElementById("result").innerHTML = "Sorry, your browser
does not support web storage...";
   }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter will
continue to count (is not reset).</p>
</body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (sessionStorage.clickcount) {
            sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
        } else {
            sessionStorage.clickcount = 1;
        }
        document.getElementById("result").innerHTML = "You have clicked the
button " + sessionStorage.clickcount + " time(s) in this session.";
    } else {
        document.getElementById("result").innerHTML = "Sorry, your browser
does not support web storage...";
    }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is
reset.</p>
</body>
</html>
```

HTML5 Web Workers

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.

HTML Web Workers Example

The example below creates a simple web worker that count numbers in the background:

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

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support Web Workers.</p>

<script>
var w;

function startWorker() {
    if(typeof(Worker) !== "undefined") {
        if(typeof(w) == "undefined") {
            w = new Worker("demo_workers.js");
        }
        w.onmessage = function(event) {
            document.getElementById("result").innerHTML = event.data;
        };
    } else {
        document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Workers...";
    }
}

function stopWorker() {
    w.terminate();
    ;
}
</script>

</body>
</html>
```

Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it:

```
if (typeof(Worker) !== "undefined") {
    // Yes! Web worker support!
    // Some code.....
} else {
    // Sorry! No Web Worker support..
}
```

Create a Web Worker File

Now, let's create our web worker in an external JavaScript.

Here, we create a script that counts. The script is stored in the "demo_workers.js" file:

```
var i = 0;

function timedCount() {
   i = i + 1;
   postMessage(i);
   setTimeout("timedCount()",500);
}

timedCount();
```

The important part of the code above is the postMessage() method - which is used to post a message back to the HTML page.

Note: Normally web workers are not used for such simple scripts, but for more CPU intensive tasks.

Create a Web Worker Object

Now that we have the web worker file, we need to call it from an HTML page.

The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo_workers.js":

```
if (typeof(w) == "undefined") {
   w = new Worker("demo_workers.js");
}
```

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

```
w.onmessage = function(event){
   document.getElementById("result").innerHTML = event.data;
};
```

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

Terminate a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.

To terminate a web worker, and free browser/computer resources, use the terminate() method:

w.terminate();

Reuse the Web Worker

If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

w = undefined;

Full Web Worker Example Code
We have already seen the Worker code in the .js file. Below is the code for the HTML page:

Example

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

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support Web Workers.</p>

<script>
var w;

function startWorker() {
    if(typeof(Worker) !== "undefined") {
        if(typeof(w) == "undefined") {
            w = new Worker("demo_workers.js");
        }
        w.onmessage = function(event) {
            document.getElementById("result").innerHTML = event.data;
        };
    } else {
        document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Workers...";
    }
}

function stopWorker() {
    w.terminate();
    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

HTML5 Server-Sent Events

Server-Sent Events allow a web page to get updates from a server.

Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Receive Server-Sent Event Notifications

The EventSource object is used to receive server-sent event notifications:

Example

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

<h1>Getting server updates</h1>
<div id="result"></div>

<script>
if(typeof(EventSource) !== "undefined") {
   var source = new EventSource("demo_sse.php");
   source.onmessage = function(event) {
      document.getElementById("result").innerHTML += event.data + "<br>";
   };
} else {
   document.getElementById("result").innerHTML = "Sorry, your browser does not support server-sent events...";
}
</script>

</body>
</html>
```

Example explained:

- Create a new EventSource object, and specify the URL of the page sending the updates (in this example "demo_sse.php")
- Each time an update is received, the onmessage event occurs
- When an onmessage event occurs, put the received data into the element with id="result"
-

Check Server-Sent Events Support

In the tryit example above there were some extra lines of code to check browser support for server-sent events:

```
if(typeof(EventSource) !== "undefined") {
    // Yes! Server-sent events support!
    // Some code.....
} else {
    // Sorry! No server-sent events support..
}
```

Server-Side Code Example

For the example above to work, you need a server capable of sending data updates (like PHP or ASP).

The server-side event stream syntax is simple. Set the "Content-Type" header to "text/event-stream". Now you can start sending event streams.

Code in PHP (demo_sse.php):

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

Code in ASP (VB) (demo_sse.asp):

```
<%
Response.ContentType = "text/event-stream"
Response.Expires = -1
Response.Write("data: The server time is: " & now())
Response.Flush()
%>
```

Code explained:

Set the "Content-Type" header to "text/event-stream"
Specify that the page should not cache
Output the data to send (Always start with "data: ")
Flush the output data back to the web page

The EventSource Object

In the examples above we used the onmessage event to get messages. But other events are also available:

Events	Description
onopen	When a connection to the server is opened

onmessage When a message is received
onerror When an error occurs

HTML Comments

Hidden comments

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

<!-- This is a comment -->
<p>This is a paragraph.</p>
<!-- Comments are not displayed in the browser -->

</body>
</html>
```

Conditional comments

```
<!--[if IE 5]>This is IE 5<br><![endif]-->
<!--[if IE 6]>This is IE 6<br><![endif]-->
```

Comments for debugging

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

Removing the underline from links

```
<a href="html_images.asp" style="text-decoration:none">HTML
Images</a>
```

A mailto link

```
<p>
This is an email link:
<a href="mailto:someone@example.com?Subject=Hello%20again"
target="_top">
Send Mail</a>
</p>
```