Inline-block

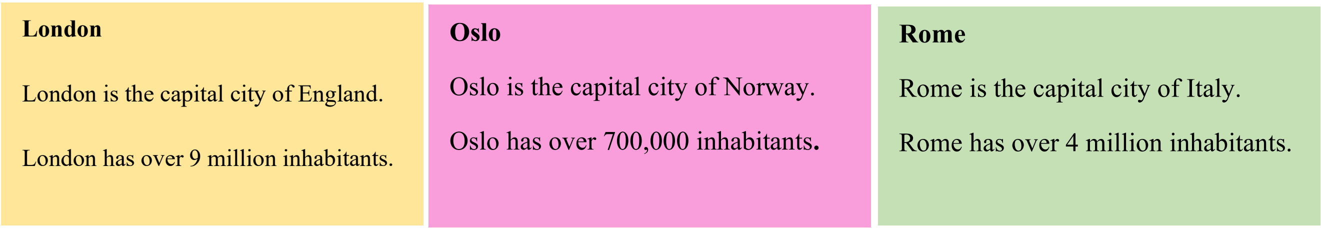
If you change the **<div>** element's display property from block to **inline-block**, the **<div>** elements will no longer add a line break before and after, and will be displayed side by side instead of on top of each other.

Ex: How to use display: inline-block to align div elements side by side

Input:

```
<style>
div {
  width: 30%;
  display: inline-block;
}
</style>
```

Output:



London

London is the capital city of England.

London has over 9 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 700,000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has over 4 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**.

Example

How to use **flex** to align div elements side by side.

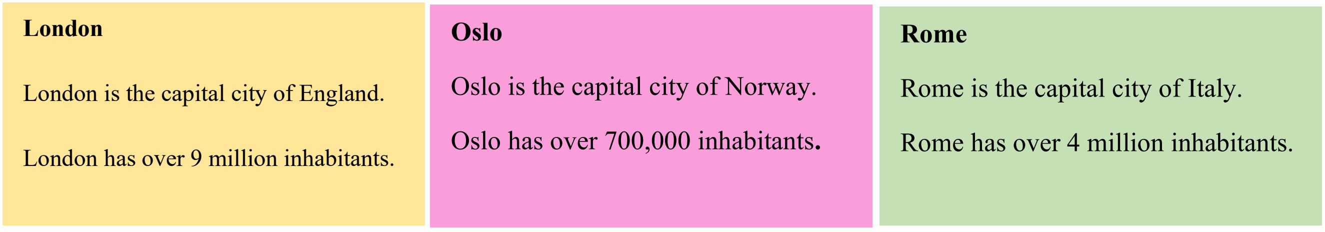
Input:

```
<style>
.my container {
  display: flex;
}
.my container > div {
  width:33%;
}
</style>
```

Output:

Flexbox Example

Align three DIV elements side by side.



London

London is the capital city of England.

London has over 9 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 700,000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has over 4 million inhabitants.

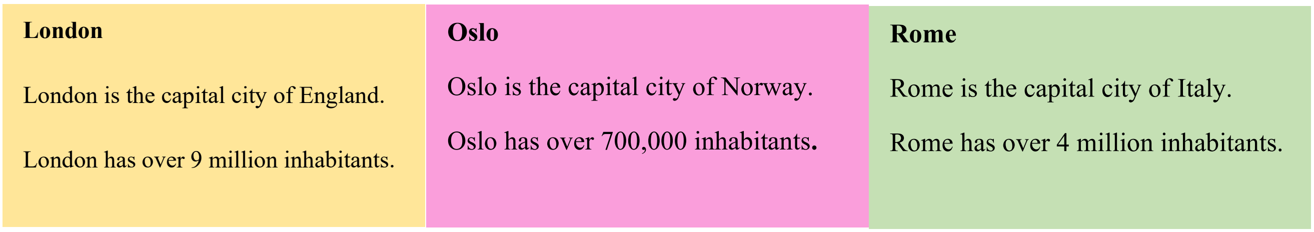
Grid

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.

Input: How to use grid to align <div> elements side by side:

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: 33% 33% 33%;
}
</style>
```

Output:



London

London is the capital city of England.

London has over 9 million inhabitants.

Oslo

Oslo is the capital city of Norway.

Oslo has over 700,000 inhabitants.

Rome

Rome is the capital city of Italy.

Rome has over 4 million inhabitants.

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**.**

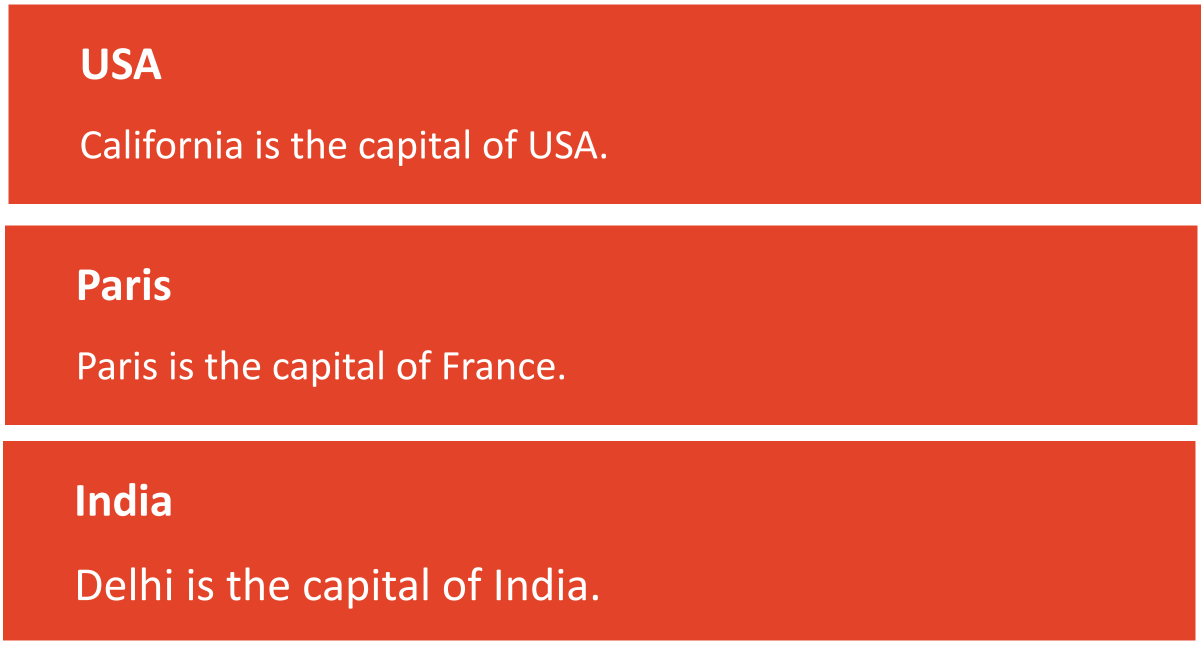
The **class** attribute is often used to point to a class name in a style sheet. It can also be used by a **JavaScript** to access and manipulate elements with the specific class name.

Input:

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

```
<div class="Task">
    <h2>USA</h2>
    <p>California is the capital of USA. </p>
</div>
<div class="Task">
  <h2>Paris</h2>
  <p>Paris is the capital of France. </p>
</div>
  <div class="Task">
  <h2>India</h2>
<p>Delhi is the capital of India. </p>
</div>
</body>
</html>
```

Output:



USA

California is the capital of USA.

Paris

Paris is the capital of France.

India

Delhi is the capital of India.

2. 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

<u>Input:</u>

```
<!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>
```

<u>Output:</u>

My Important Heading

This is some important text.

The Syntax For Class

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

EX:

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

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.

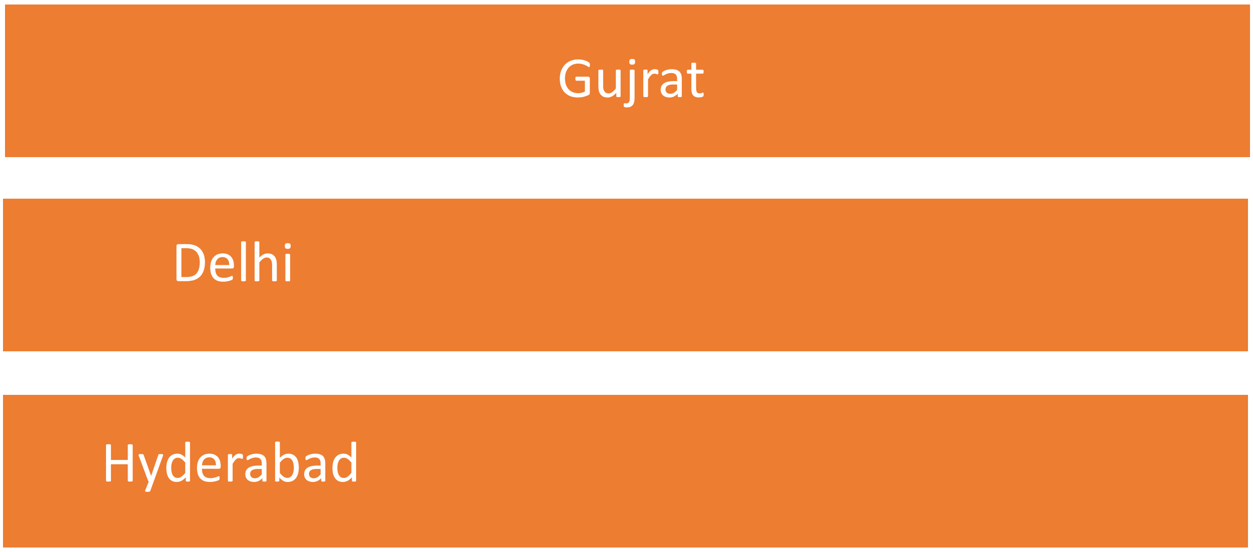
In the following example, the first **<h2>** element belongs to both the city class and also to the main class, and will get the CSS styles from both of the classes:

Example

```
<!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, Gujrat also belongs to the "main" class, which centre-
aligns the text. </p>


<h2 class="city main">Gujrat </h2>
```

```
<h2 class="city">Delhi</h2>

<h2 class="city">Hyderabad</h2>

</body>

</html>
```

<u>Output:</u>



```
Gujrat

Delhi

Hyderabad
```

<u>Different Elements Can Share Same Class</u>

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.

<u>Ex</u>:
```
<h2 class="city">Delhi</h2>
 <p class="city">Delhi is the capital of India <p>.
```

<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.

<u>Example:</u>

```
<!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">Delhi</h2>

<p>Delhi is the capital of India. </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>
```

Output:

Use of The class Attribute in JavaScript

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

Hide Elements

Delhi

Delhi is the capital of India.

Paris

Paris is the capital of France.

Tokyo

Tokyo is the capital of Japan.

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. 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.

Input:

```
<!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">Main Header</h1>

</body>

</html>
```

Output:

The id Attribute

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

Main Header

Difference Between Class and ID

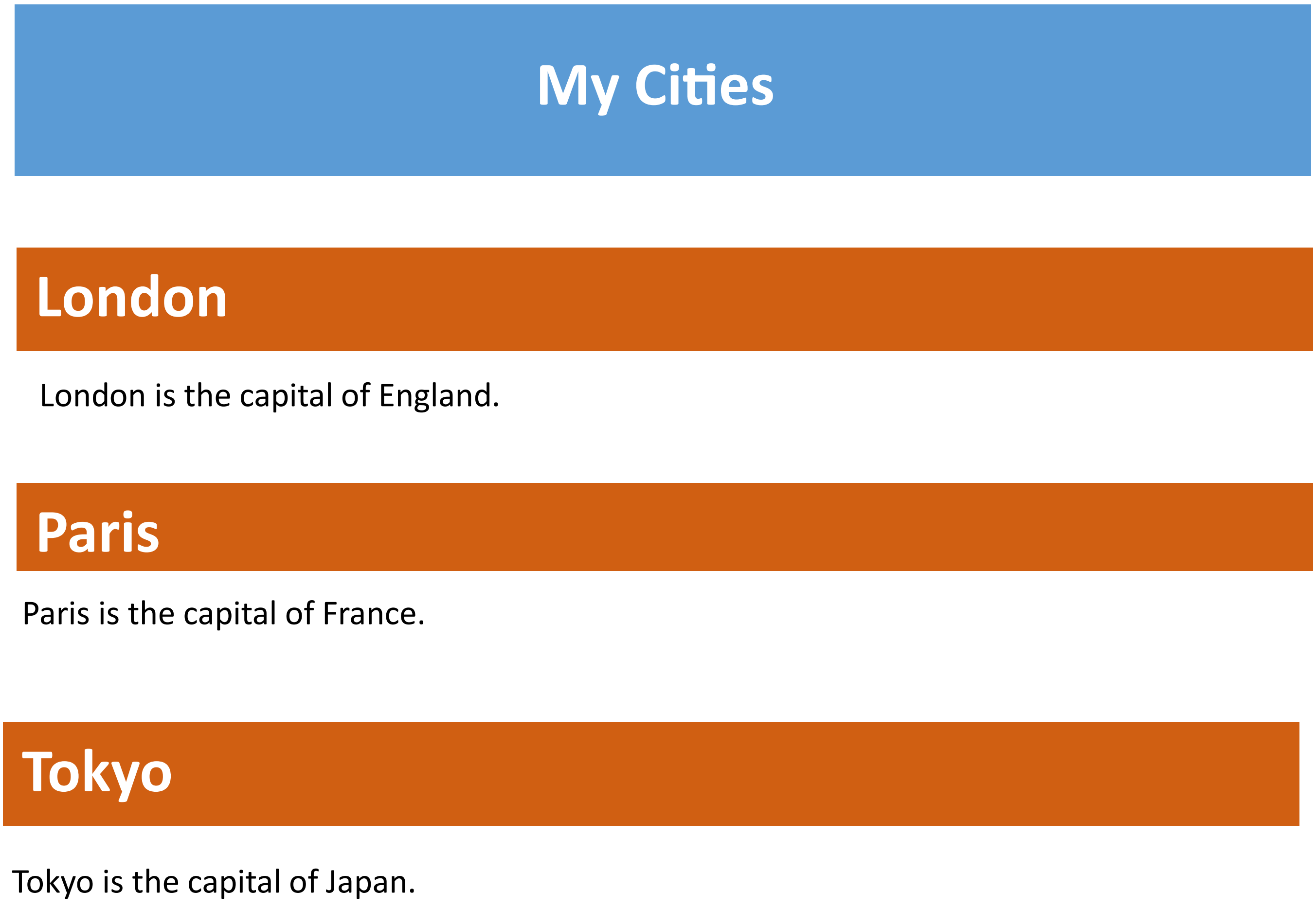
A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page.

<u>Input:</u>

```
<!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>
```

Output:



My Cities

London

London is the capital of England.

Paris

Paris is the capital of France.

Tokyo

Tokyo is the capital of Japan.

HTML Bookmarks with ID and Links

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

Example:

First, create a bookmark with the **id attribute.**

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

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

Input:

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

Output:

Jump to Chapter 4

Jump to Chapter 10

Chapter 1

This chapter explains ba bla bla

Chapter 2

This chapter explains ba bla bla

Chapter 3

This chapter explains ba bla bla

Chapter 4

This chapter explains ba bla bla

Chapter 5

This chapter explains ba bla bla

Chapter 6

This chapter explains ba bla bla

Chapter 7

This chapter explains ba bla bla

Chapter 8

This chapter explains ba bla bla

Chapter 9

This chapter explains ba bla bla

Chapter 10

This chapter explains ba bla bla

Chapter 11

This chapter explains ba bla bla

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.

Ex:

```
<!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>
```

Output:

Using The id Attribute in JavaScript

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

Hello World!

Change Text

HTML Iframes

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

HTML Iframe Syntax

The HTML <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

Syntax

```
<iframe src="url " title="description"></iframe>
```

Iframe - Set Height and Width

Use the **heigh**t and **width attributes** to specify the size of the iframe. The height and width are specified in pixels by default.

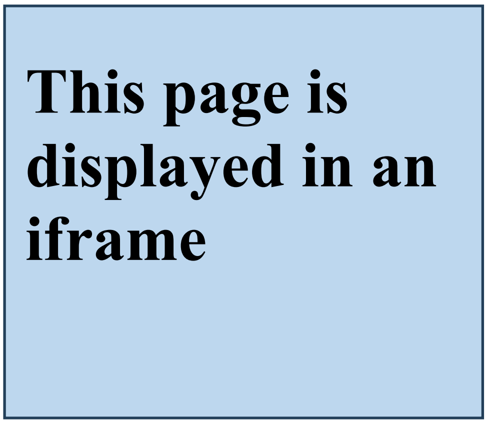
Input:

```
<!DOCTYPE html>

<html>

<body>

<h2>HTML Iframes</h2>

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

<iframe src="demo_iframe.html" height="200" width="300" title="Iframe Example"></iframe>

</body>

</html>
```

Output:

HTML Iframes

You can use the height and width attributes to specify the size of the iframe



This page is displayed in an iframe

Iframe - Remove the Border

By default, an iframe has a border around it. To remove the border, add the style attribute and use the CSS border property.

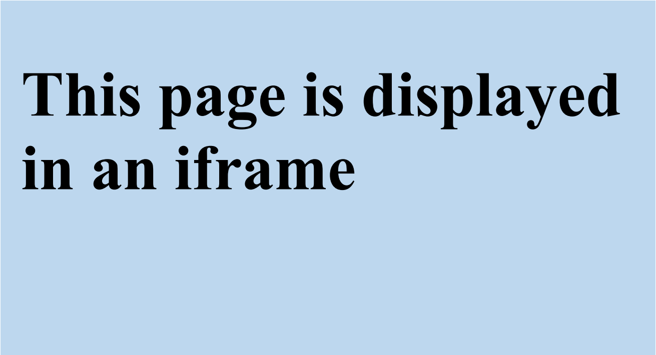
Input:

```
<!DOCTYPE html>

<html>

<body>

<h2>Remove the Iframe Border</h2>

<p>To remove the default border of the iframe, use CSS</p>

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


</body>

</html>
```

Output:

Remove the Iframe Border

To remove the default border of the iframe, use CSS.



This page is displayed in an iframe

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

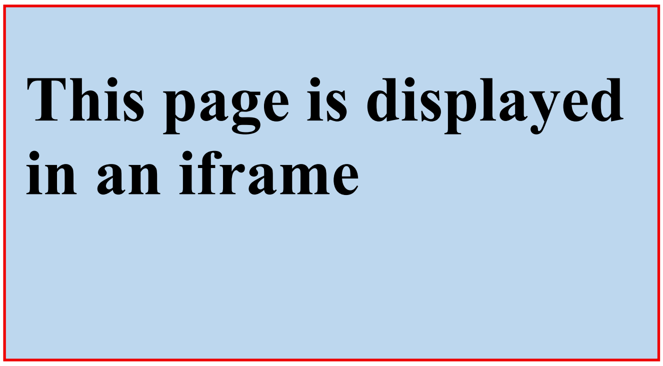
<u>Input:</u>

```
<!DOCTYPE html>

<html>

<body>

<h2>Custom Iframe Border</h2>

<p>With CSS, you can also change the size, style and color of the iframe's border. </p>

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


</body>

</html>
```

<u>Output:</u>

Custom Iframe Border

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



This page is displayed in an iframe

Iframe - Target for a Link

An iframe can be used as the target frame for a link.
The target attribute of the link must refer to the name attribute of the iframe.

Input:

```
<!DOCTYPE html>

<html>

<body>

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


<iframe src="demo_iframe.html" name="iframe_a"
height="300px" width="100%" title="Iframe
Example"></iframe>
```

```
<p><a href="https://www.youtube.com"
target="iframe_a">www.youtube.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>
```

Output:

Iframe - Target for a Link

This page is displayed in an iframe

youtube.com

When the target attribute of a link matches the name of an iframe, the link will open in the iframe.

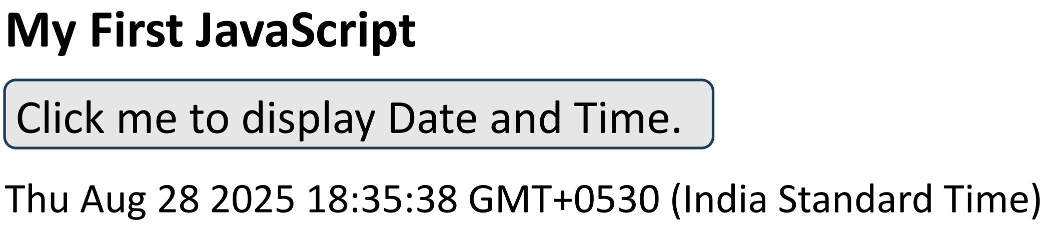
HTML JavaScript

JavaScript makes HTML pages more dynamic and interactive.

Ex:

```
<!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>
```

Output:



My First JavaScript

Click me to display Date and Time.

Thu Aug 28 2025 18:35:38 GMT+0530 (India Standard Time)

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".**

Input:

```
<!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>
```

Output:

Use JavaScript to Change Text

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

Hello JavaScript!

A Taste of JavaScript

Here are some examples of what JavaScript can do.

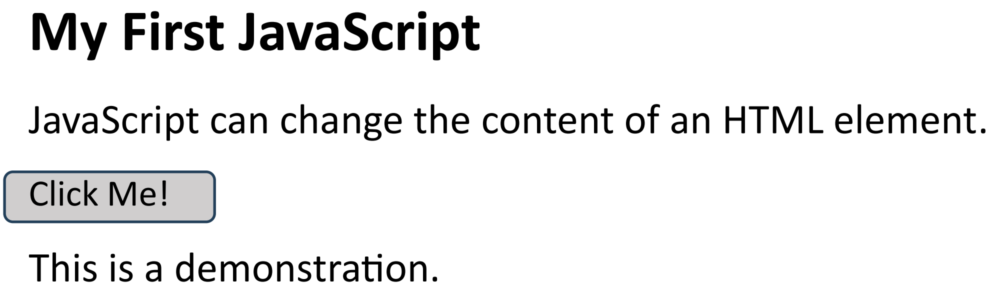
1.JavaScript can change content.

Input:

```
<!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>
```

Output:



My First JavaScript

JavaScript can change the content of an HTML element.

This is a demonstration.

2. JavaScript can change styles

Input:

```
<!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>
```

Output:

My First JavaScript

JavaScript can change the style of an HTML element.

Click on me

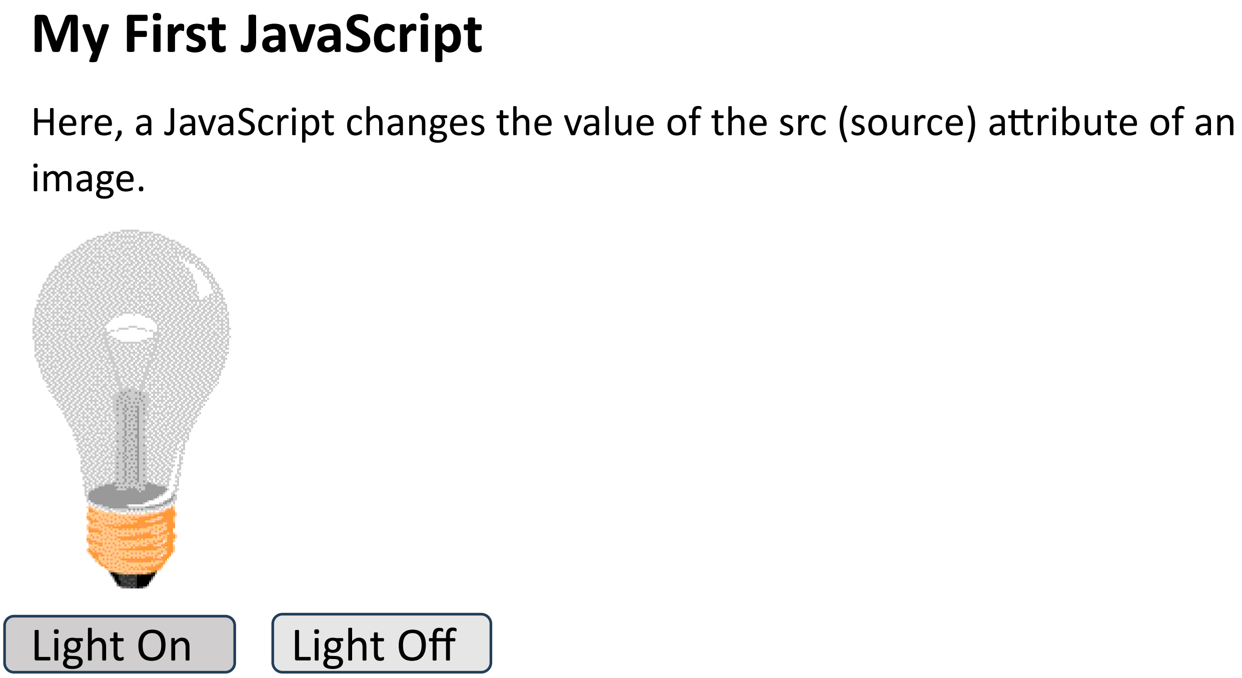
3. JavaScript can change attributes.

Input:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<p>Here, a 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>
```

Output:



My First JavaScript

Here, a JavaScript changes the value of the src (source) attribute of an image.

The HTML <noscript> Tag

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

<u>Input:</u>

```
<!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>
```

<u>Output:</u>

Hello JavaScript!

A browser without support for JavaScript will show the text written inside the noscript element.

<u>HTML File Paths</u>

File Path Examples

HTML File Paths

A file path describes the location of a file in a web site's folder structure. File paths are used when linking to external files, like.

- **Web pages**
- **Images**
- **Style sheets**
- **Java Scripts**

1.Absolute File Paths

1.An absolute file path is the full URL to a file.

Input:

```
<!DOCTYPE html>

<html>

<body>

<h2>Using a Full URL File Path</h2>

<img src="https://www.google.com/images/picture.jpg" alt="Mountain" style="width:300px">

</body>

</html>
```

Output:

```
Using a Full URL File Path
```

2.Relative File Paths

A relative file path points to a file relative to the current page. In the following example, the file path points to a file in the images folder located at the root of the current web.

Input:

```
<!DOCTYPE html>
<html>
<body>
<h2>Using a Relative File Path</h2>
<img src="/images/picture.jpg" alt="Mountain"
style="width:300px">
</body>
</html>
```

Output:

Using a Relative File Path

HTML - The Head Element

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

The HTML <head> Element

The **<head>** element is a container for metadata (data about data) and is placed between the **<html> tag** and the **<body> tag**. HTML metadata is data about the HTML document. Metadata is not displayed on the page. Metadata typically define the **document title, character set, styles, scripts, and other meta information**.

The HTML <title> Element

The **<title>** element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab. The **<title>** element is required in HTML documents!

The 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 favourites
- displays a title for the page in search engine-results

The HTML <style> Element

The **<style>** element is used to define style information for a single HTML page.

Input:

```
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
  <style>
    body {background-color: powder blue;}
    h1 {color: red;}
    p {color: blue;}
  </style>
</head>
<body>
```

The HTML <link> Element

The **<link>** element defines the relationship between the current document and an external resource. The **<link>** tag is most often used to link to external style sheets.

Ex:

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

The HTML <meta> Element

The **<meta>** element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings. The metadata will not be displayed on the page, but is used by browsers **(how to display content or reload page), by search engines (keywords), and other web services.**

Ex:

1.Define the character set used.

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

2.Define keywords for search engines.

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

3.Define a description of your web page.

<meta name="description**" content="**Html tutorials**">**

4.Define the author of a page.

```
<meta name="author" content="John Doe">
```

5.Refresh document every 30 seconds.

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

6.Setting the viewport to make your website look good on all devices.

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

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.

Ex:

```
<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.

The HTML <script> Element

The **<script>** element is used to define client-side Java Scripts. The following JavaScript writes "Hello JavaScript!" into an HTML element with **id=**"demo":

Input:

```
<!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>
```

Output:

My Web Page

A Paragraph

Try it

The HTML <base> Element

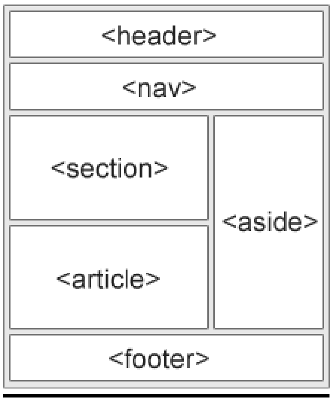
The **<base>** element specifies the base URL and/or target for all relative URLs in a page. The **<base>** tag must have either an **href** or a target attribute present, or both. There can only be one single **<base>** element in a document!

HTML Layout Elements and Techniques

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

HTML Layout Elements

HTML has several semantic elements that define the different parts of a web page:



- **<header>** - Defines a header for a document or a section
- **<nav>** - Defines a set of navigation links
- **<section>** - Defines a section in a document
- **<article>** - Defines 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.

HTML Layout Techniques

There are four different techniques to create multicolumn layouts. Each technique has its pros and cons.

- **CSS framework**
- **CSS float property**
- **CSS flexbox**
- **CSS grid**

1.CSS Frameworks

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

2.CSS Float Layout

It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work. Disadvantages: Floating elements are tied to the document flow, which may harm the flexibility.

3.CSS Flexbox Layout

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

4. 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).

Setting The Viewport

To create a responsive website, add the following **<meta>** tag to all your web pages.

Ex: <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.

Responsive Images

Responsive images are images that scale nicely to fit any browser size. Using the **width Property**

If the CSS **width property** is set to **100%**, the image will be responsive and scale up and down.

Ex: <img src="img_girl.jpg**" style="**width:100%;**">**

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.

Ex:

```
<img src="img_girl.jpg" style="maxwidth:100%;height:auto;">
```

Show Different Images Depending on Browser Width

The HTML **<picture>** element allows you to define different images for different browser window sizes. Resize the browser window to see how the image below changes depending on the width.

Input:

```
<!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>
```

Output:

Show Different Images Depending on Browser Width

Resize the browser width and the image will change at 600px and 1500px.

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.

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

Hello World

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

Media Queries

In addition to resize text and images, it is also common to use media queries in responsive web pages. With media queries you can define completely different styles for different browser sizes.

Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stack vertically on small screens.

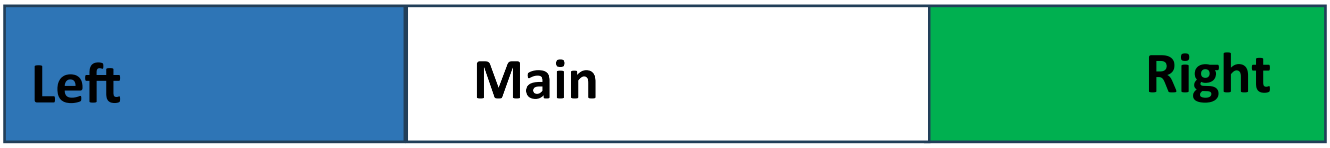
Input:

```
<!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 break point at 800px: */

@media screen and (max-width: 800px) {
  . left, . main, .right {
    width: 100%;
  }
}
</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>
```

Output:



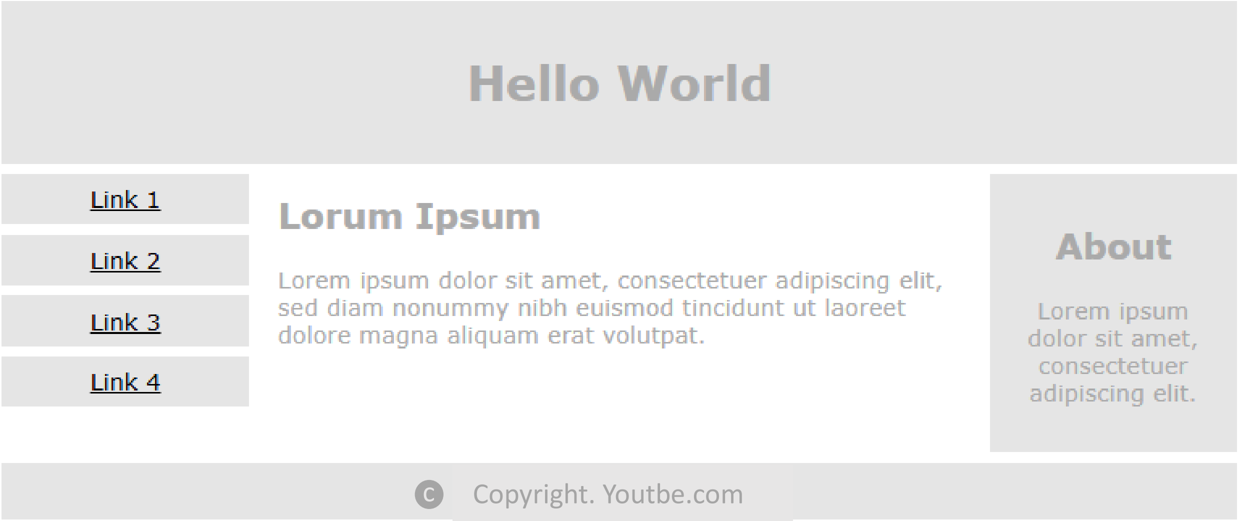
Responsive Web Page - Full Example

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

Ex:

```
<!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 Youtube.com</div>
</body>
</html>
```

Output:



Hello World

Link 1

Link 2

Link 3

Link 4

Lorum Ipsum

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

About

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Ⓒ Copyright. Youtbe.com

HTML Computer Code Elements

HTML contains several elements for defining user input and computer code.

Input:

```
<!DOCTYPE html>
<html>
<body>
<h2>Computer Code</h2>
<p>Some programming code</p>
<code>
x = 5;
y = 6;
z = x + y;
</code>
</body>
</html>
```

Output:

Computer Code

Some programming code

x = 5; y = 6; z = x + y;

HTML <kbd> For Keyboard Input

The HTML **<kbd>** element is used to define keyboard input. The content inside is displayed in the browser's default monospace font.

Example

1.Define some text as keyboard input in a document.

Input:

```
<!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>
```

Output:

```
The kbd Element

The kbd element is used to define keyboard input

Save the document by pressing Ctrl + S
```

2.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**.

Input:

```
<!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>
```

Output:

The samp Element

The samp element is used to define sample output from a computer program.

Message from my computer

File not found.
Press F1 to continue

3. 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**.

Input:

```
<!DOCTYPE html>

<html>

<body>

<h2>The code Element</h2>

<p>Programming code example</p>

<code>

x = 5;

y = 6;

z = x + y;

</code>

</body>

</html>
```

Output:

```
The code Element

Programming code example:

x = 5; y = 6; z = x + y;
```

Preserve Line-Breaks

Notice that the **<code>** element does NOT preserve extra whitespace and line-breaks. To preserve extra whitespace and line-breaks, you can put the **<code>** element inside a **<pre>** element.

Input:

```
<!DOCTYPE html>

<html>

<body>

<p>The code element does not preserve whitespace and line-breaks. </p>

<p>To fix this, you can put the code element inside a pre element. </p>

<pre>

<code>

x = 5;

y = 6;

z = x + y;

</code>

</pre>

</body>

</html>
```

Output:

The code element does not preserve whitespace and line-breaks.

To fix this, you can put the code element inside a pre element.

x = 5;

y = 6;

z = x + y;

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.

Input:

```
<!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>
```

<u>Output:</u>

The var Element

The area of a triangle is: *1/2* x *b* x *h*, where *b* is the base, and *h* is the vertical height.

<u>HTML Semantic Elements</u>

Semantic elements = elements with a meaning.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

<u>Examples</u>: of **non-semantic** elements: **<div> and <span>** - Tells nothing about its content.

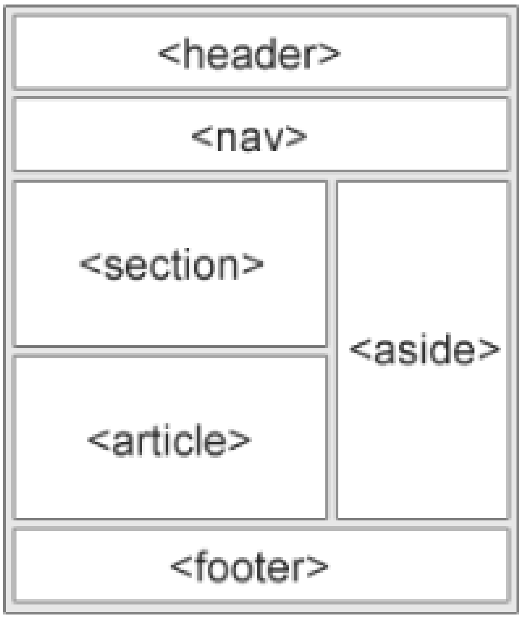
<u>Examples:</u> of **semantic** elements: **<img>, <table>, and <article>** - Clearly defines its content.

<u>Semantic Elements in HTML</u>

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 several semantic elements that can be used to define different parts of a web page.

- **<article>**
- **<aside>**
- **<details>**
- **<fig caption>**
- **<figure>**
- **<footer>**
- **<header>**
- **<main>**
- **<mark>**
- **<nav>**
- **<section>**
- **<summary>**
- **<time>**



HTML <section> Element

The **<section>** element defines a section in a document.

Examples of where a **<section>** element can be used:

- **Chapters**
- **Introduction**
- **News items**
- **Contact information**

Input:

```
<!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>
```

Output:

WWF

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.

WWF's Panda symbol

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.

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**

<u>Input:</u>

```
<!DOCTYPE html>
<html>
<body>
<h1>The article element</h1>
<article>
  <h2>Google Chrome</h2>
  <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today! </p>
</article>
<article>
  <h2>Mozilla Firefox</h2>
  <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018. </p>
</article>
<article>
  <h2>Microsoft Edge</h2>
  <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer. </p>
</article>
```

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

Output:

The article element

Google Chrome

Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!

Mozilla Firefox

Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.

Microsoft Edge

Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.

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.

HTML <footer> Element

The **<footer>** element defines a footer for a document or section. A **<footer>** element typically contains.

- **authorship information**
- **copyright information**
- **contact information**
- **sitemap**
- **back to top links**
- **related documents**

You can have several **<footer>** elements in one document.

Input:

```
<!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>
```

Output:

Author: Hege Refsnes

hege@example.com

HTML <nav> Element

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

Input:

```
<!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>
```

Output:

HTML | CSS | JavaScript | jQuery

HTML <aside> Element

The **<aside>** element defines some content aside from the content it is placed in (like a sidebar). The **<aside>** content should be indirectly related to the surrounding content.

Ex:

```
<aside>
<h4>Epcot Center</h4>
<p>Epcot is a theme park at Walt Disney World Resort featuring
exciting attractions, international pavilions, award-winning
fireworks and seasonal special events. </p>
</aside>
```

HTML <figure> and <figcaption> Elements

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

The **<img>** element defines the actual image/illustration.

Input:

```
<!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>
```

Output:

Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.

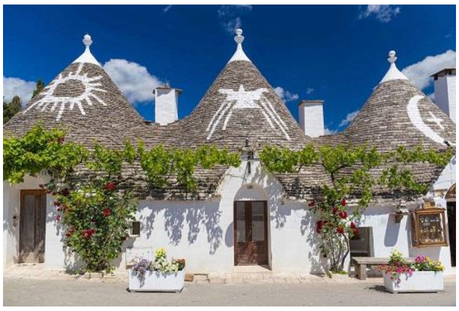


Fig.1 - Trulli, Puglia, Italy.

Semantic Elements in HTML

Tag	Description
<article>	Defines independent, self-contained content
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

HTML Style Guide

Consistent, clean, and tidy HTML code makes it easier for others to read and understand your code. Here are some guidelines and tips for creating good HTML code.

Always Declare Document Type

Always declare the document type as the first line in your document. The correct document type for HTML is

<!DOCTYPE html>

Use Lowercase Element Names

HTML allows mixing uppercase and lowercase letters in element names. However, we recommend using lowercase element names, because.

- **Mixing uppercase and lowercase names looks bad.**
- **Developers normally use lowercase names.**
- **Lowercase looks cleaner.**
- **Lowercase is easier to type.**

Close All HTML Elements

In HTML, you do not have to close all elements (for example the <p> element). However, we strongly recommend closing all HTML elements, like this.

Use Lowercase Attribute Names

HTML allows mixing uppercase and lowercase letters in attribute names. However, we recommend using lowercase attribute names, because:

- **Mixing uppercase and lowercase names looks bad**
- **Developers normally use lowercase names**
- **Lowercase looks cleaner**
- **Lowercase is easier to type**

Always Quote Attribute Values

HTML allows attribute values without quotes. However, we recommend quoting attribute values, because.

- **Developers normally quote attribute values**
- **Quoted values are easier to read**
- **You MUST use quotes if the value contains spaces**

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.

Spaces and Equal Signs

HTML allows spaces around equal signs. But space-less is easier to read and groups entities better together.

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.

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**

Omitting <html> and <body>?

An HTML page will validate without the **<html>** and **<body>** tags.

However, we strongly recommend to always add
the **<html>** and **<body>** tags!

Omitting **<body>** can produce errors in older browsers.

Omitting **<html>** and **<body>** can also crash DOM and XML software.

Omitting <head>?

The HTML **<head>** tag can also be omitted. Browsers will add all elements before <body>, to a default **<head>** element.

Close Empty HTML Elements?

In HTML, it is optional to close empty elements. 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.

Meta Data

To ensure proper interpretation and correct search engine indexing, both the language and the character encoding **<meta charset="*UTF-8*">** should be defined as early as possible in an HTML document.

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. 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">
```

- **Place the opening bracket on the same line as the selector**
- **Use one space before the opening bracket**
- **Use two spaces of indentation**
- **Use semicolon after each property-value pair, including the last**
- **Only use quotes around values if the value contains spaces**
- **Place the closing bracket on a new line, without leading spaces**

Loading JavaScript in HTML

Use simple syntax for loading external scripts
(the type attribute is not necessary).

```
<script src="myscript.js">
```

Accessing HTML Elements with JavaScript

Using "untidy" HTML code can result in JavaScript errors. These two JavaScript statements will produce different results.

Example:

```
getElementById("Demo").innerHTML = "Hello";

getElementById("demo").innerHTML = "Hello";
```

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.youtube.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.