HTML-HyperText Markup Language.

Heading Tag: <h1>... to<h6>
specify the size for any heading with the style attribute
<h1 style="font-size:60px;">Heading 1</h1>

Paragraph Tag: <p>...</p>

Link / Anchor Tag: <a href="...{URL of the page}"> </a>

Image Tag: <img src="...{path}" alt"...{alternate text for an image}" width="..." height="..">
#note: There are two ways to specify the URL in the src attribute:
1. Absolute URL - Links to an external image that is hosted on another website. Example: src="https://www.w3schools.com/images/img_girl.jpg".
Notes: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.
2. Relative URL - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page.
Example: src="img_girl.jpg". If the URL begins with a slash, it will be relative to the domain. Example: src="/images/img_girl.jpg".
Tip: It is almost always best to use relative URLs. They will not break if you change domain.

Horizontal Rule Tag: Used to get a horizontal line on a webpage <hr>

Line Break Tag: Used to break a line <br>

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

style Attribute: <tagname style="property:value;">
The property is a CSS property. The value is a CSS value.
The style attribute is used to add styles to an element, such as color, font, size, and more.
Some styles:
Background color: <body style="background-color:powderblue;">
Text Color: <h1 style="color:blue;">This is a heading</h1>
Fonts: <h1 style="font-family:verdana;">This is a heading</h1>
Textsize: <h1 style="font-size:300%;">This is a heading</h1>
Text Alignment: <h1 style="text-align:center;">Centered Heading</h1>

Formatting Tags:
<b> - Bold text
<strong> - Important text
<i> - Italic text
<em> - Emphasised text
<mark> - Marked text / Highlighted text
<small> - Smaller text
<del> - Deleted text
<ins> - Inserted text/Underline text
<sub> - Subscript text
<sup> - Superscript text

Quotation & Citation Tags:
<blockquote></blockquote> - a section that is quoted from another source browsers usually indent blockquote elements.
<q></q> - defines a short quotation / insert quotation marks around the quotation.
<abbr title="...">....</abbr> - It defines an abbreviation or an acronym.
<address>...</address> - it defines the contact information for the author
<cite>...</cite> - tag defines the title of a creative work. #Note: A person's name is not the title of a work.
<bdo dir="rtl / ltr">...</bdo> Bi-Directional Override.(Reverses the text)
# Note: Here, <i>, <cite>, <em> tags give the same output. But they are used for different purposes.
Comment Tag:
<!-- Write your comments here -->
HTML comments are not displayed in the browser, but they can help document your HTML source code.
These comments are used to add comments, hide content, hide inline content.

Colors in HTML:
Background color: <h1 style="background-color:DodgerBlue;">Hello World</h1>
Text color: <h1 style="color:Tomato;">Hello World</h1>
Border color: <h1 style="border:2px solid Tomato;">Hello World</h1>.
• RGB Color:
rgb(red, green, blue)
Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.
(0,0,0) -> Black.
(255,255,255) -> White.
56 x 256 x 256 = 16777216 possible colors!
Shades of gray are often defined using equal values for all three parameters, Like; rgb(60, 60, 60), rgb(240, 240, 240) ... etc.
• RGBA Color:
RGBA color values are an extension of RGB color values with an Alpha channel - which specifies the opacity for a color.
Syntax: rgba(red, green, blue, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all).
eg :- rgba(255, 99, 71, 0), rgba(255, 99, 71, 0.2), rgba(255, 99, 71, 1)

• Hex Color:
#rrggbb
Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).
For example, #ff0000 is displayed as red, because red is set to its highest value (ff), and the other two (green and blue) are set to 00.
Another example, #00ff00 is displayed as green, because green is set to its highest value (ff), and the other two (red and blue) are set to 00.
To display black, set all color parameters to 00, like this: #000000.
To display white, set all color parameters to ff, like this: #ffffff.

• HSL Color:
a color can be specified using hue, saturation, and lightness (HSL) in the form:
hsl(hue, saturation, lightness)
Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.
Lightness is also a percentage value. 0% is black, and 100% is white.

• HSLA Color:
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).

HTML Styles - CSS:
CSS stands for Cascading Style Sheets.
Cascading Style Sheets (CSS) is used to format the layout of a webpage.
With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!.
# (Tip: The word cascading means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!).
CSS can be added to HTML documents in 3 ways:
Inline - by using the style attribute inside HTML elements.
Internal - by using a <style> element in the <head> section.
External - by using a <link> element to link to an external CSS file.

• 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.
eg: <h1 style="color:blue;">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.

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

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

In style.css file:

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

# Note: The most common way to add CSS, is to keep the styles in external CSS files. With an external style sheet, you can change the look of an entire web site, by changing one file !

• CSS Colors, Fonts & Sizes :-

  Property:Value;

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

  eg: color: blue;

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

  eg: font-family: verdana;

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

  eg: font-size: 300%;

• CSS Borders :-The CSS border property defines a border around an HTML element.

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

```
   p {
  border: 2px solid powderblue; (border: how_many_pixels type_of_border color_of_border;)
  }
```

• CSS Padding :- The CSS padding property defines a padding (space) between the text and the border.

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

• CSS Margin:- The CSS margin property defines a margin (space) outside the border.

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

HTML Links - Hyperlinks:

Links are found in nearly all web pages. Links allow users to click their way from page to page.
You can click on a link and jump to another document.
When you move the mouse over a link, the mouse arrow will turn into a little hand.
# Note: A link does not have to be text. A link can be an image or any other HTML element!
The HTML <a> tag defines a hyperlink. It has the following syntax:

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

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.
The link text is the part that will be visible to the reader.
Clicking on the link text, will send the reader to the specified URL address.
By default, links will appear as follows in all browsers:
An unvisited link is underlined and blue
A visited link is underlined and purple
An active link is underlined and red

• The target Attribute:
By default, the linked page will be displayed in the current browser window.
To change this, you must specify another target for the link.
The target attribute specifies where to open the linked document.
The target attribute can have one of the following values:
_self - Default. Opens the document in the same window/tab as it was clicked.
_blank - Opens the document in a new window or tab.
_parent - Opens the document in the parent frame.
_top - Opens the document in the full body of the window.
eg: <a href="https://www.w3schools.com/" target="_blank">Visit My Website</a>.

• Absolute URL vs. Relative URL:
Example above is using an absolute URL (a full web address) in the href attribute.
A local link (a link to a page within the same website) is specified with a relative URL (without the "https://www" part).

```
eg: <h2>Absolute URL</h2>
<p><a href="https:// www.MyWebsite.org/">Click Here</a></p>
<h2>Relative URL</h2>
<p><a href="html_images.asp">HTML Images</a></p>
```

• Use an Image as a Link:

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

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

- Link to an Email Address:

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

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

- Button as a Link:

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

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

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

- 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.

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

- HTML Link Colors:

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

You can change the link state colors, by using CSS:

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

```
 <style>
 a:link {
 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>
```

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

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

• Create Bookmarks:
HTML links can be used to create bookmarks, so that readers can jump to specific parts of a web page.
Bookmarks can be useful if a web page is very long.
To create a bookmark - first create the bookmark, then add a link to it.
When the link is clicked, the page will scroll down or up to the location with the bookmark.
eg:
First, use the id attribute to create a bookmark:

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

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

Example:

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

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

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

The HTML <img> tag is used to embed an image in a web page.
Images are not technically inserted into a web page; images are linked to web pages. The <img> tag
creates a holding space for the referenced image.
The <img> tag is empty, it contains attributes only, and does not have a closing tag.
The <img> tag has two required attributes:
src - Specifies the path to the image.
alt - Specifies an alternate text for the image.
The src attribute:-
The required src attribute specifies the path (URL) to the image.
Note: When a web page loads, it is the browser, at that moment, that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise your visitors will get a broken link icon.
The broken link icon and the alt text are shown if the browser cannot find the image.
The alt attribute:-

The required alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).
Tip: A screen reader is a software program that reads the HTML code, and allows the user to "listen" to the content. Screen readers are useful for people who are visually impaired or learning disabled.

- Image Size - Width and Height or Style? :-

The width, height, and style attributes are all valid in HTML.
However, it is better to use the style attribute. It prevents styles sheets from changing the size of images:
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  width: 100%;
}
</style>
</head>
<body>
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
</body>
</html>
```

- Images in Another Folder:-

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

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

- Images on Another Server/Website:-

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

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

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

- Animated Images:-

HTML allows animated GIFs:

```
<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:

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

- Image Floating:-

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

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

- Common Image formats:-

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

- Image Maps:-

With HTML image maps, you can create clickable areas on an image.
The HTML <map> tag defines an image map. An image map is an image with clickable areas. The areas are defined with one or more <area> tags.

```html
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">
<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="Coffee" href="coffee.htm">
</map>
```

The idea behind an image map is that you should be able to perform different actions depending on where in the image you click.
To create an image map you need an image, and some HTML code that describes the clickable areas.
Create Image:
The image is inserted using the <img> tag. The only difference from other images is that you must add a use map attribute:

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

The use map value starts with a hashtag # followed by the name of the image map, and is used to create a relationship between the image and the image map.
Tip: You can use any image as an image map!
Create Image Map:
Create Image Map
Then, add a <map> element.
The <map> element is used to create an image map, and is linked to the image by using the required name attribute:

```html
<map name="workmap">
```

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

The Areas:
Then, add the clickable areas.
A clickable area is defined using an <area> element.
Shape
You must define the shape of the clickable area, and you can choose one of these values:
rect - defines a rectangular region
circle - defines a circular region
poly - defines a polygonal region
default - defines the entire region
You must also define some coordinates to be able to place the clickable area onto the image.
• Shape="rect"
To add a rectangle:
the coordinates 34,44 is located 34 pixels from the left margin and 44 pixels from the top
The coordinates 270,350 is located 270 pixels from the left margin and 350 pixels from the top
The coordinates for shape="rect" come in pairs, one for the x-axis and one for the y-axis.
eg:
```
<area shape="rect" coords="x,y{from left margin},  x,y{from right margin}"
href="image_name.htm">
<area shape="rect" coords="34, 44,  270, 350" href="computer.htm">
```
• Shape="circle"
To add a circle area, first locate the coordinates of the center of the circle:
Then specify the radius of the circle
eg:
```
<area shape="circle" coords="x,y{Coordinates}, r{Radius of circle}" href="coffee.htm">
```
• Shape="poly"
The shape="poly" contains several coordinate points, which creates a shape formed with straight lines (a polygon).
This can be used to create any shape.
We have to find the x and y coordinates for all edges of the poly
The coordinates come in pairs, one for the x-axis and one for the y-axis
eg:
```
<area shape="poly" coords="x,y...,x,y{we get so many pairs based on image shape}"
href="Image_name.htm">
<area shape="poly"
coords="140,121,181,116,204,160,204,222,191,270,140,329,85,355,58,352,37,322,40,259,
103,161,128,147" href="croissant.htm">
```
Image Map and JavaScript:
A clickable area can also trigger a JavaScript function.
Add a click event to the <area> element to execute a JavaScript function:
Example
Here, we use the on click attribute to execute a JavaScript function when the area is clicked:
```
<map name="workmap">
  <area shape="circle" coords="337,300,44" href="coffee.htm" onclick="myFunction()">
</map>
<script>
function myFunction() {
  alert("You clicked the coffee cup!");
}
</script>
```

• Background Images:

A background image can be specified for almost any HTML element.
To add a background image on an HTML element, use the HTML style attribute and the CSS background-image property:

```
<p style="background-image: url('imagename.format');">
eg:<p style="background-image: url('img_girl.jpg');">
```

• Background Image on a Page:-
If you want the entire page to have a background image, you must specify the background image on the <body> element:

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

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

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

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

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

• Background Cover:-
If you want the background image to cover the entire element, you can set the background-size property to cover.
Also, to make sure the entire element is always covered, set the background-attachment property to fixed:
This way, the background image will cover the entire element, with no stretching (the image will keep its original proportions):

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

• Background Stretch:-

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

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

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

HTML Picture Element:

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

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

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

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

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

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

• When to use the Picture Element:-

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

1. Bandwidth

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

2. Format Support

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

The browser will use the first image format it recognizes:

```
eg:<picture>
```

```
  <source srcset="img_avatar.png">
  <source srcset="img_girl.jpg">
  <img src="img_beatles.gif" alt="Beatles" style="width:auto;">
</picture>
```
# Note: The browser will use the first <source> element with matching attribute values, and ignore any following <source> elements.

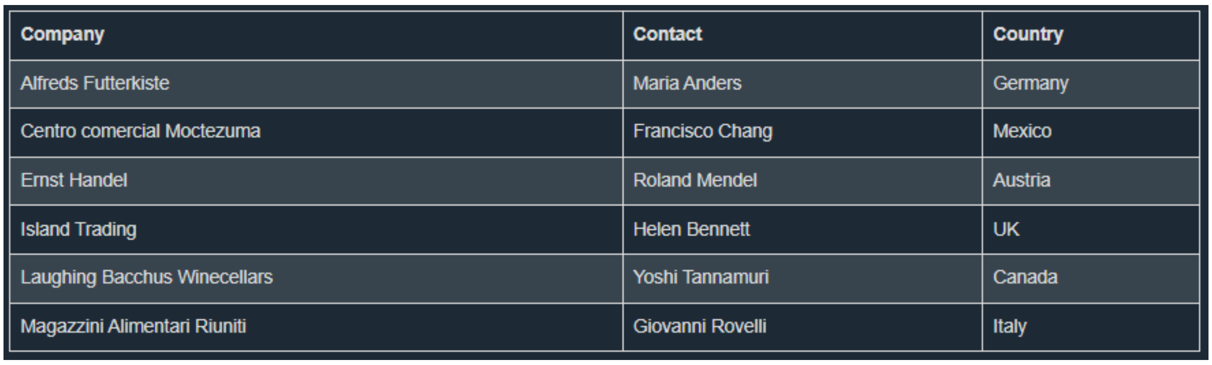
HTML Favicon:
A favicon is a small image displayed next to the page title in the browser tab.
Tip: A favicon is a small image, so it should be a simple image with high contrast.
To add a favicon to your website, either save your favicon image to the root directory of your webserver, or create a folder in the root directory called images, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".
Next, add a <link> element to your "index.html" file, after the <title> element, like this:
```
<!DOCTYPE html>
<html>
<head>
  <title>My Page Title</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```
Now, save the "index.html" file and reload it in your browser. Your browser tab should now display your favicon image to the left of the page title.
Browsers like Microsoft Edge, Chrome, FireFox, Opera, Safari supports .Ico, .png, .gif, .jpeg, .svg.

HTML Page Title:
Every web page should have a page title to describe the meaning of the page.
The <title> element adds a title to your page:
```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Tutorial</title>
</head>
<body>
The content of the document......
</body>
</html>
```
The title should describe the content and the meaning of the page.
The page title is very important for search engine optimization (SEO). The text is used by search engine algorithms to decide the order when listing pages in search results.
The <title> element:
defines a title in the browser toolbar
provides a title for the page when it is added to favourites

displays a title for the page in search engine-results
So, try to make the title as accurate and meaningful as possible!

HTML Tables:
HTML tables allow web developers to arrange data into rows and columns.
Define an HTML Table:-
A table in HTML consists of table cells inside rows and columns.

- <u>A simple HTML table:</u>

eg:
```
<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>
</table>
```


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

- <u>Table Cells:-</u>

Each table cell is defined by a <td> and a </td> tag.
td stands for table data.
Everything between <td> and </td> are the content of the table cell.
```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
```

```
</table>
```
# Note: A table cell can contain all sorts of HTML elements: text, images, lists, links, other tables, etc.

• <u>Table Rows:-</u>

Each table row starts with a <tr> and ends with a </tr> tag.
tr stands for table row.
eg:
```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
  </tr>
</table>
```
You can have as many rows as you like in a table; just make sure that the number of cells are the same in each row.
# Note: There are times when a row can have less or more cells than another. You will learn about that in a later chapter.

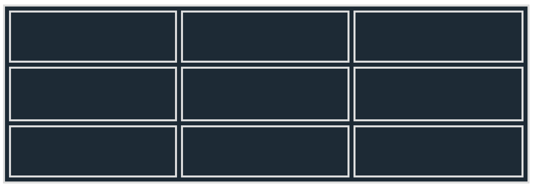
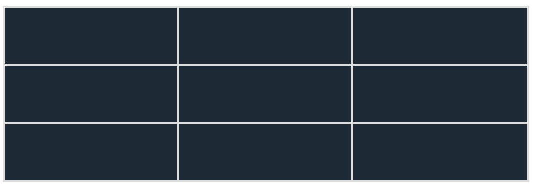
• <u>Table Headers:-</u>

Sometimes you want your cells to be table header cells. In those cases use the <th> tag instead of the <td> tag:
th stands for table header.
```
<table>
  <tr>
    <th>Person 1</th>
    <th>Person 2</th>
    <th>Person 3</th>
  </tr>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
  </tr>
</table>
```
By default, the text in <th> elements are bold and centered, but you can change that with CSS.

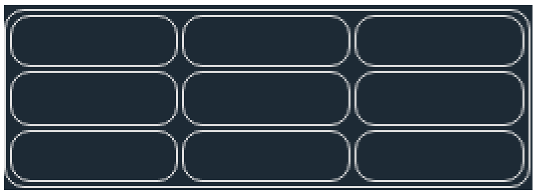
• <u>Table Tags:-</u>

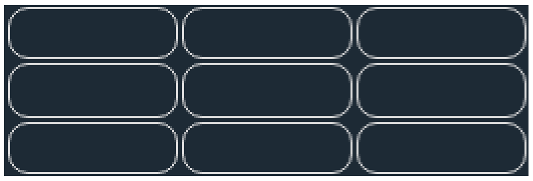
<table>Defines a table

<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

• <u>Table Borders:-</u>
HTML tables can have borders of different styles and shapes.
To add a border, use the CSS border property on table, th, and td elements.
eg:
```
table, th, td {
  border: 1px solid black;
}
```

• <u>Collapsed Table Borders:-</u>
To avoid having double borders set the CSS border-collapse property to collapse.
eg:
```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

• <u>Style Table Borders:-</u>
If you set a background color of each cell, and give the border a white color (the same as the document background), you get the impression of an invisible border.
eg:
```
table, th, td {
  border: 1px solid white;
  border-collapse: collapse;
}
th, td {
  background-color: #96D4D4;
}
```

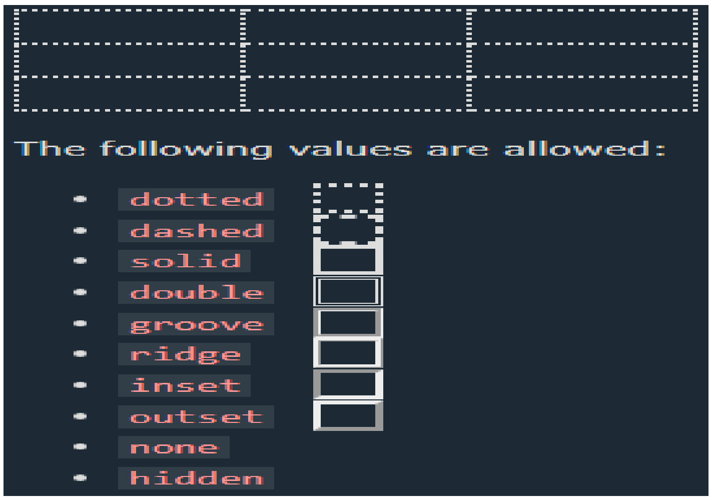
- <u>Round Table Borders:-</u>

With the border-radius property, the borders get rounded corners
eg:
```
table, th, td {
  border: 1px solid black;
  border-radius: 10px;
}
```


Skip the border around the table by leaving out table from the css selector.
eg:
```
th, td {
  border: 1px solid black;
  border-radius: 10px;
}
```


- <u>Some more types of Table Borders:-</u>

With the border-style property, you can set the appearance of the border.
dotted, dashed, solid, double, groove, ridge, inset, outset, none, hidden.
eg:
```
th, td {
  border-style: dotted;
}
```
Border Color:-
With the border-color property, you can set the color of the border.
```
 th, td {
  border-color: #96D4D4;
}
```

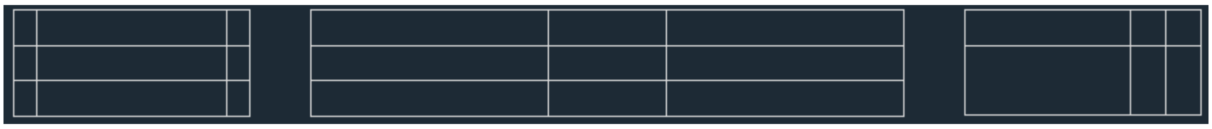


The following values are allowed:

- dotted
- dashed
- solid
- double
- groove
- ridge
- inset
- outset
- none
- hidden

• Table Sizes:-

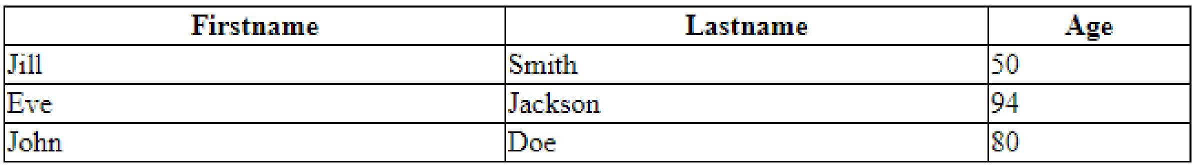
HTML tables can have different sizes for each column, row or the entire table.
Use the style attribute with the width or height properties to specify the size of a table, row or column.



• Table Width:-

To set the width of a table, add the style attribute to the <table> element.
eg:
Set the width of the table to 100%:

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



Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Note: Using a percentage as the size unit for a width means how wide will this element be compared to its parent element, which in this case is the <body> element.

- Table Column Width:-

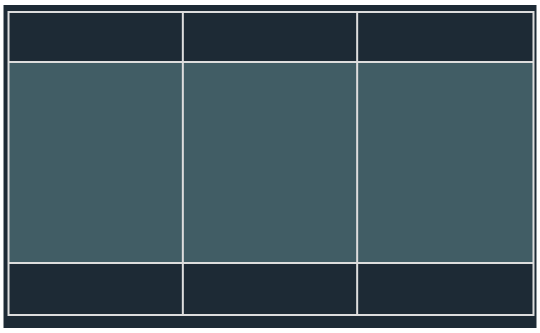
To set the size of a specific column, add the style attribute on a <th> or <td> element.
eg:
Set the width of the first column to 70%:

```
<table style="width:100%">
  <tr>
    <th style="width:70%">Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

- Table Row Height:-

To set the height of a specific row, add the style attribute on a table row element.
eg:
Set the height of the second row to 200 pixels:

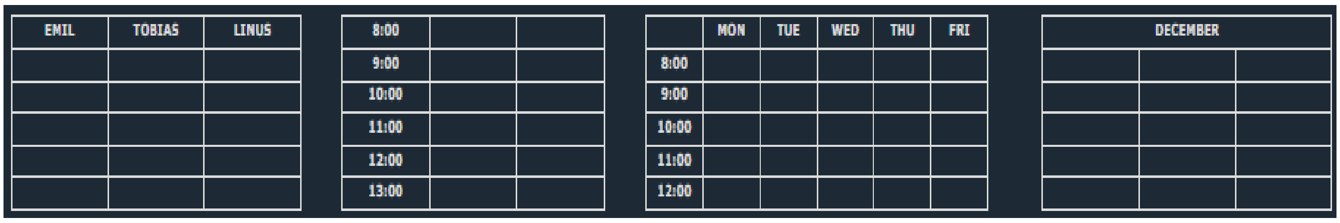
```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr style="height:200px">
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr></table>
```



• <u>Table Headers:-</u>

HTML tables can have headers for each column or row, or for many columns/rows.

Table Headers:-



EMIL	TOBIAS	LINUS

8:00		
9:00		
10:00		
11:00		
12:00		
13:00		

	MON	TUE	WED	THU	FRI
8:00					
9:00					
10:00					
11:00					
12:00					

DECEMBER		

HTML tables can have headers for each column or row, or for many columns/rows.

• <u>Headers:-</u>

Table headers are defined with the elements. Each th element represents a table cell

eg:

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

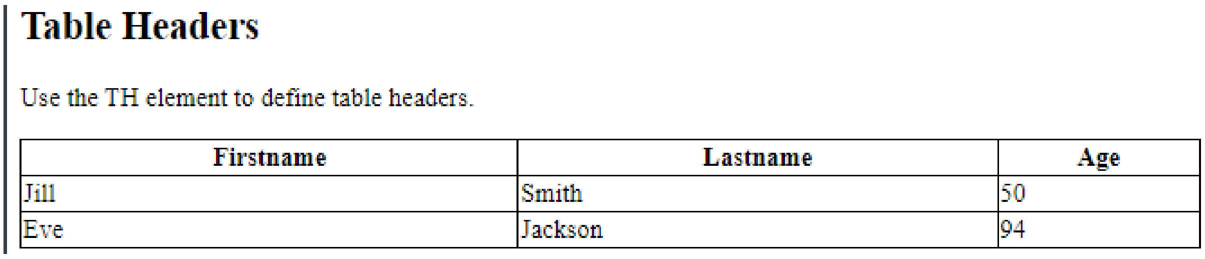


Table Headers

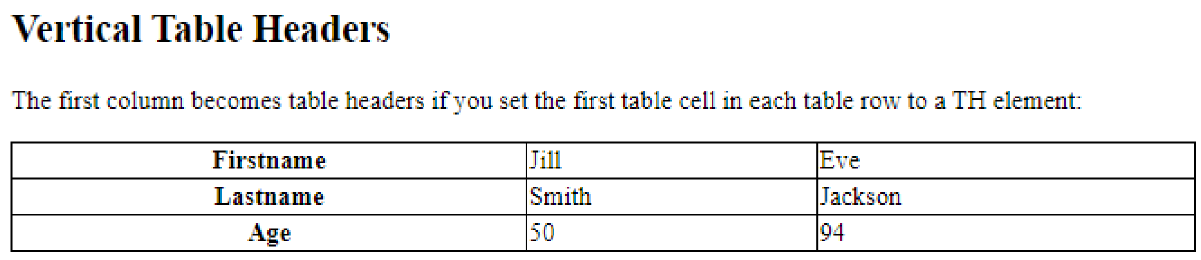
Use the TH element to define table headers.

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

• <u>Vertical Table Headers:-</u>

To use the first column as table headers, define the first cell in each row as a <th> element:

```
<table>
  <tr>
    <th>Firstname</th>
    <td>Jill</td>
    <td>Eve</td>
  </tr>
  <tr>
    <th>Lastname</th>
    <td>Smith</td>
    <td>Jackson</td>
  </tr>
  <tr>
    <th>Age</th>
    <td>94</td>
    <td>50</td>
  </tr>
</table>
```



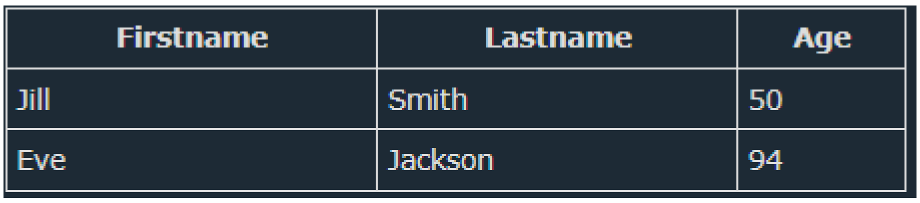
Vertical Table Headers

The first column becomes table headers if you set the first table cell in each table row to a TH element:

Firstname	Jill	Eve
Lastname	Smith	Jackson
Age	50	94

• <u>Align Table Headers:-</u>

By default, table headers are bold and centered:

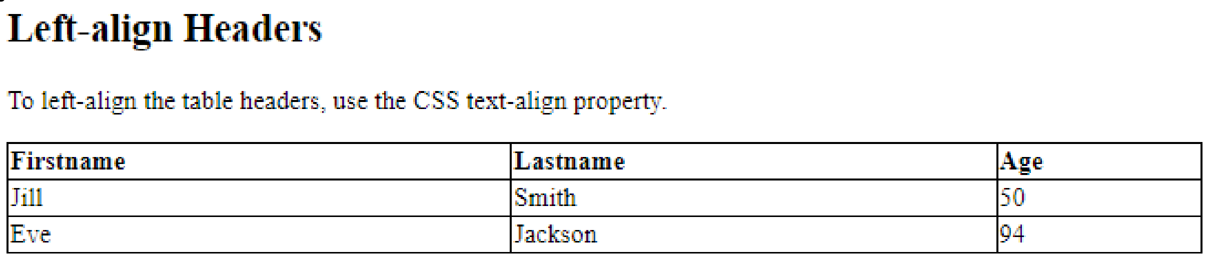


Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

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

eg:

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



Left-align Headers

To left-align the table headers, use the CSS text-align property.

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

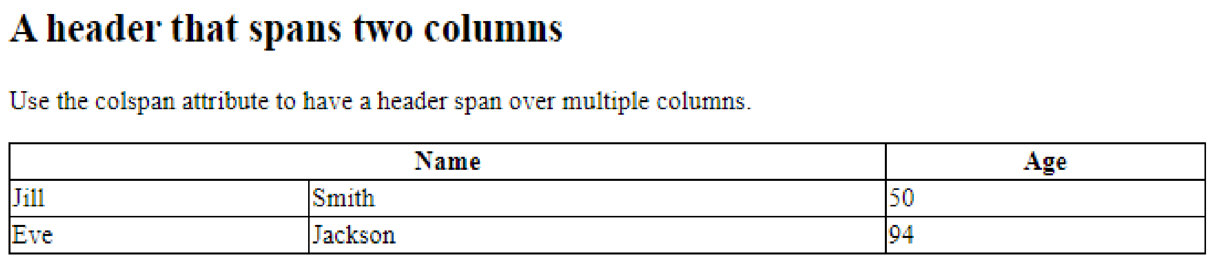
• <u>Header for Multiple Columns:-</u>

You can have a header that spans over two or more columns.

To do this, use the colspan attribute on the <th> element:

```
<table>
```

```
  <tr>
    <th colspan="2">Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```



A header that spans two columns

Use the colspan attribute to have a header span over multiple columns.

Name		Age
Jill	Smith	50
Eve	Jackson	94

- Table Caption:-

You can add a caption that serves as a heading for the entire table.
To add a caption to a table, use the <caption> tag:
eg:

```
<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>
```

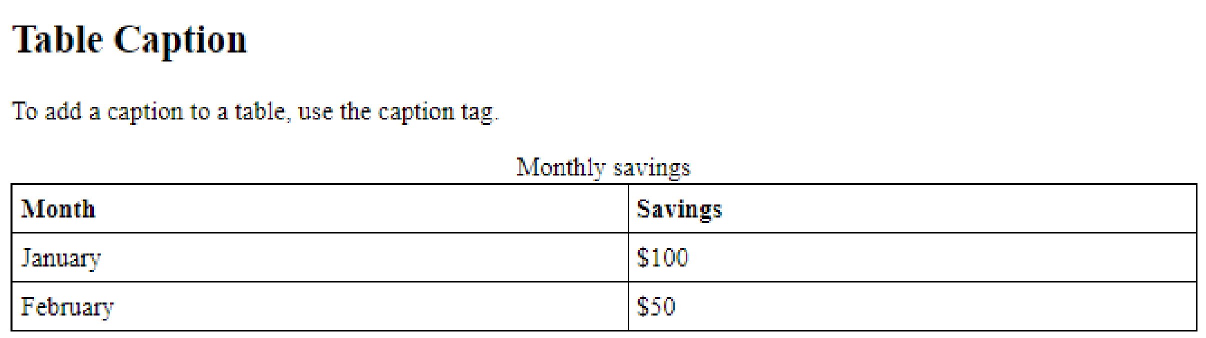


Table Caption

To add a caption to a table, use the caption tag.

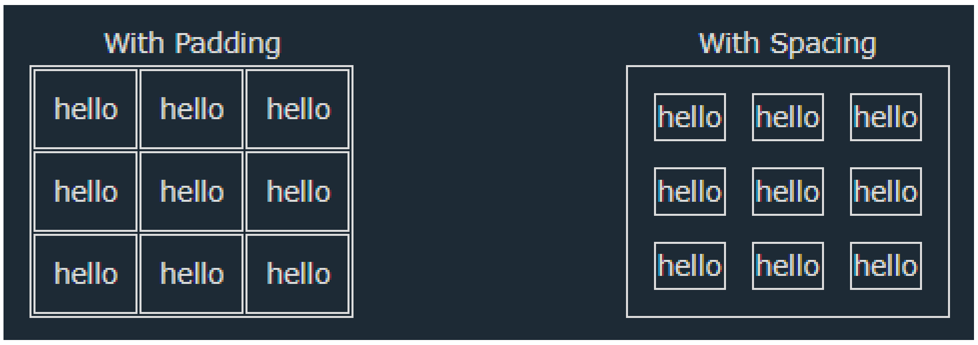
Monthly savings

Month	Savings
January	$100
February	$50

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

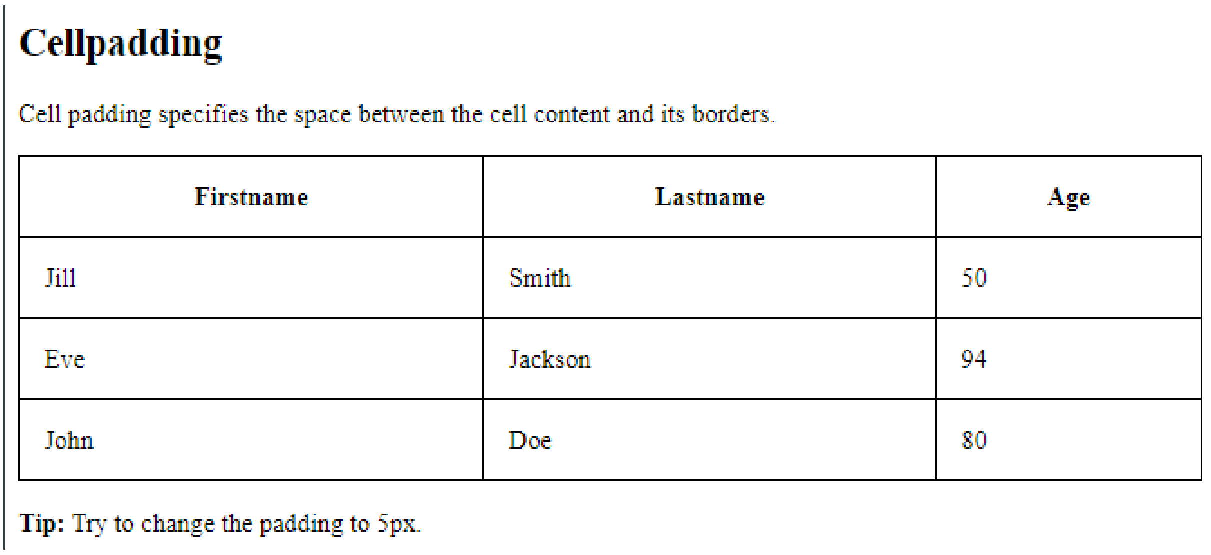
- Padding & Spacing:-

HTML tables can adjust the padding inside the cells, and also the space between the cells.



- Cell Padding:-

Cell padding is the space between the cell edges and the cell content.
By default the padding is set to 0.
To add padding on table cells, use the CSS padding property:



Cellpadding

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

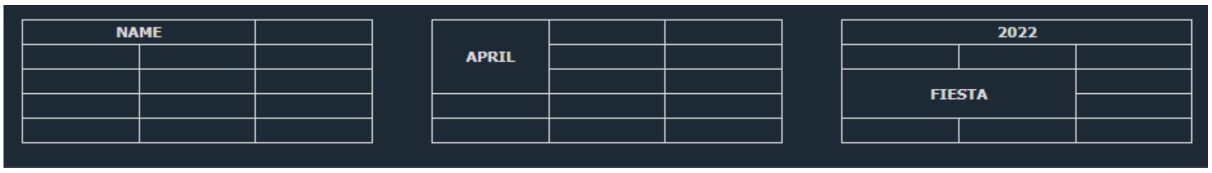
Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Tip: Try to change the padding to 5px.

eg:
```
th, td {
  padding: 15px;
}
```
To add padding only above the content, use the padding-top property.
And the others sides with the padding-bottom, padding-left, and padding-right properties:
```
th, td {
  padding-top: 10px;
  padding-bottom: 20px;
  padding-left: 30px;
  padding-right: 40px;
}
```

• <u>Table Colspan & Rowspan:-</u>

HTML tables can have cells that span over multiple rows and/or columns.



• <u>Colspan:-</u>

To make a cell span over multiple columns, use the colspan attribute

eg:

```
<table>
  <tr>
    <th colspan="2">Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>43</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>57</td>
  </tr>
</table>
```

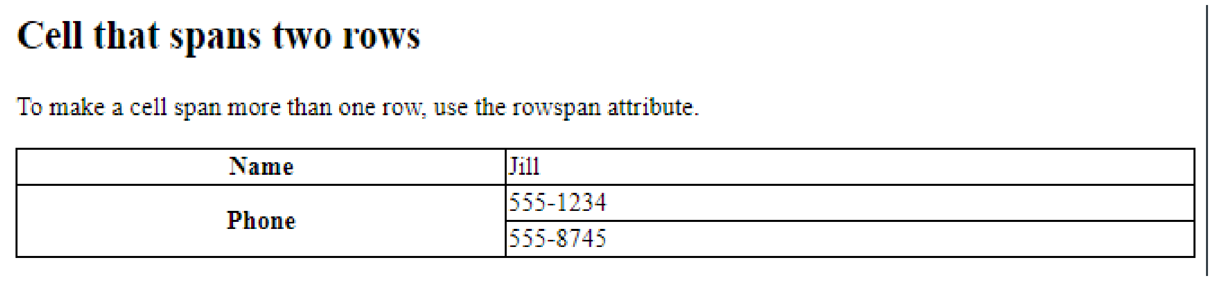
# Note: The value of the colspan attribute represents the number of columns to span.

• <u>Rowspan:-</u>

To make a cell span over multiple rows, use the rowspan attribute:

eg:

```
<table>
  <tr>
    <th>Name</th>
    <td>Jill</td>
  </tr>
  <tr>
    <th rowspan="2">Phone</th>
    <td>555-1234</td>
  </tr>
  <tr>
    <td>555-8745</td>
</tr>
</table>
```

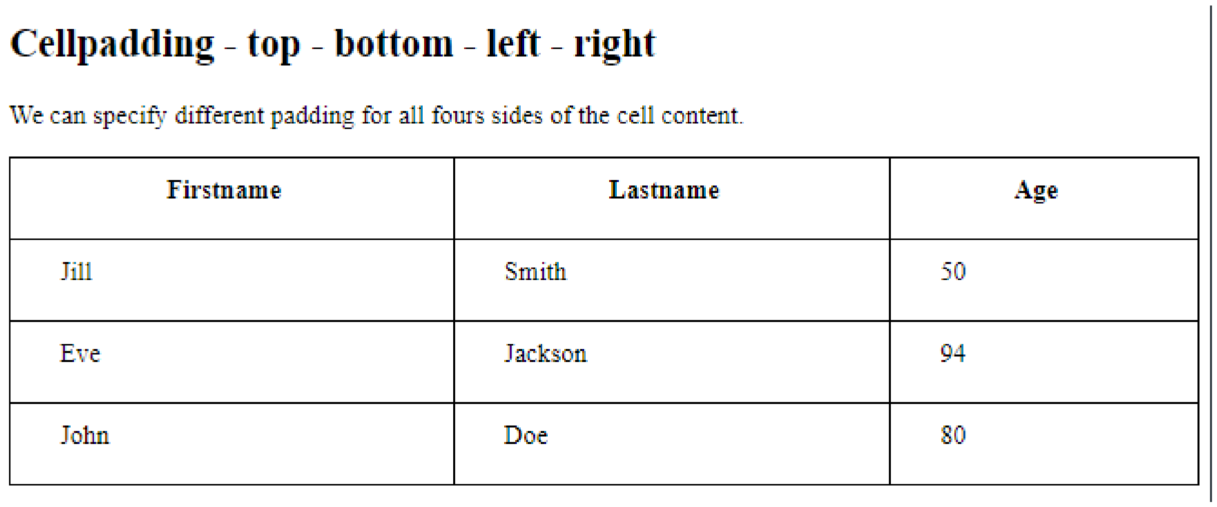


Cell that spans two rows

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

Name	Jill
Phone	555-1234
	555-8745

# Note: The value of the rowspan attribute represents the number of rows to span.



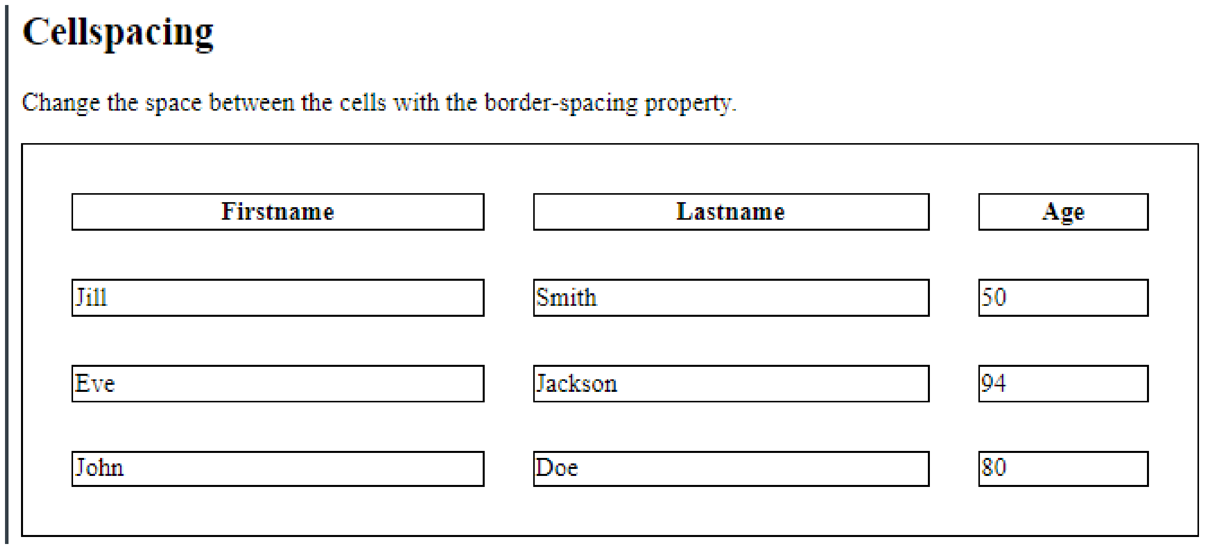
Cellpadding - top - bottom - left - right

We can specify different padding for all fours sides of the cell content.

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

• Cell Spacing:-
Cell spacing is the space between each cell.
By default the space is set to 2 pixels.
To change the space between table cells, use the CSS border-spacing property on the table element:

```
table {
  border-spacing: 30px;
}
```



Cellspacing

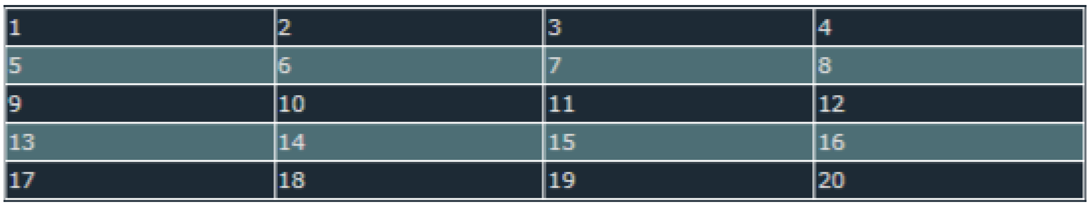
Change the space between the cells with the border-spacing property.

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Table Styling:-
Use CSS to make your tables look better.
• Table - Horizontal Zebra Stripes:-

If you add a background color on every other table row, you will get a nice zebra stripes effect.
To style every other table row element, use the :nth-child(even) selector like this:
eg:

```
tr:nth-child(even) {
  background-color: #D6EEEE;
}
```



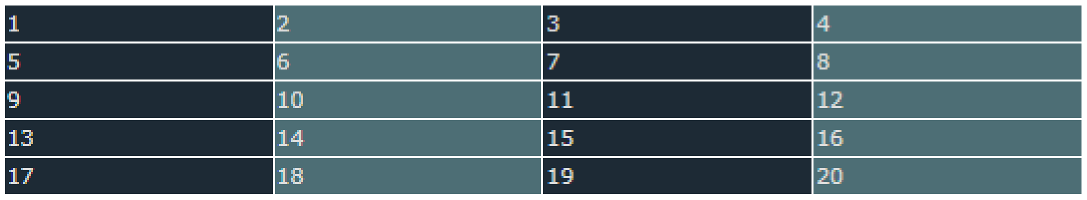
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

# Note: If you use (odd) instead of (even), the styling will occur on row 1,3,5 etc. instead of 2,4,6 etc.

• <u>Table - Vertical Zebra Stripes:-</u>

To make vertical zebra stripes, style every other column, instead of every other row.
Set the :nth-child(even) for table data elements like this:
eg:

```
td:nth-child(even), th:nth-child(even) {
  background-color: #D6EEEE;
}
```

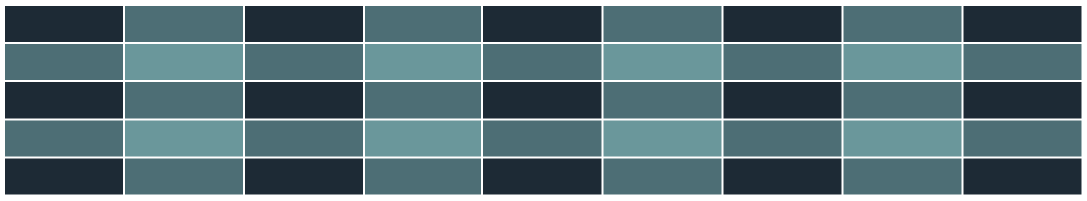


1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

# Note: Put the :nth-child() selector on both th and td elements if you want to have the styling on both headers and regular table cells.

• <u>Combine Vertical and Horizontal Zebra Stripes:-</u>

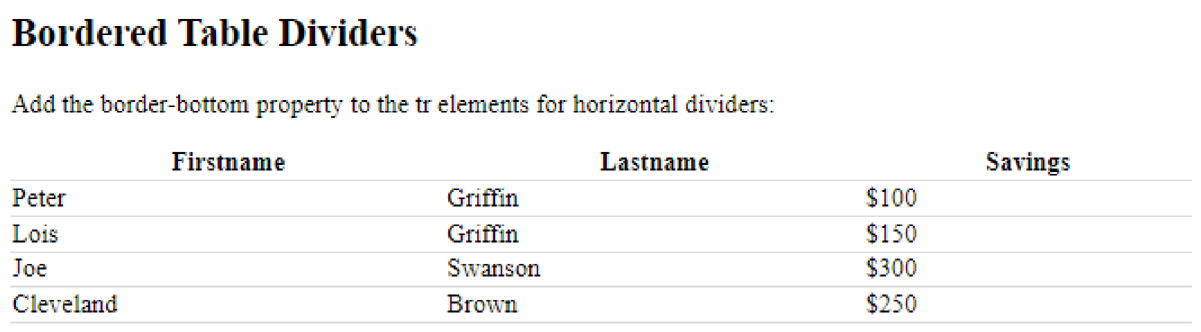
You can combine the styling from the two examples above and you will have stripes on every other row and every other column.
If you use a transparent color you will get an overlapping effect.



Use an rgba() color to specify the transparency of the color:
eg:

```
tr:nth-child(even) {
  background-color: rgba(150, 212, 212, 0.4);
}
th:nth-child(even),td:nth-child(even) {
  background-color: rgba(150, 212, 212, 0.4);}
```

• <u>Horizontal Dividers:-</u>

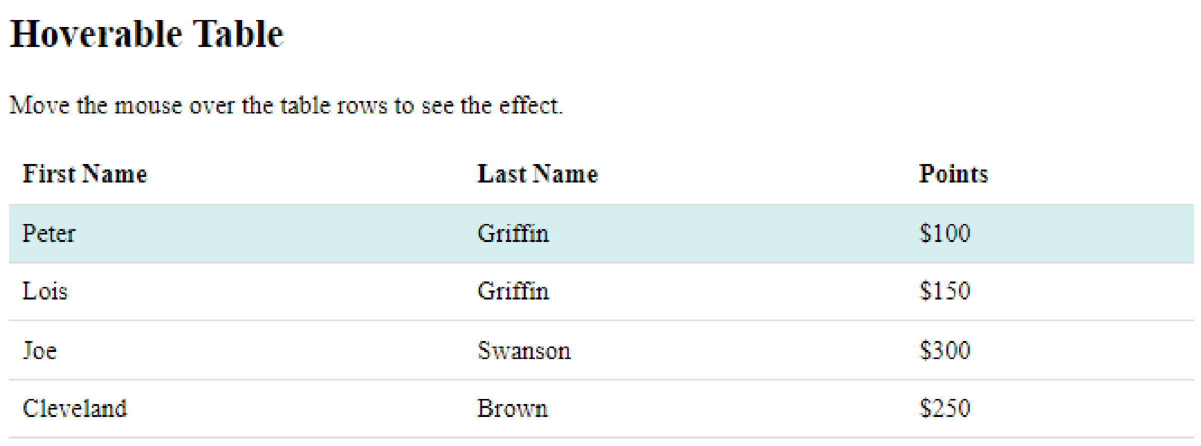
If you specify borders only at the bottom of each table row, you will have a table with horizontal dividers.
Add the border-bottom property to all tr elements to get horizontal dividers:
eg:
```
tr {
  border-bottom: 1px solid #ddd;
}
```


Bordered Table Dividers

Add the border-bottom property to the tr elements for horizontal dividers:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

- <u>Hoverable Table:-</u>

Use the :hover selector on tr to highlight table rows on mouse over:
eg:
```
tr:hover {background-color: #D6EEEE;}
```


Hoverable Table

Move the mouse over the table rows to see the effect.

First Name	Last Name	Points
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

- <u>Table Colgroup:</u>

The <colgroup> element is used to style specific columns of a table.
If you want to style the two first columns of a table, use the <colgroup> and <col> elements.
The <colgroup> element should be used as a container for the column specifications.
Each group is specified with a <col> element.
The span attribute specifies how many columns that get the style.
The style attribute specifies the style to give the columns.
eg:
```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Colgroup</h2>
<p>Add the a colgroup with a col element that spans over two columns to define a style for
the two columns:</p>
<table style="width: 100%;">
<colgroup>
  <col span="2" style="background-color: #D6EEEE">
</colgroup>
<tr>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
```

```
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</table>
</body>
</html>
```

# Note: The <colgroup> tag must be a child of a <table> element and should be placed before any other table elements, like <thead>, <tr>, <td> etc., but after the <caption> element, if present.

- Legal CSS:

There is only a very limited selection of CSS properties that are allowed to be used in the colgroup:

width property
visibility property
background properties
border properties

All other CSS properties will have no effect on your tables.

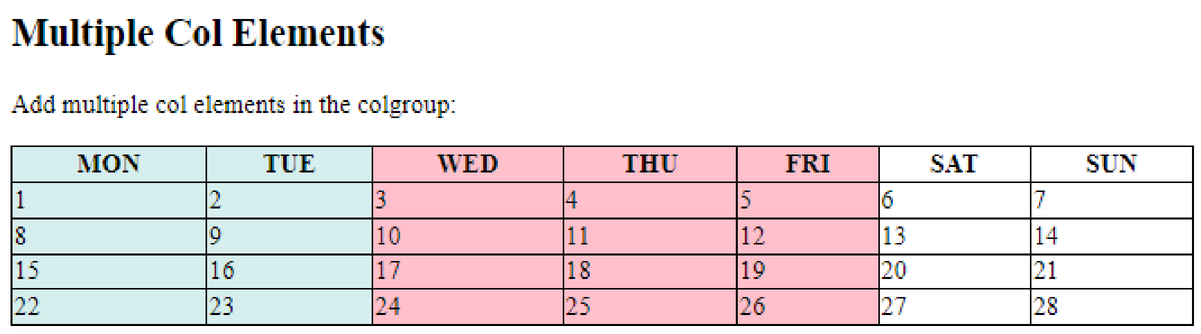
- Multiple Col Elements:

If you want to style more columns with different styles, use more <col> elements inside the <colgroup>:

eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Multiple Col Elements</h2>
<p>Add multiple col elements in the colgroup:</p>
<table style="width: 100%;">
  <colgroup>
    <col span="2" style="background-color: #D6EEEE">
    <col span="3" style="background-color: pink">
  </colgroup>
<tr>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</table>
</body>
```

```
</html>
```



Multiple Col Elements

Add multiple col elements in the colgroup:

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

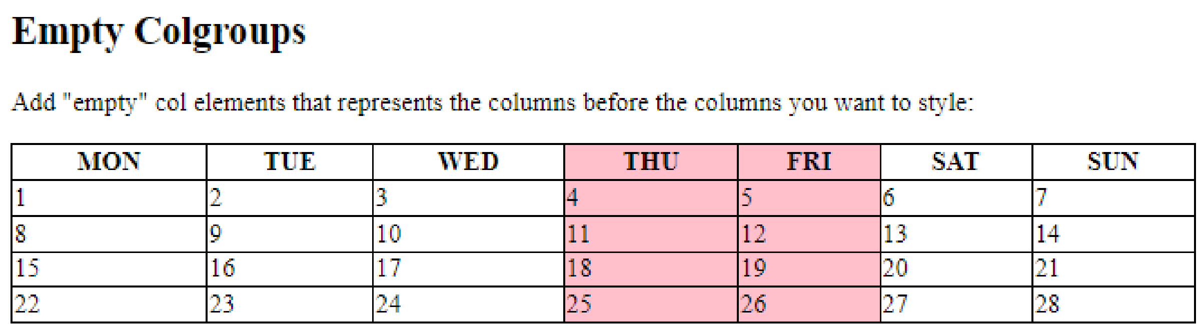
- Empty Colgroups:

If you want to style columns in the middle of a table, insert a "empty" <col> element (with no styles) for the columns before:

eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Empty Colgroups</h2>
<p>Add "empty" col elements that represents the columns before the columns you want to
style:</p>
<table style="width: 100%;">
<colgroup>
  <col span="3">
  <col span="2" style="background-color: pink">
</colgroup>
<tr>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
```

```
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</table>
</body>
</html>
```



Empty Colgroups

Add "empty" col elements that represents the columns before the columns you want to style:

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

- Hide columns:

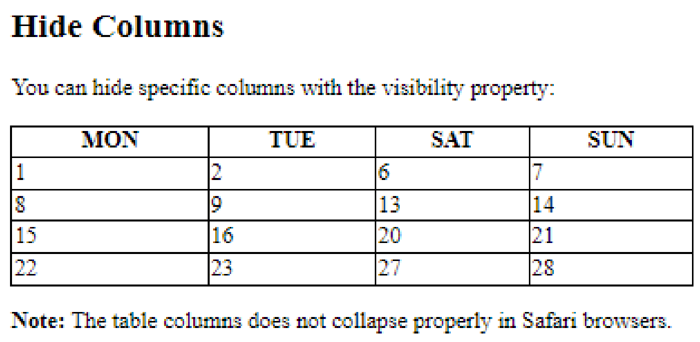
You can hide columns with the visibility: collapse property:

eg:

```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Hide Columns</h2>
<p>You can hide specific columns with the visibility property:</p>
<table style="width: 100%;">
<colgroup>
    <col span="2">
    <col span="3" style="visibility: collapse">
  </colgroup>
<tr>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
```

```
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</table>
<p><b>Note:</b> The table columns do not collapse properly in Safari browsers.</p>
</body>
</html>
```



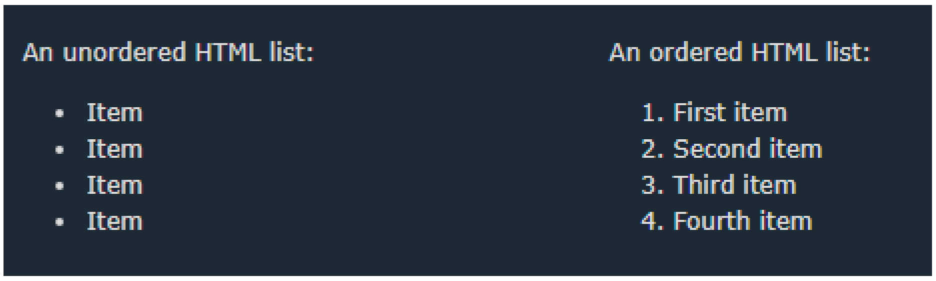
Hide Columns

You can hide specific columns with the visibility property:

MON	TUE	SAT	SUN
1	2	6	7
8	9	13	14
15	16	20	21
22	23	27	28

Note: The table columns does not collapse properly in Safari browsers.

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



An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

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

• <u>Unordered List:</u>
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:
eg:

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

• Ordered 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:
eg:

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

• Description LIst:

HTML also supports description lists.
A description list is a list of terms, with a description of each term.
The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:
eg:

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

Unordered HTML Lists:

The HTML <ul> tag defines an unordered (bulleted) 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:
eg:

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

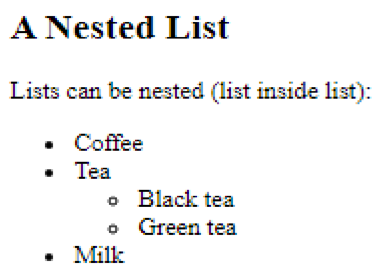
Unordered HTML List - Choose List Item Marker:-
The CSS "list-style-type: " property is used to define the style of the list item marker. It can have one of the following values:

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

eg:
 Disc:-

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

```
</ul>
```
Circle:-
```
<ul style="list-style-type:circle;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```
Square:-
```
<ul style="list-style-type:square;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```
None:-
```
<ul style="list-style-type:none;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```
Nested HTML LIsts:
Lists can be nested (list inside list):



A Nested List

Lists can be nested (list inside list):

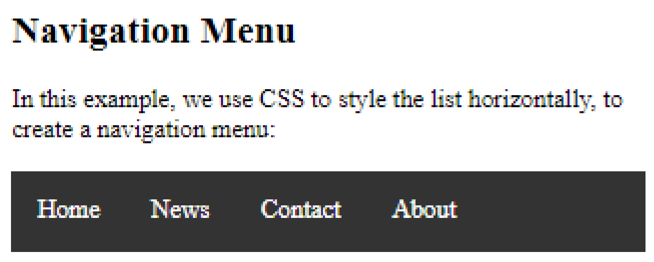
- Coffee
- Tea
  - Black tea
  - Green tea
- Milk

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```
# Note: A list item (<li>) can contain a new list, and other HTML elements, like images and links, etc.

Horizontal List with CSS:

HTML lists can be styled in many different ways with CSS.
One popular way is to style a list horizontally, to create a navigation menu:
eg:
```
<!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>
```


Navigation Menu

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

Ordered HTML Lists:
The HTML <ol> tag defines an ordered list. An ordered list can be numerical or alphabetical.
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:
eg:

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

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

eg:

Numbers:-
```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
UpperCase Letters:-
```
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
LowerCase Letters:-
```
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
UpperCase Roman Numbers:-
```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
LowerCase Roman Numbers:-
```
<ol type="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
Control List Counting:

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

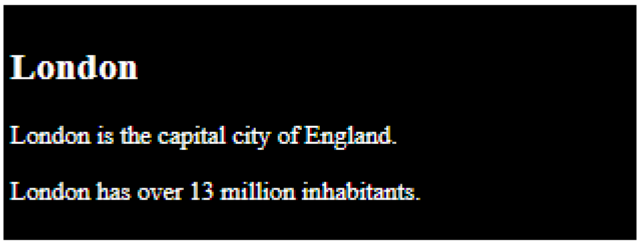
eg:

```
<ol start="50">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```
Nested HTML Lists:
Lists can be nested (list inside list):
eg:
```
<ol>
  <li>Coffee</li>
  <li>Tea
    <ol>
      <li>Black tea</li>
      <li>Green tea</li>
    </ol>
  </li>
  <li>Milk</li>
</ol>
```
# Note: A list item (<li>) can contain a new list, and other HTML elements, like images and links, etc.

HTML Block and Inline Elements
Every HTML element has a default display value, depending on what type of element it is.
The two most common display values are block and inline.
• Block level elements:-
A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.
A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
Two commonly used block elements are: <p> and <div>.
The <p> element defines a paragraph in an HTML document.
The <div> element defines a division or a section in an HTML document.
The <p> element is a block-level element.
The <div> element is a block-level element.
Here are the block-level elements in HTML:
<address>,<article>,<aside>,<blockquote>,<canvas>,<dd>,<div>,<dl>,<dt>,<fieldset>,<figcaption>,<figure>,<footer>,<form,><h1>-<h6>,<header>,<hr>,<li>,<main>,<nav>,<noscript>,<ol>,<p>,<pre>,<section>,<table>,<tfoot>,<ul>,<video>.
• Inline Elements:-
An inline element does not start on a new line.
An inline element only takes up as much width as necessary.
Here are the inline elements in HTML:
<a>,<abbr>,<acronym>,<b>,<bdo>,<big>,<br>,<button>,<cite>,<code>,<dfn>,<em>,<i>,<img>,
<input>,<kbd>,<label>,<map>,<object>,<output>,<q>,<samp>,<script>,<select>,<small>,<span>,<strong>,<sub>,<sup>,<textarea>,<time>,<tt>,<var>.
#Note: An inline element cannot contain a block-level element!.
The Div Element:-

The <div> element is often used as a container for other HTML elements.
The <div> element has no required attributes, but style, class and id are common.
When used together with CSS, the <div> element can be used to style blocks of content.
• Div as a Container:
The <div> element is often used to group sections of a web page together.
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
div{
padding:5px;
background-color:black;
color:white;
}
</style>
</head>
<body>
<div >
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 13 million inhabitants.</p>
</div>
</body>
</html>
```



```
London

London is the capital city of England.

London has over 13 million inhabitants.
```

<u>• Center align a Div Element:</u>
If you have a <div> element that is not 100% wide, and you want to center-align it, set the CSS margin property to auto.
eg:

```
<!DOCTYPE html>
<html>
<style>
div {
  width: 300px;
  margin: auto;
  background-color: black;
  color:white;
  padding:5px;
}
</style>
<body>
```

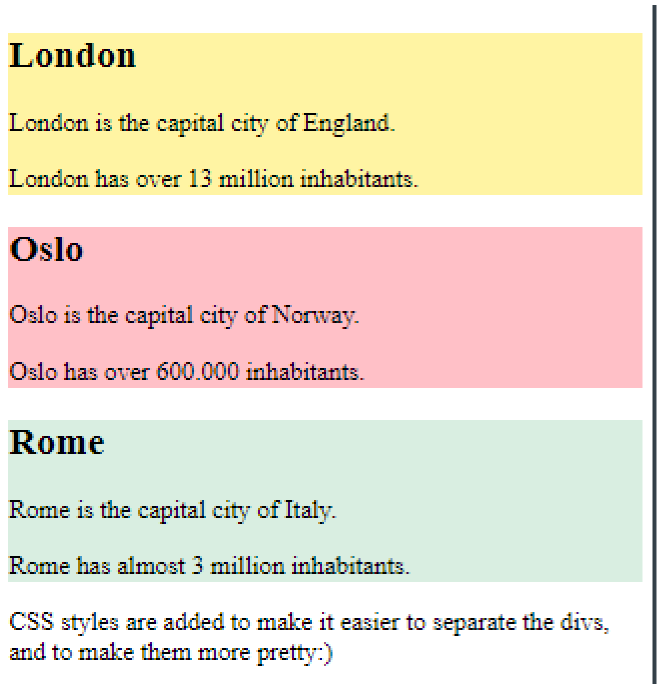
```
<div>
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 13 million inhabitants.</p>
</div>
</body>
</html>
```

• <u>Multiple Div Elements:</u>

You can have many <div> containers on the same page.

eg:

```
<!DOCTYPE html>
<html>
<body>
<div style="background-color:#FFF4A3;">
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 13 million inhabitants.</p>
</div>
<div style="background-color:#FFC0C7;">
  <h2>Oslo</h2>
  <p>Oslo is the capital city of Norway.</p>
  <p>Oslo has over 600.000 inhabitants.</p>
</div>
<div style="background-color:#D9EEE1;">
  <h2>Rome</h2>
  <p>Rome is the capital city of Italy.</p>
  <p>Rome has almost 3 million inhabitants.</p>
</div>
<p>CSS styles are added to make it easier to separate the divs, and to make them more
pretty:)</p>
</body>
</html>
```


London

London is the capital city of England.

London has over 13 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

CSS styles are added to make it easier to separate the divs, and to make them more pretty:)

• Aligining Div Elements side by Side:

When building web pages, you often want to have two or more <div> elements side by side.
There are different methods for aligning elements side by side, all include some CSS styling.
We will look at the most common methods:

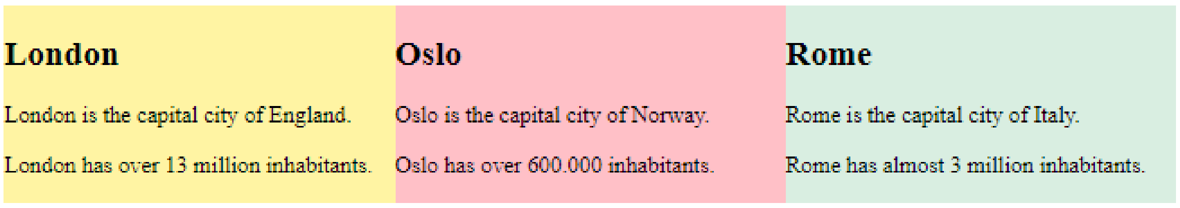
Float:

The CSS float property was not originally meant to align <div> elements side-by-side, but has been used for this purpose for many years.
The CSS float property is used for positioning and formatting content and allow elements float next to each other instead of on top of each other.

eg:

```
<!DOCTYPE html>
<html>
<style>
div.mycontainer {
  width:100%;
  overflow:auto;
}
div.mycontainer div {
  width:33%;
  float:left;
}
</style>
<body>
<div class="mycontainer">
  <div style="background-color:#FFF4A3;">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
    <p>London has over 13 million inhabitants.</p>
  </div>
  <div style="background-color:#FFC0C7;">
```

```
    <h2>Oslo</h2>
    <p>Oslo is the capital city of Norway.</p>
    <p>Oslo has over 600.000 inhabitants.</p>
  </div>
  <div style="background-color:#D9EEE1;">
    <h2>Rome</h2>
    <p>Rome is the capital city of Italy.</p>
    <p>Rome has almost 3 million inhabitants.</p>
  </div>
</div>
</body>
</html>
```



London

London is the capital city of England.

London has over 13 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

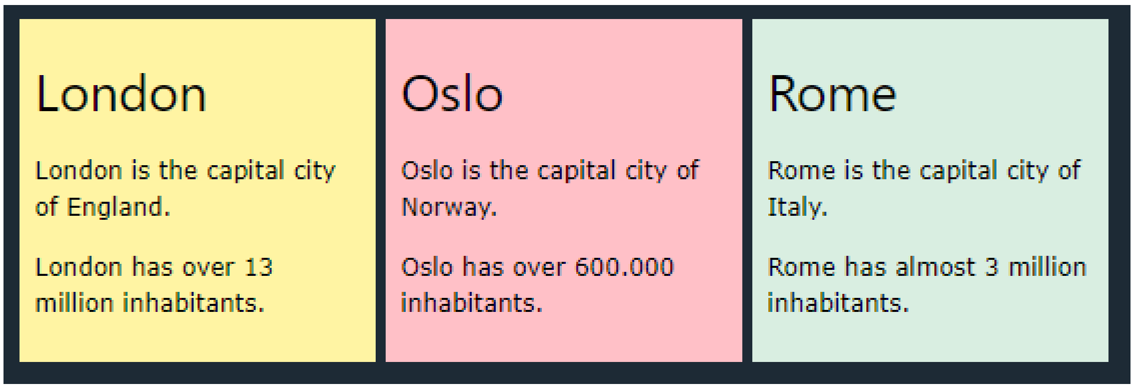
Inline-Block:

If you change the <div> element's display property from block to inline-block, the <div> elements will no longer add a line break before and after, and will be displayed side by side instead of on top of each other.

eg:

```
<!DOCTYPE html>
<html>
<style>
div {
  width:30%;
  display:inline-block;
}
</style>
<body>
<div style="background-color:#FFF4A3;">
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 13 million inhabitants.</p>
</div>
<div style="background-color:#FFC0C7;">
  <h2>Oslo</h2>
  <p>Oslo is the capital city of Norway.</p>
  <p>Oslo has over 600.000 inhabitants.</p>
</div>
<div style="background-color:#D9EEE1;">
  <h2>Rome</h2>
  <p>Rome is the capital city of Italy.</p>
  <p>Rome has almost 3 million inhabitants.</p>
</div>
```

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



London

London is the capital city of England.

London has over 13 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

Flex:

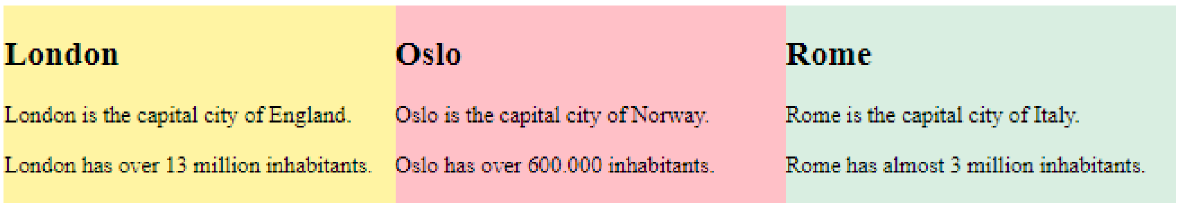
The CSS Flexbox Layout Module was introduced to make it easier to design flexible responsive layout structure without using float or positioning.

To make the CSS flex method work, surround the <div> elements with another <div> element and give it the status as a flex container.

eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
.mycontainer {
  display: flex;
}
.mycontainer > div {
  width:33%;
}
</style>
</head>
<body>
<div class="mycontainer">
  <div style="background-color:#FFF4A3;">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
    <p>London has over 13 million inhabitants.</p>
  </div>
  <div style="background-color:#FFC0C7;">
    <h2>Oslo</h2>
    <p>Oslo is the capital city of Norway.</p>
    <p>Oslo has over 600.000 inhabitants.</p>
  </div>
  <div style="background-color:#D9EEE1;">
    <h2>Rome</h2>
    <p>Rome is the capital city of Italy.</p>
    <p>Rome has almost 3 million.</p>
  </div>
```

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


London

London is the capital city of England.

London has over 13 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

Rome

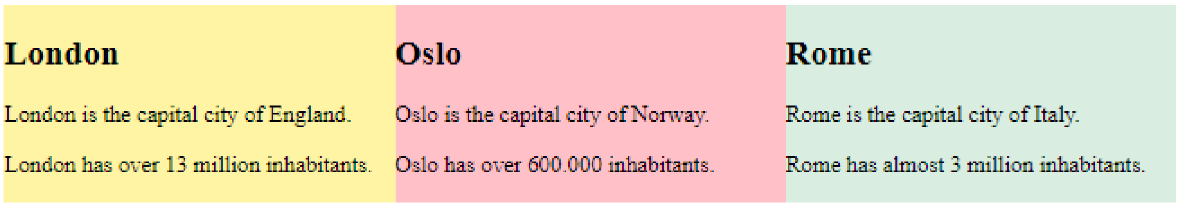
Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

Grid:

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.
Sounds almost the same as flex, but has the ability to define more than one row and position each row individually.
The CSS grid method requires that you surround the <div> elements with another <div> element and give the status as a grid container, and you must specify the width of each column.
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: 33% 33% 33%;
}
</style>
</head>
<body>
<div class="grid-container">
<div style="background-color:#FFF4A3;">
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 13 million inhabitants.</p>
</div>
<div style="background-color:#FFC0C7;">
  <h2>Oslo</h2>
  <p>Oslo is the capital city of Norway.</p>
  <p>Oslo has over 600.000 inhabitants.</p>
</div>
<div style="background-color:#D9EEE1;">
  <h2>Rome</h2>
  <p>Rome is the capital city of Italy.</p>
  <p>Rome has almost 3 million inhabitants.</p>
</div>
</div>
</body>
```



London

London is the capital city of England.

London has over 13 million inhabitants.

Oslo

Oslo is the capital city of Norway.

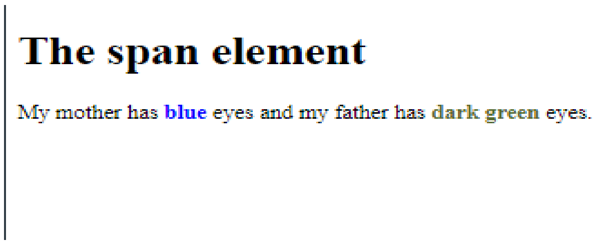
Oslo has over 600.000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

The Span Element:-
The <span> element is an inline container used to mark up a part of a text, or a part of a document.
The <span> element has no required attributes, but style, class and id are common.
When used together with CSS, the <span> element can be used to style parts of the text
eg:
<!DOCTYPE html>
<html>
<body>
<h1>The span element</h1>
<p>My mother has <span style="color:blue;font-weight:bold">blue</span> eyes and my father has <span style="color:darkolivegreen;font-weight:bold">dark green</span> eyes.</p>
</body>
</html>



The span element

My mother has **blue** eyes and my father has **dark green** eyes.

HTML Class Attribute
The HTML class attribute is used to specify a class for an HTML element.
Multiple HTML elements can share the same class.
The class attribute is often used to point to a class name in a style sheet. It can also be used by a Java Script to access and manipulate elements with the specific class name.
In the following example we have three <div> elements with a class attribute with the value of "city". All of the three <div> elements will be styled equally according to the .city style definition in the head section:
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
  background-color: tomato;
  color: white;
  border: 2px solid black;
  margin: 20px;
  padding: 20px;
}
</style>
</head>
<body>
<div class="city">
  <h2>London</h2>
  <p>London is the capital of England.</p>
</div>
<div class="city">
  <h2>Paris</h2>
  <p>Paris is the capital of France.</p>
</div>
<div class="city">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</div>
</body>
</html>
```
In the following example we have two <span> elements with a class attribute with the value of "note". Both <span> elements will be styled equally according to the .note style definition in the head section:

eg:
```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>
<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>
</body>
</html>
```
#Tip: The class attribute can be used on any HTML element.

#Note: The class name is case sensitive!

<u>The Syntax for Class:-</u>

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

eg:
```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>
</head>
<body>
<h2 class="city">London</h2>
<p>London is the capital of England.</p>
<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>
<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
</body>
</html>
```

Multiple Classes:

HTML elements can belong to more than one class.
To define multiple classes, separate the class names with a space, e.g. <div class="city main">. The element will be styled according to all the classes specified.
In the following example, the first <h2> element belongs to both the city class and also to the main class, and will get the CSS styles from both of the classes:
eg:

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

Different Elements Can Share Same Class:

Different HTML elements can point to the same class name.

In the following example, both <h2> and <p> point to the "city" class and will share the same style:
eg:
```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>
</head>
<body>
<h2>Different Elements Can Share Same Class</h2>
<p>Even if the two elements do not have the same tag name, they can both point to the
same class, and get the same CSS styling:</p>
<h2 class="city">Paris</h2>
<p class="city">Paris is the capital of France.</p>
</body>
</html>
```
<u>Use of The class Attribute in JavaScript</u>
The class name can also be used by JavaScript to perform certain tasks for specific elements.
JavaScript can access elements with a specific class name with the getElementsByClassName() method:
eg:
```
<!DOCTYPE html>
<html>
<body>
<h2>Use of The class Attribute in JavaScript</h2>
<p>Click the button to hide all elements with class name "city":</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>
```

HTML ID Attribute

The HTML id attribute is used to specify a unique id for an HTML element.
You cannot have more than one element with the same id in an HTML document.

<u>Using the ID Attribute:</u>

The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.
The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.
The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces { }.
In the following example we have an <h1> element that points to the id name "myHeader". This <h1> element will be styled according to the #myHeader style definition in the head section:
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>
</head>
<body>
<h2>The id Attribute</h2>
<p>Use CSS to style an element with the id "myHeader":</p>
<h1 id="myHeader">My Header</h1>
</body>
</html>
```

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

<u>Difference Between Class and ID:</u>

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

```
<!DOCTYPE html>
<html>
<head>
<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>
</head>
<body>
<h2>Difference Between Class and ID</h2>
<p>A class name can be used by multiple HTML elements, while an id name must only be
used by one HTML element within the page:</p>
<!-- An element with a unique id -->
<h1 id="myHeader">My Cities</h1>
<!-- Multiple elements with same class -->
<h2 class="city">London</h2>
<p>London is the capital of England.</p>
<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>
<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
</body>
</html>
```

<u>HTML Bookmarks with ID and Links:</u>

HTML bookmarks are used to allow readers to jump to specific parts of a webpage.
Bookmarks can be useful if your page is very long.
To use a bookmark, you must first create it, and then add a link to it.
Then, when the link is clicked, the page will scroll to the location with the bookmark.
eg:

```
<!DOCTYPE html>
<html>
<body>
<p><a href="#C4">Jump to Chapter 4</a></p>
<p><a href="#C10">Jump to Chapter 10</a></p>
<h2>Chapter 1</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 2</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 3</h2>
<p>This chapter explains ba bla bla</p>
<h2 id="C4">Chapter 4</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 5</h2>
<p>This chapter explains ba bla bla</p>
```

```
<h2>Chapter 6</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 7</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 8</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 9</h2>
<p>This chapter explains ba bla bla</p>
<h2 id="C10">Chapter 10</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 11</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 12</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 13</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 14</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 15</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 16</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 17</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 18</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 19</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 20</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 21</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 22</h2>
<p>This chapter explains ba bla bla</p>
<h2>Chapter 23</h2>
<p>This chapter explains ba bla bla</p>
</body>
</html>
```

Using The id Attribute in JavaScript:

The id attribute can also be used by JavaScript to perform some tasks for that specific element.

JavaScript can access an element with a specific id with the getElementById() method:

eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>Using The id Attribute in JavaScript</h2>
```

```
<p>JavaScript can access an element with a specified id by using the getElementById()
method:</p>
<h1 id="myHeader">Hello World!</h1>
<button onclick="displayResult()">Change text</button>
<script>
function displayResult() {
  document.getElementById("myHeader").innerHTML = "Have a nice day!";
}
</script>
</body>
</html>
```

HTML Iframes
An HTML iframe is used to display a web page within a web page.
<u>Syntax:-</u> `<iframe src="url" title="description"></iframe>`
The HTML <iframe> tag specifies an inline frame.
An inline frame is used to embed another document within the current HTML document.
#Tip: It is a good practice to always include a title attribute for the <iframe>. This is used by screen readers to read out what the content of the iframe is.
<u>Set Height and Width:</u>
Use the height and width attributes to specify the size of the iframe and we can add style also.
The height and width are specified in pixels by default:
eg:

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

```
<iframe src="demo_iframe.htm" style="border:none;" title="Iframe Example"></iframe>
```

We can also style it

```
<iframe src="demo_iframe.htm" style="border:2px solid red;" title="Iframe
Example"></iframe>
```

<u>Target for a Link:</u>
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:
eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>Iframe - Target for a Link</h2>
<iframe src="demo_iframe.htm" name="iframe_a" height="300px" width="100%"
title="Iframe Example"></iframe>
<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>
<p>When the target attribute of a link matches the name of an iframe, the link will open in
the iframe.</p>
</body>
```

HTML Java Script
JavaScript makes HTML pages more dynamic and interactive.
eg:
```
<!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>
```
The HTML <script> Tag:
The HTML <script> tag is used to define a client-side script (JavaScript).
The <script> element either contains script statements, or it points to an external script file through the src attribute.
Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
To select an HTML element, JavaScript most often uses the document.getElementById() method.
This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":
eg:
```
<!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>
```
<u>What Java Script can DO ? ?...:</u>
<u>• Java Script can change Content:</u>
```
<!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 styles:

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

• JavaScript can change attributes:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<p>Here, JavaScript changes the value of the src (source) attribute of an image.</p>
<script>
function light(sw) {
  var pic;
  if (sw == 0) {
    pic = "pic_bulboff.gif"
  } else {
    pic = "pic_bulbon.gif"
  }
  document.getElementById('myImage').src = pic;
}
</script>
<img id="myImage" src="pic_bulboff.gif" width="100" height="180">
<p>
<button type="button" onclick="light(1)">Light On</button>
<button type="button" onclick="light(0)">Light Off</button>
</p>
</body>
</html>
```

HTML <noscript> Tag:

The HTML <noscript> tag defines an alternate content to be displayed to users that have disabled scripts in their browser or have a browser that doesn't support scripts:

eg:

```
<!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>
```

HTML File Paths
A file path describes the location of a file in a web site's folder structure.

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page.
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current folder.
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web.
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder.

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
<u>Absolute File Paths:</u>
An absolute file path is the full URL to a file:
eg:
<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">
<u>Relative File Paths:</u>
A relative file path points to a file relative to the current page.
In the following example, the file path points to a file in the images folder located at the root of the current web:
<img src="/images/picture.jpg" alt="Mountain">
In the following example, the file path points to a file in the images folder located in the current folder:

<img src="images/picture.jpg" alt="Mountain">
In the following example, the file path points to a file in the images folder located in the folder one level up from the current folder:
<img src="../images/picture.jpg" alt="Mountain">
Best Practice:
It is best practice to use relative file paths (if possible).
When using relative file paths, your web pages will not be bound to your current base URL.
All links will work on your own computer (localhost) as well as on your current public domain and your future public domains.

HTML Head Element
The HTML <head> element is a container for the following elements: <title>, <style>, <meta>, <link>, <script>, and <base>.

• 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 defines the document title, character set, styles, scripts, and other meta information.
• Title Element:
The <title> element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.
The <title> element is required in HTML documents!
The content of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.
The <title> element:
defines a title in the browser toolbar
provides a title for the page when it is added to favorites
displays a title for the page in search engine-results
So, try to make the title as accurate and meaningful as possible!
• Style Element:
The <style> element is used to define style information for a single HTML page.
eg:
```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```
• Link Element:
The <link> element defines the relationship between the current document and an external resource.
The <link> tag is most often used to link to external style sheets.
eg:

```
<link rel="stylesheet" href="mystyle.css">
```
• Meta Element:
The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.
The metadata will not be displayed on the page, but is used by browsers (how to display content or reload page), by search engines (keywords), and other web services.
eg's:
~ Define the character set used:
```
<meta charset="UTF-8">
```
~ Define keywords for search engines:
```
<meta name="keywords" content="HTML, CSS, JavaScript">
```
~ Define a description of your web page:
```
<meta name="description" content="Free Web tutorials">
```
~ Define the author of a page:
```
<meta name="author" content="John Doe">
```
~ Refresh document every 30 seconds:
```
<meta http-equiv="refresh" content="30">
```
~ Setting the viewport to make your website look good on all devices:
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```
eg:
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <meta name="author" content="John Doe">
</head>
<body>
<p>All meta information goes inside the head section.</p>
</body>
</html>
```
• Setting The ViewPort:
The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.
You should include the following <meta> element in all your web pages:
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```
This gives the browser instructions on how to control the page's dimensions and scaling.
The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.
• Script Element:
The <script> element is used to define client-side JavaScripts.
The following JavaScript writes "Hello JavaScript!" into an HTML element with id="demo":
eg:
```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>Page Title</title>
  <script>
  function myFunction() {
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  }
  </script>
</head>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

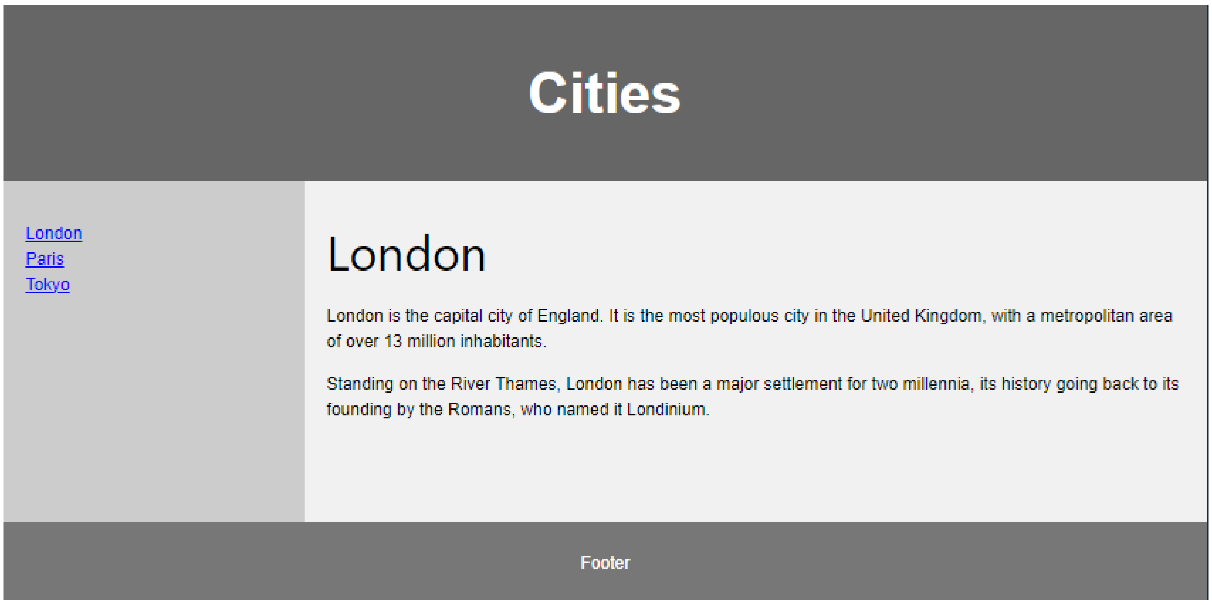
• Base Element:

The <base> element specifies the base URL and/or target for all relative URLs in a page.
The <base> tag must have either an href or a target attribute present, or both.
There can only be one single <base> element in a document!
eg:

```
<!DOCTYPE html>
<html>
<head>
  <base href="https://www.w3schools.com/" target="_blank">
</head>
<body>
<h1>The base element</h1>
<p><img src="images/stickman.gif" width="24" height="39" alt="Stickman"> - Notice that we
have only specified a relative address for the image. Since we have specified a base URL in
the head section, the browser will look for the image at
"https://www.w3schools.com/images/stickman.gif".</p>
<p><a href="tags/tag_base.asp">HTML base tag</a> - Notice that the link opens in a new
window, even if it has no target="_blank" attribute. This is because the target attribute of the
base element is set to "_blank".</p>
</body>
</html>
```

HTML Layouts & Techniques

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



Cities

London
Paris
Tokyo

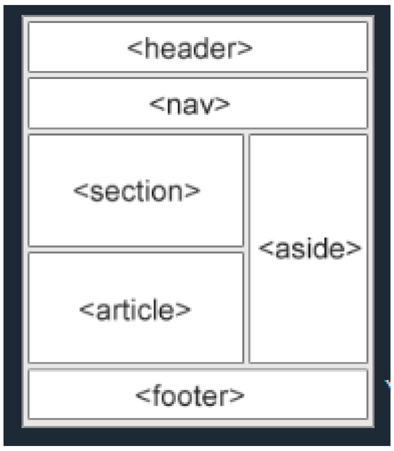
London

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

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

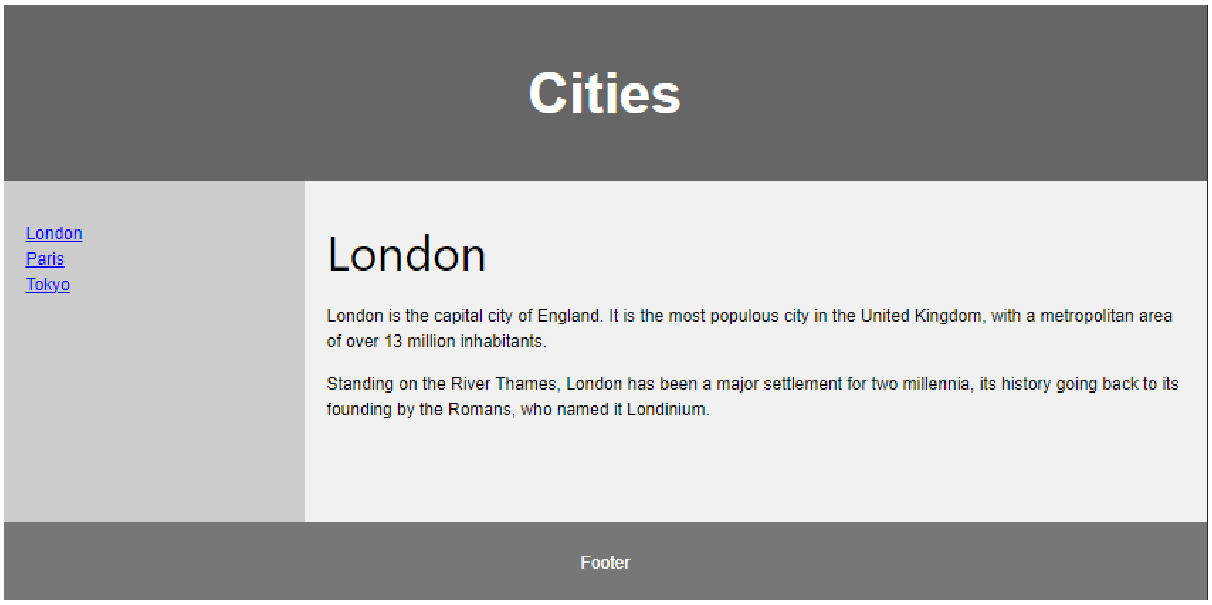
Footer

<u>Layout Elements:</u>
HTML has several semantic elements that define the different parts of a web page:



HTML5 Semantic Elements
<header> - Defines a header for a document or a section
<nav> - Defines a set of navigation links
<section> - Defines a section in a document
<article> - Defines an independent, self-contained content
<aside> - Defines content aside from the content (like a sidebar)
<footer> - Defines a footer for a document or a section
<details> - Defines additional details that the user can open and close on demand
<summary> - Defines a heading for the <details> element
<u>Layout Techniques:</u>
There are four different techniques to create multicolumn layouts. Each technique has its pros and cons:
CSS framework.
CSS float property.
CSS flexbox.
CSS grid.
<u>• CSS FrameWork:</u>

If you want to create your layout fast, you can use a CSS framework, like W3.CSS or Bootstrap.
• CSS Float Layout(Float property):
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. {We can learn more about this in CSS float and clear chapter}


Cities

London
Paris
Tokyo

London

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

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

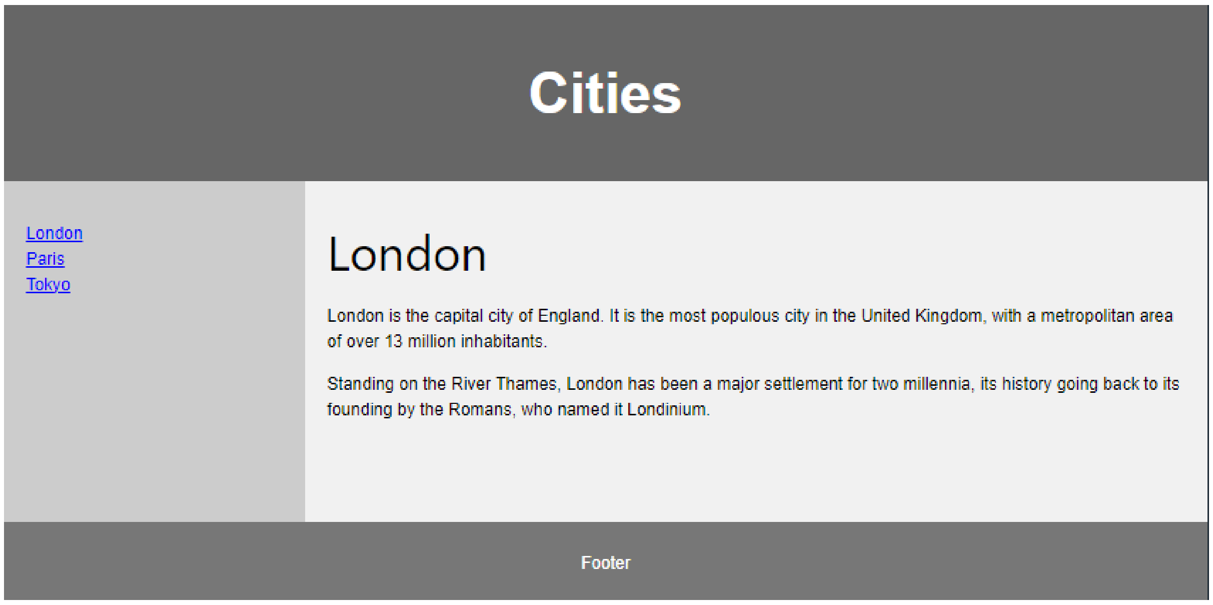
Footer

eg:
```
<!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>
```

• CSS Flexbox Layout:
Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.



Cities

London
Paris
Tokyo

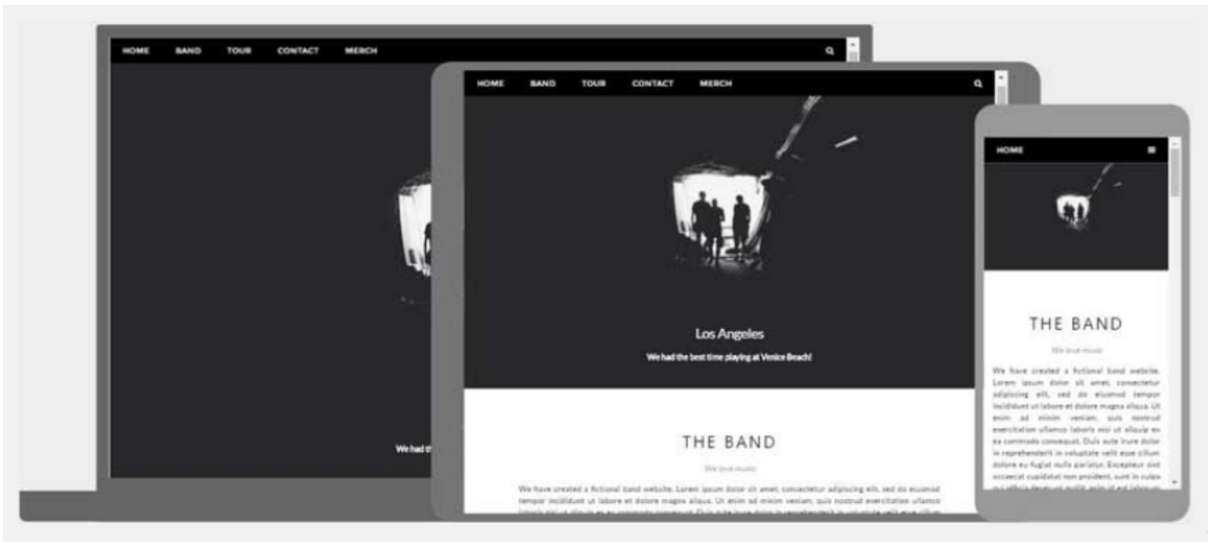
London

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

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

Footer

• CSS Grid Layout:
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

HTML Responsive Web Design
Responsive web design is about creating web pages that look good on all devices!
A responsive web design will automatically adjust for different screen sizes and viewports.


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):
<u>• Setting The ViewPort:</u>
To create a responsive website, add the following <meta> tag to all your web pages:
<meta name="viewport" content="width=device-width, initial-scale=1.0">
This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.
<u>• Responsive Images:</u>
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
<img src="img_girl.jpg" style="width:100%;">
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:
<img src="img_girl.jpg" style="max-width:100%;height:auto;">
<u>• Show Different Images Depending on Browser Width</u>
The HTML <picture> element allows you to define different images for different browser window sizes.
eg:
```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
```

```
<h2>Show Different Images Depending on Browser Width</h2>
<p>Resize the browser width and the image will change at 600px and 1500px.</p>
<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_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
</body>
</html>
```

• 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:

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

eg:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h1 style="font-size:10vw;">Responsive Text</h1>
<p style="font-size:5vw;">Resize the browser window to see how the text size scales.</p>
<p style="font-size:5vw;">Use the "vw" unit when sizing the text. 10vw will set the size to 10% of the viewport width.</p>
<p>Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.</p>
</body>
</html>
```

#Note: 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.
eg:(Try it in mobile and PC, See the difference)

```
<!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: #04AA6D;
  padding: 20px;
  float: left;
  width: 20%; /* The width is 20%, by default */
}
/* Use a media query to add a breakpoint 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>
```

• Responsive WebPage Full 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>
```

HTML Computer Code Elements
HTML contains several elements for defining user input and computer code.
• HTML <kbd> For Keyboard Input:
The HTML <kbd> element is used to define keyboard input. The content inside is displayed in the browser's default monospace font.
eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>The kbd Element</h2>
<p>The kbd element is used to define keyboard input:</p>
<p>Save the document by pressing <kbd>Ctrl + S</kbd></p>
</body>
</html>
```

• HTML <samp> For Program Output:
The HTML <samp> element is used to define sample output from a computer program. The content inside is displayed in the browser's default monospace font.
eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>The samp Element</h2>
<p>The samp element is used to define sample output from a computer program.</p>
<p>Message from my computer:</p>
<p><samp>File not found.<br>Press F1 to continue</samp></p>
</body>
</html>
```

• HTML <code> For Computer Code:
The HTML <code> element is used to define a piece of computer code. The content inside is displayed in the browser's default monospace font.
eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>Computer Code</h2>
<p>Some programming code:</p>
<code>
```

```
class Hello <br/>
{<br/>
public static void main(String args[])<br/>
{<br/>
  System.out.print("I Rule The World...");<br/>
}<br/>
}<br/>
</code>
</body>
</html>
```

#Note: that the <code> element does not preserve extra whitespace and line-breaks.
To fix this, you can put the <code> element inside a <pre> element or keep <br> to each line.

• HTML <var> For Variables:

The HTML <var> element is used to define a variable in programming or in a mathematical expression. The content inside is typically displayed in italic.
eg:

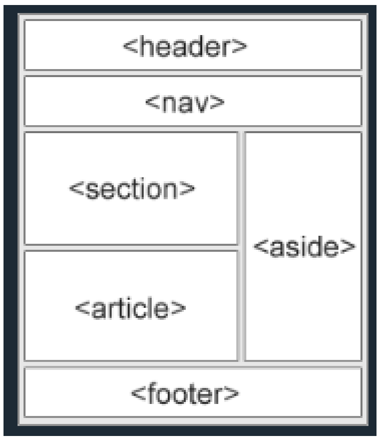
```
<!DOCTYPE html>
<html>
<body>
<h2>The var Element</h2>
<p>The area of a triangle is: 1/2 x <var>b</var> x <var>h</var>, where <var>b</var> is the base, and <var>h</var> is the vertical height.</p>
</body>
</html>
```

HTML Semantic Elements

Semantic elements = elements with a meaning.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.
Examples of non-semantic elements: <div> and <span> - Tells nothing about its content.
Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.
Semantic Elements in HTML
Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.
In HTML there are some semantic elements that can be used to define different parts of a web page:



<article>
<aside>
<details>
<figcaption>
<figure>
<footer>
<header>
<main>
<mark>
<nav>
<section>
<summary>
<time>

• HTML <section> Element:

The <section> element defines a section in a document.
According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."
Examples of where a <section> element can be used:
Chapters
Introduction
News items
Contact information
A web page could normally be split into sections for introduction, content, and contact information.
eg:

```
<!DOCTYPE html>
<html>
<body>
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>
<section>
  <h1>WWF's Panda symbol</h1>
  <p>The Panda has become the symbol of WWF. The well-known panda logo of WWF
originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the
London Zoo in the same year of the establishment of WWF.</p>
</section>
</body>
</html>
```

• HTML <article> Element:

The <article> element specifies independent, self-contained content.
An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.
Examples of where the <article> element can be used:
Forum posts
Blog posts
User comments
Product cards
Newspaper articles
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
.all-browsers {
  margin: 0;
  padding: 5px;
  background-color: lightgray;
}
.all-browsers > h1, .browser {
  margin: 10px;
  padding: 5px;
}
.browser {
  background: white;
}
.browser > h2, p {
  margin: 4px;
  font-size: 90%;
}
</style>
</head>
<body>
<article class="all-browsers">
  <h1>Most Popular Browsers</h1>
  <article class="browser">
    <h2>Google Chrome</h2>
    <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is
the world's most popular web browser today!</p>
  </article>
  <article class="browser">
```

```
    <h2>Mozilla Firefox</h2>
    <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has
been the second most popular web browser since January, 2018.</p>
  </article>
  <article class="browser">
    <h2>Microsoft Edge</h2>
    <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft
Edge replaced Internet Explorer.</p>
  </article>
</article>
</body>
</html>
```

• HTML Header Element:

The <header> element represents a container for introductory content or a set of navigational links.
A <header> element typically contains:
one or more heading elements (<h1> - <h6>)
logo or icon
authorship information
# Note: You can have several <header> elements in one HTML document. However, <header> cannot be placed within a <footer>, <address> or another <header> element.
eg:

```
<!DOCTYPE html>
<html>
<body>
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment, and build
a future in which humans live in harmony with nature.</p>
</article>
</body>
</html>
```

• HTML Footer Element:

The <footer> element defines a footer for a document or section.
A <footer> element typically contains:
authorship information
copyright information
contact information
sitemap
back to top links
related documents
# Note: You can have several <footer> elements in one document.
eg:

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

```
<body>
<footer>
  <p>Author: Hege Refsnes</p>
  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
</body>
</html>
```

• <u>HTML Nav Element:</u>

The <nav> element defines a set of navigation links.

# Note: That NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major blocks of navigation links.

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

eg:

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

• <u>HTML Aside Element:</u>

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The <aside> content should be indirectly related to the surrounding content.

eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
aside {
  width: 30%;
  padding-left: 15px;
  margin-left: 15px;
  float: right;
  font-style: italic;
  background-color: lightgray;
}
</style>
</head>
<body>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot
was amazing! I had a great summer together with my family!</p>
<aside>
```

```
<p>The Epcot center is a theme park at Walt Disney World Resort featuring exciting
attractions, international pavilions, award-winning fireworks and seasonal special
events.</p>
</aside>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot
was amazing! I had a great summer together with my family!</p>
<p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot
was amazing! I had a great summer together with my family!</p>
</body>
</html>
```

• <u>HTML Figure and Figcaption Elements:</u>

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
The <figcaption> tag defines a caption for a <figure> element. The <figcaption> element can be placed as the first or as the last child of a <figure> element.
The <img> element defines the actual image/illustration.
eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>Places to Visit</h2>
<p>Puglia's most famous sight is the unique conical houses (Trulli) found in the area around
Alberobello, a declared UNESCO World Heritage Site.</p>
<figure>
  <img src="pic_trulli.jpg" alt="Trulli" style="width:100%">
  <figcaption>Fig.1 - Trulli, Puglia, Italy.</figcaption>
</figure>
</body>
</html>
```

• <u>HTML Details Tag:</u>

The <details> tag specifies additional details that the user can open and close on demand.
The <details> tag is often used to create an interactive widget that the user can open and close. By default, the widget is closed. When open, it expands, and displays the content within.
Any sort of content can be put inside the <details> tag.
Tip: The <summary> tag is used in conjunction with <details> to specify a visible heading for the details.
Attribute can apply:

open	open	Specifies that the details should be visible (open) to the user

eg:

```
<html>
<style>
details > summary {
  padding: 4px;
  width: 200px;
  background-color: #eeeeee;
  border: none;
  box-shadow: 1px 1px 2px #bbbbbb;
  cursor: pointer;
}
details > p {
  background-color: #eeeeee;
  padding: 4px;
  margin: 0;
  box-shadow: 1px 1px 2px #bbbbbb;
}
</style>
<body>
<details>
  <summary>Epcot Center</summary>
  <p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and seasonal special events.</p>
</details>
</body>
</html>
```

• HTML Main Tag:

The <main> tag specifies the main content of a document.
The content inside the <main> element should be unique to the document. It should not contain any content that is repeated across documents such as sidebars, navigation links, copyright information, site logos, and search forms.
# Note: There must not be more than one <main> element in a document. The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.
eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
main {
  margin: 0;
  padding: 5px;
  background-color: lightgray;
}
main > h1, p, .browser {
  margin: 10px;
  padding: 5px;
}
.browser {
  background: white;
}
.browser > h2, p {
  margin: 4px;
  font-size: 90%;
}
</style>
</head>
```

```
<body>
<h1>The main element - Styled with CSS</h1>
<main>
  <h1>Most Popular Browsers</h1>
  <p>Chrome, Firefox, and Edge are the most used browsers today.</p>
  <article class="browser">
    <h2>Google Chrome</h2>
    <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is
the world's most popular web browser today!</p>
  </article>
  <article class="browser">
    <h2>Mozilla Firefox</h2>
    <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has
been the second most popular web browser since January, 2018.</p>
  </article>
  <article class="browser">
    <h2>Microsoft Edge</h2>
    <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft
Edge replace Internet Explorer.</p>
  </article>
</main>
</body>
</html>
```

• HTML Mark Tag:

The <mark> tag defines text that should be marked or highlighted.

eg:

```
<!DOCTYPE html>
<html>
<head>
<style>
mark {
  background-color: yellow;
  color: black;
}
</style>
</head>
<body>
<p>A mark element is displayed like this:</p>
<mark>Highlighted text!!</mark>
<p>Change the default CSS settings to see the effect.</p>
</body>
</html>
```

• HTML Summary Tag:

The <summary> tag defines a visible heading for the <details> element. The heading can be clicked to view/hide the details.

# Note: The <summary> element should be the first child element of the <details> element.

eg:

```
<html>
```

```
<style>
details > summary {
  padding: 4px;
  width: 200px;
  background-color: #eeeeee;
  border: none;
  box-shadow: 1px 1px 2px #bbbbbb;
  cursor: pointer;
}
details > p {
  background-color: #eeeeee;
  padding: 4px;
  margin: 0;
  box-shadow: 1px 1px 2px #bbbbbb;
}
</style>
<body>
<details>
  <summary>Epcot Center</summary>
  <p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and seasonal special events.</p>
</details>
</body>
</html>
```

• HTML Time Tag:

The <time> tag defines a specific time (or datetime).

The datetime attribute of this element is used to translate the time into a machine-readable format so that browsers can offer to add date reminders through the user's calendar, and search engines can produce smarter search results.

eg:

```
<p>Open from <time>10:00</time> to <time>21:00</time> every weekday.</p>
<p>I have a date on <time datetime="2008-02-14 20:00">Valentines day</time>.</p>
```

HTML Style Guide

A consistent, clean, and tidy HTML code makes it easier for others to read and understand your code.

Here are some guidelines and tips for creating good HTML code.

• Always Declare Document Type:

Always declare the document type as the first line in your document.

The correct document type for HTML is:

• Use Lowercase Element Names:

HTML allows mixing uppercase and lowercase letters in element names.

However, it is recommended that using lowercase element names, because:

Mixing uppercase and lowercase names looks bad

Developers normally use lowercase names

Lowercase looks cleaner

Lowercase is easier to write

~Good:

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

~Bad:

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

• Close All HTML Elements:

In HTML, you do not have to close all elements (for example the <p> element).
However, it is strongly recommended that closing all HTML elements, like this:

~Good:

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

~Bad:

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

• Use Lowercase Attribute Names:

HTML allows mixing uppercase and lowercase letters in attribute names.
However, it is recommended that using lowercase attribute names, because:
Mixing uppercase and lowercase names looks bad
Developers normally use lowercase names
Lowercase looks cleaner
Lowercase is easier to write

~Good:

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

~Bad:

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

• Always Quote Attribute Values:

HTML allows attribute values without quotes.
However, it is recommended that quoting attribute values, because:
Developers normally quote attribute values
Quoted values are easier to read
You MUST use quotes if the value contains spaces

~Good:

```
<table class="striped">
```

~Bad:

```
<table class=striped>
```

~Very Bad: This will not work, because the value contains spaces.

```
<table class=table striped>
```

• Always Specify alt, width, and height for Images:

Always specify the alt attribute for images. This attribute is important if the image for some reason cannot be displayed.

Also, always define the width and height of images. This reduces flickering, because the browser can reserve space for the image before loading.
~Good:
```
<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">
```
~Bad:
```
<img src="html5.gif">
```
• Spaces and Equal Signs:
HTML allows spaces around equal signs. But space-less is easier to read and groups entities better together.
~Good:
```
<link rel="stylesheet" href="styles.css">
```
~Bad:
```
<link rel = "stylesheet" href = "styles.css">
```
• Avoid Long Code Lines:
When using an HTML editor, it is NOT convenient to scroll right and left to read the HTML code.
Try to avoid too long code lines.
• Blank Lines and Indentation:
Do not add blank lines, spaces, or indentations without a reason.
For readability, add blank lines to separate large or logical code blocks.
For readability, add two spaces of indentation. Do not use the tab key.
~Good:
```
<body>

<h1>Famous Cities</h1>

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

<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United
Kingdom.</p>

<h2>Paris</h2>
<p>Paris is the capital of France. The Paris area is one of the largest population centers in
Europe.</p>

</body>
```
~Bad:
```
<body>
<h1>Famous Cities</h1>
<h2>Tokyo</h2><p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and
the most populous metropolitan area in the world.</p>
<h2>London</h2><p>London is the capital city of England. It is the most populous city in the
United Kingdom.</p>
<h2>Paris</h2><p>Paris is the capital of France. The Paris area is one of the largest
population centers in Europe.</p>
```

```
</body>
```
~Good Table Example:
```
<table>
  <tr>
    <th>Name</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```
~Good List Example:
```
<ul>
  <li>London</li>
  <li>Paris</li>
  <li>Tokyo</li>
</ul>
```
• Never Skip the <title> Element:

The <title> element is required in HTML.
The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.
The <title> element:
defines a title in the browser toolbar
provides a title for the page when it is added to favourites
displays a title for the page in search-engine results
So, try to make the title as accurate and meaningful as possible:
```
<title>HTML Style Guide and Coding Conventions</title>
```
• Omitting <html> and <body>?:

An HTML page will validate without the <html> and <body> tags.
However, it is strongly recommended to always add the <html> and <body> tags!
Omitting <body> can produce errors in older browsers.
Omitting <html> and <body> can also crash DOM and XML software.

• Omitting Head Element:

The HTML <head> tag can also be omitted.
Browsers will add all elements before <body>, to a default <head> element.
However, it is recommended using the <head> tag.

• Close Empty HTML Elements:

In HTML, it is optional to close empty elements.
~Allowed:
```
<meta charset="utf-8">
```
~Also Allowed:
```
<meta charset="utf-8" />
```

If you expect XML/XHTML software to access your page, keep the closing slash (/), because it is required in XML and XHTML.

• Add the lang Attribute:

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

eg:

```
<!DOCTYPE html>
<html lang="en-us">
<head>
  <title>Page Title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

• Meta Data:

To ensure proper interpretation and correct search engine indexing, both the language and the character encoding <meta charset="charset"> should be defined as early as possible in an HTML document.

eg:

```
<!DOCTYPE html>
<html lang="en-us">
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
</head>
```

• Setting The Viewport:

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

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

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

• 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 with two spaces.

• Using Style Sheets:

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

<link rel="stylesheet" href="styles.css">
Short CSS rules can be written compressed, like this:
p.intro {font-family:Verdana;font-size:16em;}
Long CSS rules should be written over multiple lines:

```
body {
  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

• Loading Java Script in HTML:

Use simple syntax for loading external scripts (the type attribute is not necessary):
<script src="myscript.js">

• Use lower case File Names:

Some web servers (Apache, Unix) are case sensitive about file names: "london.jpg" cannot be accessed as "London.jpg".
Other web servers (Microsoft, IIS) are not case sensitive: "london.jpg" can be accessed as "London.jpg".
If you use a mix of uppercase and lowercase, you have to be aware of this.
If you move from a case-insensitive to a case-sensitive server, even small errors will break your web!
To avoid these problems, always use lowercase file names!

• File Extensions:

HTML files should have a .html extension (.htm is allowed).
CSS files should have a .css extension.
JavaScript files should have a .js extension.

• Differences Between .htm and .html?:

There is no difference between the .htm and .html file extensions!
Both will be treated as HTML by any web browser and web server.

• Default Filenames:

When a URL does not specify a filename at the end (like "https://www.w3schools.com/"), the server just adds a default filename, such as "index.html", "index.htm", "default.html", or "default.htm".
If your server is configured only with "index.html" as the default filename, your file must be named "index.html", and not "default.html".
However, servers can be configured with more than one default filename; usually you can set up as many default filenames as you want.

HTML Entities
Reserved characters in HTML must be replaced with entities:
< (less than) = &lt;
> (greater than) = &gt;

• HTML Character Entities:
Some characters are reserved in HTML.
If you use the less than (<) or greater than (>) signs in your HTML text, the browser might mix them with tags.
Entity names or entity numbers can be used to display reserved HTML characters.
Entity names look like this
&entity_name;
Entity numbers look like this:
&#entity_number;
To display a less than sign (<) we must write: &lt; or &#60;
Entity names are easier to remember than entity numbers.
• Non-breaking Space
A commonly used HTML entity is the non-breaking space: &nbsp;
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.
eg:
§ 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 &nbsp; character entity.
The non-breaking hyphen (&#8209;) is used to define a hyphen character (-) that does not break into a new line.
• Some Useful HTML Character Entities:

Result	Description	Name
	non-breaking space	&nbsp;
<	less than	&lt;
>	greater than	&gt;
&	ampersand	&amp;
"	double quotation mark	&quot;
'	single quotation mark	&apos;
¢	cent	&cent;
£	pound	&pound;
¥	yen	&yen;
€	euro	&euro;
©	copyright	&copy;
®	trademark	&reg;

# Note:Entity names are case sensitive.
• Combining Diacritical Marks:
A diacritical mark is a "glyph" added to a letter.
Some diacritical marks, like grave (̀) and acute (́) are called accents.
Diacritical marks can be used in combination with alphanumeric characters to produce a character that is not present in the character set (encoding) used in the page.

Mark	Character	Construct	Result
`	a	a&#768;	à
´	a	a&#769;	á
ˆ	a	a&#770;	â
˜	a	a&#771;	ã
`	O	O&#768;	Ò
´	O	O&#769;	Ó
ˆ	O	O&#770;	Ô
˜	O	O&#771;	Õ

HTML Symbols:
Symbols or letters that are not present on your keyboard can be added to HTML using entities.
Many mathematical, technical, and currency symbols are not present on a normal keyboard.
To add such symbols to an HTML page, you can use the entity name or the entity number (a decimal or a hexadecimal reference) for the symbol:
eg:
```
<p>I will display &euro;</p>
<p>I will display &#8364;</p>
<p>I will display &#x20AC;</p>
```
To know more about HTML Symbols please refer to these links!
• Some Mathematical Symbols Supported by HTML:
https://www.w3schools.com/charsets/ref_utf_math.asp
• Some Greek Letters Supported by HTML:
https://www.w3schools.com/charsets/ref_utf_greek.asp
• Arrows Supported by HTML:
https://www.w3schools.com/charsets/ref_utf_arrows.asp
• Symbols Supported by HTML:
https://www.w3schools.com/charsets/ref_utf_symbols.asp
• Currency symbols Supported by HTML:
https://www.w3schools.com/charsets/ref_utf_currency.asp

HTML Emojis
Emojis are characters from the UTF-8 character set: 😄 😍 💗
• HTML Smiley Emojis:
https://www.w3schools.com/charsets/ref_emoji_smileys.asp
• HTML Transport Emojis:
https://www.w3schools.com/charsets/ref_emoji_transport.asp
• HTML Office Emojis:
https://www.w3schools.com/charsets/ref_emoji_office.asp
• HTML People Emojis:
https://www.w3schools.com/charsets/ref_emoji_body.asp
• HTML Animal Emojis:
https://www.w3schools.com/charsets/ref_emoji_animals.asp

What are Emojis?
Emojis look like images, or icons, but they are not.
They are letters (characters) from the UTF-8 (Unicode) character set.
UTF-8 covers almost all of the characters and symbols in the world.

- <u>The HTML charset Attribute:</u>

To display an HTML page correctly, a web browser must know the character set used in the page.
This is specified in the <meta> tag:

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

If not specified, UTF-8 is the default character set in HTML.

- <u>UTF-8 Characters:</u>

Many UTF-8 characters cannot be typed on a keyboard, but they can always be displayed using numbers (called entity numbers):
eg:
A is &#65
B is &#66
C is &#67
eg:

```
<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<body>
<p>I will display A B C</p>
<p>I will display &#65; &#66; &#67;</p>
</body>
</html>
```

The <meta charset="UTF-8"> element defines the character set.
The characters A, B, and C, are displayed by the numbers 65, 66, and 67.
To let the browser understand that you are displaying a character, you must start the entity number with &# and end it with ; (semicolon).

- <u>Emoji Characters:</u>

Emojis are also characters from the UTF-8 alphabet:
😄 is 128516
😍 is 128525
💗 is 128151
eg:

```
<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<body>
<h1>My First Emoji</h1>
<p>&#128512;</p>
</body>
</html>
```

Since Emojis are characters, they can be copied, displayed, and sized just like any other character in HTML.
eg:

```
<!DOCTYPE html>
```

```
<html>
<meta charset="UTF-8">
<body>
<h1>Sized Emojis</h1>
<p style="font-size:48px">
&#128512; &#128516; &#128525; &#128151;
</p>
</body>
</html>
```

HTML Encoding (Character Sets)

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

• HTML Charset Attribute:

The character set is specified in the <meta> tag:

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

The HTML5 specification encourages web developers to use the UTF-8 character set.
UTF-8 covers almost all of the characters and symbols in the world!
Please Refer to this link for UTF Characters:
https://www.w3schools.com/charsets/ref_utf_basic_latin.asp

• The ASCII Character Set:

ASCII was the first character encoding standard for the web. It defined 128 different characters that could be used on the internet:
English letters (A-Z)
Numbers (0-9)
Special characters like ! $ + - ( ) @ < >

• The ANSI Character Set:

ANSI (Windows-1252) was the original Windows character set:
Identical to ASCII for the first 127 characters
Special characters from 128 to 159
Identical to UTF-8 from 160 to 255

• The ISO-8859-1 Character Set:

ISO-8859-1 was the default character set for HTML 4. This character set supported 256 different character codes. HTML 4 also supported UTF-8.
Identical to ASCII for the first 127 characters
Does not use the characters from 128 to 159
Identical to ANSI and UTF-8 from 160 to 255
HTML 4 Example

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

HTML 5 Example

```
<meta charset="ISO-8859-1">
```

• The UTF-8 Character Set:

is identical to ASCII for the values from 0 to 127
Does not use the characters from 128 to 159
Identical to ANSI and 8859-1 from 160 to 255
Continues from the value 256 to 10 000 characters

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

HTML Uniform Resource Locators (URL)
A URL is another word for a web address.
A URL can be composed of words (e.g. w3schools.com), or an Internet Protocol (IP) address (e.g. 192.68.20.50).
Most people enter the name when surfing, because names are easier to remember than numbers.
• <u>URL - Uniform Resource Locator:</u>
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
• <u>Common URL Schemes:</u>

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

• <u>URL Encoding:</u>
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.
• <u>ASCII Encoding Examples:</u>
Your browser will encode input, according to the character-set used in your page.
The default character-set in HTML5 is UTF-8.
for more encoding reference visit this: https://www.w3schools.com/tags/ref_urlencode.asp

HTML Versus XHTML
XHTML is a stricter, more XML-based version of HTML.
What is XHTML?
XHTML stands for EXtensible HyperText Markup Language

XHTML is a stricter, more XML-based version of HTML
XHTML is HTML defined as an XML application
XHTML is supported by all major browsers
Why XHTML?
XML is a markup language where all documents must be marked up correctly (be "well-formed").
XHTML was developed to make HTML more extensible and flexible to work with other data formats (such as XML). In addition, browsers ignore errors in HTML pages, and try to display the website even if it has some errors in the markup. So XHTML comes with a much stricter error handling.

• The Most Important Differences from HTML:

<!DOCTYPE> is mandatory
The xmlns attribute in <html> is mandatory
<html>, <head>, <title>, and <body> are mandatory
Elements must always be properly nested
Elements must always be closed
Elements must always be in lowercase
Attribute names must always be in lowercase
Attribute values must always be quoted
Attribute minimization is forbidden

• XHTML - <!DOCTYPE ....> Is Mandatory:

An XHTML document must have an XHTML <!DOCTYPE> declaration.
The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document.
eg:
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Title of document</title>
</head>
<body>
  some content here...
</body>
</html>
```

• XHTML Elements Must be Properly Nested:

In XHTML, elements must always be properly nested within each other, like this:
~Correct:
```
<b><i>Some text</i></b>
```
~Wrong:
```
<b><i>Some text</b></i>
```

• XHTML Elements Must Always be Closed:

In XHTML, elements must always be closed, like this:
~Correct:
```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```
~Wrong:
```
<p>This is a paragraph
```

<p>This is another paragraph

• XHTML Empty Elements Must Always be Closed:

In XHTML, empty elements must always be closed, like this:
~Correct:
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
~Wrong:
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">

• XHTML Elements Must be in Lowercase:

In XHTML, element names must always be in lowercase, like this:
~Correct:
<body>
<p>This is a paragraph</p>
</body>
~Wrong:
<BODY>
<P>This is a paragraph</P>
</BODY>

• XHTML Attribute Values Must be Quoted:

In XHTML, attribute values must always be quoted, like this:
~Correct:
<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>
~Wrong:
<a href=https://www.w3schools.com/html/>Visit our HTML tutorial</a>

• XHTML Attribute Minimization is Forbidden:

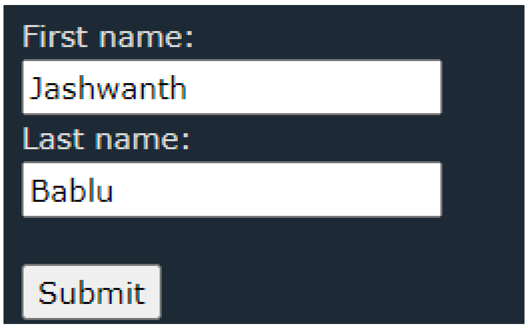
In XHTML, attribute minimization is forbidden
~Correct:
<input type="checkbox" name="vehicle" value="car" checked="checked" />
<input type="text" name="lastname" disabled="disabled" />
~Wrong:
<input type="checkbox" name="vehicle" value="car" checked />
<input type="text" name="lastname" disabled />

HTML Forms

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

• The Form Element:

The HTML <form> element is used to create an HTML form for user input:
eg:
<form>
...
form elements
...
</form>



The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

• The Input Element:

The HTML <input> element is the most used form element.
An <input> element can be displayed in many ways, depending on the type attribute.
Here are some examples:

Type	Description
<input type="text">	Displays a single-line text input field.
<input type="radio">	Displays a radio button (for selecting one of many choices).
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices).
<input type="submit">	Displays a submit button (for submitting the form).
<input type="button">	Displays a clickable button.

• The Text Fields:

The <input type="text"> defines a single-line input field for text input.
eg:
A form with input fields for text:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

# Note: The form itself is not visible. Also note that the default width of an input field is 20 characters.

• The <label> Element:

Notice the use of the <label> element in the example above.
The <label> tag defines a label for many form elements.
The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focuses on the input element.
The <label> element also helps users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.
The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

• The Radio Buttons:

The <input type="radio"> defines a radio button.
Radio buttons let a user select ONE of a limited number of choices.
eg:
A form with radio buttons:

```
<p>Choose your favourite Web language:</p>
<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

• The Check Boxes:

The <input type="checkbox"> defines a checkbox.
Checkboxes let a user select ZERO or MORE options of a limited number of choices.
eg:
A form with checkboxes:

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

• The Submit Button:

The <input type="submit"> defines a button for submitting the form data to a form-handler.
The form-handler is typically a file on the server with a script for processing input data.
The form-handler is specified in the form's action attribute.
A form with a submit button:

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

• The Name Attribute for <input>:

Notice that each input field must have a name attribute to be submitted.
If the name attribute is omitted, the value of the input field will not be sent at all.
eg:

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" value="John"><br><br>
  <input type="submit" value="Submit">
</form>
```

HTML Form Attributes

This chapter describes the different attributes for the HTML <form> element.

• <u>The Action Attribute:</u>

The action attribute defines the action to be performed when the form is submitted.
Usually, the form data is sent to a file on the server when the user clicks on the submit button.
In the example below, the form data is sent to a file called "action_page.php". This file contains a server-side script that handles the form data:
eg:
On submit, send form data to "action_page.php":

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

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

• <u>The Target Attribute:</u>

The target attribute specifies where to display the response that is received after submitting the form.
The target attribute can have one of the following values:

Value	Description
_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window
framename	The response is displayed in a named iframe

The default value is _self which means that the response will open in the current window.
eg: <form action="/action_page.php" target="_blank">

• <u>The Method Attribute:</u>

The method attribute specifies the HTTP method to be used when submitting the form data.
The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").
The default HTTP method when submitting form data is GET.
eg:
~This example uses the GET method when submitting the form data:
<form action="/action_page.php" method="get">
~This example uses the POST method when submitting the form data:
<form action="/action_page.php" method="post">

# Notes on GET:

Appends the form data to the URL, in name/value pairs
NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
The length of a URL is limited (2048 characters)

Useful for form submissions where a user wants to bookmark the result
GET is good for non-secure data, like query strings in Google
# Notes on POST:
Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
POST has no size limitations, and can be used to send large amounts of data.
Form submissions with POST cannot be bookmarked
# Tip: Always use POST if the form data contains sensitive or personal information!
• The Autocomplete Attribute:
The autocomplete attribute specifies whether a form should have autocomplete on or off.
When autocomplete is on, the browser automatically completes values based on values that the user has entered before.
eg:
A form with autocomplete on:

```
<form action="/action_page.php" autocomplete="on">
```

• Novalidate Attribute:
The novalidate attribute is a boolean attribute.
When present, it specifies that the form-data (input) should not be validated when submitted.
eg:
A form with a novalidate attribute:

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

HTML Form Elements
This chapter describes all the different HTML form elements.
The HTML <form> Elements
The HTML <form> element can contain one or more of the following form elements:
<input>
<label>
<select>
<textarea>
<button>
<fieldset>
<legend>
<datalist>
<output>
<option>
<optgroup>
• The <input> element:
One of the most used form elements is the <input> element.
The <input> element can be displayed in several ways, depending on the type attribute.
eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>The input Element</h2>
<form action="/action_page.php">
```

```
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

• The <label> element:

The <label> element defines a label for several form elements.
The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.
The <label> element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.
The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

• The <select> element:

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

```
<!DOCTYPE html>
<html>
<body>
<h2>The select Element</h2>
<p>The select element defines a drop-down list:</p>
<form action="/action_page.php">
  <label for="jeeps">Choose a Jeep:</label>
  <select id="cars" name="cars">
    <option value="Boa">Boa</option>
    <option value="Thar">Thar</option>
    <option value="Alfred">Alfered</option>
    <option value="Sandman">Sandman</option>
  </select>
  <input type="submit">
</form>
</body>
</html>
```

The <option> element 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:

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

~Visible Values
Use the size attribute to specify the number of visible values:
eg:

```
<label for="cars">Choose a car:</label>
<select id="cars" 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:
eg:
```
<!DOCTYPE html>
<html>
<body>
<h2>Allow Multiple Selections</h2>
<p>Use the multiple attribute to allow the user to select more than one value.</p>
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" 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><br><br>
  <input type="submit">
</form>
<p>Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.</p>
</body>
</html>
```
• The<textarea> element:

The <textarea> element defines a multi-line input field (a text area)
```
<!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">Please give your Feed Back
here🙏</textarea>
  <br><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:
eg:
```
<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:
eg:
```
<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.

• The <fieldset> and <legend> elements:

The <fieldset> element is used to group related data in a form.
The <legend> element defines a caption for the <fieldset> element.

eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>Grouping Form Data with Fieldset</h2>
<p>The fieldset element is used to group related data in a form, and the legend element
defines a caption for the fieldset element.</p>
<form action="/action_page.php">
  <fieldset>
    <legend>Details:</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
</body>
</html>
```

• The <datalist> element:

The <datalist> element specifies a list of pre-defined options for an <input> element.
Users will see a drop-down list of the pre-defined options as they input data.
The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.

eg:

```
<!DOCTYPE html>
<html>
<body>
<h2>The datalist Element</h2>
<p>The datalist element specifies a list of pre-defined options for an input element.</p>
<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Edge">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
  <input type="submit">
```

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

• The<output> element:

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

```
<!DOCTYPE html>
<html>
<body>
<h2>The output Element</h2>
<p>The output element represents the result of a calculation.</p>
<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>
</body>
</html>
```

HTML Input Types

This chapter describes the different types for the HTML <input> element.
Here are the different input types you can use in HTML:

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
```

```
<input type="text">
<input type="time">
<input type="url">
<input type="week">
```
# Tip: The default value of the type attribute is "text".

• Input Type Text:

<input type="text"> defines a single-line text input field.

eg:
```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```
• Input Type Password:

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

eg:
```
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd">
</form
```
# Note: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.
```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```
• Input Type Reset:

<input type="reset"> defines a reset button that will reset all form values to their default values.

eg:
```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
</form>
```

# Note:If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

<u>• Input Type Radio:</u>

<input type="radio"> defines a radio button.
Radio buttons let a user select ONLY ONE of a limited number of choices.
eg:

```
<p>Choose your favourite Web language:</p>
<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

<u>• Input Type Checkbox:</u>

<input type="checkbox"> defines a checkbox.
Checkboxes let a user select ZERO or MORE options of a limited number of choices.
eg:

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

<u>• Input Type Button:</u>

<input type="button"> defines a button.
eg:

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

<u>• Input Type Color:</u>

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.
eg:

```
<form>
  <label for="favcolor">Select your favorite color:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

<u>• Input Type Date:</u>

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.
eg:

```
<form>
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday">
</form>
```

~You can also use the min and max attributes to add restrictions to dates:

eg:
```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02">
</form>
```
• <u>Input Type Datetime-Local:</u>
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.
eg:
```
<form>
  <label for="birthdaytime">Birthday (date and time):</label>
  <input type="datetime-local" id="birthdaytime" name="birthdaytime">
</form>
```
• <u>Input Type Email:</u>
The <input type="email"> is used for input fields that should contain an e-mail address.
Depending on browser support, the e-mail address can be automatically validated when submitted.
Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.
eg:
```
<form>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email">
</form>
```
• <u>Input Type Image:</u>
The <input type="image"> defines an image as a submit button.
The path to the image is specified in the src attribute.
eg:
```
<form>
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```
• <u>Input Type File:</u>
The <input type="file"> defines a file-select field and a "Browse" button for file uploads.
eg:
```
<form>
  <label for="myfile">Select a file:</label>
  <input type="file" id="myfile" name="myfile">
</form>
```
• <u>Input Type Hidden:</u>
The <input type="hidden"> defines a hidden input field (not visible to a user).
A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted.
A hidden field often stores what database record that needs to be updated when the form is submitted.

# Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!
eg:

```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
```

• Input Type Month:
The <input type="month"> allows the user to select a month and year.
Depending on browser support, a date picker can show up in the input field.
eg:

```
<form>
  <label for="bdaymonth">Birthday (month and year):</label>
  <input type="month" id="bdaymonth" name="bdaymonth">
</form>
```

• Input Type Number:
The <input type="number"> defines a numeric input field.
You can also set restrictions on what numbers are accepted.
The following example displays a numeric input field, where you can enter a value from 1 to 5:

```
<form>
  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

• Input Restrictions:
Here is a list of some common input restrictions:

Attribute	Description
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

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:

eg:
```
<form>
  <label for="quantity">Quantity:</label>
  <input type="number" id="quantity" name="quantity" min="0" max="100" step="10"
value="30">
</form>
```
• Input Type Range:

The <input type="range"> defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the min, max, and step attributes:

eg:
```
<!DOCTYPE html>
<html>
<body>
<h2>Range Field</h2>
<p>Depending on browser support: The input type "range" can be displayed as a slider
control.</p>
<form action="/action_page.php" method="get">
  <label for="vol">Volume (between 0 and 50):</label>
  <input type="range" id="vol" name="vol" min="0" max="50">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```
• Input Type Search:

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

eg:
```
<form>
  <label for="gsearch">Search Google:</label>
  <input type="search" id="gsearch" name="gsearch">
</form>
```
• Input Type Tel:

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

eg:
```
<form>
  <label for="phone">Enter your phone number:</label>
  <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```
• Input Type Time:

The <input type="time"> allows the user to select a time (no time zone).
Depending on browser support, a time picker can show up in the input field.

eg:
```
<form>
  <label for="appt">Select a time:</label>
  <input type="time" id="appt" name="appt">
</form>
```

• Input Type Url:

The <input type="url"> is used for input fields that should contain a URL address.
Depending on browser support, the url field can be automatically validated when submitted.
Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

eg:

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

• Input Type Week:

The <input type="week"> allows the user to select a week and year.
Depending on browser support, a date picker can show up in the input field.

eg:

```
<form>
  <label for="week">Select a week:</label>
  <input type="week" id="week" name="week">
</form>
```

HTML Input Attributes

This chapter describes the different attributes for the HTML <input> element.

• The value Attribute:

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

eg:

Input fields with initial (default) values:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

• The readonly Attribute:

The input readonly attribute specifies that an input field is read-only.
A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).
The value of a read-only input field will be sent when submitting the form!

eg:

A read-only input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" readonly><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

• The disabled Attribute:

The input disabled attribute specifies that an input field should be disabled.
A disabled input field is unusable and un-clickable.
The value of a disabled input field will not be sent when submitting the form!
eg:
A disabled input field:
```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```
• The size Attribute

The input size attribute specifies the visible width, in characters, of an input field.
The default value for size is 20.
# Note: The size attribute works with the following input types: text, search, tel, url, email, and password.
eg:
Set a width for an input field:
```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" size="4">
</form>
```
• The maxlength Attribute:

The input maxlength attribute specifies the maximum number of characters allowed in an input field.
# Note: When a maxlength is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.
eg:
Set a maximum length for an input field:
```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" maxlength="4" size="4">
</form>
```
• The min and max Attributes:

The input min and max attributes specify the minimum and maximum values for an input field.
The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.
# Tip: Use the max and min attributes together to create a range of legal values.
eg:
Set a max date, a min date, and a range of legal values:
```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02"><br><br>
  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

• The multiple Attribute:

The input multiple attribute specifies that the user is allowed to enter more than one value in an input field.

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

eg:

A file upload field that accepts multiple values:

```
<form>
  <label for="files">Select files:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```

• The pattern Attribute:

The input pattern attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.

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

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

eg:

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

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

• The placeholder Attribute:

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

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

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

eg:

An input field with a placeholder text:

```
<form>
  <label for="phone">Enter a phone number:</label>
  <input type="tel" id="phone" name="phone"
  placeholder="123-45-678"
  pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

• The required Attribute:

The input required attribute specifies that an input field must be filled out before submitting the form.

The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

eg:
A required input field:
```
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
</form>
```
• The Step Attribute:

The input step attribute specifies the legal number intervals for an input field.
Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.
# Tip: This attribute can be used together with the max and min attributes to create a range of legal values.
The step attribute works with the following input types: number, range, date, datetime-local, month, time and week.
eg:
An input field with a specified legal number intervals:
```
<form>
  <label for="points">Points:</label>
  <input type="number" id="points" name="points" step="3">
</form>
```
# Note: Input restrictions are not foolproof, and JavaScript provides many ways to add illegal input. To safely restrict input, it must also be checked by the receiver (the server)!

• The autofocus Attribute:

The input autofocus attribute specifies that an input field should automatically get focus when the page loads.
eg:
Let the "First name" input field automatically get focus when the page loads:
```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" autofocus><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```
• The height and width Attributes:

The input height and width attributes specify the height and width of an <input type="image"> element.
# Tip: Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. Without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).
eg:
```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```

• The list Attribute:
The input list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.
eg:
An <input> element with pre-defined values in a <datalist>:
```
<form>
  <input list="browsers">
  <datalist id="browsers">
    <option value="Edge">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```
• The autocomplete Attribute:
The input autocomplete attribute specifies whether a form or an input field should have autocomplete on or off.
Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.
The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.
eg:
An HTML form with autocomplete on, and off for one input field:
```
<form action="/action_page.php" autocomplete="on">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" autocomplete="off"><br><br>
  <input type="submit" value="Submit">
</form>
```
# Tip: In some browsers you may need to activate an autocomplete function for this to work (Look under "Preferences" in the browser's menu).

HTML Canvas Graphics
The HTML <canvas> element is used to draw graphics on a web page.
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.

• Add a JavaScript:

After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

eg:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
</script>
```

To Draw a Circle:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
</script>
```

To Draw a Text:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</script>
```

To Draw a Stroke Text:

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Jashwanth.",10,50);
</script>
</body>
</html>
```

• Gradients:

To Draw Linear Gradient:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
```

```
// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

To Draw Circular Gradient:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// Create gradient
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

<u>To Draw an Image:</u>

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img, 10, 10);
</script>
```

HTML SVG Graphics

SVG defines vector-based graphics in XML format.

What is SVG?

SVG stands for Scalable Vector Graphics
SVG is used to define graphics for the Web
SVG is a W3C recommendation

• <u>The HTML <svg> Element:</u>

The HTML <svg> element is a container for SVG graphics.
SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

• <u>SVG Circle:</u>

```
<!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>
```

• <u>SVG Rectangle:</u>

```
<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:
```
<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:
```
<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:
```
<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">Jash</text>
Sorry, your browser does not support inline SVG.
</svg>
```
• Differences Between SVG and Canvas:

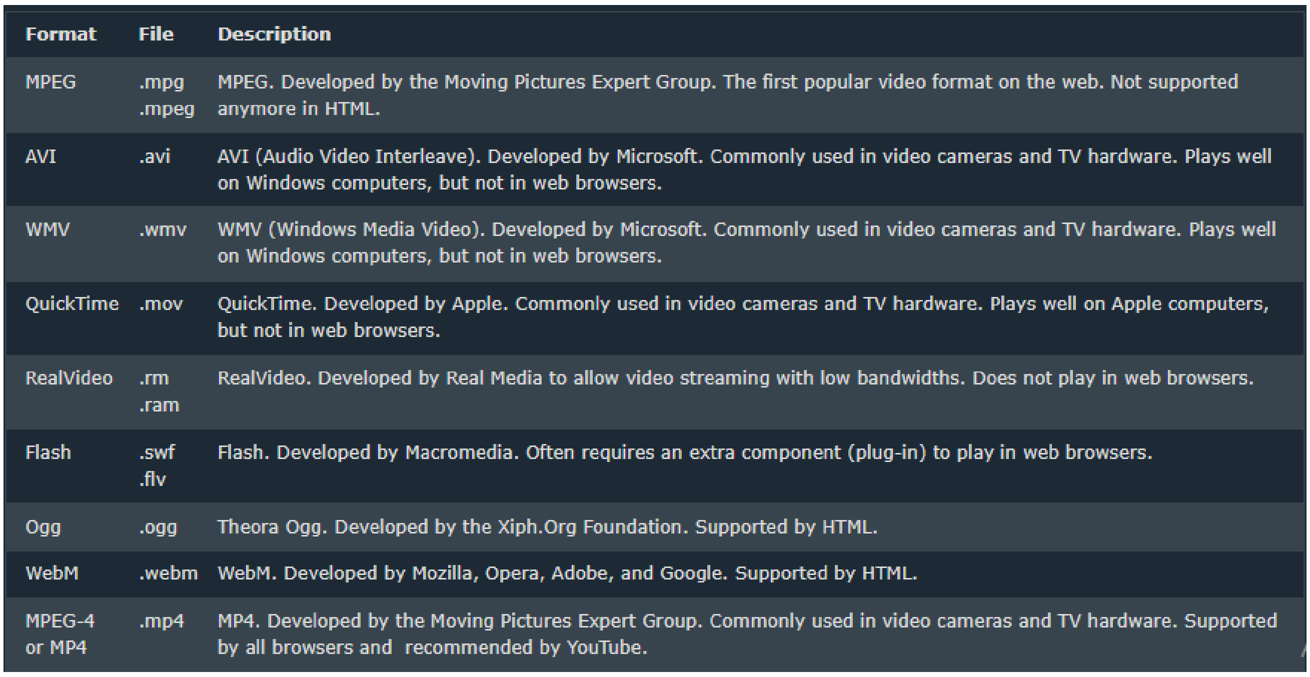
SVG is a language for describing 2D graphics in XML.
Canvas draws 2D graphics, on the fly (with JavaScript).
SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
In SVG, each drawn shape is remembered as an object. 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:

~Canvas
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
~SVG
Resolution independent
Support for event handlers

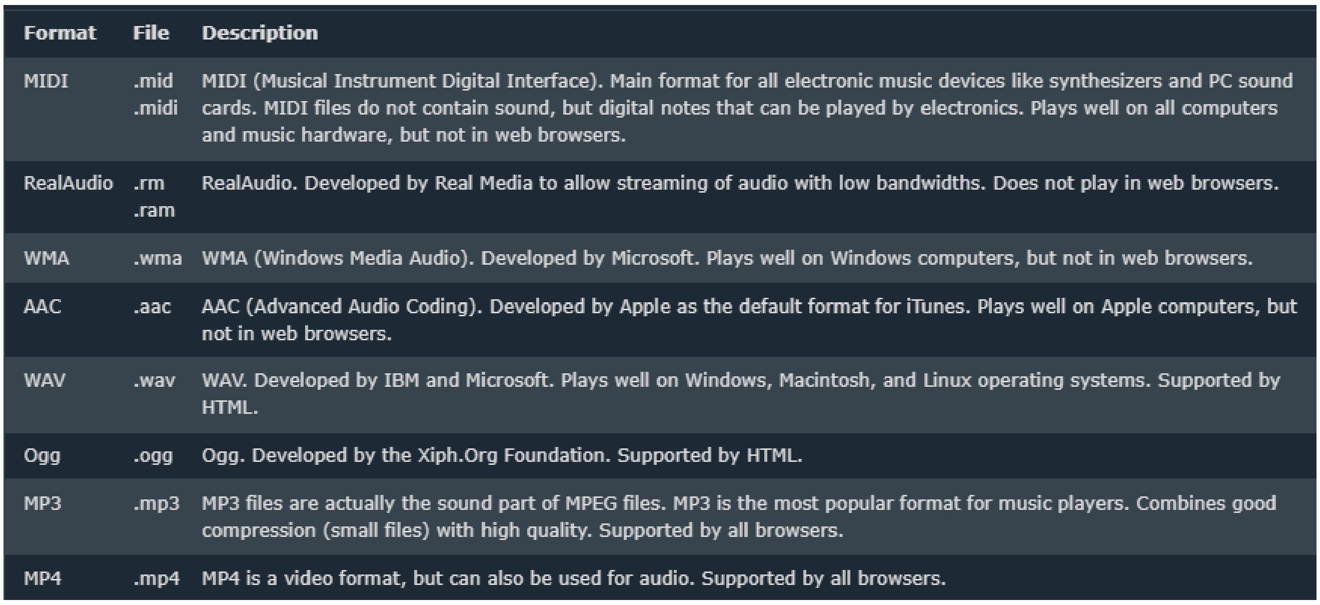
Best suited for applications with large rendering areas (Google Maps)
Slow rendering if complex (anything that uses the DOM a lot will be slow)
Not suited for game applications

HTML Multimedia
Multimedia on the web is sound, music, videos, movies, and animations.
What is Multimedia?
Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more.
Web pages often contain multimedia elements of different types and formats.
<u>• Multimedia Formats:</u>
Multimedia elements (like audio or video) are stored in media files.
The most common way to discover the type of a file, is to look at the file extension.
Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.
<u>• Common Video Formats:</u>
There are many video formats out there.
The MP4, WebM, and Ogg formats are supported by HTML.
The MP4 format is recommended by YouTube.



Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Not supported anymore in HTML.
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers.
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. Does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
WebM	.webm	WebM. Developed by Mozilla, Opera, Adobe, and Google. Supported by HTML.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Commonly used in video cameras and TV hardware. Supported by all browsers and recommended by YouTube.

# Note: Only MP4, WebM, and Ogg video are supported by the HTML standard.
<u>• Common Audio Formats:</u>
MP3 is the best format for compressed recorded music. The term MP3 has become synonymous with digital music.
If your website is about recorded music, MP3 is the choice.



Format	File	Description
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. Supported by all browsers.

# Note: Only MP3, WAV, and Ogg audio are supported by the HTML standard.

HTML Video

The HTML <video> element is used to show a video on a web page.

• The HTML <video> element:

To show a video in HTML, use the <video> element.

eg:

```
<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>
```

How does 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.

```
<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>
```

# Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add muted after autoplay to let your video start playing automatically (but muted):

eg:

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

• <u>HTML Video - Methods, Properties, and Events:</u>

The HTML DOM defines methods, properties, and events for the <video> element.
This allows you to load, play, and pause videos, as well as setting duration and volume.
There are also DOM events that can notify you when a video begins to play, is paused, etc.
eg:

```
<!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 HTML 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>
```

HTML Audio

The HTML <audio> element is used to play an audio file on a web page.

• The HTML <audio> Element:

To play an audio file in HTML, use the <audio> element.

eg:

```
<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>
```

HTML Audio - How It Works?

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

• HTML <audio> Autoplay:

To start an audio file automatically, use the autoplay attribute.

eg:

```
<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

# Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add muted after autoplay to let your audio file start playing automatically (but muted):

eg:

```
<audio controls autoplay muted>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

• HTML Audio - Methods, Properties, and Events:

The HTML DOM defines methods, properties, and events for the <audio> element.

This allows you to load, play, and pause audios, as well as set duration and volume.

There are also DOM events that can notify you when an audio begins to play, is paused, etc.

HTML Plug-ins

Plug-ins are computer programs that extend the standard functionality of the browser.

Plug-ins were designed to be used for many different purposes:

To run Java applets

To run Microsoft ActiveX controls

To display Flash movies

To display maps

To scan for viruses

To verify a bank id

# Warning !
Most browsers no longer support Java Applets and Plug-ins.
ActiveX controls are no longer supported in any browsers.
The support for Shockwave Flash has also been turned off in modern browsers.
• The <object> Element:
The <object> element is supported by all browsers.
The <object> element defines an embedded object within an HTML document.
It was designed to embed plug-ins (like Java applets, PDF readers, and Flash Players) in web pages, but can also be used to include HTML in HTML:
eg:
```
<object width="100%" height="500px" data="snippet.html"></object>
```
Or images if you like:
```
<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.
eg:
```
<embed src="audi.jpeg">
```
# Note that the <embed> element does not have a closing tag. It can not contain alternative text.
The <embed> element can also be used to include HTML in HTML:
eg:
```
<embed width="100%" height="500px" src="snippet.html">
```

HTML YouTube Videos
The easiest way to play videos in HTML is to use YouTube.
Struggling with Video Formats?
Converting videos to different formats can be difficult and time-consuming.
An easier solution is to let YouTube play the videos in your web page.
• YouTube Video Id:
YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.
You can use this id, and refer to your video in the HTML code.
• Playing a YouTube Video in HTML:
To play your video on a web page, do the following:
Upload the video to YouTube
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)
eg:
```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```
• YouTube Autoplay + Mute:

You can let your video start playing automatically when a user visits the page, by adding autoplay=1 to the YouTube URL. However, automatically starting a video is annoying for your visitors!
# Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.
Add mute=1 after autoplay=1 to let your video start playing automatically (but muted).

• YouTube - Autoplay + Muted:

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>
```

• YouTube Playlist:

A comma separated list of videos to play (in addition to the original URL).

• YouTube Loop:

Add loop=1 to let your video loop forever.
Value 0 (default): The video will play only once.
Value 1: The video will loop (forever).
eg:

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1">
</iframe>
```

• YouTube Controls:

Add controls=0 to not display controls in the video player.
Value 0: Player controls does not display.
Value 1 (default): Player controls display.
eg:

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

HTML Geolocation API

The HTML Geolocation API is used to locate a user's position.

• Locate the User's Position:

The HTML Geolocation API is used to get the geographical position of a user.
Since this can compromise privacy, the position is not available unless the user approves it.
# Note: Geolocation is most accurate for devices with GPS, like smartphones.

• 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.
eg:

```
<script>
const 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>
```

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.
eg:

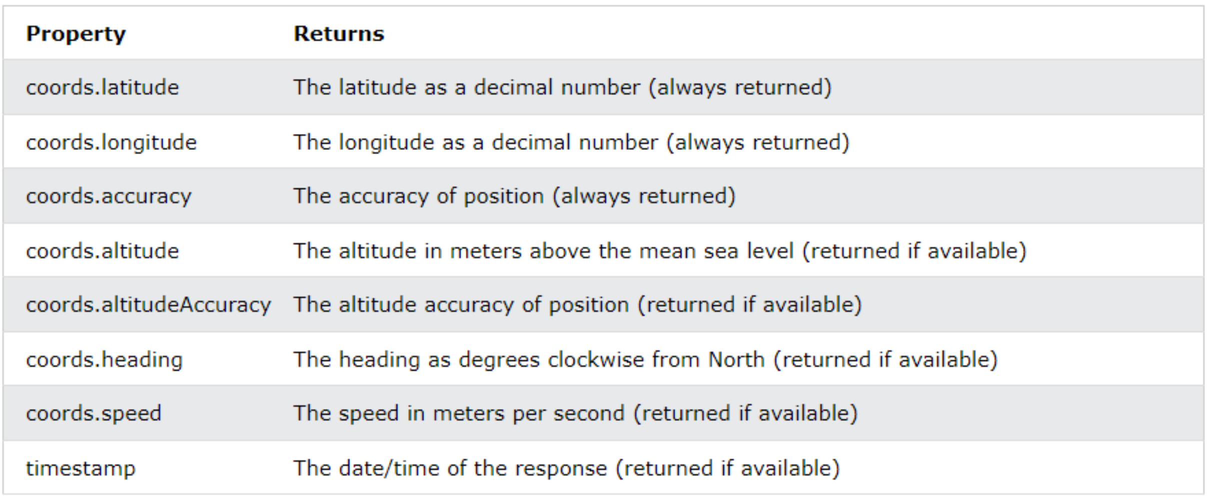
```
function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML = "The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
  }
}
```

• Location-specific Information:

This page has demonstrated how to show a user's position on a map.
Geolocation is also very useful for location-specific information, like:
Up-to-date local information
Showing Points-of-interest near the user
Turn-by-turn navigation (GPS)

• The getCurrentPosition() Method - Return Data:

The getCurrentPosition() method returns an object on success. The latitude, longitude and accuracy properties are always returned. The other properties are returned if available.



Property	Returns
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

• Geolocation Object - Other interesting Methods:

The Geolocation object also has other interesting methods:
watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
clearWatch() - Stops the watchPosition() method.
The example below shows the watchPosition() method. You need an accurate GPS device to test this (like smartphone):
eg:

```
<script>
const 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>
```

HTML Drag and Drop API

In HTML, any element can be dragged and dropped.
Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.
eg:

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336"
height="69">
</body>
</html>
```

It might seem complicated, but lets go through all the different parts of a drag and drop event.

• Make an Element Draggable:

First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

• What to Drag - ondragstart and setData():

Then, specify what should happen when the element is dragged.
In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.
The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
```

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

• Where to Drop - ondragover:

The ondragover event specifies where the dragged data can be dropped.
By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.
This is done by calling the event.preventDefault() method for the ondragover event:

```
event.preventDefault()
```

• Do the Drop - ondrop:

When the dragged data is dropped, a drop event occurs.
In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

Code explained:

Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)
Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method
The dragged data is the id of the dragged element ("drag1")
Append the dragged element into the drop element
eg:

```
<!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>
```

HTML Web Storage API
HTML web storage; better than cookies.

What is HTML Web Storage?
With web storage, web applications can store data locally within the user's browser.
Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

<u>• HTML Web Storage Objects:</u>

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..
}
```

<u>• The localStorage Object:</u>

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.
eg:

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

Example explained:
Create a localStorage name/value pair with name="lastname" and value="Smith"
Retrieve the value of "lastname" and insert it into the element with id="result"
The example above could also be written like this:

```
// Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
```

The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

# Note: Name/value pairs are always stored as strings. Remember to convert them to another format when needed!

The following example counts the number of times a user has clicked a button. In this code the value string is converted to a number to be able to increase the counter:
eg:

```
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).";
```

<u>• The sessionStorage Object:</u>

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:
eg:

```
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.";
```

HTML Web Workers API

A web worker is a JavaScript running in the background, without affecting the performance of the page.

What is a Web Worker?

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.
A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.
The example below creates a simple web worker that count numbers in the background.
eg:

```
<!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>
```

• 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:
eg:
```
<!DOCTYPE html>
<html>
<body>
<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<script>
var w;
function startWorker() {
  if (typeof(Worker) !== "undefined") {
    if (typeof(w) == "undefined") {
      w = new Worker("demo_workers.js");
    }
    w.onmessage = function(event) {
      document.getElementById("result").innerHTML = event.data;
    };
  } else {
    document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";
  }
}
function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>
</body>
</html>
```
• Web Workers and the DOM:

Since web workers are in external files, they do not have access to the following JavaScript objects:
The window object
The document object
The parent object

HTML SSE API

Server-Sent Events (SSE) allow a web page to get updates from a server.

• Server-Sent Events - One Way Messaging:

A server-sent event is when a web page automatically gets updates from a server.
This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.
Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

• Receive Server-Sent Event Notifications:

The EventSource object is used to receive server-sent event notifications:
eg:

```
var source = new EventSource("demo_sse.php");
source.onmessage = function(event) {
  document.getElementById("result").innerHTML += event.data + "<br>";
};
```

Example explained:
Create a new EventSource object, and specify the URL of the page sending the updates (in this example "demo_sse.php")
Each time an update is received, the onmessage event occurs
When an onmessage event occurs, put the received data into the element with id="result"

• Check Server-Sent Events Support:

In the tryit example above there were some extra lines of code to check browser support for server-sent events:

```
if(typeof(EventSource) !== "undefined") {
  // 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

Resources :-
~~~~~~~~~~~
Attributes reference: https://www.w3schools.com/tags/ref_attributes.asp
Element reference: https://www.w3schools.com/tags/ref_byfunc.asp
Tag reference: https://www.w3schools.com/tags/default.asp
HTML colour names supported by all browsers:
https://www.w3schools.com/colors/colors_names.asp

❤️ ALL IZZ WELL ❤️

By:
– N.JASHWANTH.
~~~~~~~~~~~