Frontend Programming

Frontend programming involves creating the user interface and user experience elements of a website or web application. This includes everything that users interact with directly in their web browser, such as layout, design, and interactivity. The primary technologies used in frontend development are HTML, CSS, and JavaScript, with frameworks and libraries like Bootstrap enhancing these core technologies.

In the context of embedded full-stack IoT (Internet of Things) development, frontend programming plays a crucial role in creating user interfaces that allow users to interact with IoT devices. These interfaces can range from simple web dashboards to complex web applications that monitor and control IoT devices. Key technologies used include HTML, CSS, JavaScript, and various frameworks and libraries that facilitate the development of responsive, user-friendly interfaces.

In the realm of embedded full-stack IoT applications, creating an effective and engaging user interface is crucial for monitoring and controlling IoT devices. The frontend development process leverages a combination of HTML, CSS, JavaScript, and Bootstrap to achieve this goal.

- **HTML (HyperText Markup Language)** provides the foundational structure of the web page, defining the layout and elements such as headings, paragraphs, buttons, forms, and data displays that users interact with.
- **CSS (Cascading Style Sheets)** is used to style these HTML elements, ensuring that the web page is visually appealing and user-friendly. CSS also handles responsive design, making sure the interface works seamlessly across different devices, from desktops to mobile phones.
- **JavaScript** adds interactivity and dynamic functionality to the web page. It allows for real-time updates, user input handling, and communication with backend systems through APIs or WebSockets, essential for the dynamic control and monitoring of IoT devices.
- **Bootstrap** is a popular front-end framework that offers pre-designed components and a robust grid system. It streamlines the development process, enabling developers to create modern, responsive web pages quickly and efficiently.

By combining these technologies, developers can create sophisticated, interactive, and responsive user interfaces that enhance the usability and effectiveness of IoT applications. This integration ensures that the web pages not only look good but also provide seamless communication with backend systems, enabling efficient management and control of IoT devices.

HTML

HTML (HyperText Markup Language) is the most widely used language for creating webpages and serves as the fundamental building block of the Web. It used to structure and organize the content on a web page. It uses various tags to define the different elements on a page, such as headings, paragraphs, and links.

What is HTML?

HTML stands for HyperText Markup Language. It is a markup language, not a programming language, which means it is used to "mark up" a text document with tags that tell a web browser how to structure and display the content.

Key Concepts of HTML

HyperText : HyperText refers to the way web pages (HTML documents) are linked together. The links available on a webpage are called Hypertext. These links allow users to navigate to different parts of the same web page or to different web pages altogether.

Markup Language : A markup language is a computer language used to add structure and formatting to a text document. Markup languages use a system of tags to define the structure and content of a document. These tags are interpreted by web browsers to display the document in a specific way.

HTML Features

HTML is a text-based language with several powerful features that make it essential for creating web pages:

- **Standard Language:** HTML is the standard language used for creating and structuring web pages. It organizes content using elements such as headings, paragraphs, lists, and tables.
- **Media Support:** HTML supports a wide range of media types, including text, images, audio, and video, making web pages more engaging and interactive.
- **Flexibility:** HTML can be used with other technologies, such as CSS (Cascading Style Sheets) and JavaScript, to add additional features and functionality to a web page.

- **Cross-Browser Compatibility:** HTML is compatible with all web browsers, ensuring that web pages are displayed consistently across a variety of platforms and devices.
- **Open and Standardized:** HTML is an open and standardized language, continuously updated and improved by a community of developers and experts.

HTML provides the essential framework for building web pages, offering a structured way to present content and integrate multimedia. By understanding and utilizing HTML, developers can create robust and accessible web pages that are the cornerstone of modern web development.

HTML History

HTML has evolved significantly since the early days of the World Wide Web. Here are key milestones in its development:

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

Basic Structure of HTML Document

```
<!DOCTYPE html>
<html>
<head>
<title>This is document title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>Document content goes here.....</p>
</body>
```

To check the result of this HTML code, or let's save it in an HTML file test.htm using your favourite text editor. Finally open it using a web browser like Internet Explorer or Google Chrome, or Firefox etc. It must show the following output:

HTML elements are hierarchical, which means that they can be nested inside each other to create a tree-like structure of the content on the web page.

This hierarchical structure is called the DOM (Document Object Model), and it is used by the web browser to render the web page.

In this example, the html element is the root element of the hierarchy and contains two child elements: head and body. The head element, in turn, contains a child element called the title, and the body element contains child elements: h1 and p.

HTML Tags

HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing tag**</html>** and **<body>** tag has its closing tag **</body>** tag etc.

In above example of HTML document uses the following tags

- **<!DOCTYPE html>:** This declaration defines the document type and version of HTML being used. In this case, it specifies that the document is an HTML5 document.
- **<html>:** This is the root element of an HTML document. All other HTML elements are contained within this tag.
- **<head>:** The head element contains meta-information about the HTML document, such as its title, character set, styles, scripts, and other meta information.
- **<title>This is document title</title>:** The title element sets the title of the HTML document, which is displayed in the browser's title bar or tab. In this case, the title of the document is "This is document title".
- **<body>:** The body element contains the content of the HTML document that is visible to users. All visible HTML elements such as text, images, links, tables, etc., are placed inside the body element.
- **<h1>This is a heading</h1>:** The h1 element defines a top-level heading. Headings are used to define sections of content. In this case, the heading text is "This is a heading".
- ** Document content goes here..... :** The p element defines a paragraph. Paragraphs are used to group blocks of text. In this case, the paragraph text is "Document content goes here.....".

HTML Document Structure

```
Document declaration tag

<html>

<head>

Document header related tags

</head>
```

The <!DOCTYPE> Declaration

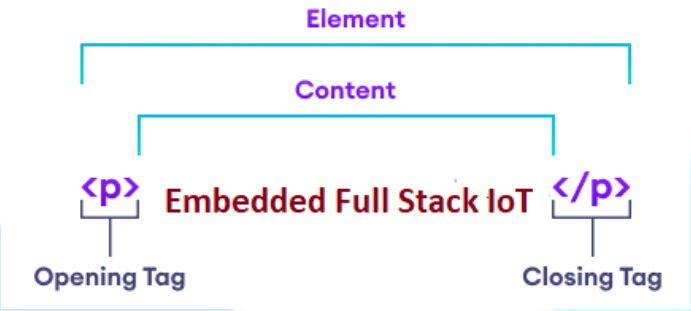
The <!DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration:

```
<! DOCTYPE html>
```

There are many other declaration types which can be used in HTML document depending on what version of HTML is being used. We will see more details on this while discussing <!DOCTYPE...> tag along with other HTML tags.

HTML elements

HTML elements consist of several parts, including the opening and closing tags, the content, and the attributes.



Here,

- **The opening tag:** This consists of the element name, wrapped in angle brackets. It indicates the start of the element and the point at which the element's effects begin.
- **The closing tag:** This is the same as the opening tag, but with a forward slash before the element name. It indicates the end of the element and the point at which the element's effects stop.
- **The content:** This is the content of the element, which can be text, other elements, or a combination of both.
- **The element:** The opening tag, the closing tag, and the content together make up the element.

HTML (HyperText Markup Language) provides a variety of elements to structure and present content on the web. Here's a list of some common HTML elements:

1. **Text Formatting:**
    - <h1> to <h6>: Headings (from highest importance to lowest).
    - <p>: Paragraph.
    - : Strong emphasis.
    - : Emphasis.
    - : Generic inline container.
    -
: Line break.
    - <hr>: Horizontal rule.
2. **Lists:**
    - : Unordered list.
    - : Ordered list.
    - : List item.
    - <dl>: Description list.
    - <dt>: Description term.
    - <dd>: Description details.
3. **Links and Anchors:**
    - <a>: Anchor.
    - <link>: External resource link (typically used for stylesheets).
    - <nav>: Navigation links container.
4. **Images and Multimedia:**
    - <img>: Image.
    - <audio>: Audio player.
    - <video>: Video player.
5. **Tables:**
    - <table>: Table container.
    - <tr>: Table row.
    - <th>: Table header cell.
    - <td>: Table data cell.
    - <caption>: Table caption.

6. **Forms:**

- <form>: Form container.
- <input>: Input control.
- <textarea>: Text input area.
- <select>: Dropdown selection.
- <button>: Button.
- <label>: Form field label.
- <fieldset>: Group of form controls.
- <legend>: Title for <fieldset>.

7. **Sections and Grouping:**

- : Generic container.
- <section>: Section of a document.
- <header>: Header section.
- <footer>: Footer section.
- <main>: Main content area.
- <article>: Article or independent content.
- <aside>: Sidebar content.
- <nav>: Navigation links.

8. **Semantic Elements:**

- <header>: Defines a header for a document or a section.
- <footer>: Defines a footer for a document or a section.
- <main>: Defines the main content of a document.
- <article>: Defines an article.
- <section>: Defines a section in a document.
- <aside>: Defines content aside from the content (like a sidebar).
- <details>: Defines additional details that the user can view or hide.
- <summary>: Defines a heading for the <details> element.

9. **Meta Information:**

- <meta>: Metadata that provides information about the HTML document.
- <title>: Title of the document (displayed in the browser's title bar).

Self Closing Tags

Self-closing tags, also known as void elements or empty elements, are HTML elements that do not require closing tags because they don't contain any content. Instead, they self-terminate with a trailing slash (/) immediately before the closing angle bracket (>).

Here's an explanation of self-closing tags with some examples:

1. **Image Tag (`<img>`):**
    - The <img> tag is used to embed images in an HTML document.
    - It doesn't contain any content, so it's self-closing.
    - Example:
2. **Line Break Tag (`<br>`):**
    - The
 tag inserts a single line break.
    - It doesn't require a closing tag since it doesn't contain any content.
    - Example: First Line
Second Line
3. **Horizontal Rule Tag (<hr>):**
    - The <hr> tag inserts a horizontal rule to separate content.
    - Like
, it doesn't contain content and is self-closing.
    - Example: <hr>
4. **Input Tag (<input>):**
    - The <input> tag creates form controls like text fields, checkboxes, radio buttons, etc.
    - It doesn't contain content and is self-closing.
    - Example: <input type="text" name="username" />
5. **Meta Tag (<meta>):**
    - The <meta> tag provides metadata about the HTML document.
    - It's self-closing and commonly used to specify character encoding, viewport settings, etc.
    - Example: <meta charset="UTF-8" />
6. **Embedded Content Tags (<audio>, <video>, <source>):**
    - Tags like <audio>, <video>, and <source> for media embedding are also self-closing.
    - Example: <audio controls><source src="audio.mp3" /></audio>

Self-closing tags make HTML cleaner and more concise, especially for elements that don't contain content. They're essential for adhering to HTML standards and ensuring compatibility across different browsers and platforms.

Classifications of HTML Elements

In HTML, elements are categorized as either inline or block elements. Understanding the difference between these types is essential for effective web design and layout.

Block Elements

Block elements are typically used to create the structure of a webpage. They start on a new line and take up the full width available (stretching out to the left and right as far as they can).

Characteristics of Block Elements:

- Always start on a new line.
- Take up the full width available by default.
- Can contain other block elements and inline elements.
- Examples include: , <p>, <h1> through <h6>, <ul>, <ol>, <li>, <header>, <footer>, <section>, <article>, and more.

Inline Elements

Inline elements do not start on a new line and only take up as much width as necessary. They are typically used for smaller pieces of content within block elements.

Characteristics of Inline Elements:

- Do not start on a new line.
- Only take up as much width as necessary.
- Cannot contain block elements but can contain other inline elements and text.
- Examples include: , <a>, <img>, , ,
, <i>, <b>, and more.

Differences Between Block and Inline Elements

- **Block Elements**: Take up the full width available and start on a new line. They can contain both block and inline elements.
- **Inline Elements**: Take up only as much width as necessary and do not start on a new line. They can contain other inline elements and text but not block elements.

HTML Attributes

HTML elements can have attributes, which provide additional information about the element. They are specified in the opening tag of the element and take the form of name-value pairs.

Example:

```
<a href="http://example.com"> Example </a>
```

Here,

The href is an attribute. It provides the link information about the <a> tag. In the above example,

- href - the name of attribute
- https://www.programiz.com - the value of attribute

HTML attributes are mostly optional.

The various HTML attributes along with their descriptions

Attribute	Description	Example
Global Attributes		
id	Specifies a unique id for an HTML element.	Header Section
class	Specifies one or more class names for an element (used for CSS and JavaScript).	This is an introductory paragraph.
style	Specifies inline CSS styles for an element.	Blue Heading
title	Provides extra information about an element (displayed as a tooltip).	<abbr title="HyperText Markup Language">HTML</abbr>
data-*	Custom data attributes used to store private data.	User Data
Event Attributes		
onclick	Script to be run when an element is clicked.	<button onclick="alert('Button clicked!')">Click Me</button>
onmouseover	Script to be run when the mouse pointer is moved over an element.	Hover over this text.

Attribute	Description	Example
onmouseout	Script to be run when the mouse pointer is moved out of an element.	Move the mouse away from this text.
Anchor (a) Attributes		
href	Specifies the URL of the page the link goes to.	Visit Example
target	Specifies where to open the linked document.	Open in New Tab
Image (img) Attributes		
src	Specifies the path to the image.	<img src="image.jpg" alt="Sample Image">
alt	Provides alternative text for an image if it cannot be displayed.	<img src="image.jpg" alt="A beautiful scenery">
width	Specifies the width of the image.	<img src="image.jpg" alt="Sample Image" width="500">
height	Specifies the height of the image.	<img src="image.jpg" alt="Sample Image" height="600">
Input (input) Attributes		
type	Specifies the type of input element.	<input type="text" placeholder="Enter your name">
placeholder	Provides a short hint that describes the expected value of the input field.	<input type="text" placeholder="Enter your name">
value	Specifies the initial value of the input field.	<input type="text" value="John Doe">
name	Specifies the name of the input element.	<input type="text" name="username">
required	Specifies that an input field must be filled out before submitting the form.	<input type="email" required>
Form (form) Attributes		
action	Specifies where to send the form-data when a form is submitted.	<form action="/submit-form" method="post">

Attribute	Description	Example
method	Specifies the HTTP method to be used when sending form-data.	<form action="/submit-form" method="post">
Table (table) Attributes		
border	Specifies whether the table cells should have borders.	<table border="1"> <tr> <th>Header</th> <th>Header</th> </tr> <tr> <td>Data</td> <td>Data</td> </tr> </table>

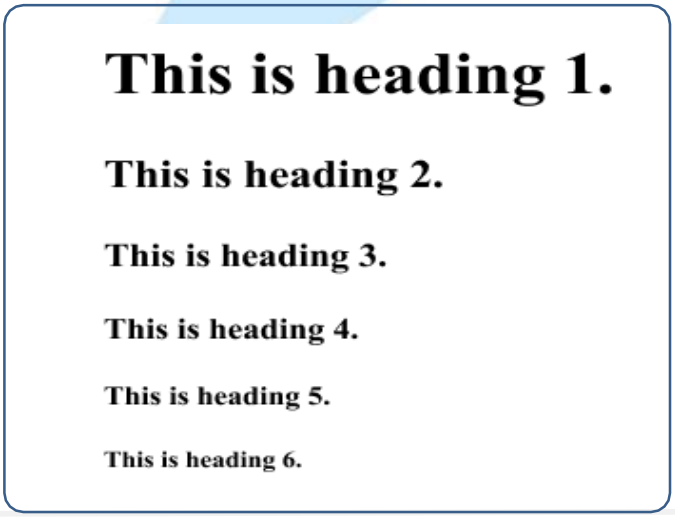
HTML Headings

The HTML heading tags (<h1> to <h6>) are used to add headings to a webpage.

For example,

```
<h1>This is heading 1.</h1>
<h2>This is heading 2.</h2>
<h3>This is heading 3.</h3>
<h4>This is heading 4.</h4>
<h5>This is heading 5.</h5>
<h6>This is heading 6.</h6>
```

The browser output is



In the example, we have used tags <h1> to <h6> to create headings of varying sizes and importance.

The <h1> tag denotes the most important heading on a webpage. Similarly, <h6> denotes the least important heading.

The difference in sizes of heading tags comes from the browser's default styling. And, you can always change the styling of heading tags, including font size, using CSS.

Note: Do not use heading tags to create large texts. It's because search engines like Google use heading tags to understand what a page is about.

h1 Tag is Important

The HTML <h1> tag defines the most important heading in a webpage.

Although it's possible to include multiple <h1> tags in a webpage, the general practice is to use a single <h1> tag (usually at the beginning of the page).

The <h1> tag is also important for SEO. Search engines (such as Google) treat content inside <h1> with greater importance compared to other tags.

Headings are block-level elements

The Headings <h1> to <h6> tags are block-level elements. They start on a new line and take up the full width of their parent element.

For example,

```
<h1>Headings</h1> <h2>are</h2> <h3>fun!</h3>
```

The browsers output

Headings

are

fun!

HTML Paragraphs

The HTML tag is used to create paragraphs.

For example,

```
<p>HTML is fun to learn.</p>
```

Browser Output

HTML is fun to learn.

A paragraph starts with the and ends with the tag. HTML paragraphs always start on a new line.

```
<p>Paragraph 1: Short Paragraph</p>

<p>Paragraph 2: Long Paragraph, this is a long paragraph with more text to fill an entire line in the website.</p>
```

Browser Output

Paragraph 1: Short Paragraph
Paragraph 2: Long Paragraph, this is a long paragraph with more text to fill an entire line in the website.

HTML Paragraphs and Spaces

Paragraphs automatically remove extra spaces and lines from our text.

For example,

```
<p>
The paragraph tag removes      all extra spaces.


The paragraph tag also removes all extra lines.
</p>
```

Browser Output

The paragraph tag removes all extra spaces. The paragraph tag also removes all extra lines.

Here, the output

- remove all the extra spaces between words
- remove extra lines between sentences

Adding Line Breaks in Paragraphs

We can use the HTML line break tag,
, to add line breaks within our paragraphs.

For example,

```
<p>We can use the <br> HTML br tag <br> to add a line break.</p>
```

Browser Output

We can use the
HTML br tag
to add a line break without creating a new paragraph

Note: The
 tag does not need a closing tag like the <p> tag.

Pre-formatted Text in HTML

The paragraphs cannot preserve extra lines and space. If we need to create content that uses multiple spaces and lines, we can use the HTML <pre> tag.

The <pre> tag creates preformatted text. Preformatted texts are displayed as written in the HTML file. For example,

```
<pre>
 This text
  will be
  displayed


   in the same manner
    as it is written
</pre>
```

Browser Output

```
This text
  will be
    displayed

   in the same manner
     as it is written
```

Other Elements Inside Paragraphs

It's possible to include one HTML element inside another. This is also true for <p> tags. For example,

```
<p>
We can use other tags like <strong>the strong tag to emphasize text</strong>
</p>
```

Browser Output

We can use other tags like **the strong tag to emphasize text**

In the above example, we have used the tag inside the <p> tag. Browsers automatically make the contents inside tags bold.

Paragraph is Block-level

The <p> tag is a block-level element. It starts on a new line and takes up the full width (of its parent element).

Add Extra Space Inside Paragraphs

As discussed earlier, we cannot normally add extra empty spaces inside paragraphs. However, we can use a certain HTML entity called non-breaking space to add extra spaces.

For example,

```
<p>Extra space    inside a paragraph</p>
```

Browser Output

Extra space inside a paragraph

Here,   is an HTML entity, which is interpreted as a space by browsers. This allows us to create multiple spaces inside paragraphs and other HTML tags.

HTML Formatting Elements

Formatting elements were designed to display special types of text:

- **<b> - Bold text :**
  The HTML <b> element defines bold text, without any extra importance.
  For example:

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

- ** - Important text :**
  The HTML element defines text with strong importance. The content inside is typically displayed in bold.
  For example:

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

- **<i> - Italic text :**
  The HTML <i> element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.
  For example:

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

- ** - Emphasized text :**
  The HTML element defines emphasized text. The content inside is typically displayed in italic.
  For example:

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

- **<mark> - Marked text** :
  The HTML <mark> element defines text that should be marked or highlighted
  For example:

```
<p>This is <mark>highlighted</mark> text.</p>
```

- **<small> - Smaller text :**
  The HTML <small> element defines smaller text
  For example:

```
<p>This is <small>smaller</small> text.</p>
```

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

```
<p>This is <del>deleted</del> text.</p>
```

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

```
<p>This is <ins>inserted</ins> text.</p>
```

- **<sub> - Subscript text :**
  The HTML <sub> element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H2O
  For example:

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

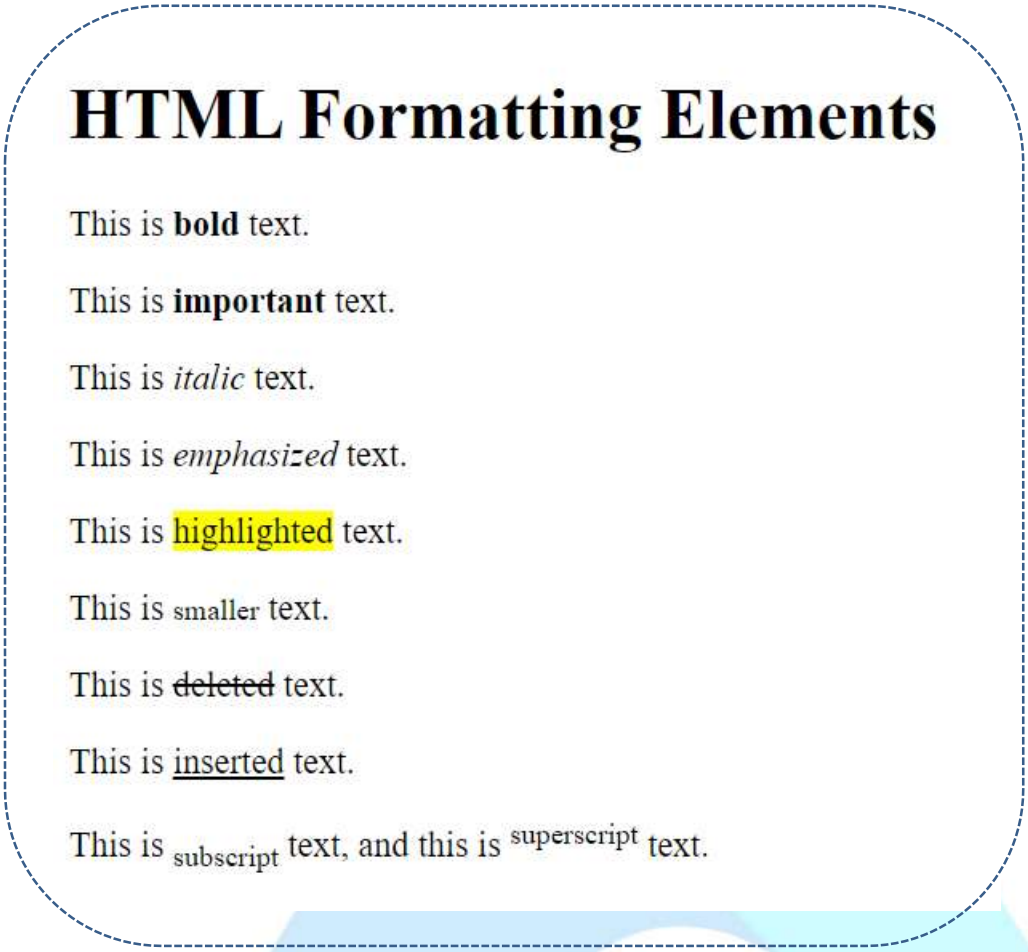
- **<sup> - Superscript text :**
  The HTML <sup> element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW[1]
  For example:

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

Examples in Context of all of these formatting elements

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Formatting Elements</title>
</head>
<body>
    <h1>HTML Formatting Elements</h1>

    <p>This is <b>bold</b> text.</p>
    <p>This is <strong>important</strong> text.</p>
    <p>This is <i>italic</i> text.</p>
    <p>This is <em>emphasized</em> text.</p>
    <p>This is <mark>highlighted</mark> text.</p>
    <p>This is <small>smaller</small> text.</p>
    <p>This is <del>deleted</del> text.</p>
    <p>This is <ins>inserted</ins> text.</p>
    <p>This is <sub>subscript</sub> text, and this is <sup>superscript</sup> text.</p>
</body>
</html>
```

The browser output:



HTML Comments

HTML comments are used to add notes or explanations within the HTML code that are not displayed in the web browser. Comments are useful for documentation, reminders, and for temporarily disabling parts of the code during development and debugging.

Here is how to create and use HTML comments

The syntax for an single line HTML comment is:

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

The syntax for an multi line HTML comment is:

```
<!--
This is a multi-line comment.
It can span multiple lines.
-->
```

HTML Quotations

The HTML Quotation elements are used to insert quoted texts in a web page, that is the portion of texts different from the normal texts in the web page. Below are some of the most used quotation elements of the HTML.

Tag	Description
<abbr>	Defines abbreviation or acronym.
	Defines contact info for the author/owner of a document.
	Defines text direction, left-to-right or right-to-left.
	Defines a section quoted from another source.
	Defines the title of a work, book, article, or publication.
<q>	Defines short inline quotation, enclosed in quotation marks.

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Quotations</title>
   </head>
   <body>
      <!--Normal text-->
      <p>Internet of Things</p>
      <!--Inside <bdo> tag-->
      <p> <bdo dir="rtl">Internet of Things</bdo> </p>
      <p> Embedded Full Stack <abbr title="Internet of Things">IoT</abbr></p>
      <address>
         <p> Address:<br />
          MSK Solutions,<br/>
          #475,Sangolli Rayanna Nagar, Dharwad </p>
      </address>
      <p>As Mahatma Gandhi once said, <q cite="https://www.example.com/quote">Be
the change you wish to see in the world.</q></p>
      <blockquote cite="https://www.example.com"> This is a long quotation from a
source. </blockquote>
   </body>
</html>
```

Browser output:

Internet of Things

sgnihT fo tenretnI

Embedded Full Stack IoT

Address:
MSK Solutions,
#475,Sangolli Rayanna Nagar, Dharwad

As Mahatma Gandhi once said, “Be the change you wish to see in the world.”

> This is a long quotation from a source.

HTML Colors

HTML colors are used in web design to enhance the visual appeal of web pages. Colors in HTML can be specified in various formats: by name, by hexadecimal value, by RGB value, by RGBA value, by HSL value, and by HSLA value.

Here's an overview of these formats:

1. Named Colors

HTML supports 140 named colors, like red, blue, green, black, white, gray, aqua, fuchsia, etc.

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

<p style="background-color: blue;">This is a blue background.</p>
```

2. Hexadecimal Colors

Colors can be specified using a hex code, which is a combination of three pairs of hexadecimal digits, representing the red, green, and blue components.

```
<p style="color: #FF0000;">This is red text.</p>

<p style="background-color: #0000FF;">This is a blue background.</p>
```

3. RGB Colors

The RGB color model represents colors using their red, green, and blue components. Each component can have a value between 0 and 255.

```
<p style="color: rgb(255, 0, 0);">This is red text.</p>

<p style="background-color: rgb(0, 0, 255);">This is a blue background.</p>
```

4. RGBA Colors

RGBA is similar to RGB but includes an alpha channel, which represents the opacity. The alpha value ranges from 0 (completely transparent) to 1 (completely opaque).

```
<p style="color: rgba(255, 0, 0, 0.5);">This is semi-transparent red text.</p>

<p style="background-color: rgba(0, 0, 255, 0.5);">This is a semi-transparent blue background.</p>
```

5. HSL Colors

HSL stands for Hue, Saturation, and Lightness. Hue is a degree on the color wheel (0-360), Saturation is a percentage (0%-100%), and Lightness is a percentage (0%-100%).

```
<p style="color: hsl(0, 100%, 50%);">This is red text.</p>

<p style="background-color: hsl(240, 100%, 50%);">This is a blue background.</p>
```

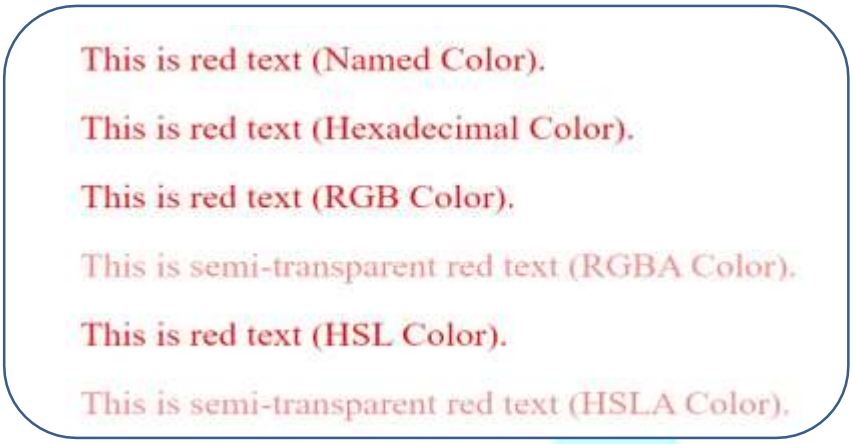
6. HSLA Colors

HSLA is similar to HSL but includes an alpha channel for opacity.

```
<p style="color: hsla(0, 100%, 50%, 0.5);">This is semi-transparent red text.</p>

<p style="background-color: hsla(240, 100%, 50%, 0.5);">This is a semi-transparent blue background.</p>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Colors Example</title>
</head>
<body>
    <p style="color: red;">This is red text (Named Color).</p>
    <p style="color: #FF0000;">This is red text (Hexadecimal Color).</p>
    <p style="color: rgb(255, 0, 0);">This is red text (RGB Color).</p>
    <p style="color: rgba(255, 0, 0, 0.5);">This is semi-transparent red text (RGBA Color).</p>
    <p style="color: hsl(0, 100%, 50%);">This is red text (HSL Color).</p>
    <p style="color: hsla(0, 100%, 50%, 0.5);">This is semi-transparent red text (HSLA Color).</p>
</body>
</html>
```

Browser Output:



HTML Images

The HTML <img> tag is used to embed an image in web pages by linking them. It creates a placeholder for the image, defined by attributes like src, width, height, and alt, and does not require a closing tag.

There are 2 ways to insert the images into a webpage:

- By providing a full path or address (URL) to access, an internet file.
- By providing the file path relative to the location of the current web page file.

Syntax :

```
<img src = "url"  alt = "some_text"  width=" width_value " height=" height_value ">
```

```
<img src="file_path" alt="some_text" width="width_value" height="height_value">
```

- **src:** The source attribute specifies the path to the image. This can be a relative path (to an image file in your project) or an absolute URL (to an image hosted online).
- **alt:** The alternative text attribute provides a textual description of the image. This text is displayed if the image cannot be loaded and is also used for accessibility purposes.
- **width:** Specifies the width of the image in pixels or percentage.
- **height:** Specifies the height of the image in pixels or percentage.

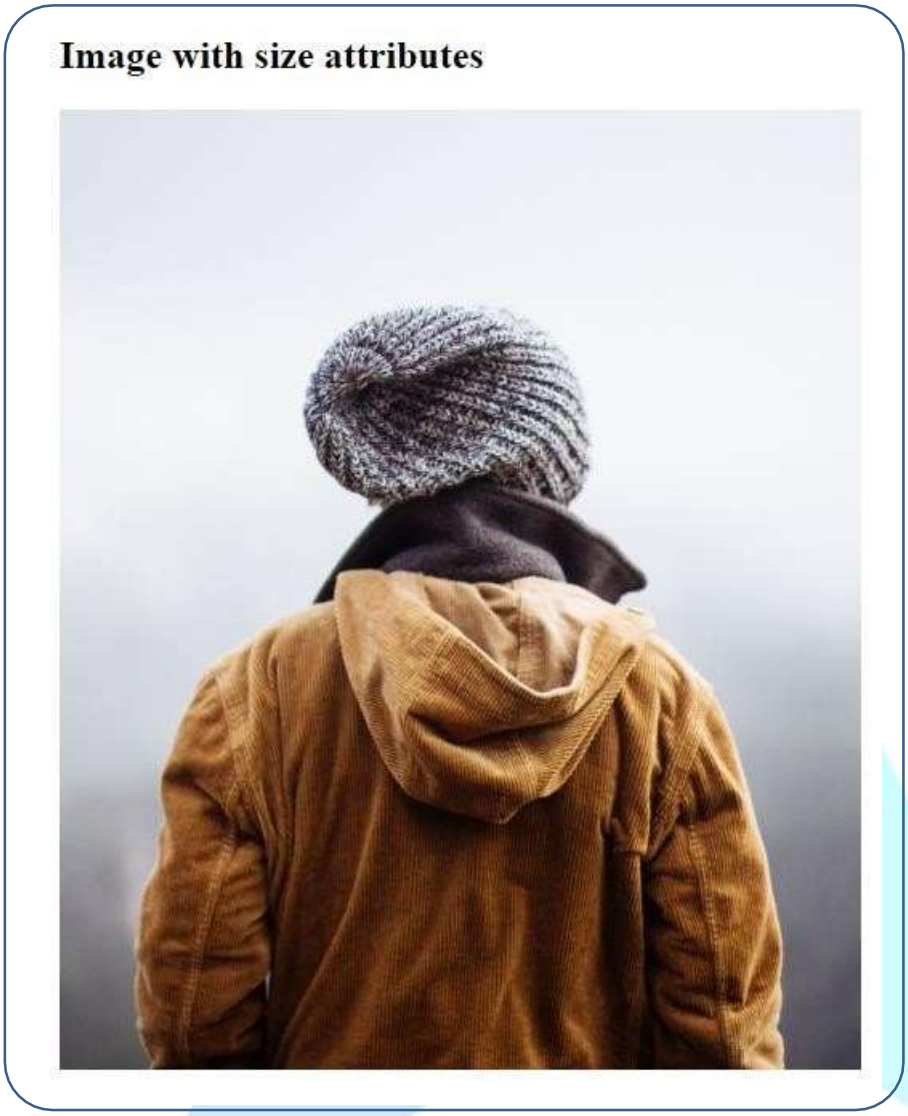
Common Image Format:

Here is the commonly used image file format that is supported by all the browsers.

S.No.	Abbreviation	File Type	Extension
1.	PNG	Portable Network Graphics.	.png
2.	JPEG.		.jpg, .jpeg, .jfif, .pjpeg, .pjp
3.	SVG		.svg.
4.	GIF		.gif
5.	ICO		
6.			

```
<!DOCTYPE html>
<html>
<body>
<h2>Image with  size attributes </h2>
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
</body>
</html>
```

Browser output:



HTML Links Hyperlinks

HTML links, or hyperlinks, connect web pages. They're created using the <a> tag , which stands for "anchor." with the href attribute, which specifies the destination URL. Users can click on links to navigate between different pages or resources.

Syntax:

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

Note: A hyperlink can be represented by an image or any other HTML element, not just text.

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

target Attribute:

Attribute	Description
_blank	
_self	
_parent	
_top	
framename	

Link to an External Website

```
<a href="https://www.google.com">Visit Google</a>
```

Link to an Internal Page

Assuming your file structure is:

```
/project
    /about.html
    index.html
```

```
<a href="about.html">About Us</a>
```

Link to a Section Within the Same Page

First, create an anchor in the page:

```
<h2 id="section1">Section 1</h2>
```

Then link to this section:

```
<a href="#section1">Go to Section 1</a>
```

Link with an Image

```
<a href="https://www.example.com">
  <img src="https://via.placeholder.com/150" alt="Example Image">
</a>
```

Opening Links in a New Tab

Use the target attribute with the value _blank:

```
<a href="https://www.example.com" target="_blank">Visit Example.com</a>
```

Adding a Tooltip to a Link

Use the title attribute:

```
<a href="https://www.example.com" title="Go to Example.com">Visit Example.com</a>
```

Link with Email

Use the mailto: protocol:

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

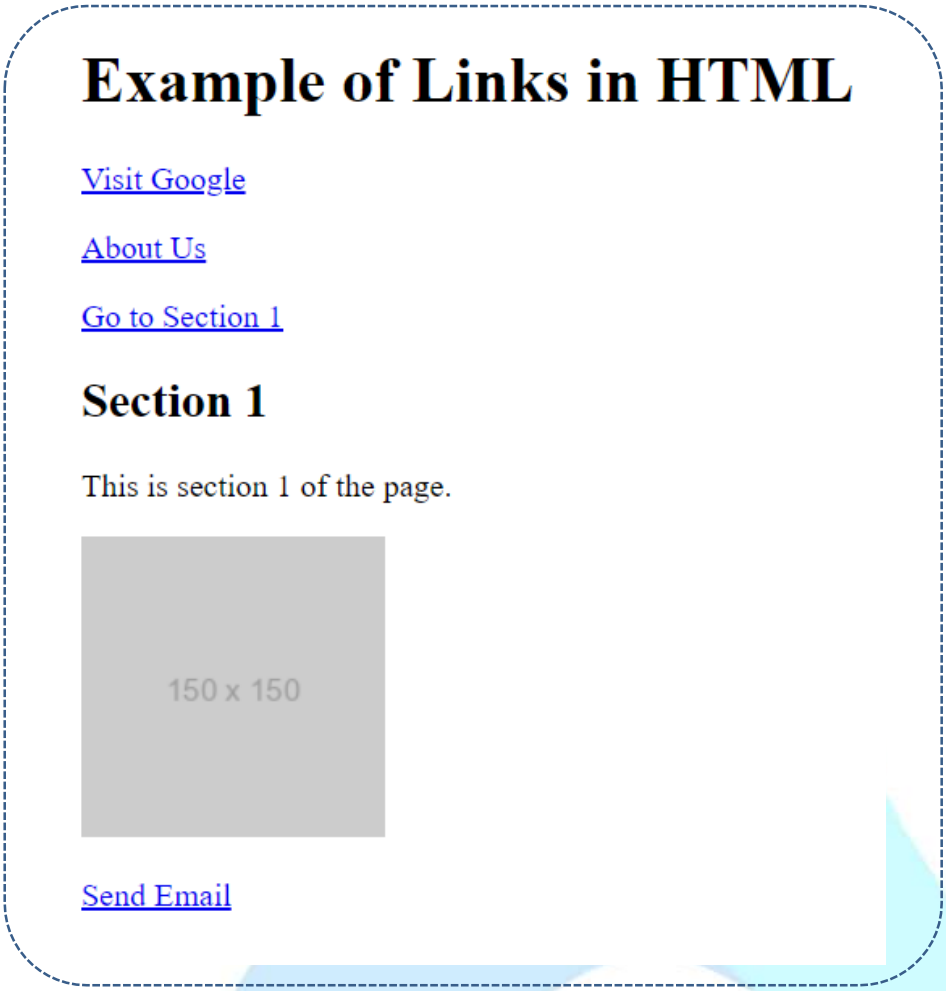
Example of a Basic Link

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Links Example</title>
</head>
<body>
    <h1>Example of a Link in HTML</h1>
    <a href="https://www.example.com">Visit Example.com</a>
</body>
</html>
```

The Browser output:

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Links Example</title> </head>
<body>
    <h1>Example of Links in HTML</h1>
    <!-- External Link -->
    <p><a href="https://www.google.com" target="_blank" title="Visit Google">Visit Google</a></p>
    <!-- Internal Link -->
    <p><a href="about.html">About Us</a></p>
    <!-- Link to a Section -->
    <p><a href="#section1">Go to Section 1</a></p>
    <!-- Section for Internal Link -->
    <h2 id="section1">Section 1</h2>
    <p>This is section 1 of the page.</p>
    <!-- Link with an Image -->
    <p><a href="https://www.example.com">
        <img src="https://via.placeholder.com/150" alt="Example Image">
    </a></p>
    <!-- Email Link -->
    <p><a href="mailto:someone@example.com">Send Email</a></p>
</body>
</html>
```

The Browser output:



HTML Tables

An HTML Table is an arrangement of data in rows and columns in tabular format. Tables are useful for various tasks, such as presenting text information and numerical data. A table is a useful tool for quickly and easily finding connections between different types of data. Tables are also used to create databases.

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

- **Table Cells :** Table Cell are the building blocks for defining the Table. It is denoted with <td> as a start tag & </td> as a end tag.

  Syntax : `</td> Content...</td>`

- **Table Rows :** The rows can be formed with the help of combination of Table Cells. It is denoted by <tr> and </tr> tag as a start & end tags.

  Syntax : `</tr> Content...</tr>`

- **Table Headers :** The Headers are generally use to provide the Heading. The Table Headers can also be used to add the heading to the Table. This contains the <th> & </th> tags.

  Syntax : `</th> Content...</th>`

- **Adding a border to an HTML Table :** A border is set using the CSS border property. If you do not specify a border for the table, it will be displayed without borders.

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

- **Adding Collapsed Borders in an HTML Table:** For borders to collapse into one border, add the CSS border-collapse property.

Syntax :

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

- **Adding Cell Padding in an HTML Table :** Cell padding specifies the space between the cell content and its borders. If we do not specify a padding, the table cells will be displayed without padding.

  Syntax :

```
th, td {
    padding: 20px;
}
```

- **Adding Border Spacing in an HTML Table :** Border spacing specifies the space between the cells. To set the border-spacing for a table, we must use the CSS border-spacing property.

  Syntax

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

Tags used in HTML Tables

HTML Tags	Descriptions
<table>	Defines the structure for organizing data in rows and columns within a web page.
<tr>	
<th>	
<td>	
<caption>	
<thead>	
<col>	Defines attributes for table columns that can be applied to multiple columns at once.
<colgroup>	Groups together a set of columns in a table to which you can apply formatting or properties collectively.

Example for HTML Table

```
<!DOCTYPE html>
<html>
<head>
   <style>  table, th , td { border: 1px solid black;} </style>
</head>
<body>
   <table style="width:100%; text-align :center;">
      <tr>
         <th>First Name</th>
         <th>Last Name</th>
         <th>Age</th>
      </tr>
      <tr>
         <td>Priya</td>
         <td>Sharma</td>
         <td>24</td>
      </tr>
      <tr>
         <td>Arun</td>
         <td>Singh</td>
         <td>32</td>
      </tr>
      <tr>
         <td>Sam</td>
         <td>Watson</td>
         <td>41</td>
      </tr>
   </table>
</body>
</html>
```

Browser Output:

Firstname	Lastname	Age
Priya	Sharma	24
Arun	Singh	32
Sam	Watson	41

HTML Lists

HTML lists are used to display related information in an easy-to-read and concise way as lists.

We can use three types of lists to represent different types of data in HTML:

1. Unordered List
2. Ordered List
3. Description List <dl>

Unordered List

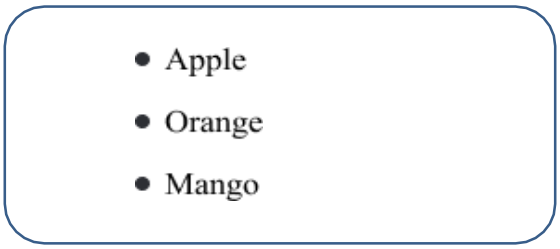
The unordered list is used to represent data in a list for which the order of items does not matter.

In HTML, we use the <ul> tag to create unordered lists. Each item of the list must be a <li> tag which represents list items. For example,

```
<ul>
        <li>Apple</li>
        <li>Orange</li>
        <li>Mango</li>
</ul>
```

Here, Apple, Orange, and Mango are the list item

Browser Output

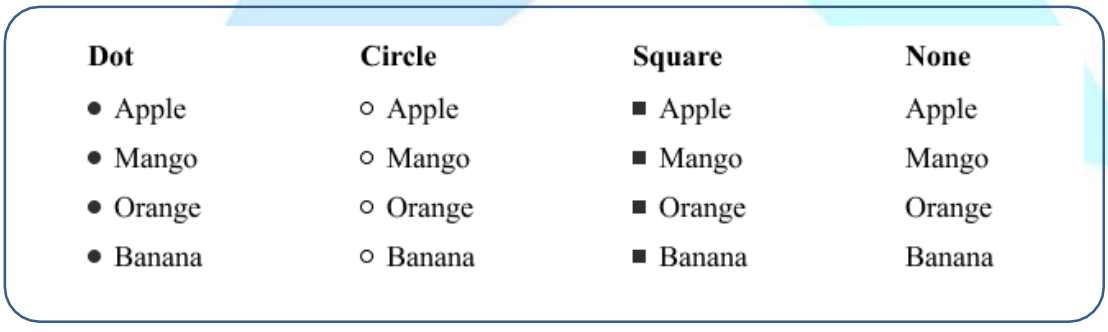


Unordered HTML List - Choose List Item Marker

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

Examples for all the marker types



Nesting Lists

In HTML, we can create a nested list by adding one list inside another.

Example for nested unordered list:

```
<ul>
  <li>
   Coffee
        <ul>
              <li>Cappuccino</li>
              <li>Americano</li>
              <li>Espresso</li>
        </ul>
  </li>
  <li>
   Tea
      <ul>
           <li>Milk Tea</li>
            <li>Black Tea</li>
      </ul>
  </li>
  <li>Milk</li>
</ul>
```

Browser Output:

- Coffee
  - Cappuccino
  - Americano
  - Espresso
- Tea
  - Milk Tea
  - Black Tea
- Milk

Ordered List

The ordered list is used to represent data in a list for which the order of items has significance.

The <ol> tag is used to create ordered lists. Similar to unordered lists, each item in the ordered list must be a <li> tag. For example,

```
<ol>
        <li>Ready</li>
        <li>Set</li>
        <li>Go</li>
</ol>
```

Browser Output

1. Ready
2. Set
3. Go

Ordered HTML List - The Type Attribute

The type attribute of the <ol> tag, defines the type of the list item marker:

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

Examples of all ordered number type lists:

type = "1"	type = "a"	type = "A"	type = "i"	type = "I"
1. Name	a. Name	A. Name	i. Name	I. Name
2. Address	b. Address	B. Address	ii. Address	II. Address
3. Phone Number	c. Phone Number	C. Phone Number	iii. Phone Number	III. Phone Number

start Attribute

We use the start attribute to change the starting point for the numbering of the list. For example,

```
<ol start='5'>
    <li>Harry</li>
    <li>Ron</li>
    <li>Sam</li>
</ol>
```

Browser Output

5. Harry
6. Ron
7. Sam

Here, we change the starting value of the list to 5.

This attribute also works with other types. For example,

```
<ol type="a" start='5'>
        <li>Harry</li>
        <li>Ron</li>
        <li>Sam</li>
</ol>
```

Browser Output

e. Harry
f. Ron
g. Sam

Similarly, we can use the start attribute along with all other types.

reversed Attribute

We can use the reversed attribute on the ordered list to reverse the numbering on the list. For example,

```
<ol reversed>
        <li>Cat</li>
        <li>Dog</li>
        <li>Elephant</li>
        <li>Fish</li>
</ol>
```

Browser Output

4. Cat
3. Dog
2. Elephant
1. Fish

Here, we can see the order of the list is reversed, the first list item is numbered 4 and the last is numbered 1.

Similarly, the reversed attribute can also be used with other types and in conjunction with the start attribute. For example,

```
<ol reversed type="I" start="10">
        <li>Cat</li>
        <li>Dog</li>
        <li>Elephant</li>
        <li>Fish</li>
</ol>
```

Browser Output

X. Cat
IX. Dog
VIII. Elephant
VII. Fish

In the above example, we use the upper-case roman numeral type and start at 10 and reverse the order of the numbers.

Description List

The HTML description list is used to represent data in the name-value form. We use the <dl> tag to create a definition list and each item of the description list has two elements:

- term/title - represented by the <dt> tag
- description of the term - represented by the <dd> tag

Let's see an example,

```
<dl>
        <dt>HTML</dt>
        <dd>Hyper-Text Markup Language</dd>
        <dt>CSS</dt>
        <dd>Cascading StyleSheets</dd>
        <dt>JS</dt>
        <dd>Javascript</dd>
</dl>
```

Browser Output

HTML
Hyper-Text Markup Language
CSS
Cascading StyleSheets
JS
Javascript

Here, it consists of two different types of list items:

- <dt> - defines the terms/name
- <dd> - defines the description/value of the term/name

The description list includes two related values, hence it can also be used to store items in key/value pairs.

Since the description list includes the definition of a term, it is also known as the definition list.

Multiple Terms and Multiple Definitions

The definition list is used to display data in a key/value format, where the <dt> tag indicates the key elements and the <dd> tag element indicates the value (definition) of the key.

However, while creating a description list, it's not necessary that a single <dt> tag (key) should have a single <dd> tag (value). We can have any combination of the <dt> and <dd> elements.

Possible Combinations of terms and term descriptions:

- Single term (<dt>) with a single definition (<dd>).
- Multiple terms (<dt>) with a single definition (<dd>).
- Multiple definitions (<dd>) with a single term (<dt>).

Let's take a look at a few examples,

1. Single Term and Description

```
<dl>
        <dt>HTML</dt>
        <dd>HyperText Markup Language</dd>
        <dt>CSS</dt>
        <dd>Cascading Style Sheets</dd>
</dl>
```

Browser Output

HTML
HyperText Markup Language
CSS
Cascading Style Sheets

Here, we can see a single term is followed by a single description.

2. Single Term and Multiple Description

Sometimes, we can come across data where there are multiple descriptors that fit the same term, like a product and its features list.

In such a case, rather than listing the product name multiple times before each feature, we can follow the single term <dt> with multiple feature/description <dd> to present it better. For example,

```
<dl>
        <dt>HTML</dt>
        <dd>HyperText Markup Language.</dd>
        <dd>Used to create Websites.</dd>
        <dd>Released in 1993.</dd>
</dl>
```

Browser Output

HTML
HyperText Markup Language.
Used to create Websites.
Released in 1993.

Here, we can see that the single term <dt>HTML</dt> is described by multiple definitions <dd>

3. Multiple Term and Single Description

Sometimes, we come across data where multiple keys may have similar values. Like multiple programming languages can feature the same feature sets.

In such cases, rather than repeating the key/value pair, we can group several keys <dt> followed by a single description <dd> such that the single description describes many terms,

```
<dl>
        <dt>HTML</dt>
        <dd>is markup language</dd>
        <dt>Python</dt>
        <dt>Java</dt>
        <dd>are programming languages.</dd>
</dl>
```

Browser Output

HTML
    is markup language
Python
Java
    are programming languages.

Here, we can see that multiple terms <dt>Python</dt> and <dt>Java</dt> share the same description <dd>are programming languages.</dd>.

A description list is the best choice when we want to express the semantically correct format of title-description elements in our HTML. We can also denote the relation between pairs using Description lists.

HTML Iframes

An HTML iframe is used to display a web page within a web page. HTML iframes offer a powerful way to embed external content, such as videos, maps, or other webpages, directly into your own webpage. An iframe is an HTML document embedded inside another HTML document. The <iframe> tag specifies the URL of the embedded content, **allowing for seamless integration of external resources.**

Syntax:

```
<iframe src="url" width="width_value" height="height_value" title="description"></iframe>
```

- src: Specifies the URL of the page to embed.
- width: Specifies the width of the iframe.
- height: Specifies the height of the iframe.
- title: Provides an accessible name for the iframe (important for accessibility).

Attributes Value:

It contains a single value URL that specifies the URL of the document that is embedded in the iframe. There are two types of URL links which are listed below:

	It points to other files of the same web page.

Example of iframe

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Iframe Example</title>
</head>
<body>
    <h1>Embedding a Website with an Iframe</h1>
    <iframe src="https://www.example.com" width="600" height="400" title="Example Website"></iframe>
</body>
</html>
```

Browser output:

HTML Forms

An HTML form is a section of a document designed to collect user inputs through various interactive controls. These controls include text fields, password fields, checkboxes, radio buttons, submit buttons, and menus, among others. HTML forms are essential for gathering data from site visitors, such as names, email addresses, passwords, and phone numbers, which are then sent to the server for processing.

Why Use HTML Forms?

HTML forms are crucial when there is a need to collect data from users. For example, if a user wants to purchase items online, they must fill out a form with their shipping address and credit/debit card details to ensure the item is sent to the correct address. By utilizing the <form> element, HTML forms act as a powerful tool for collecting user input through a variety of interactive controls. These controls range from text fields, numeric inputs, email fields, password fields, to checkboxes, radio buttons, and submit buttons.

In essence, an HTML form serves as a versatile container for numerous input elements, thereby enhancing user interaction. Whether it's for signing up for a newsletter, making a purchase, or providing feedback, HTML forms play a vital role in the user experience on a website.

HTML Form Syntax

```
<form action="server url" method="get|post">
  //input controls e.g. textfield, textarea, radiobutton, button
</form>
```

HTML Form Elements

The HTML <form> comprises several elements, each serving a unique purpose. For instance, the <label> element is used to define labels for other <form> elements. The <input> element, on the other hand, is versatile and can be used to capture various types of input data such as text, password, email, and more, simply by altering its type attribute.

List of HTML form elements

Elements	Descriptions
<label>	
<input>	
<button>	
<select>	
<textarea>	
<fieldset>	
<legend>	
<output>	
<option>	It is used to define options in a drop-down list.
<optgroup>	It is used to define group-related options in a drop-down list.

Commonly Used Input Types in HTML Forms

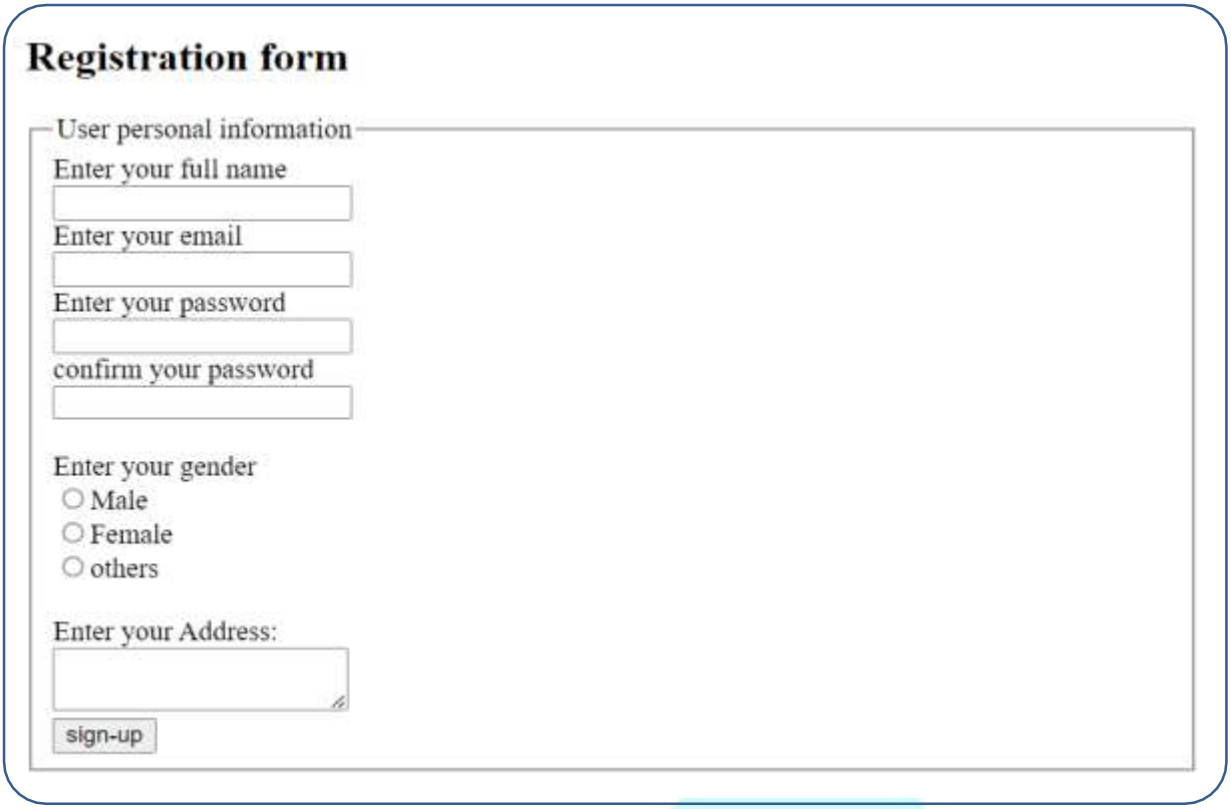
In HTML forms, various input types are used to collect different types of data from users. Here are some commonly used input types:

Input Type	Description
<input type=”text“>	Defines a one-line text input field
<input type=”password“>	
<input type=”submit“>	
<input type=”radio“>	
	Allows the user to select a date from a calendar.
<input type=”time“>	Allows the user to select a time.
<input type=”file“>	Allows the user to select a file to upload.

Example for HTML Form

```
<!DOCTYPE html>
<html>
<head>
 <title>Form in HTML</title>
</head>
<body>
   <h2>Registration form</h2>
  <form>
   <fieldset>
     <legend>User personal information</legend>
     <label>Enter your full name</label><br>
     <input type="text" name="name"><br>
      <label>Enter your email</label><br>
      <input type="email" name="email"><br>
      <label>Enter your password</label><br>
      <input type="password" name="pass"><br>
      <label>confirm your password</label><br>
      <input type="password" name="pass"><br>
      <br><label>Enter your gender</label><br>
      <input type="radio" id="gender" name="gender" value="male"/>Male  <br>
      <input type="radio" id="gender" name="gender" value="female"/>Female <br/>
      <input type="radio" id="gender" name="gender" value="others"/>others <br/>
       <br>Enter your Address:<br>
      <textarea></textarea><br>
      <input type="submit" value="sign-up">
   </fieldset>
 </form>
</body>
</html>
```

Browser Output:



HTML Form Attributes

Attribute	Description
accept-charset	Specifies the character encodings used for form submission
action	Specifies where to send the form-data when a form is submitted
autocomplete	Specifies whether a form should have autocomplete on or off
enctype	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
method	Specifies the HTTP method to use when sending form-data
name	Specifies the name of the form
novalidate	Specifies that the form should not be validated when submitted
rel	Specifies the relationship between a linked resource and the current document
target	Specifies where to display the response that is received after submitting the form

CSS

Cascading Style Sheets (CSS) is a styling language used to define the presentation and layout of web pages. It allows web developers to control the appearance of web documents (HTML, XML) by specifying styles for various elements on a page. CSS can change the color, size, and style of text, backgrounds, and links. It is indispensable for creating interesting layouts and enhancing user experience.

Key Features of CSS:

1. **Separation of Content and Style**: CSS separates the visual presentation of a webpage from its content. This separation makes it easier to maintain and update web pages, as changes to the style can be made in a single CSS file rather than in multiple HTML files.
2. **Styling Elements**: CSS can be used to style a wide range of HTML elements. For instance, it can modify text attributes (such as font, size, and color), change background colors or images, and style links. This capability helps create visually appealing web pages that are more engaging for users.
3. **Layout Control**: CSS is essential for defining the layout of web pages. It allows developers to create complex layouts using features like flexbox and grid, making it easier to design responsive websites that look good on various screen sizes and devices.
4. **Animations and Transitions**: CSS also supports animations and transitions, which can be used to create dynamic effects on web pages. This includes animating elements, creating hover effects, and adding smooth transitions between states.
5. **Selectors and Specificity**: CSS uses selectors to target HTML elements for styling. Understanding selectors and their specificity is crucial for applying styles correctly and avoiding conflicts between different styles.
6. **Media Queries**: CSS supports media queries, which enable the creation of responsive designs. Media queries allow developers to apply different styles based on the characteristics of the device, such as its width, height, and orientation.

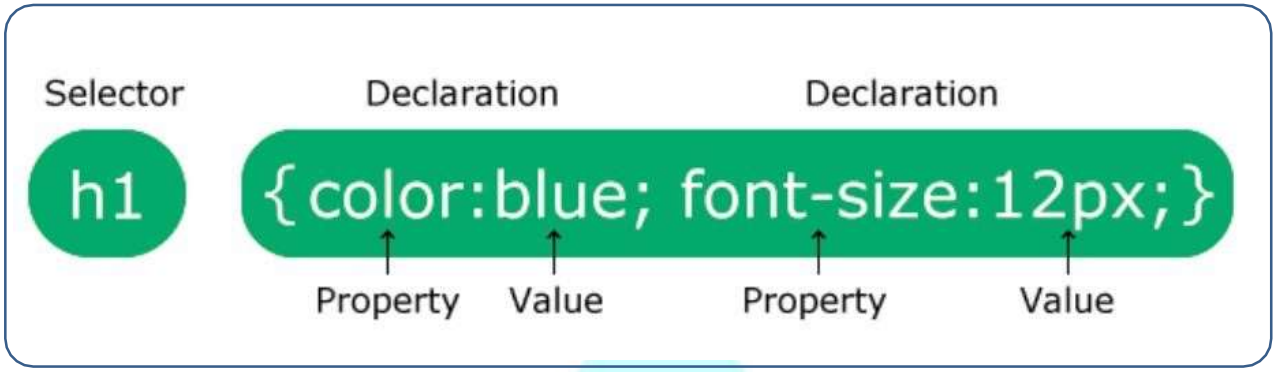
Example Use Cases:

- **Text Styling**: Change the font, size, and color of text to improve readability and match the design of the website.
- **Backgrounds**: Add background colors or images to enhance the visual appeal of sections or the entire page.
- **Navigation**: Create styled navigation bars and sidebars to improve user experience and ease of navigation.
- **Layouts**: Design complex layouts with columns, grids, and flexible boxes to present content in an organized manner.

CSS is an essential component of web development, alongside HTML and JavaScript. HTML provides the structure and content of a webpage, while CSS enhances the visual presentation of that content through various styles. By mastering CSS, developers can create visually stunning and user-friendly websites that provide a better experience for visitors.

CSS Syntax

A CSS rule consists of a selector and a declaration block.



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
              color: red;
              text-align: center;
            }
        </style>
    </head>
    <body>

        <p>Hello World!</p>
        <p>These paragraphs are styled with CSS.</p>

    </body>
</html>
```

Browser Output

CSS Selectors:

CSS selectors are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector
6. CSS Child Selector
7. CSS Descendant selectors
8. CSS Adjacent sibling selectors
9. CSS General sibling selectors
10. CSS Attribute selector

CSS Element Selector:

The element selector selects the HTML element by name.

```
<!DOCTYPE html>
<html>
        <head>
                <style> p{ text-align: center; color: blue; } </style>
        </head>
        <body>
                <p>This style will be applied on every paragraph.</p>
                <p > Me too! </p>
                <p>And me! </p>
        </body>
</html>
```

Browser output

This style will be applied on every paragraph.

Me too!

And me!

CSS Id Selector:

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Let?s take an example with the id "para1".

```
<!DOCTYPE html>
<html>
 <head>
        <style>  #para1 { text-align: center;  color: blue; }  </style>
 </head>
 <body>
        <p id="para1">Hello Everyone</p>
        <p>This paragraph will not be affected.</p>
 </body>
</html>
```

Browser Output

Hello Everyone

This paragraph will not be affected.

CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name. A class name should not be started with a number.

Example with a class "center".

```
<!DOCTYPE html>
<html>
<head>
<style>
        .center {
            text-align: center;
            color: blue;
        }
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Browser output:

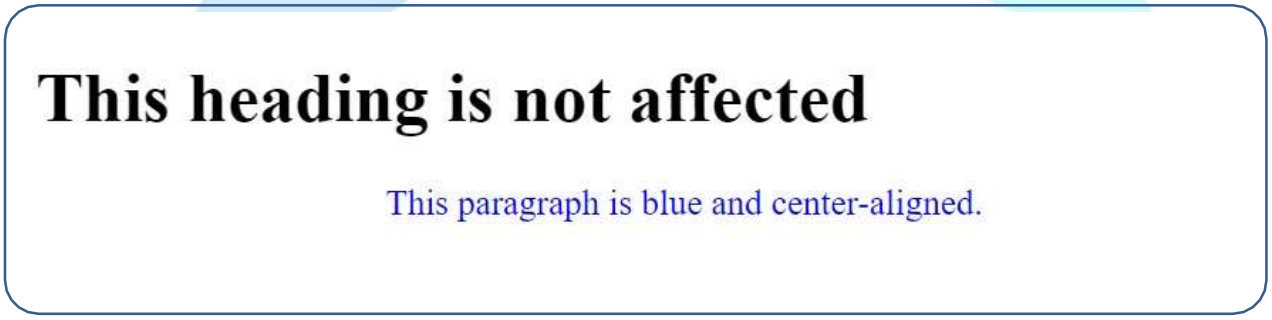
CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector. It is also called element class selector.

Let's see an example.

```
<!DOCTYPE html>
<html>
<head>
<style>
        p.center {
            text-align: center;
            color: blue;
        }
</style>
</head>
<body>
<h1 class="center">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Browser output:

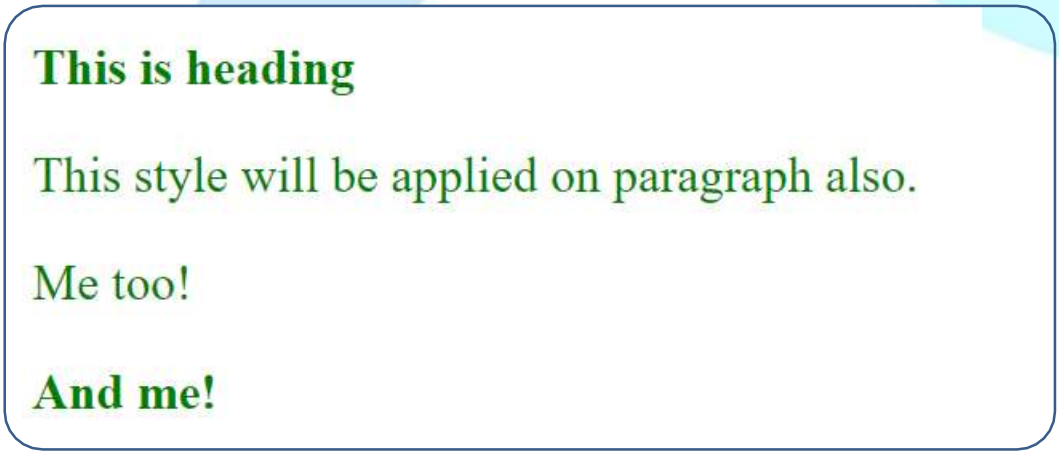


CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<!DOCTYPE html>
<html>
<head>
<style>
        * {
          color: green;
          font-size: 20px;
        }
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p >Me too!</p>
<h3>And me!</h3>
</body>
</html>
```

Browser Output:



CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

```
h1 {
    text-align: center;
    color: blue;
}
h2 {
    text-align: center;
    color: blue;
}
p {
    text-align: center;
    color: blue;
}
```

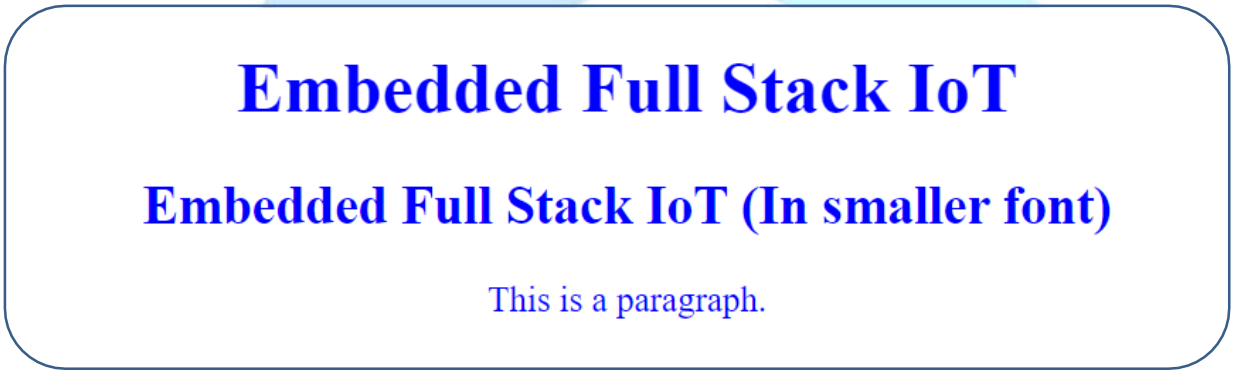
As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

```
h1,h2,p {
    text-align: center;
    color: blue;
}
```

Let's see the full example of CSS group selector.

```
<!DOCTYPE html>
<html>
<head>
<style>
        h1, h2, p {
            text-align: center;
            color: blue;
        }
</style>
</head>
<body>
<h1> Embedded Full Stack IoT </h1>
<h2> Embedded Full Stack IoT (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Browser Output:

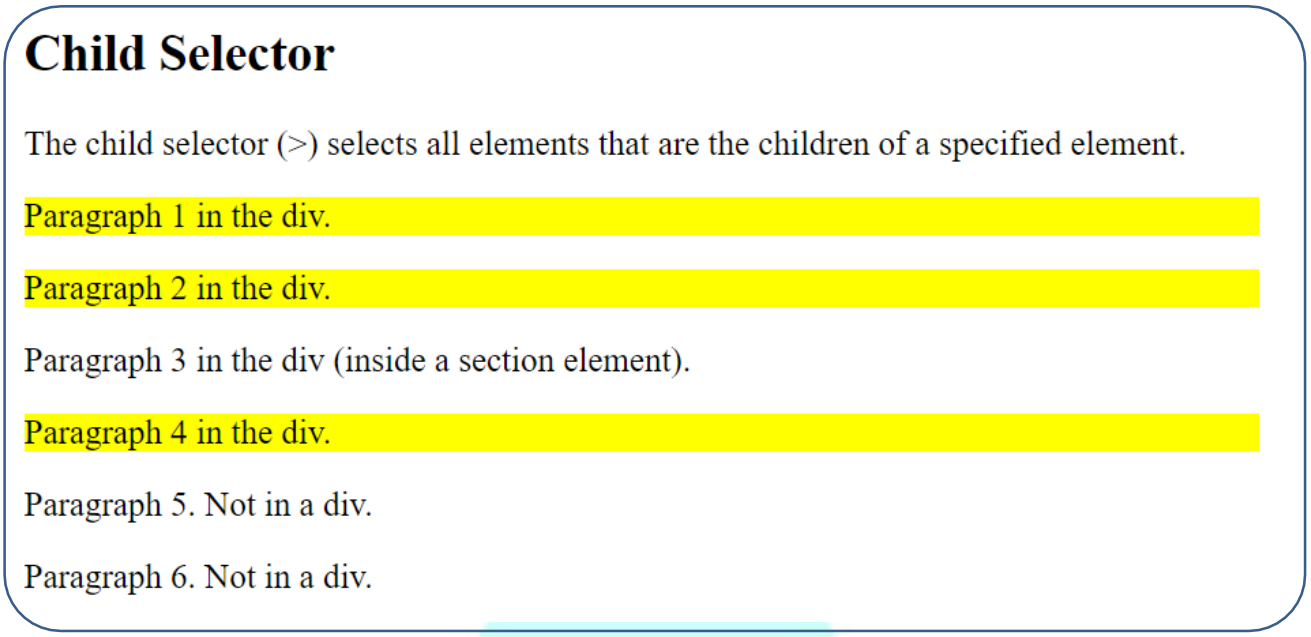


CSS Child Selector:

The child selector selects all elements that are the children of a specified element. It gives the relation between two elements. The element > element selector selects those elements which are the children of the specific parent. The operand on the left side of > is the parent and the operand on the right is the children element.

```
<!DOCTYPE html>
<html>
<head>
<style>
        div > p {
          background-color: yellow;
        }
</style>
</head>
<body>
<h2>Child Selector</h2>
<p>The child selector (>) selects all elements that are the children of a specified element.</p>
<div>
    <p>Paragraph 1 in the div.</p>
    <p>Paragraph 2 in the div.</p>
          <section>
            <!-- not Child but Descendant -->
            <p>Paragraph 3 in the div (inside a section element).</p>
          </section>
    <p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
</body>
</html>
```

Browser output:

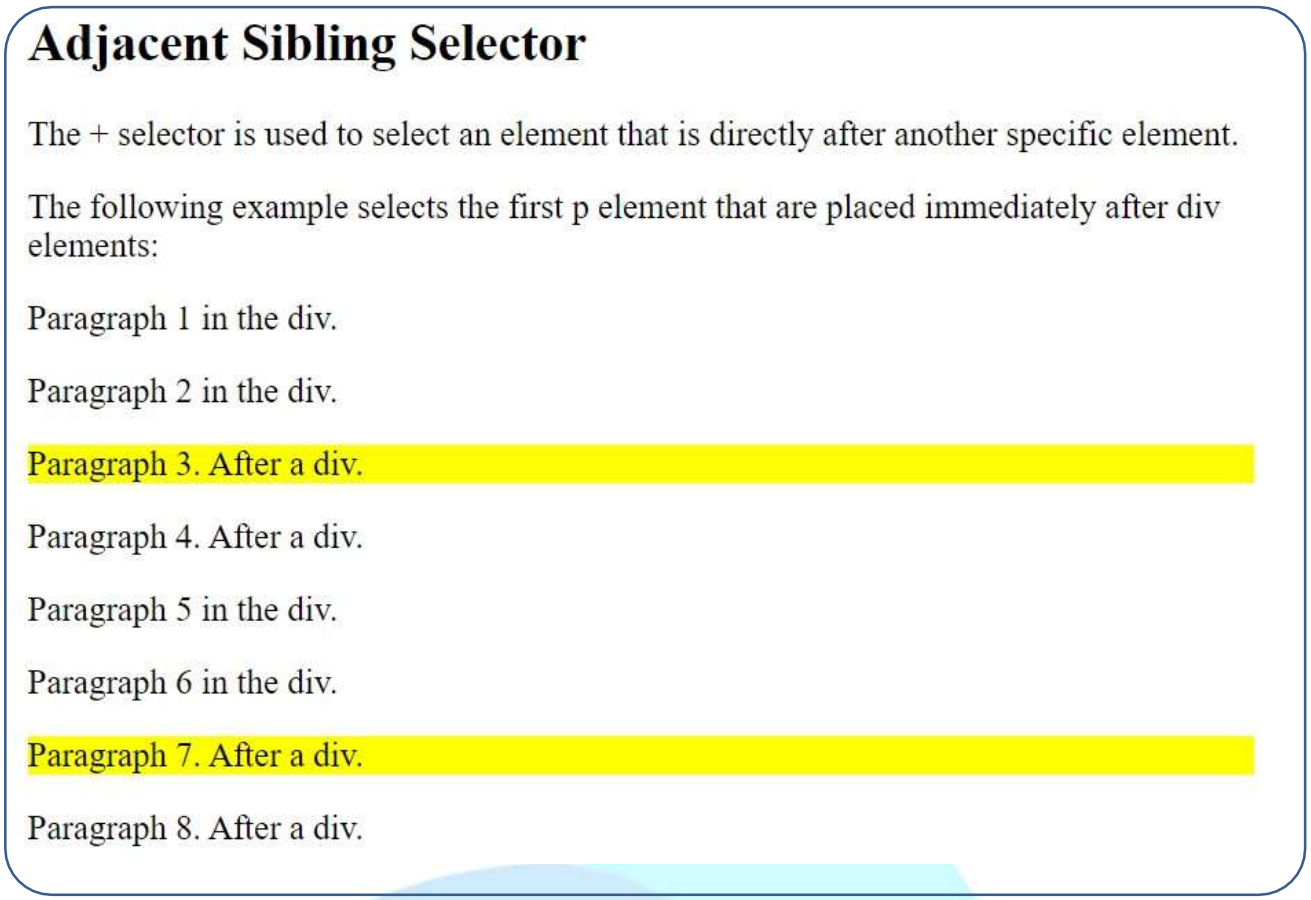


CSS Adjacent sibling selectors

The CSS adjacent sibling selector is used to select the adjacent sibling of an element. Sibling elements must have the same parent element, and "adjacent" means "immediately following". It is used to select only those elements which immediately follow the first selector. The + sign is used as a separator. For example, the direct next element is selected here with the adjacent sibling selector concept

```
<!DOCTYPE html>
<html>
<head>
<style>
        div + p {
          background-color: yellow;
        }
</style>
</head>
<body>
<h2>Adjacent Sibling Selector</h2>
<p>The + selector is used to select an element that is directly after another specific element.</p>
<p>The following example selects the first p element that are placed immediately after div elements:</p>
        <div>
          <p>Paragraph 1 in the div.</p>
          <p>Paragraph 2 in the div.</p>
        </div>
<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>
        <div>
          <p>Paragraph 5 in the div.</p>
          <p>Paragraph 6 in the div.</p>
        </div>
<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>
</body>
</html>
```

Browser output

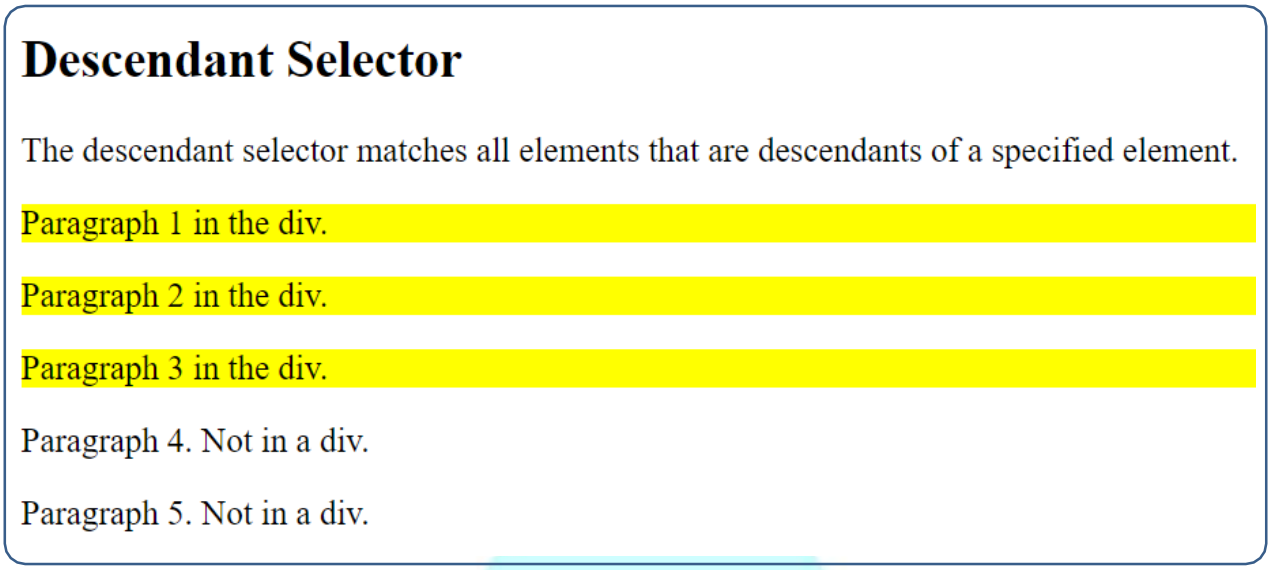


Descendant Selector

The descendant selector matches all elements that are descendants of a specified element. Descendant selector is used to select all the elements which are child of the element (not a specific element). It selects the elements inside the elements i.e it combines two selectors such that elements matched by the second selector are selected if they have an ancestor element matching the first selector.

```
<!DOCTYPE html>
<html>
<head>
<style>
        div p {
          background-color: yellow;
        }
</style>
</head>
<body>
<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are descendants of a specified
element.</p>
        <div>
          <p>Paragraph 1 in the div.</p>
          <p>Paragraph 2 in the div.</p>
          <section> <p>Paragraph 3 in the div.</p> </section>
        </div>
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
</body>
</html>
```

Browser Output:



CSS Attribute Selectors

These are used to style HTML elements with specific attributes. It is possible to style HTML elements that have specific attributes or attribute values. The [attribute] selector is used to select elements with a specified attribute.

```
<!DOCTYPE html>
<html>
<head>
<style>
        a[target] {
          background-color: yellow;
        }
</style>
</head>
<body>
<h2>CSS [attribute] Selector</h2>
<p>The links with a target attribute gets a yellow background:</p>
<a href="https://msksolutions.in/">msksolutions.in</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
</body>
</html>
```

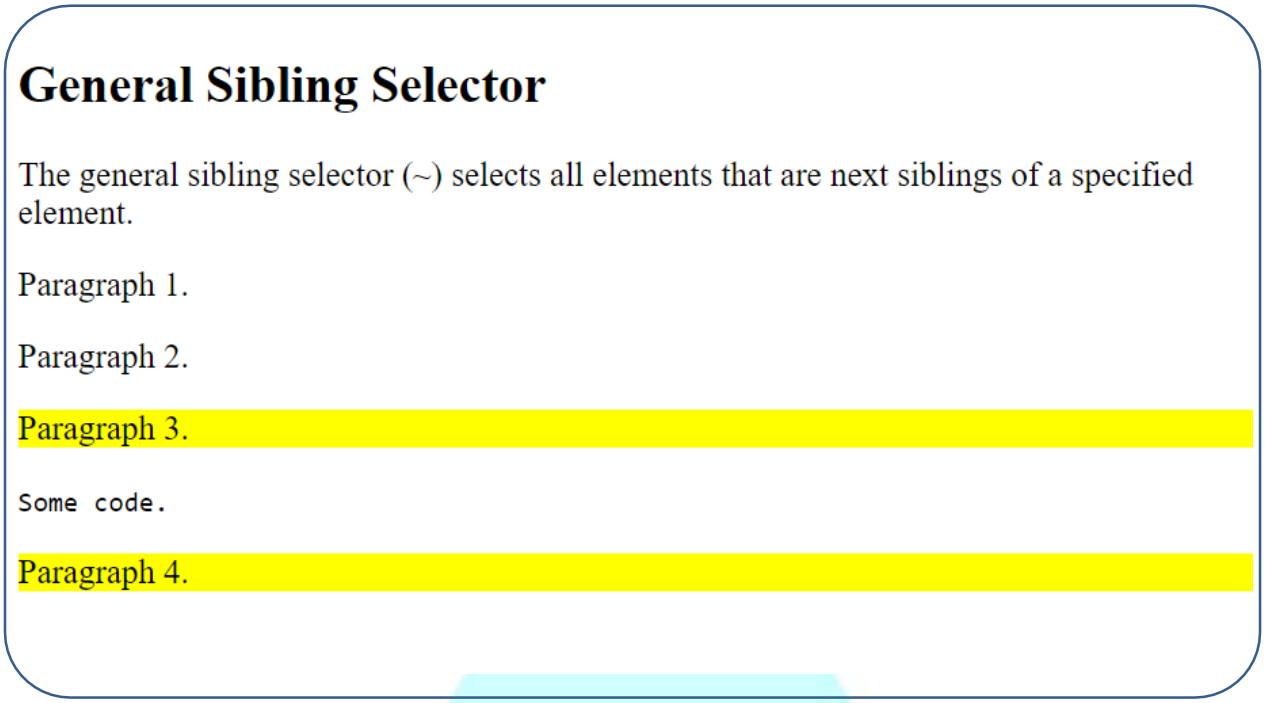
Browser output:

CSS General sibling selectors

The general sibling selector selects all elements that are next siblings of a specified element. The general sibling selector is used to select the element that follows the first selector element and also shares the same parent as the first selector element. This can be used to select a group of elements that share the same parent element.

```
<!DOCTYPE html>
<html>
<head>
<style>
        div ~ p {
          background-color: yellow;
        }
</style>
</head>
<body>
<h2>General Sibling Selector</h2>
<p>The general sibling selector (~) selects all elements that are next siblings of a specified element.</p>
<p>Paragraph 1.</p>
<div>
  <p>Paragraph 2.</p>
</div>
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
</body>
</html>
```

Browser Output:



Types of CSS (Cascading Style Sheet)

Cascading Style Sheets (CSS) is a language used to style web pages that contain HTML elements, defining how elements are displayed on webpages, including layout, colors, fonts, and other properties of the elements on a web page. CSS works by targeting HTML elements and applying style rules to define how they should be displayed, including properties like color, size, layout, and positioning.

There are three types of CSS which are given below:

1. Inline CSS
2. Internal or Embedded CSS
3. External CSS

Inline CSS

Inline CSS is a method of applying styling directly to individual HTML elements using the "style" attribute within the HTML tag, allowing for specific styling of individual elements within the HTML document, overriding any external or internal styles.

Example for Inline CSS

```
<!DOCTYPE html>
<html>
<body>
        <h1 style="color:blue; text-align:center;">This is a heading</h1>
        <p style="color:red;">This is a paragraph.</p>
</body>
</html>
```

Browser Output:

This is a heading

This is a paragraph.

Internal or Embedded CSS:

Internal or Embedded CSS is defined within the HTML document's <style> element. It applies styles to specified HTML elements, The CSS rule set should be within the HTML file in the head section i.e. the CSS is embedded within the <style> tag inside the head section of the HTML file.

```
<!DOCTYPE html>
<html>
<head>
<style>
        body {
          background-color: linen;
        }
        h1 {
          color: maroon;
          margin-left: 40px;
        }
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Browser Output:

External CSS:

External CSS contains separate CSS files that contain only style properties with the help of tag attributes (For example class, id, heading, … etc). CSS property is written in a separate file with a .css extension and should be linked to the HTML document using a link tag. It means that, for each element, style can be set only once and will be applied across web pages.

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

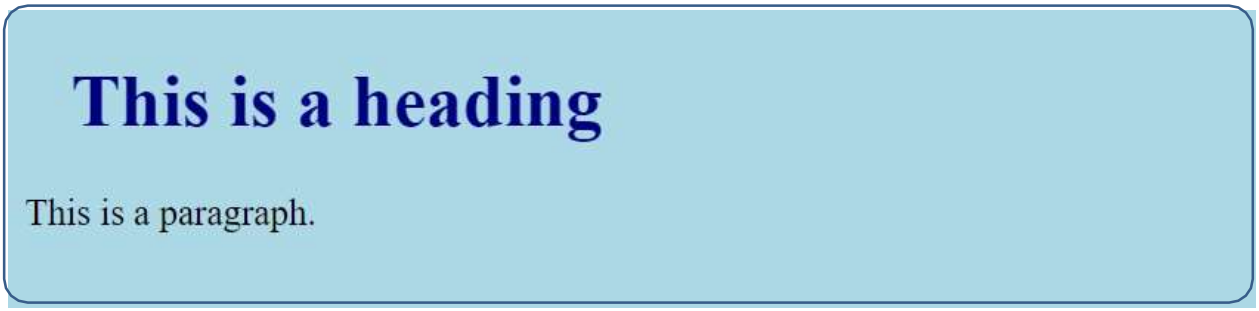
"mystyle.css"

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

Example for external CSS

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Browser output



External style sheets have the following advantages over internal and inline styles:

- one change to the style sheet will change all linked pages
- you can create classes of styles that can then be used on many different HTML elements
- consistent look and feel across multiple web pages
- improved load times because the css file is downloaded once and applied to each relevant page as needed

Differences between Inline, Internal, and External CSS:

Feature	Inline CSS	Internal CSS	External CSS
Location	It is used within HTML tag using the style attribute.	It is used within <head> section of HTML document.	It is used in a separate .css file.
Selector Scope	Affects a single element or a group of elements.		Affects multiple HTML documents or an entire website.
Reusability	Not reusable. Styles need to be repeated for each element.		Can be reused on multiple HTML documents or an entire website.
Priority	Highest priority. Overrides internal and external styles.		
	Inline styles increase the HTML file size, which can affect the page load time.		
Maintainability	Not easy to maintain. Changes need to be made manually to each element.	Relatively easy to maintain. Changes need to be made in one place in the <head> section.	Easiest to maintain. Changes need to be made in one place in the external .css file

CSS Background

CSS background property is used to define the background effects on element. There are 8 CSS background properties that affects the HTML elements:

1. background-color
2. background-image
3. background-position
4. background-size
5. background-repeat
6. background-origin
7. background-clip
8. background-attachment

CSS background-color:

The background-color property sets the background color of an element. The background of an element is the total size of the element, including padding and border (but not the margin).

Example for CSS background-color:

```
<!DOCTYPE html>
<html>
<head>
<style>
        body {
          background-color: coral;
        }
</style>
</head>
<body>
<h1>The background-color Property</h1>
<p>Here the background color can be specified with a color name coral.</p>
</body>
</html>
```

Browser output:

CSS background-image

The background-image property is used to set an image as a background of an element. The background-image property sets one or more background images for an element. By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally. The background of an element is the total size of the element, including padding and border (but not the margin).

```
<!DOCTYPE html>
<html>
<head>
<style>
        body {
          background-image: url("paper.gif");
          background-color: #cccccc;
         }
</style>
</head>
<body>
<h1>The background-image Property</h1>
<p>Hello World!</p>
</body>
</html>
```

Browser output:

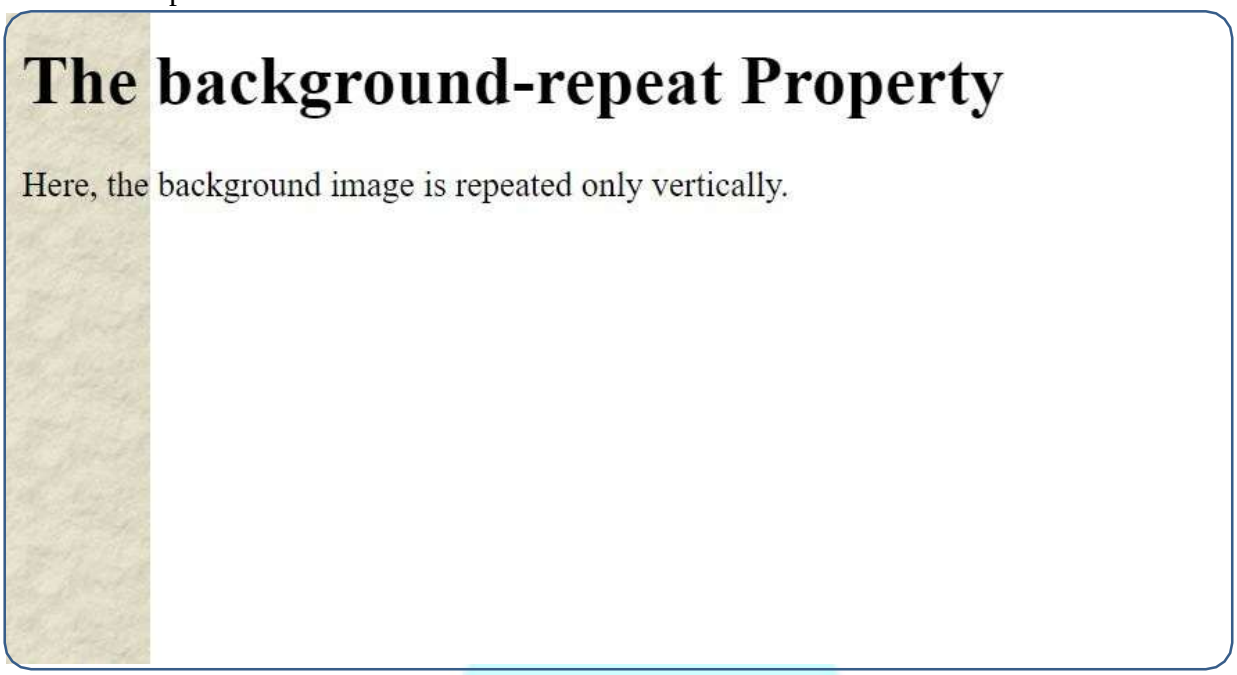
CSS background-repeat

The background-repeat property sets if/how a background image will be repeated. By default, a background-image is repeated both vertically and horizontally. The background image is placed according to the background-position property. If no background-position is specified, the image is always placed at the element's top left corner.

Example for background-repeat: repeat-y;

```
<!DOCTYPE html>
<html>
<head>
<style>
        body {
          background-image: url("paper.gif");
          background-repeat: repeat-y;
         }
</style>
</head>
<body>
<h1>The background-repeat Property</h1>
<p>Here, the background image is repeated only vertically.</p>
</body>
</html>
```

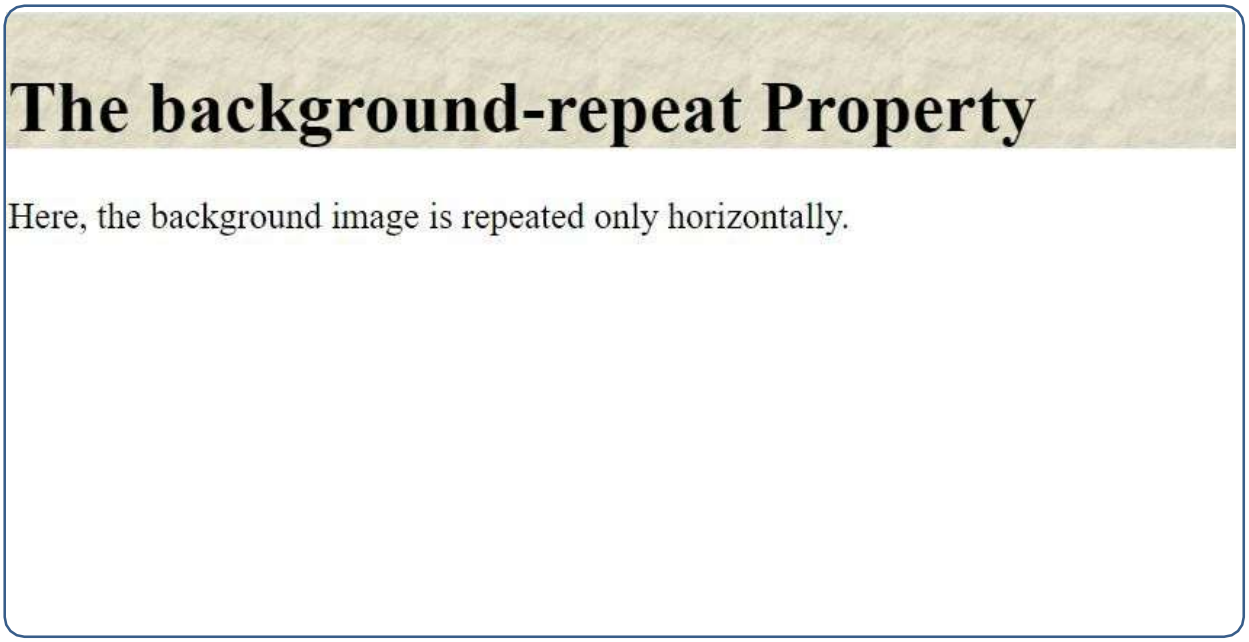
Browser Output:



Example for background-repeat: repeat-x;

```
<!DOCTYPE html>
<html>
<head>
<style>
        body {
          background-image: url("paper.gif");
          background-repeat: repeat-x;
        }
</style>
</head>
<body>
<h1>The background-repeat Property</h1>
<p>Here, the background image is repeated only horizontally.</p>
</body>
</html>
```

Browser output:



CSS background-attachment

The background-attachment property is used to specify if the background image is fixed or scroll with the rest of the page in browser window. If you set fixed the background image then the image will not move during scrolling in the browser.

Example with fixed background image.

```
body {
  background-image: url("img_tree.gif");
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

Example with fixed background image.

```
body {
  background-image: url("img_tree.gif");
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

CSS background-position

The background-position property sets the starting position of a background image. By default, the background image is placed on the top-left of the webpage.

You can set the following positions:

1. center
2. top
3. bottom
4. left
5. right

Example for background-position: center;

```
<!DOCTYPE html>
<html>
<head>
<style>
    body {
      background-image: url('w3css.gif');
      background-repeat: no-repeat;
      background-attachment: fixed;
      background-position: center;
    }
</style>
</head>
<body>
<h1>The background-position Property</h1>
<p>Here, the background image will be positioned in the center of the element (in this case, the body element).</p>
</body>
</html>
```

Browser Output:

CSS background-size

The background-size property specifies the size of the background images.

```
#example1 {
  border: 2px solid black;
  padding: 25px;
  background: url(mountain.jpg);
  background-repeat: no-repeat;
  background-size: auto;
}

#example2 {
  border: 2px solid black;
  padding: 25px;
  background: url(mountain.jpg);
  background-repeat: no-repeat;
  background-size: 300px 100px;
}
```

CSS background-origin

The background-origin property specifies the origin position (the background positioning area) of a background image. This property has no effect if background-attachment is "fixed".

```
#example1 {
  border: 10px dashed black;
  padding: 25px;
  background: url(paper.gif);
  background-repeat: no-repeat;
  background-origin: padding-box;
}
#example2 {
  border: 10px dashed black;
  padding: 25px;
  background: url(paper.gif);
  background-repeat: no-repeat;
  background-origin: border-box;
}
#example3 {
  border: 10px dashed black;
  padding: 25px;
  background: url(paper.gif);
  background-repeat: no-repeat;
  background-origin: content-box;
}
```

CSS background-clip

The background-clip property defines how far the background (color or image) should extend within an element.

```
#example1 {
  border: 10px dotted black;
  padding: 15px;
  background: lightblue;
  background-clip: border-box;
}

#example2 {
  border: 10px dotted black;
  padding: 15px;
  background: lightblue;
  background-clip: padding-box;
}

#example3 {
  border: 10px dotted black;
  padding: 15px;
  background: lightblue;
  background-clip: content-box;
}
```

Example of CSS background Properties

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Background Property Example</title>
    <style>
        .background-example {
            /* Background shorthand property */
            background:
                #ffffff                  /* background-color */
                url('example.jpg')          /* background-image */
                no-repeat                /* background-repeat */
                fixed                    /* background-attachment */
                center center / cover       /* background-position / background-size */
                border-box               /* background-origin */
                content-box              /* background-clip */
                rgba(255, 255, 255, 0.8);     /* background-blend-mode */
            width: 300px;
            height: 300px;
            border: 1px solid #000;
        }
    </style>
</head>
<body>
    <div class="background-example"></div>
</body>
</html>
```

CSS Borders

CSS borders are essential elements in websites, representing the edges of various components and elements. CSS Borders refer to the lines that surround elements, defining their edges. Borders can be styled, colored, and sized using CSS properties such as border style, border color, border width, and border radius. borders can be styled with the top border, the right border, the bottom border, and the left border.

CSS provides several properties to customize borders:

1. **border-style:** Determines the type of border (e.g., solid, dashed, dotted).
2. **border-width:** Sets the width of the border (in pixels, points, or other units).
3. **border-color:** Specifies the border color.
4. **border-radius:** Creates rounded corners for elements

Ways to Style Border in CSS

The CSS border property enables the styling of an element's border by setting its width, style, and color, allowing for customizable visual boundaries in web design.

1. Border Style

- CSS border-top style Property
- border-right-style Property
- border-bottom-style Property
- border-left-style Property

2. Border Width

3. border-top-width Property
4. border-right-width Property
5. border-bottom-width Property
6. border-left-width Property

3. Border Color

- border-top-color Property
- border-right-color Property
- border-bottom-color Property
- border-left-color Property

4. Border individual sides

5. Border radius property

Common Border Styles

The border-style property specifies the type of border. None of the other border properties will work without setting the border style.

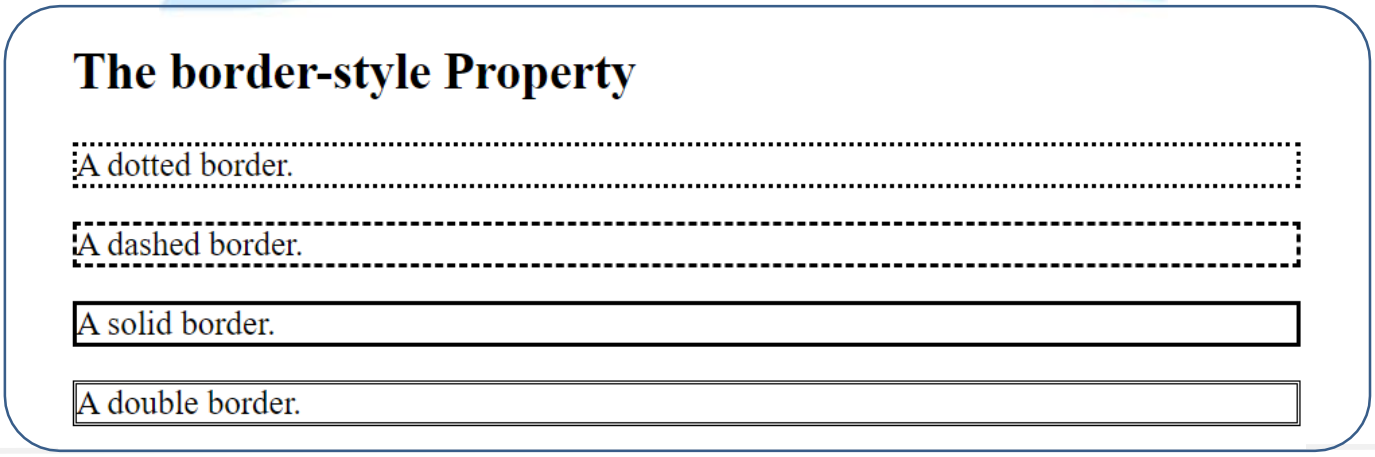
Following are the types of borders:

- **Dotted**: Creates a series of dots.
- **Dashed**: Forms a dashed line.
- **Solid**: Produces a continuous line.
- **Double**: Renders two parallel lines.
- **Groove** and **Ridge**: Create 3D grooved and ridged effects.
- **Inset** and **Outset**: Add 3D inset and outset borders.
- **None**: Removes the border.
- **Hidden**: Hides the border.

Examples of CSS border Style

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p.dotted {border-style: dotted; }

    p.dashed {border-style: dashed;}

    p.solid { border-style: solid; }

    p.double {border-style: double;}

  </style>
</head>

<body>
  <h2>The border-style Property</h2>

  <p class="dotted">A dotted border.</p>

  <p class="dashed">A dashed border.</p>

  <p class="solid">A solid border.</p>

  <p class="double">A double border.</p>

</body>

</html>
```

Browser Output:



CSS Border Width

Border width sets the width of the border. The width of the border can be in px, pt, cm or thin, medium, and thick.

Example of Border Width

```
<!DOCTYPE html>
<html>
<head>
   <style>
      p {
         border-style: solid;
         border-width: 8px;
      }
   </style>
</head>

<body>
   <p> Embedded Full Stack IoT </p>
   <p> Border properties </p>
</body>
</html>
```

Browser Output:

Embedded Full Stack IoT

Border properties

CSS Border Color

This property is used to set the color of the border. Color can be set using the color name, hex value, or RGB value. If the color is not specified border inherits the color of the element itself.

Example of CSS Border Color

```
<!DOCTYPE html>
<html>

<head>
   <style>
      p {
         border-style: solid;
         border-color: red
      }
   </style>
</head>

<body>
  <p> Embedded Full Stack IoT </p>
    <p> color properties </p>

</body>

</html>
```

Browser Output:

Embedded Full Stack IoT

color properties

CSS Border individual sides:

Using border property, we can provide width, style, and color to all the borders separately for that we have to give some values to all sides of the border.

Example for CSS Border individual sides

```
<!DOCTYPE html>
<html>
<head>
<style>
        p {
          border-top-style: dotted;
          border-right-style: solid;
          border-bottom-style: dotted;
          border-left-style: solid;
        }
</style>
</head>
<body>
<h2>Individual Border Sides</h2>
<p>2 different border styles.</p>
</body>
</html>
```

Browser Output:

Individual Border Sides

2 different border styles.

Border radius property

The CSS border-radius property rounds the corners of an element's border, creating smoother edges, with values specifying the curvature radius.

Example of Border radius property

```
<!DOCTYPE html>
<html>
<head>
   <style>
            h1 {
               border-style: solid;
               text-align: center;
               background: green;
               border-radius: 20px;
            }
   </style>
</head>
<body>
   <h1>Embedded Full Stack IoT</h1>
</body>
</html>
```

Browser Output

CSS Margin

Margins, as defined by the CSS margin property, are the spaces created around an element, setting it apart from its neighboring elements. These margins can be individually set for each side – top, right, bottom, and left. The values for these margins can be specified in lengths (e.g., px, rem, em, ex, vh, vw, etc.), percentages (relative to the element's width), or auto (calculated by the browser). Interestingly, margins also allow negative values.

Margin Values

- Length (e.g., px, rem, em, ex, vh, vw, etc)
- Percentage (relative to the element's width)
- auto (calculated by the browser)
- margin allows negative values.

Margin Properties

- **margin-top:** Sets the top margin of an element.
- **margin-right:** Sets the right margin of an element.
- **margin-bottom:** Specifies the margin at the bottom of an element.
- **margin-left:** Determines the width of the margin on the left side of an element.

Syntax:

```
body {
   margin: value;
}
```

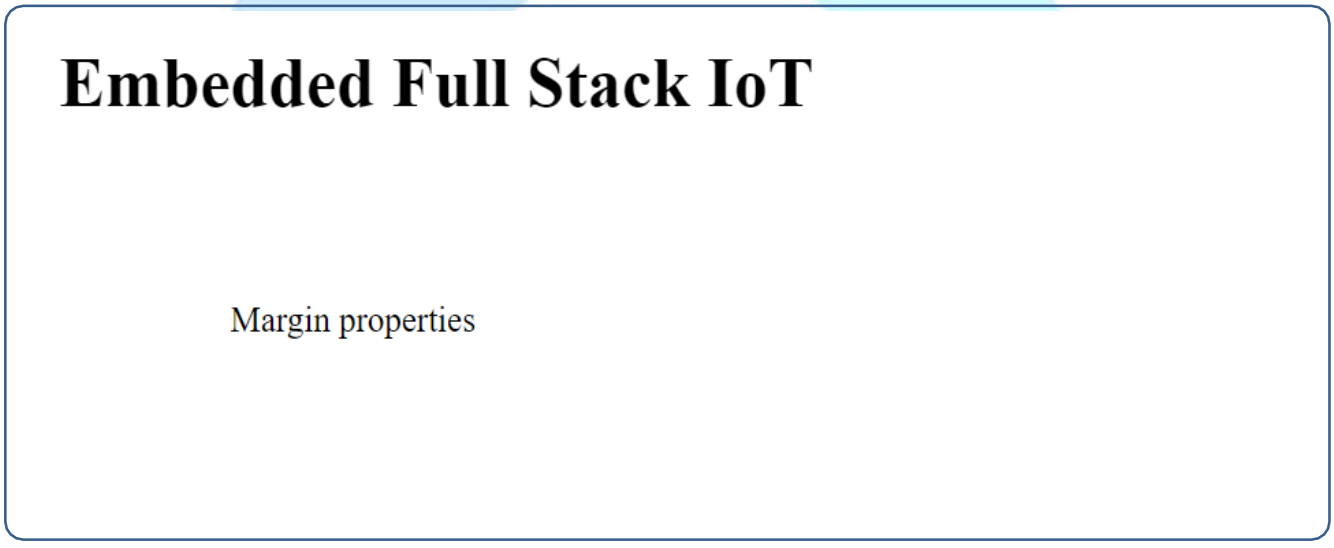
Example of margin property with 4 values:

margin: 40px 100px 120px 80px;

- top margin = 40px
- right margin = 100px
- bottom margin = 120px
- left margin = 80px

```
<!DOCTYPE html>
<html>
<head>
   <style>
            p {
               margin: 80px 100px 50px 80px;
            }
   </style>
</head>
<body>
   <h1> Embedded Full Stack IoT </h1>
   <p> Margin properties </p>
</body>
</html>
```

Browser Output:



Example of margin property with 3 values:

margin: 40px 100px 120px;

- top = 40px
- right and left = 100px
- bottom = 120px

```
<!DOCTYPE html>
<html>
<head>
   <style>
      p { margin: 80px 50px 100px; }
   </style>
</head>
<body>
   <h1> Embedded Full Stack IoT </h1>
    <p> Margin properties </p>
</body>
</html>
```

Browser output:

Example of margin property with 2 values:

margin: 40px 100px;

- top and bottom = 40px;
- left and right = 100px;

```
<!DOCTYPE html>
<html>
<head>
   <style>
      P{  margin: 100px 150px; }
   </style>
</head>
<body>
  <h1> Embedded Full Stack IoT </h1>
   <p> Margin properties </p>
</body>
</html>
```

Browser Output:

Example of margin property with 1 value:

margin: 40px;

- top, right, bottom and left = 40px

```
<!DOCTYPE html>
<html>
<head>
   <style>
      p { margin: 40px; }
   </style>
</head>
<body>
  <h1> Embedded Full Stack IoT </h1>
   <p> Margin properties </p>
</body>
</html>
```

Browser Output:

CSS Padding

CSS Padding property is used to create space between the element's content and the element's border. It only affects the content inside the element.

CSS padding is different from CSS margin as the margin is the space between adjacent element borders and padding is the space between content and element's border.

We can independently change the top, bottom, left, and right padding using padding properties.
CSS Padding Properties

CSS provides properties to specify padding for individual sides of an element which are defined as follows:

- **padding-top:** Sets the padding for the top side of the element.
- **padding-right:** Sets the padding for the right side of the element.
- **padding-bottom:** Sets the padding for the bottom side of the element.
- **padding-left:** Sets the padding for the left side of the element.

Padding properties can have the following padding values:

- Length- in cm, px, pt, etc.
- Width- % width of the element.
- inherit- inherit padding from the parent element

Syntax:

```
.myDiv {
   padding-top: 80px;
   padding-right: 100px;
   padding-bottom: 50px;
   padding-left: 80px;
}
```

Shorthand Property for Padding in CSS

The Shorthand Padding Property in CSS allows you to set the padding on all sides (top, right, bottom, left) of an element in a single line with some combinations, so we can easily apply padding to our targeted element.

There are four cases while using shorthand property:

1. If the padding property has one value.
2. If the padding property contains two values.
3. If the padding property contains three values.
4. If the padding property contains four values.

CSS Shorthand Padding Property for One Value:

If the padding property has one value, then it applies padding to all sides of an element.

For example padding: 20px applies 20 pixels of padding to all sides equally.

Syntax:

```
.element {
   /* Applies 20px padding to all sides */
   padding: 20px;
}
```

CSS Shorthand Padding Property for Two Values:

If the padding property contains two values, then the first value applies to the top and bottom padding, and the second value applies to the right and left padding.

For Example – padding: 10px 20px i.e. top and bottom padding are 10px while right and left padding is 20px.

Syntax:

```
.element {
   /* Applies 10px padding to top and bottom,
   20px padding to right and left */
   padding: 10px 20px;
}
```

CSS Shorthand Padding Property for Three Values

If the padding property contains three values, then the first value sets the top padding, the second value sets the right and left padding, and the third value sets the bottom padding.

For Example – padding: 10px 20px 30px;

- top padding is 10px
- right and left padding is 20px
- bottom padding is 30px

Syntax:

```
.element {
   /* Applies 10px padding to top,
   20px padding to right and left,
   30px padding to bottom */
   padding: 10px 20px 30px;
}
```

CSS Shorthand Padding Property Having Four Values

If the padding property contains four values, then the first value sets the top padding, the second value sets the right padding, the third value sets the bottom padding, and the fourth value sets the left padding.:

For Example – padding: 10px 20px 15px 25px;

- top padding is 10px
- right padding is 5px
- bottom padding is 15px
- left padding is 20px

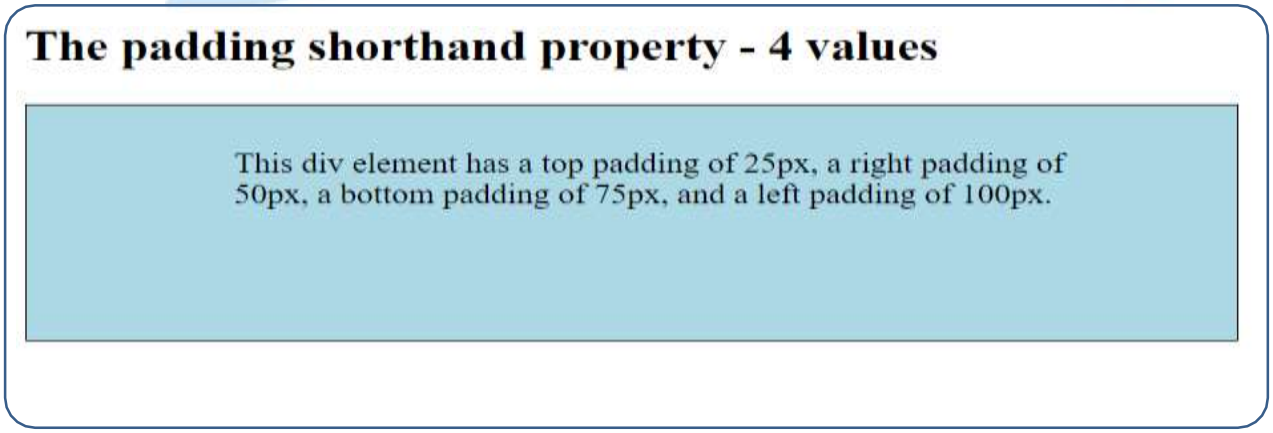
Syntax:

```
.element {
   /* Applies 10px padding to top,
   20px padding to right,
   15px padding to bottom,
   and 25px padding to left */
   padding: 10px 20px 15px 25px;
}
```

Example for the padding shorthand property with four values:

```
<!DOCTYPE html>
<html>
<head>
<style>
        div {
          border: 1px solid black;
          padding: 25px 50px 75px 100px;
          background-color: lightblue;
        }
</style>
</head>
<body>
<h2>The padding shorthand property - 4 values</h2>
<div>This div element has a top padding of 25px, a right padding of 50px, a bottom padding of 75px, and a left padding of 100px.</div>
</body>
</html>
```

Browser Output:



CSS Text

CSS Text Formatting refers to applying styles to text elements to control appearance and layout. This includes properties for color, alignment, decoration, indentation, justification, shadows, spacing, and direction. These properties enhance readability and aesthetics, improving the presentation of textual content on web pages.

Syntax:

The Syntax to write this property:

```
Selector {
   property-name : /*value*/
}
```

CSS Text Formatting Properties:

These are the following text formatting properties.

Property	
text-color	Sets color of text decorations like overlines, underlines, and line-throughs.
text-decoration-line	Sets various text decorations like underline, overline, line-through.
text-decoration-style	Combines text-decoration-line and text-decoration-color properties.

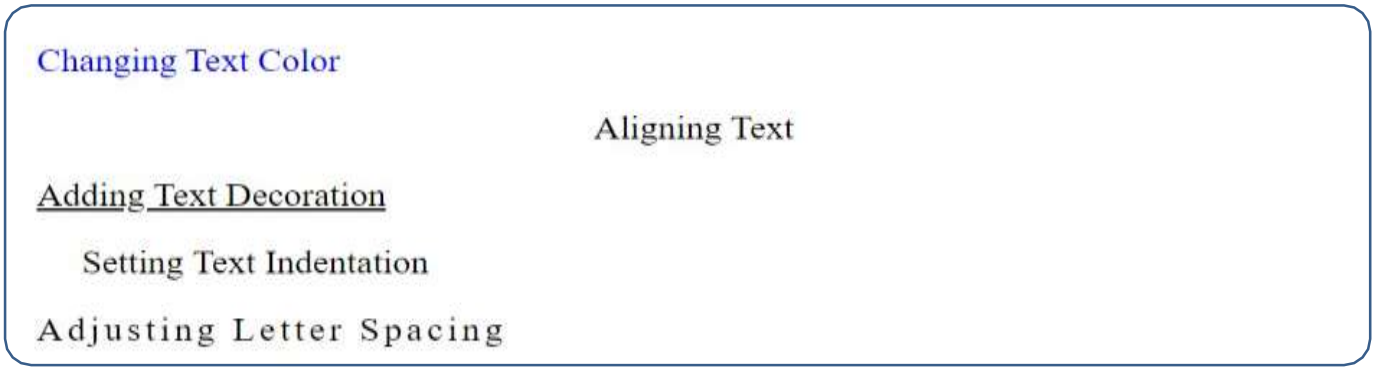
Property	Description
text-indent	Indents first line of paragraph.
text-justify	Justifies text by spreading words into complete lines.
text-overflow	Specifies hidden overflow text.
text-transform	
text-shadow	
letter-spacing	
line-height	
direction	
word-spacing	

Example for CSS Text Formatting

In this example we demonstrates basic text formatting using CSS. It includes paragraphs with different styles: changing color, aligning center, adding underline, setting indentation, and adjusting letter spacing for improved readability and aesthetics.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Basic Text Formatting</title>
        <style>
            .text-color {color: blue;}
            .text-align-center {text-align: center;}
            .text-decoration {text-decoration: underline;}
            .text-indent {text-indent: 20px;}
            .letter-spacing {letter-spacing: 2px;}
        </style>
</head>
<body>
    <p class="text-color">Changing Text Color</p>
    <p class="text-align-center">Aligning Text</p>
    <p class="text-decoration">Adding Text Decoration</p>
    <p class="text-indent">Setting Text Indentation</p>
    <p class="letter-spacing">Adjusting Letter Spacing</p>
</body>
</html>
```

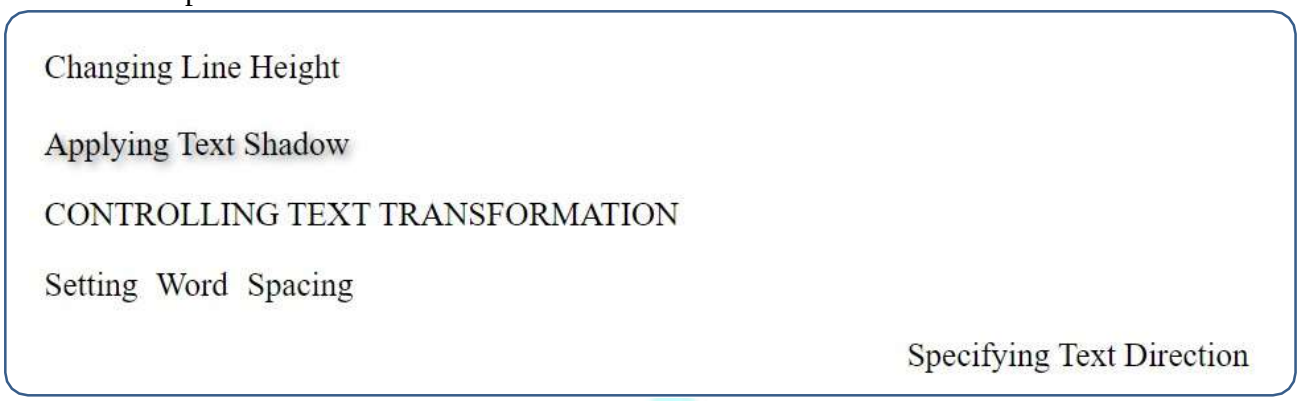
Browser Output:



Example for CSS text advanced properties

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Advanced Text Formatting</title>
         <style>
            .line-height {line-height: 1.5;}
            .text-shadow {text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);}
            .text-transform {text-transform: uppercase;}
            .word-spacing {word-spacing: 5px;}
            .text-direction {direction: rtl;}
         </style>
</head>
<body>
   <p class="line-height">Changing Line Height</p>
   <p class="text-shadow">Applying Text Shadow</p>
   <p class="text-transform">Controlling Text Transformation</p>
   <p class="word-spacing">Setting Word Spacing</p>
   <p class="text-direction">Specifying Text Direction</p>
</body> </html>
```

Browser output



CSS Fonts

CSS fonts offer a range of options to style the text content within HTML elements. It gives you the ability to control different aspects of fonts, including font family, font size, font weight, font style, and more.

There are many font properties in CSS which are mentioned and briefly discussed below:

- **CSS font-family Property:** The font-family property specifies the font of an element.
- **CSS font-style Property:** If we want to give design to any type of text then we can make use of CSS font-style property.
- **CSS font-weight Property:** The font-weight property of the CSS is used to set the weight or thickness of the font being used with the HTML Text.
- **CSS font-variant Property:** The font-variant property is used to convert all lowercase letters into uppercase letters.
- **CSS font-size Property:** The font-size property in CSS is used to set the font size of the text in an HTML document.
- **CSS font-stretch Property:** The font-stretch property in CSS is used to set the text wider or narrower.
- **CSS font-kerning Property:** This property is used to control the usage of the Kerning Information that has been stored in the Font.

Example for few CSS Font properties.

```
<!DOCTYPE html>
<html>
<head>
   <title>CSS Font</title>
   <style>
            .iot {
               font-family: "Arial, Helvetica, sans-serif";
               font-size: 60px;
               color: #090;
               text-align: center;
            }
            .eiot {
               font-family: "Comic Sans MS", cursive, sans-serif;
               font-variant:small-caps;
               text-align: center;
            }
   </style>
</head>
<body>
   <div class="iot">Internet of Things</div>
   <div class="eiot">Embedded Full Stack IoT</div>
</body>
</html>
```

Browser Output:

Examples for CSS Font Collection :

font-family: It is used to set the font type of an HTML element. It holds several font names as a fallback system.

Syntax: font-family: "font family name";

```
<!DOCTYPE html>
<html>
<head>
   <title>CSS Font</title>
   <style>
            .iot {
                font-family: "Times New Roman";
                font-weight: bold;
                font-size: 40px;
                color: #090;
                text-align: center;
             }
            .eiot {
                font-family: "Comic Sans MS", cursive, sans-serif;
                text-align: center;
             }
   </style>
</head>
<body>
   <div class="iot">Internet of Things</div>
   <div class="eiot">Embedded Full Stack IoT</div>
</body>
</html>
```

Browser Output:

Internet of Things

Embedded Full Stack IoT

Examples for CSS Font Style Collection:

font-style: It is used to specify the font style of an HTML element. It can be "normal, italic or oblique".

Syntax: font-style: style name;

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Font</title>
  <style>
        .iot {
            font-style: normal;
            font-family: "Times New Roman";
            font-weight: bold;
            font-size: 40px;
            color: #00f;
            text-align: center;  }
         .eiot {
            font-style: italic;
            text-align: center; }
  </style></head>
<body>
  <div class="iot">Internet of Things</div>
  <div class="eiot">Embedded Full Stack IoT</div>
</body></html>
```

Browser Output:

Internet of Things

Embedded Full Stack IoT

Examples for CSS Font Weight Collection:

font-weight: It is used to set the boldness of the font. Its value can be "normal, bold, lighter, bolder".

Syntax: font-weight: font weight value;

```
<!DOCTYPE html>
<html>
<head>
   <title>CSS Font</title>
   <style>
        .iot { font-weight: bold;
                font-style: normal;
                font-family: "Times New Roman";
                font-size: 40px;
                color: #f00;
                text-align: center;  }
        .eiot { font-weight: normal;
                text-align: center;}
   </style>
</head>
<body>
   <div class="iot">Internet of Things</div>
   <div class="eiot">Embedded Full Stack IoT</div>
</body>
</html>
```

Browser Output:

Internet of Things

Embedded Full Stack IoT

Examples for CSS Font Variant Collection:

font-variant: It is used to create the small-caps effect. It can be “normal or small-caps”.

Syntax: font-variant: font variant value;

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Font</title>
  <style>
        .iot {
          font-variant: small-caps;
          font-weight: bold;
          font-family: "Times New Roman";
          font-size: 40px;
          color: #800080;
          text-align: center; }
        .eiot {
          font-variant: normal;
          text-align: center;}
  </style>
</head>
<body>
  <div class="iot">Internet of Things</div>
  <div class="eiot">Embedded Full Stack IoT</div>
</body>
</html>
```

Browser Output:

INTERNET OF THINGS

Embedded Full Stack IoT

Examples for CSS Font Size Collection:

font-size: It is used to set the font size of an HTML element. The font-size can be set in different ways like in "pixels, percentage, em or we can set values like small, large" etc.

Syntax: font-size: font size value;

```
<!DOCTYPE html>
<html>
<head>
   <title>CSS Font</title>
   <style>
                .iot{
                  font-size: 40px;
                  font-weight: bold;
                  font-family: "Times New Roman";
                  color: #800000;
                  text-align: center; }
               .eiot {
                  font-size: 1.2em;
                  text-align: center; }
   </style> </head>
<body>
   <div class="iot">Internet of Things</div>
   <div class="eiot">Embedded Full Stack IoT</div>
</body>
</html>
```

Browser Output:

Internet of Things

Embedded Full Stack IoT

CSS Icons

CSS Icons from various libraries can be effortlessly styled and customized with CSS, allowing for alterations in size, color, shadow, and more. These icons serve as intuitive graphical elements, enhancing navigation and conveying specific meanings.

There are 3 types of icon libraries available, namely

1. Font Awesome Icons
2. Google Icons
3. Bootstrap Icons

We will include the required CDN link from the available icon library, which will help us to use the pre-defined icon classes or we can customize it using the CSS.

Font Awesome Icons

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the <head> section of your HTML page:

```
<script src="https://kit.fontawesome.com/yourcode.js" crossorigin="anonymous"></script>
```

```
<!DOCTYPE html>
<html>
<head>
<title>Font Awesome Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>
<!--Get your own code at fontawesome.com-->
</head>
<body>
<h1>Font Awesome icon library</h1>

<p>Some Font Awesome icons:</p>
<i class="fas fa-cloud"></i>
<i class="fas fa-heart"></i>
<i class="fas fa-car"></i>
<i class="fas fa-file"></i>
<i class="fas fa-bars"></i>

<p>Styled Font Awesome icons (size and color):</p>
<i class="fas fa-cloud" style="font-size:24px;"></i>
<i class="fas fa-cloud" style="font-size:36px;"></i>
<i class="fas fa-cloud" style="font-size:48px;color:red;"></i>
<i class="fas fa-cloud" style="font-size:60px;color:lightblue;"></i>
</body>
</html>
```

Browser Output

Bootstrap Icons

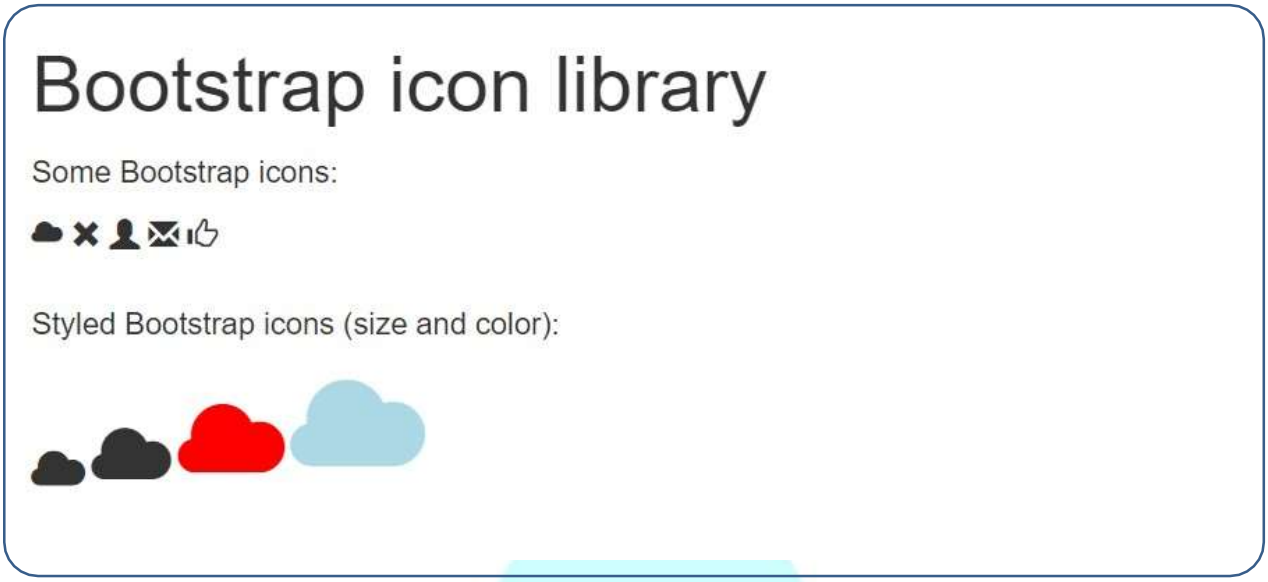
To use the Bootstrap glyphicons, add the following line inside the <head> section of your HTML page:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
<title>Bootstrap Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body class="container">
<h1>Bootstrap icon library</h1>
<p>Some Bootstrap icons:</p>
<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>
<br><br>
<p>Styled Bootstrap icons (size and color):</p>
<i class="glyphicon glyphicon-cloud" style="font-size:24px;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:36px;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:48px;color:red;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:60px;color:lightblue;"></i>
</body>
</html>
```

Browser output:



Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
<title>Google Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>
<h1>Google icon library</h1>
<p>Some Google icons:</p>
<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>
<br><br>
<p>Styled Google icons (size and color):</p>
<i class="material-icons" style="font-size:24px;">cloud</i>
<i class="material-icons" style="font-size:36px;">cloud</i>
<i class="material-icons" style="font-size:48px;color:red;">cloud</i>
<i class="material-icons" style="font-size:60px;color:lightblue;">cloud</i>
</body>
</html>
```

Browser Output:

Google icon library

Some Google icons:

Styled Google icons (size and color):

CSS Colors

CSS Colors are an essential part of web design, providing the ability to bring your HTML elements to life. This feature allows developers to set the color of various HTML elements, including font color, background color, and more.

Color Format	Description
Color Name	
RGB Format	
RGBA Format	
Hexadecimal Notation	
HSLA	The HSLA color property is similar to the HSL property, but it includes an Alpha component that specifies the transparency of elements. Syntax: h1 { color: hsla(H, S, L, A); }

CSS Opacity

The CSS opacity property is used to specify the transparency of an element. In simple word, you can say that it specifies the clarity of the image.

In technical terms, Opacity is defined as degree in which light is allowed to travel through an object.

Opacity setting is applied uniformly across the entire object and the opacity value is defined in term of digital value less than 1. The lesser opacity value displays the greater opacity. Opacity is not inherited.

Example for CSS Opacity

```
<!DOCTYPE html>
<html>
<head>
<style>
        .op1{
          opacity: 0.5;
        }
</style>
</head>
<body>
<h1>Image Transparency</h1>
<p>The opacity property specifies the transparency of an element. The lower the value,
the more transparent:</p>
<p>Image without opacity:</p>
<img src="img_forest.jpg" alt="Forest" width="170" height="100" >
<p>Image with 50% opacity:</p>
<img src="img_forest.jpg" alt="Forest" width="170" height="100" class="op1" >
</body>
</html>
```

Browser Output:

Image Transparency

The opacity property specifies the transparency of an element. The lower the value, the more transparent:

Image without opacity:

Image with 50% opacity:

Bootstrap-5 Framework

Bootstrap is a free and open-source collection of CSS and JavaScript/jQuery code used for creating dynamic website layouts and web applications. As one of the most popular front-end frameworks, Bootstrap offers a comprehensive set of predefined CSS classes and components that simplify the development of responsive, mobile-first websites.

Bootstrap's versatility makes it suitable for mobile-friendly web development. The framework ensures that web pages are adaptable to various screen sizes, providing a consistent experience across different devices. This responsiveness is achieved through a variety of classes designed to handle different layouts and elements dynamically.

The framework is available for free and can be integrated into projects in two main ways: by downloading the zip files and including Bootstrap's libraries/modules locally, or by directly linking to the online version through a CDN (Content Delivery Network).

Bootstrap 5, the latest stable version, was officially released on June 16, 2020, after several months of feature refinement. It continues the tradition of being a robust, mobile-first framework, compatible with the latest stable releases of all major browsers and operating systems. Bootstrap 5 emphasizes responsiveness by default, ensuring that websites look great on all devices, from small smartphones to large desktop monitors.

Bootstrap is an essential toolkit for modern web development, providing an array of CSS and JavaScript components that streamline the process of creating responsive, mobile-first websites and applications. Its widespread use and continuous updates make it a reliable choice for developers looking to build dynamic and accessible web projects.

Bootstrap Versions

Bootstrap 5 (released 2021) is the newest version of Bootstrap (released 2013); with new components, faster stylesheet and more responsiveness.

Bootstrap 5 supports the latest, stable releases of all major browsers and platforms. However, Internet Explorer 11 and down is not supported.

The main differences between Bootstrap 5 and Bootstrap 3 & 4, is that Bootstrap 5 has switched to vanilla JavaScript instead of jQuery.

Key Features of Bootstrap 5

- **Responsive Grid System:** Bootstrap 5 continues to use its powerful grid system based on flexbox. This system allows developers to create flexible and responsive layouts that automatically adapt to different screen sizes.

- **Predefined CSS Classes:** The framework offers a vast collection of predefined CSS classes that cover typography, forms, buttons, navigation, and other essential design elements. These classes facilitate rapid development and ensure consistent styling across different components.

- **JavaScript Components:** Bootstrap 5 includes a variety of JavaScript components such as modals, carousels, tooltips, and popovers. These components add interactivity to web pages, enhancing the user experience without requiring extensive JavaScript coding.

- **Customization and Theming:** Developers can easily customize Bootstrap 5 to match their project's branding by modifying Sass variables and using mixins. The theming system allows for the creation of custom themes, providing flexibility in design.

- **Mobile-First Approach:** Designed with a mobile-first philosophy, Bootstrap 5 ensures that web pages are optimized for mobile devices from the start. This approach helps create a seamless and user-friendly experience on smartphones, tablets, and desktops.

- **Improved Forms:** Bootstrap 5 introduces improved form controls, with enhanced styling and layout options. The new form elements are more customizable and accessible, making it easier to create complex forms.

- **Utility Classes:** A broad range of utility classes in Bootstrap 5 enables quick adjustments to layout, spacing, alignment, and other styling aspects without writing additional CSS. These utility classes simplify the process of fine-tuning designs.

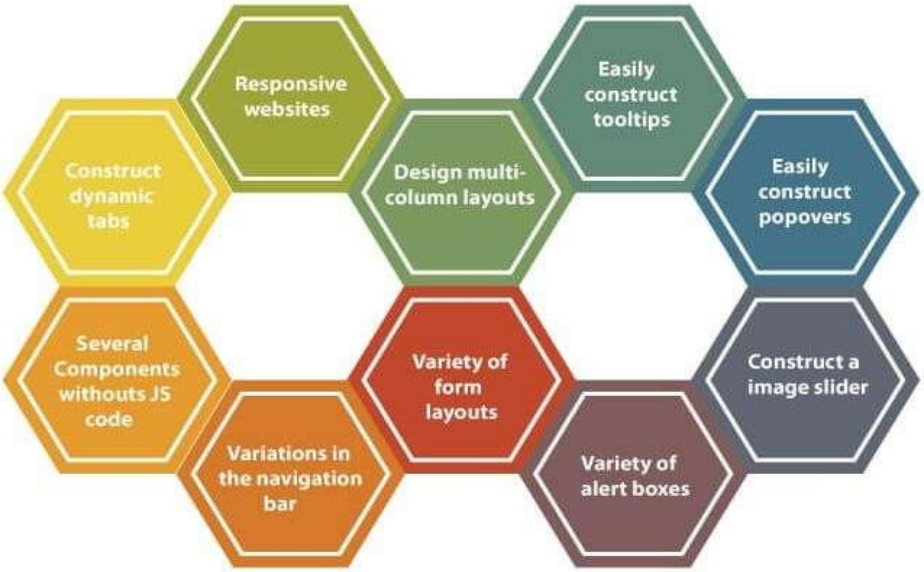
- **No More jQuery Dependency:** One of the significant changes in Bootstrap 5 is the removal of jQuery as a dependency. This shift to vanilla JavaScript reduces the overall size of the framework and improves performance.

- **Icons and SVGs:** Bootstrap 5 supports a wide range of icons through its integration with Bootstrap Icons, a separate library of high-quality SVG icons that can be easily customized and used within projects.

- **Enhanced Documentation:** The documentation for Bootstrap 5 has been significantly improved, providing detailed examples, comprehensive explanations, and robust search functionality. This makes it easier for developers to learn and implement the framework's features.

Advantages of Using Bootstrap 5

- Ease of Use: Bootstrap 5 simplifies the web development process with its extensive set of components and utilities, making it accessible even for developers with limited design skills.
- Consistency: By using predefined classes and components, Bootstrap 5 ensures a consistent look and feel across the entire website.
- Community and Support: As an open-source project with a large and active community, Bootstrap 5 benefits from continuous improvements, community support, and a wealth of shared resources.
- Cross-Browser Compatibility: The framework is tested and compatible with the latest stable releases of all major browsers, ensuring reliable performance across different platforms.



Bootstrap Containers

In Bootstrap, container is used to set the content's margins dealing with the responsive behaviors of your layout. It contains the row elements and the row elements are the container of columns (known as grid system).

Containers are the most basic layout element in Bootstrap and are required when using our default grid system. Containers are used to contain, pad, and (sometimes) center the content within them. While containers can be nested, most layouts do not require a nested container.

Bootstrap comes with three different containers:

- **.container**, which sets a max-width at each responsive breakpoint
- **.container-fluid,** which is width: 100% at all breakpoints
- **.container-{breakpoint},** which is width: 100% until the specified breakpoint

The table below illustrates how each container's max-width compares to the original .container and .container-fluid across each breakpoint.

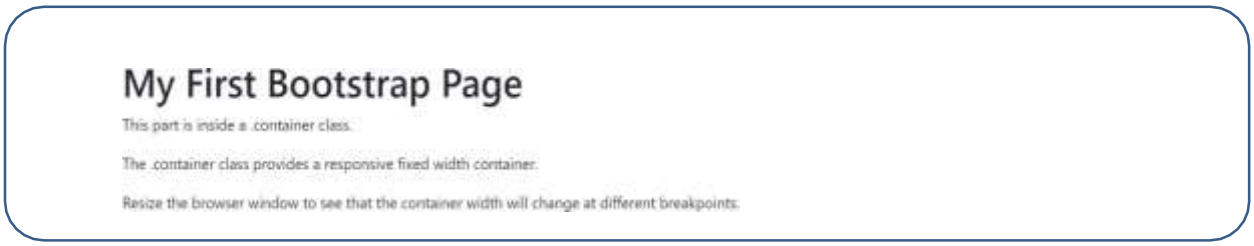
						XX-Large ≥1400px
.container						1320px
			720px			1320px
			720px			
		100%	100%	960px		
		100%	100%	100%	1140px	
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

Default container/ Fixed Container

.container class is a responsive, fixed-width container, meaning its max-width changes at each breakpoint.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This part is inside a .container class.</p>
  <p>The .container class provides a responsive fixed width container.</p>
  <p>Resize the browser window to see that the container width will change at different
breakpoints.</p>
</div>
</body>
</html>
```

Browser Output:



Responsive containers

Responsive containers allow you to specify a class that is 100% wide until the specified breakpoint is reached, after which we apply max-widths for each of the higher breakpoints. For example, .container-sm is 100% wide to start until the sm breakpoint is reached, where it will scale up with md, lg, xl, and xxl.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container-sm bg-secondary">100% wide until small breakpoint</div>
<div class="container-md bg-secondary">100% wide until medium breakpoint</div>
<div class="container-lg bg-secondary">100% wide until large breakpoint</div>
<div class="container-xl bg-secondary">100% wide until extra large breakpoint</div>
<div class="container-xxl bg-secondary">100% wide until extra extra large breakpoint</div>
</body>
</html>
```

Browser output:

Fluid containers

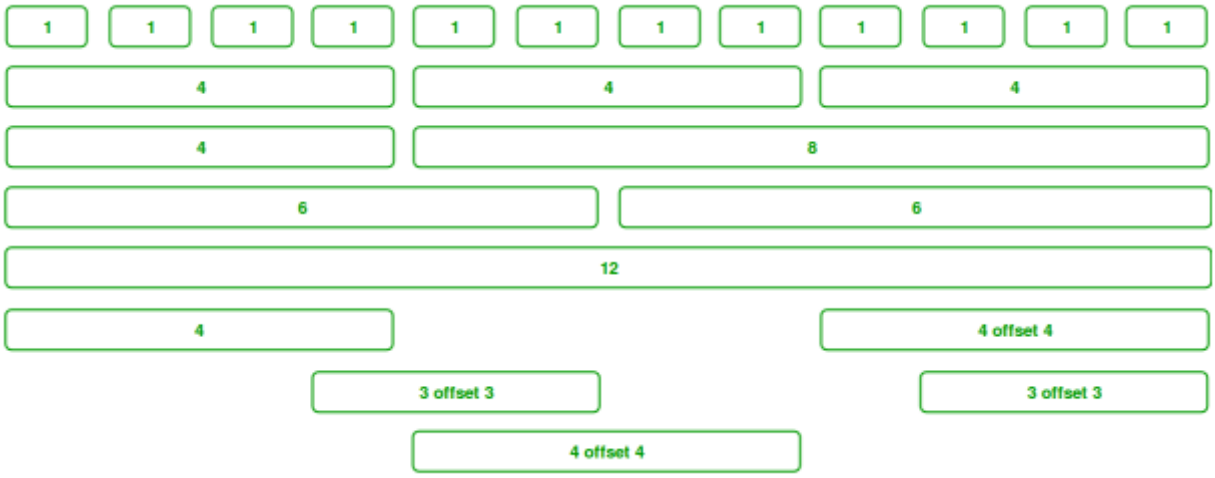
Use .container-fluid for a full width container, spanning the entire width of the viewport.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container-fluid">
  <h1>My First Bootstrap Page</h1>
  <p>This part is inside a .container-fluid class.</p>
  <p>The .container-fluid class provides a full width container, spanning the entire width of the viewport.</p>
</div>
</body>
</html>
```

Browser Output

Bootstrap Grid System

Bootstrap Grid System allows up to 12 columns across the page. You can use each of them individually or merge them together for wider columns. You can use all combinations of values summing up to 12. You can use 12 columns each of width 1, or use 4 columns each of width 3 or any other combination.



Bootstrap's grid system is responsive, and the columns will re-arrange depending on the screen size: On a big screen it might look better with the content organized in three columns, but on a small screen it would be better if the content items were stacked on top of each other.

Grid Classes

The Bootstrap grid system has four classes:

- xs (for phones - screens less than 768px wide)
- sm (for tablets - screens equal to or greater than 768px wide)
- md (for small laptops - screens equal to or greater than 992px wide)
- lg (for laptops and desktops - screens equal to or greater than 1200px wide)

The classes above can be combined to create more dynamic and flexible layouts.

Each class scales up, so if you wish to set the same widths for xs and sm, you only need to specify xs.

Grid System Rules

Some Bootstrap grid system rules:

- Rows must be placed within a .container (fixed-width) or .container-fluid (full-width) for proper alignment and padding
- Use rows to create horizontal groups of columns
- Content should be placed within columns, and only columns may be immediate children of rows
- Predefined classes like .row and .col-sm-4 are available for quickly making grid layouts
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on .rows
- Grid columns are created by specifying the number of 12 available columns you wish to span. For example, three equal columns would use three .col-sm-4
- Column widths are in percentage, so they are always fluid and sized relative to their parent element

Basic Structure of a Bootstrap Grid

The following is a basic structure of a Bootstrap grid:

```
<div class="container">
  <div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    ...
  </div>
</div>
```

So, to create the layout you want, create a container ( ). Next, create a row ( ). Then, add the desired number of columns (tags with appropriate .col-*-* classes). Note that numbers in .col-*-* should always add up to 12 for each row.

Grid Options

While Bootstrap uses ems or rems for defining most sizes, pxs are used for grid breakpoints and container widths. This is because the viewport width is in pixels and does not change with the font size.

See how aspects of the Bootstrap grid system work across multiple devices with a handy table.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Bootstrap Example for Three Equal Columns

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container-fluid">
  <h1>Hello World!</h1>
  <p>Resize the browser window to see the effect.</p>
  <p>The columns will automatically stack on top of each other when the screen is less than
768px wide.</p>
  <div class="row">
    <div class="col-sm-4" style="background-color:lavender;">.col-sm-4</div>
    <div class="col-sm-4" style="background-color:lavenderblush;">.col-sm-4</div>
    <div class="col-sm-4" style="background-color:lavender;">.col-sm-4</div>
  </div>
</div>
</body></html>
```

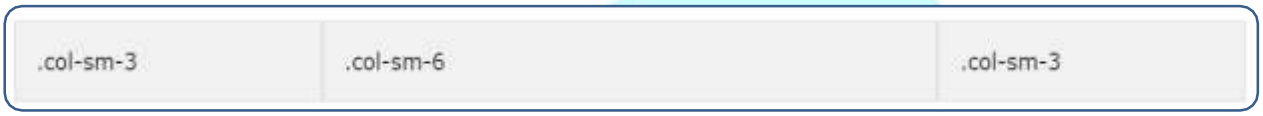
Browser Output:

Three Unequal Columns

Syntax:

```
<div class="container-fluid">

        <div class="row">
          <div class="col-sm-3">.col-sm-3</div>
          <div class="col-sm-6">.col-sm-6</div>
          <div class="col-sm-3">.col-sm-3</div>
        </div>

</div>
```

Browser Output:



Two Unequal Columns

Syntax:

```
<div class="container">

        <div class="row">
                <div class="col-sm-4">.col-sm-4</div>
                <div class="col-sm-8">.col-sm-8</div>
        </div>

</div>
```

Browser Output:

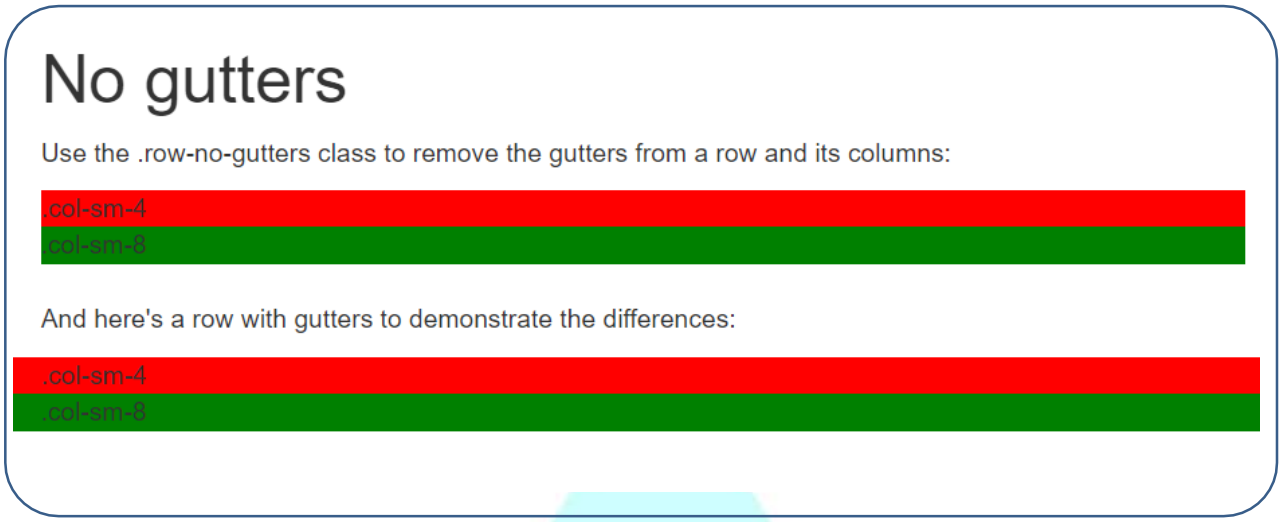
.col-sm-4	.col-sm-8

No gutters

Use the .row-no-gutters class to remove the gutters from a row and its columns:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container-fluid">
  <h1>No gutters</h1>
  <p>Use the .row-no-gutters class to remove the gutters from a row and its columns:</p>
  <div class="row row-no-gutters">
    <div class="col-sm-4" style="background-color:red;">.col-sm-4</div>
    <div class="col-sm-8" style="background-color:green;">.col-sm-8</div>
  </div>
  <br>
  <p>And here's a row with gutters to demonstrate the differences:</p>
  <div class="row">
    <div class="col-sm-4" style="background-color:red;">.col-sm-4</div>
    <div class="col-sm-8" style="background-color:green;">.col-sm-8</div>
  </div>
</div> </body> </html>
```

Browser Output:



Bootstrap Text/Typography

Bootstrap's Default Settings

- Bootstrap's global default font-size is 14px, with a line-height of 1.428.
- This is applied to the <body> element and all paragraphs (<p>).
- In addition, all <p> elements have a bottom margin that equals half their computed line-height (10px by default).

Bootstrap Typography provides a standardized and flexible approach to text styling, offering various classes for headings, paragraphs, and inline text elements. It ensures consistent typography across different devices and screen sizes, enhancing readability and aesthetics.

Typography can be used to create:

- Headings
- Subheadings
- Text and Paragraph font color, font type, and alignment
- Lists
- Other inline elements

Headings

All HTML headings, <h1> through <h6>, are available. .h1 through .h6 classes are also available, for when you want to match the font styling of a heading but cannot use the associated HTML element.

Display headings

Traditional heading elements are designed to work best in the meat of your page content. When you need a heading to stand out, consider using a **display heading**—a larger, slightly more opinionated heading style. .display-1 through .display-6 classes are available for display headings

Abbreviations

Stylized implementation of HTML's <abbr> element for abbreviations and acronyms to show the expanded version on hover. Abbreviations have a default underline and gain a help cursor to provide additional context on hover and to users of assistive technologies.

Lead

Make a paragraph stand out by adding .lead.

Some classes and Tags to implement the typography feature of bootstrap:

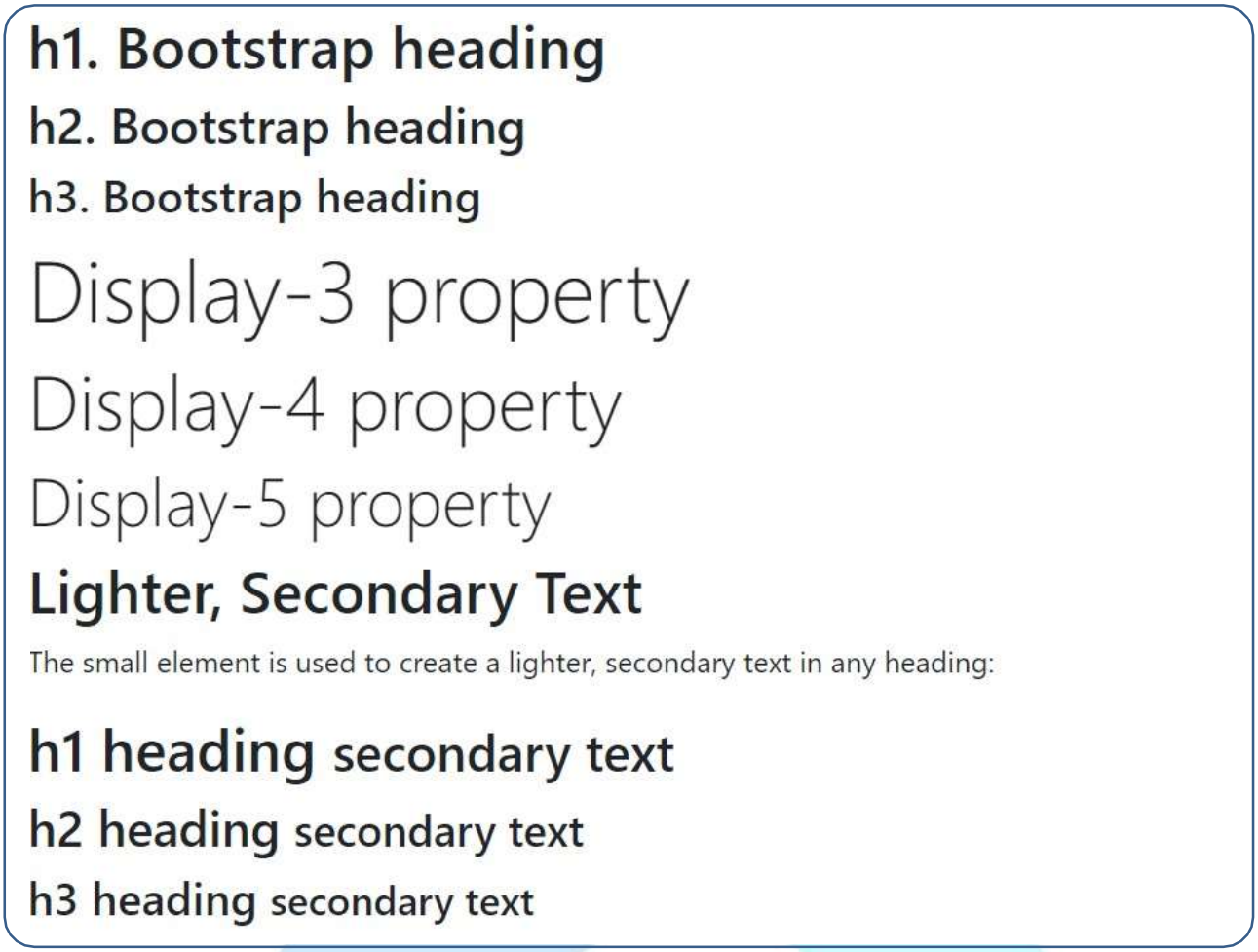
Inline Element	
lead	Makes a paragraph stand out visually
mark	Highlights the text
small	Creates secondary subheadings
initialism	Renders abbreviations in slightly smaller text size

Inline Element	Description
blockquote	Indicates quoted content
blockquote-footer	Provides footer details for identifying the source of the quote
text-center	Aligns text to the center
list-inline	
text-truncate	
text-uppercase	
text-lowercase	
text-capitalize	
pre-scrollable	
dl-horizontal	
text-left	Aligns text to the left

Example 1:
In this example it includes headings from h1 to h3 with corresponding classes, along with display classes (display-3, display-4, and display-5) for different styles and the <small> element is used for secondary text.

```
<!DOCTYPE html>
<html>
   <head>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
      <title>Text-muted</title>
   </head>
   <body>
         <p class="h1">h1. Bootstrap heading</p>
         <p class="h2">h2. Bootstrap heading</p>
         <p class="h3">h3. Bootstrap heading</p>
         <h3 class="display-3">Display-3 property</h3>
         <h3 class="display-4"> Display-4 property</h3>
         <h3 class="display-5"> Display-5 property </h3>
         <h1>Lighter, Secondary Text</h1>
         <p>The small element is used to create a lighter, secondary text in any heading:</p>
         <h1>h1 heading <small> secondary text </small> </h1>
         <h2>h2 heading <small> secondary text </small> </h2>
         <h3>h3 heading <small> secondary text </small> </h3>
   </body>
</html>
```

Browser Output

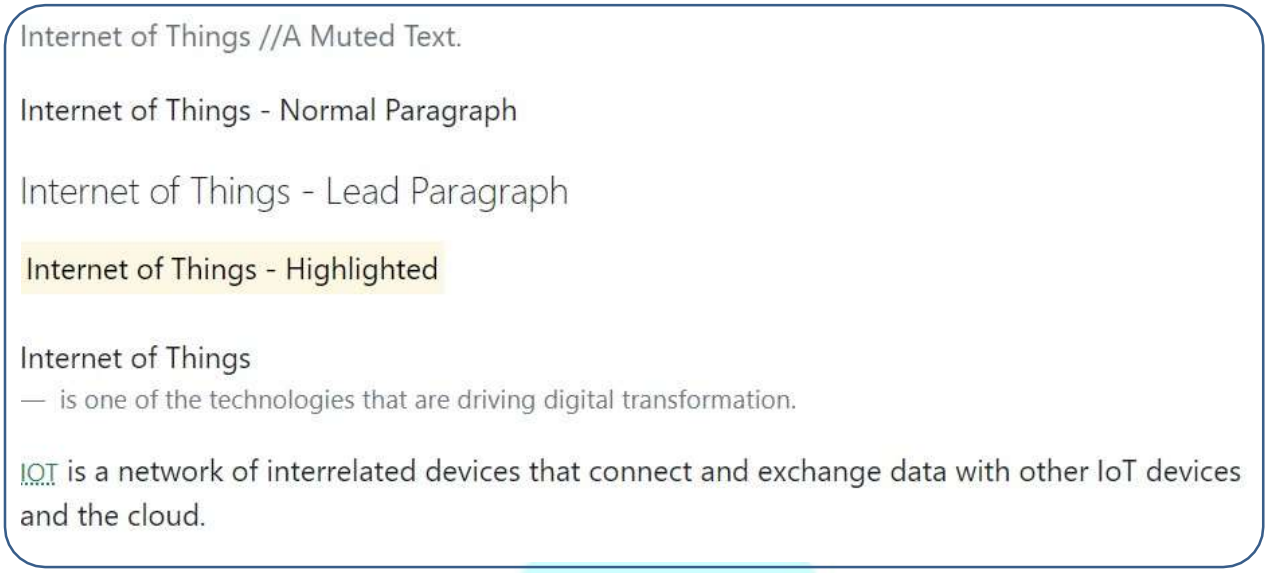


Example 2:

In this example includes muted text, lead paragraph, highlighted text using the mark element, blockquote with footer, and an abbreviation with initialism class.

```
<!DOCTYPE html>

<html>

   <head>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

      <title>Bootstrap Typography</title>

   </head>

   <body>

      <!-- Muted text-->

      <p class="text-muted">Internet of Things //A Muted Text.</p>

      <p>Internet of Things - Normal Paragraph</p>

      <!-- using lead class-->

      <p class="lead">Internet of Things - Lead Paragraph </p>

      <!-- using mark-->

      <mark>Internet of Things - Highlighted</mark> <br><br>

      <!-- using blockquote tag and blockquote-footer class.-->

      <blockquote>Internet of Things</blockquote>

            <blockquote class="blockquote-footer"> is one of the technologies that are driving
         digital transformation. </blockquote>

      <!-- using initialism class-->

         <abbr title="Internet of Things" class="text-success initialism" >IoT</abbr> is a network
         of interrelated devices that connect and exchange data with other IoT devices and the
         cloud.

   </body>

</html>
```

Browser Output



Bootstrap Jumbotron

A jumbotron was introduced in Bootstrap 3 as a big padded box for calling extra attention to some special content or information.

Bootstrap 4 Jumbotron

A jumbotron indicates a big grey box for calling extra attention to some special content or information. Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.

Example for Bootstrap 4

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Jumbotron Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container">
  <div class="jumbotron">
      <h1>Internet of Things</h1>
      <p> Internet of Things is one of the technologies that are driving digital transformation.  </p
</body>
</html>
```

Browser Output:

Full-width Jumbotron

If you want a full-width jumbotron without rounded borders, add the .jumbotron-fluid class and a .container or .container-fluid inside of it:

Example for Full width Jumbotron

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Bootstrap Example</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">
</head>
<body>
<div class="jumbotron jumbotron-fluid">
 <div class="container">
   <h1>Internet of Things</h1>
    <p> Internet of Things is one of the technologies that are driving digital transformation. </p>
 </div>
</div>
</body> </html>
```

Browser Output

Bootstrap 5 Jumbotron

Jumbotrons are no longer supported in Bootstrap 5. **However**, you can use a element and add special helper classes together with a color class to achieve the same effect:

Example for Bootstrap 5

```
<!DOCTYPE html>
<html>
<head>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container mt-3">
 <h2>Example of Jumbotron</h2>
 <div class="mt-4 p-5 bg-primary text-white rounded">
  <h1>Internet of Things</h1>
  <p> Internet of Things is one of the technologies that are driving digital transformation. </p>
 </div>
</div>
</body>
</html>
```

Browser Output:

Bootstrap 5 Colors

Text Colors

Bootstrap 5 has some contextual classes that can be used to provide "meaning through colors".

The classes for text colors are: .text-muted, .text-primary, .text-success, .text-info, .text-warning, .text-danger, .text-secondary, .text-white, .text-dark, .text-body (default body color/often black) and .text-light. You can also add 50% opacity for black or white text with the .text-black-50 or .text-white-50 classes:

Background Colors

The classes for background colors are: .bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light.

Table Colors

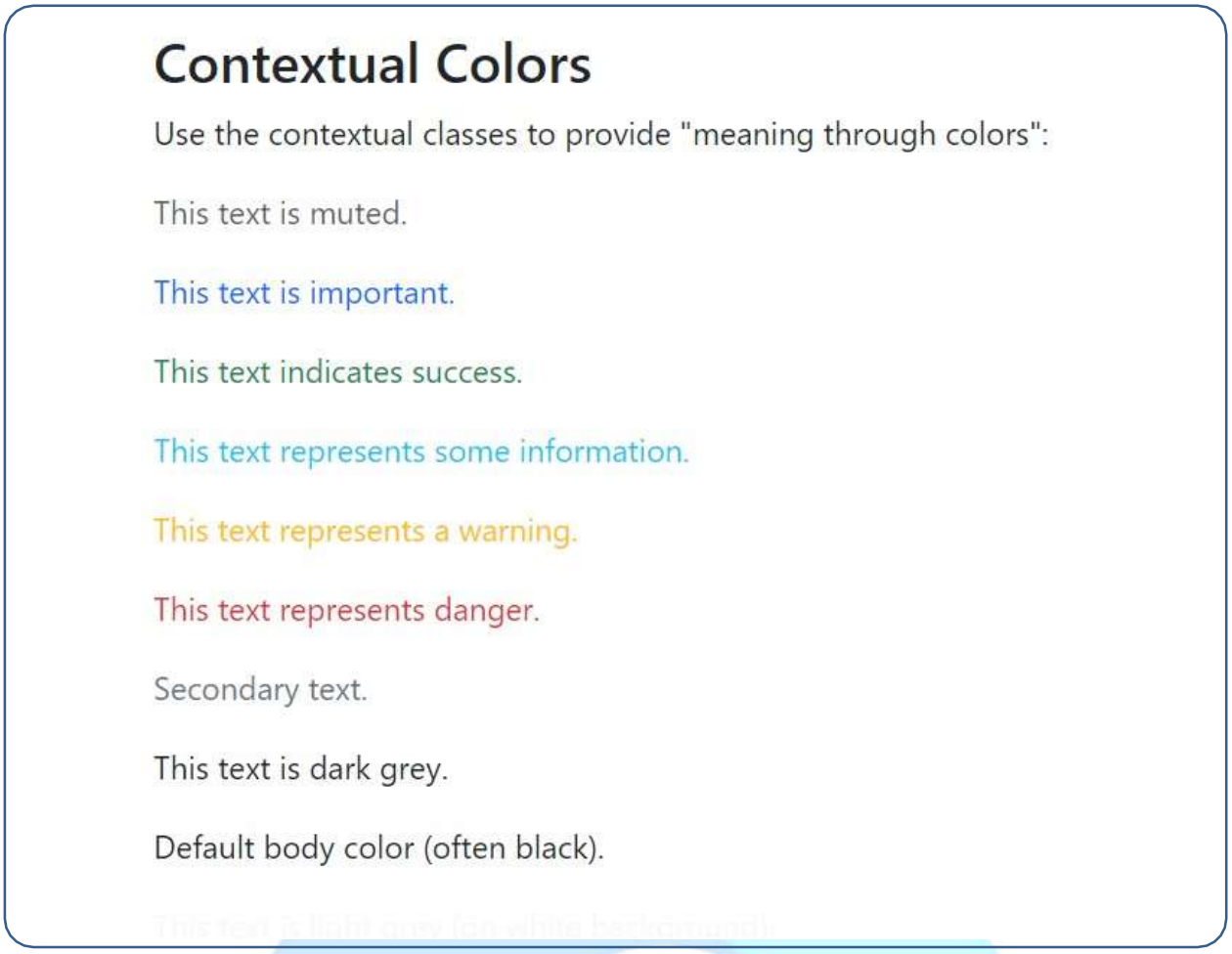
The classes for table colors are:

Class	Description
.table-primary	Blue: Indicates an important action
.table-success	Green: Indicates a successful or positive action
.table-danger	Red: Indicates a dangerous or potentially negative action
.table-info	Light blue: Indicates a neutral informative change or action
.table-warning	Orange: Indicates a warning that might need attention
.table-active	Grey: Applies the hover color to the table row or table cell
.table-secondary	Grey: Indicates a slightly less important action
.table-light	Light grey table or table row background
.table-dark	Dark grey table or table row background

Example for Bootstrap text colors

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap text colors</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container mt-3">
  <h2>Contextual Colors</h2>
  <p>Use the contextual classes to provide "meaning through colors":</p>
  <p class="text-muted">This text is muted.</p>
  <p class="text-primary">This text is important.</p>
  <p class="text-success">This text indicates success.</p>
  <p class="text-info">This text represents some information.</p>
  <p class="text-warning">This text represents a warning.</p>
  <p class="text-danger">This text represents danger.</p>
  <p class="text-secondary">Secondary text.</p>
  <p class="text-dark">This text is dark grey.</p>
  <p class="text-body">Default body color (often black).</p>
  <p class="text-light">This text is light grey (on white background).</p>
  <p class="text-white">This text is white (on white background).</p>
</div>
</body> </html>
```

Browser Output:



Example for Bootstrap Opacity Text Colors

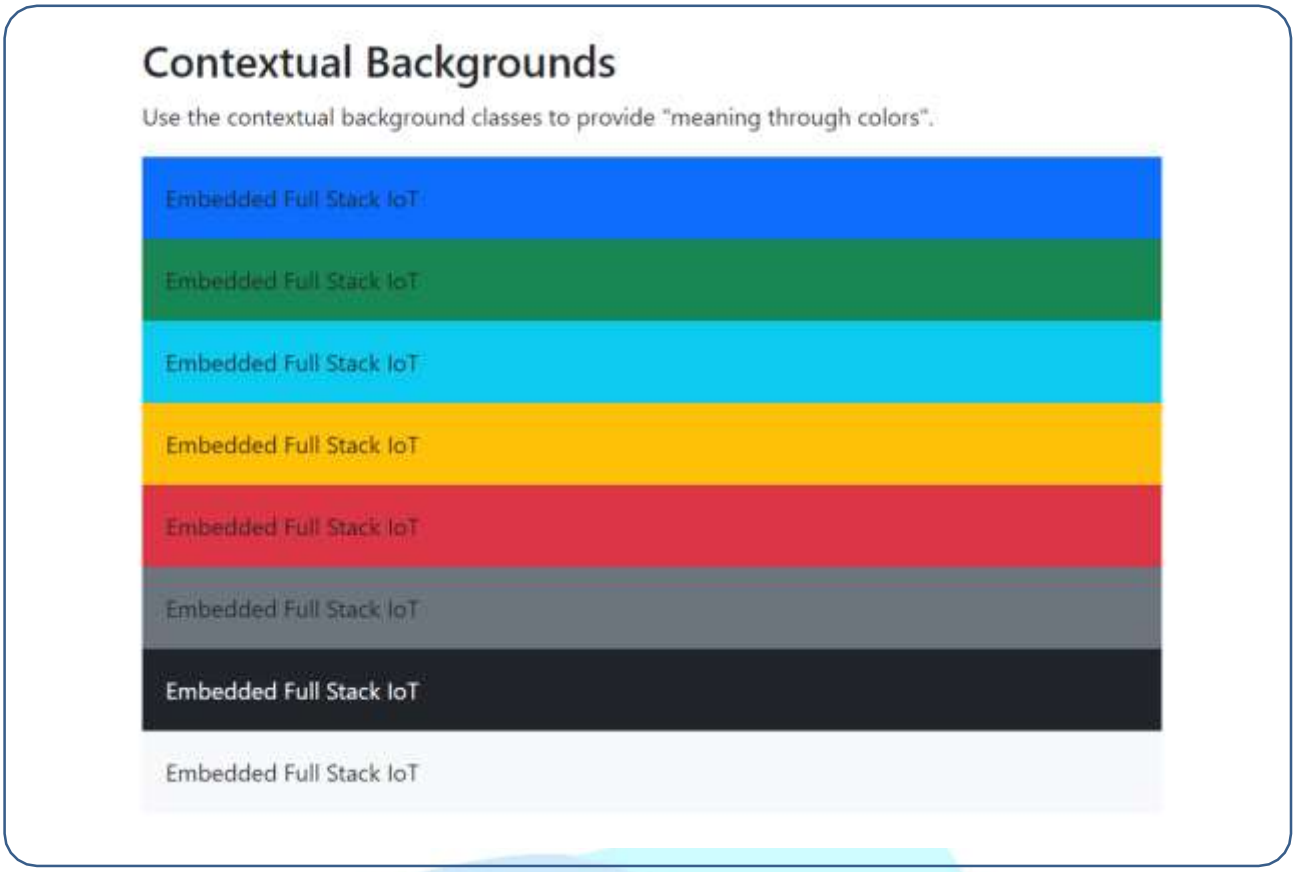
```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Example For Opacity Text Colors </title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container mt-3">
  <h2>Opacity Text Colors</h2>
  <p>Add 50% opacity for black or white text with the .text-black-50 or .text-white-50 classes:</p>
  <p class="text-black-50">Black text with 50% opacity on white background</p>
  <p class="text-white-50 bg-dark">White text with 50% opacity on black background</p>
</div>
</body> </html>
```

Browser Output:

Example for Bootstrap background colors

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example for Background Colors</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container mt-3">
  <h2>Contextual Backgrounds</h2>
  <p>Use the contextual background classes to provide "meaning through colors".</p>
  <div class="bg-primary p-3"> Embedded Full Stack IoT </div>
  <div class="bg-success p-3">Embedded Full Stack IoT</div>
  <div class="bg-info p-3">Embedded Full Stack IoT</div>
  <div class="bg-warning p-3">Embedded Full Stack IoT</div>
  <div class="bg-danger p-3">Embedded Full Stack IoT</div>
  <div class="bg-secondary p-3">Embedded Full Stack IoT</div>
  <div class="bg-dark p-3 text-white">Embedded Full Stack IoT</div>
  <div class="bg-light p-3">Embedded Full Stack IoT</div>
</div>
</body>
</html>
```

Browser Output:



Bootstrap 5 Images

Bootstrap 5 Responsive images are used to resize the images according to their parent element and screen sizes. It means, the size of the image should not overflow its parent element and will grow and shrink according to the change in the size of its parent without losing its aspect ratio.

Image Shapes

- **Rounded Corners:** The .rounded class adds rounded corners to an image:

  <img src=" " class="rounded" alt=" ">

- **Circle:** The .rounded-circle class shapes the image to a circle:

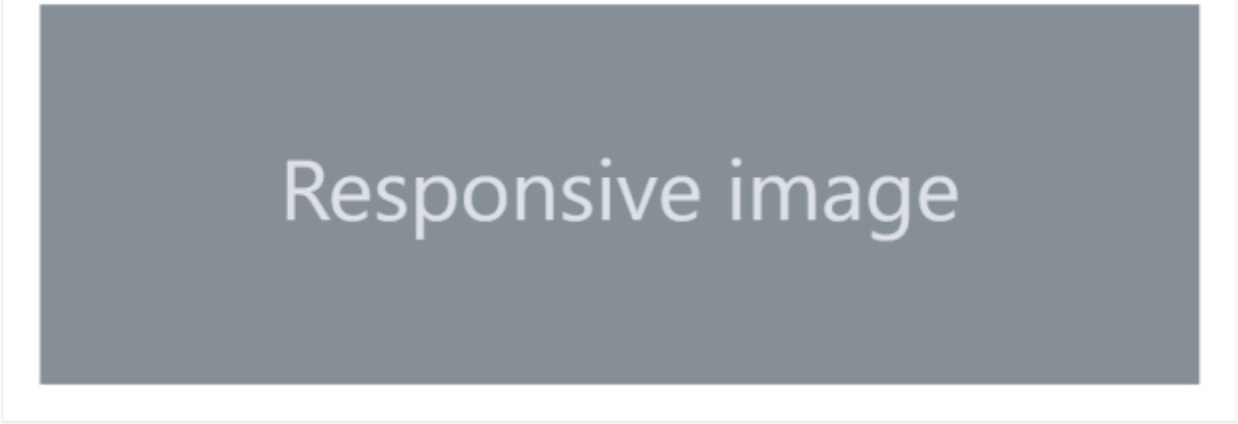
  <img src=" " class=".rounded-circle " alt=" ">

- **Thumbnail:** The .img-thumbnail class shapes the image to a thumbnail (bordered):

  <img src=" " class=".img-thumbnail " alt=" ">

Responsive images

Images in Bootstrap are made responsive with .img-fluid. This applies max-width:100%; and height: auto; to the image so that it scales with the parent element.

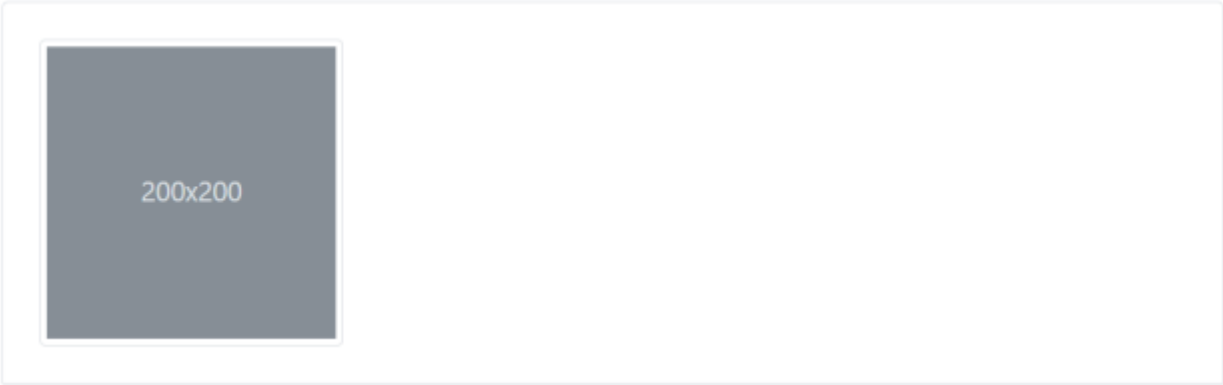



```
<img src="..." class="img-fluid" alt="...">
```

Image thumbnails

In addition to our border-radius utilities, you can use .img-thumbnail to give an image a rounded 1px border appearance.

```
<img src="..." class="img-thumbnail" alt="...">
```

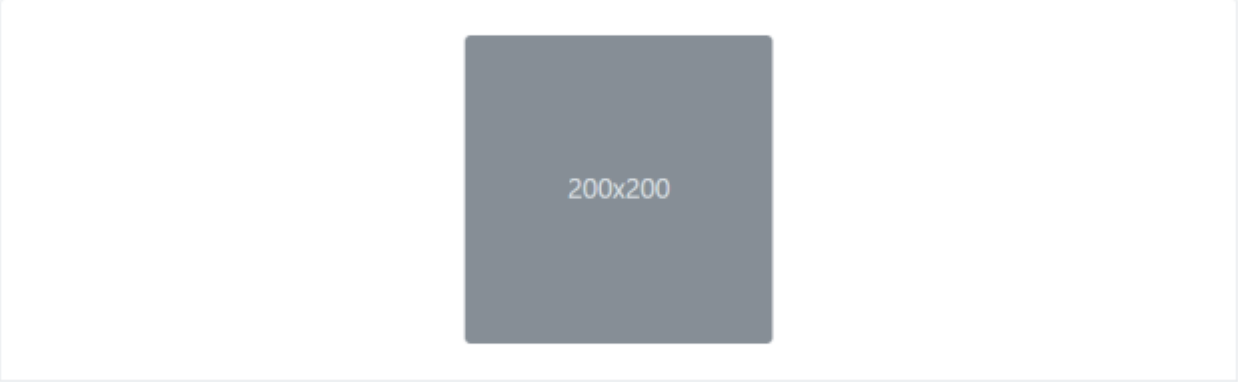


Aligning images

Align images with the helper float classes or text alignment classes. block-level images can be centered using the .mx-auto margin utility class.

```
<img src="..." class="rounded float-start" alt="...">
<img src="..." class="rounded float-end" alt="...">
```


```
<img src="..." class="rounded mx-auto d-block" alt="...">
```




```
<div class="text-center">
  <img src="..." class="rounded" alt="...">
</div>
```

Bootstrap Cards

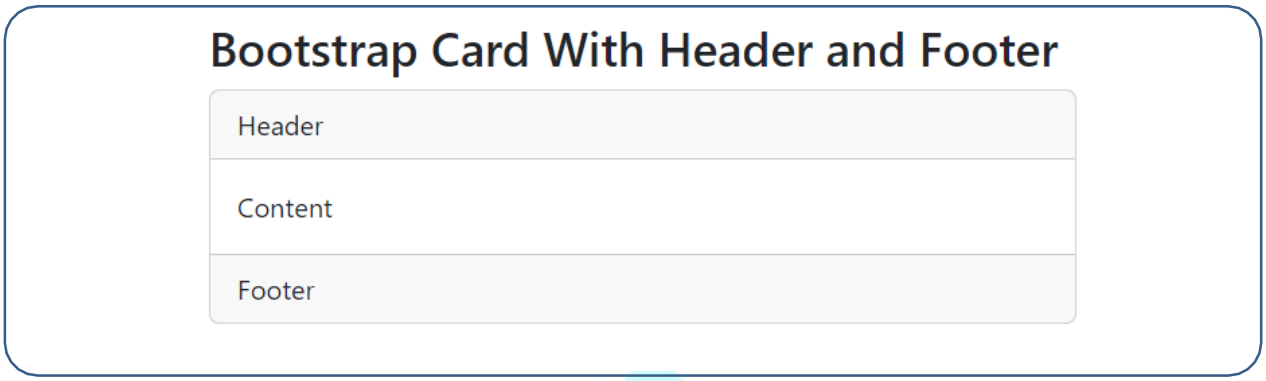
A card is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options.

It replaces the use of panels, wells and thumbnails. All of it can be used in a single container called card.

Example for bootstrap card

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Card Example</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container mt-3">
  <h2> Bootstrap Card With Header and Footer</h2>
  <div class="card">
    <div class="card-header">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
</div>
</body>
</html>
```

Browser Output:



Bootstrap Buttons

Bootstrap Buttons are pre-styled components in the Bootstrap framework, offering consistent design and functionality. They streamline development by providing ready-to-use button styles, sizes, and behaviors, ensuring a cohesive and responsive user interface across web applications with minimal CSS customization. The .btn classes are designed to be used with the <button> element.

Types: Following are the nine types of buttons available in Bootstrap 5:

- btn-primary
- btn-secondary
- btn-success
- btn-danger
- btn-warning
- btn-info
- btn-light
- btn-dark
- btn-link

The button classes can be used on <a>, <button>, or <input> elements:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Example For Bootstrap Buttons </title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container mt-3">
  <h2>Button Styles</h2>
  <button type="button" class="btn">Basic</button>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-secondary">Secondary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-dark">Dark</button>
  <button type="button" class="btn btn-light">Light</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
</body> </html>
```

Browser Output:

Outline buttons

In need of a button, but not the hefty background colors they bring? Replace the default modifier classes with the .btn-outline-* ones to remove all background images and colors on any button.

Example:

```
<button type="button" class="btn btn-outline-primary">Primary</button>
<button type="button" class="btn btn-outline-secondary">Secondary</button>
<button type="button" class="btn btn-outline-success">Success</button>
<button type="button" class="btn btn-outline-danger">Danger</button>
<button type="button" class="btn btn-outline-warning">Warning</button>
<button type="button" class="btn btn-outline-info">Info</button>
<button type="button" class="btn btn-outline-light">Light</button>
<button type="button" class="btn btn-outline-dark">Dark</button>
```

Browser Output:

Button Sizes

Bootstrap provides four button sizes:

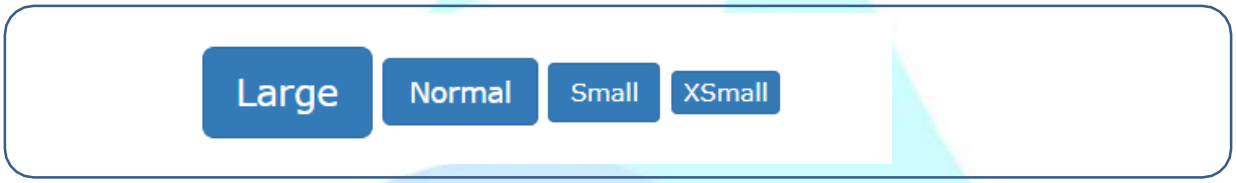
The classes that define the different sizes are:

- .btn-lg
- .btn-sm
- .btn-xs

Example:

```
<button type="button" class="btn btn-primary btn-lg">Large</button>
<button type="button" class="btn btn-primary">Normal</button>
<button type="button" class="btn btn-primary btn-sm">Small</button>
<button type="button" class="btn btn-primary btn-xs">XSmall</button>
```

Browser Output:



Block Level Buttons

A block level button spans the entire width of the parent element. Add class .btn-block to create a block level button:

Example:

```
<button type="button" class="btn btn-primary btn-block">Button 1</button>
<button type="button" class="btn btn-default btn-block">Button 2</button>
```

Browser Output:

Active/Disabled Buttons

A button can be set to an active (appear pressed) or a disabled (unclickable) state:

The class .active makes a button appear pressed, and the class .disabled makes a button unclickable:

Example:

```
<button type="button" class="btn btn-primary active">Active Primary</button>
<button type="button" class="btn btn-primary disabled">Disabled Primary</button>
```

Browser Output:

Bootstrap Forms

Bootstrap Forms make creating user input areas easy with pre-styled components. They simplify form design from basic text fields to complex layouts, ensuring consistency and mobile-friendly responsiveness for capturing user data effectively.

Bootstrap's Default Settings

Form controls automatically receive some global styling with Bootstrap:

All textual <input>, <textarea>, and <select> elements with class **.form-control** have a width of 100%.

Bootstrap Form Layouts

Bootstrap provides three types of form layouts:

- Vertical form (this is default)
- Horizontal form
- Inline form

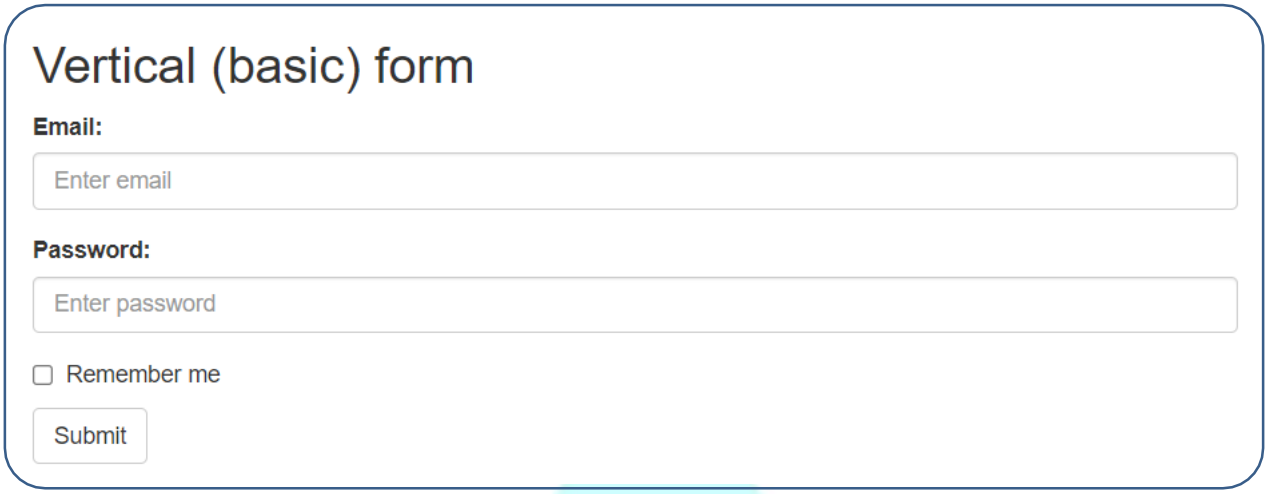
Standard rules for all three form layouts:

- Wrap labels and form controls in (needed for optimum spacing)
- Add class .form-control to all textual <input>, <textarea>, and <select> elements

Example for Bootstrap Vertical Form

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
  <h2>Vertical (basic) form</h2>
  <form action="/action_page.php">
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email"
name="email">
    </div>
    <div class="form-group">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="pwd" placeholder="Enter password"
name="pwd">
    </div>
    <div class="checkbox">
      <label><input type="checkbox" name="remember"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-default">Submit</button>
  </form>
</div> </body> </html>
```

Browser Output:



Bootstrap Inline Form

In an inline form, all of the elements are inline, left-aligned, and the labels are alongside.

Note: This only applies to forms within viewports that are at least 768px wide!

Additional rule for an inline form:

Add class .form-inline to the <form> element

Example for Bootstrap Inline Form

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
  <h2>Inline form</h2>
  <p>Make the viewport larger than 768px wide to see that all of the form elements are inline,
left aligned, and the labels are alongside.</p>
  <form class="form-inline" action="/action_page.php">
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email"
name="email">
    </div>
    <div class="form-group">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="pwd" placeholder="Enter password"
name="pwd">
    </div>
    <div class="checkbox">
      <label><input type="checkbox" name="remember"> Remember me</label>
    </div>
    <button type="submit" class="btn btn-default">Submit</button>
  </form>
</div> </body> </html>
```

Browser Output:

Bootstrap Horizontal Form

A horizontal form means that the labels are aligned next to the input field (horizontal) on large and medium screens. On small screens (767px and below), it will transform to a vertical form (labels are placed on top of each input).

Additional rules for a horizontal form:

- Add class .form-horizontal to the <form> element
- Add class .control-label to all <label> elements

Use Bootstrap's predefined grid classes to align labels and groups of form controls in a horizontal layout.

Example for bootstrap horizontal form

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title> Example for bootstrap horizontal form </title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
  <h2>Horizontal form</h2>
  <form class="form-horizontal" action="/action_page.php">
    <div class="form-group">
      <label class="control-label col-sm-2" for="email">Email:</label>
      <div class="col-sm-10">
        <input type="email" class="form-control" id="email" placeholder="Enter email"
name="email">
</div>
 </div>
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Password:</label>
      <div class="col-sm-10">
        <input type="password" class="form-control" id="pwd" placeholder="Enter password"
name="pwd">
  </div>
 </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <div class="checkbox">
          <label><input type="checkbox" name="remember"> Remember me</label>
```

```
      </div>
 </div>
</div>
   <div class="form-group">
     <div class="col-sm-offset-2 col-sm-10">
       <button type="submit" class="btn btn-default">Submit</button>
     </div>
  </div>
  </form>
</div>
 </body>
 </html>
```

Browser Output:

Bootstrap 5 Forms

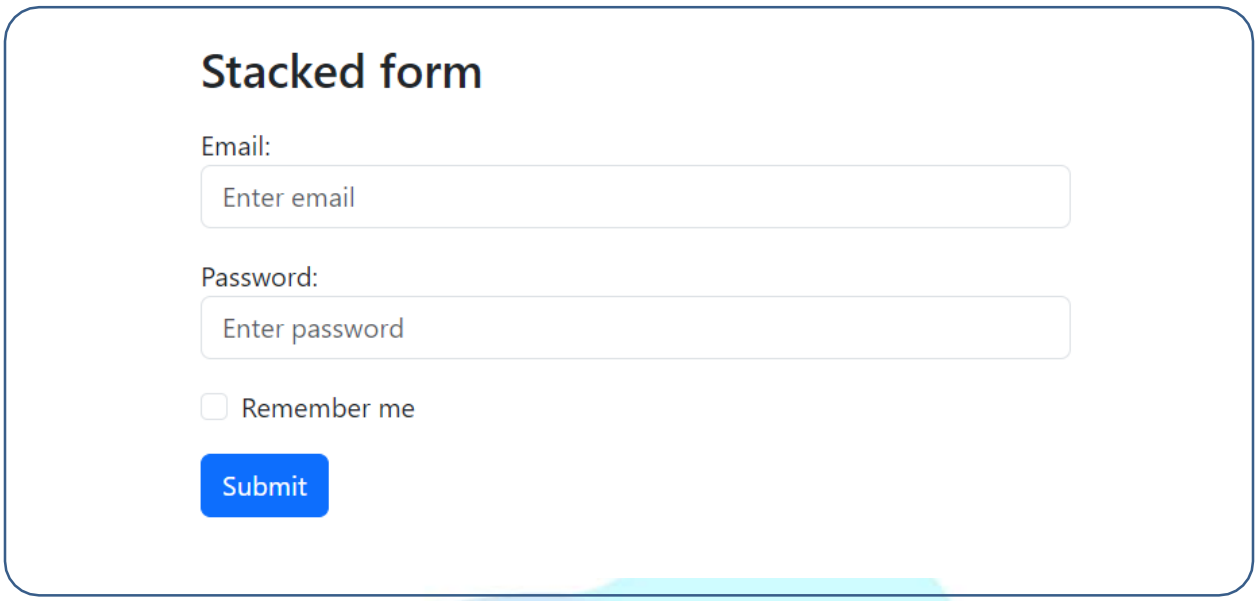
Stacked Form

All textual <input> and <textarea> elements with class .form-control get proper form styling:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container mt-3">
  <h2>Stacked form</h2>
  <form action="/action_page.php">
    <div class="mb-3 mt-3">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email"
name="email">
    </div>
    <div class="mb-3">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="pwd" placeholder="Enter password"
name="pswd">
    </div>
    <div class="form-check mb-3">
      <label class="form-check-label">
        <input class="form-check-input" type="checkbox" name="remember"> Remember me
      </label>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
```

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

Browser Output:



Bootstrap Tables

Example for Bootstrap Table

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Example For Bootstrap Table</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container mt-3">
  <h2> Bootstrap table for dark background with border </h2>
  <table class="table table-bordered table-dark">
    <thead>
      <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Email</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John</td>
        <td>Doe</td>
        <td>john@example.com</td>
```

```
    </tr>
    <tr>
      <td>Mary</td>
      <td>Moe</td>
      <td>mary@example.com</td>
    </tr>
    <tr>
      <td>July</td>
      <td>Dooley</td>
      <td>july@example.com</td>
    </tr>
  </tbody>
 </table>
</div>
</body>
</html>
```

Browser Output:

Bootstrap Navigation Bar

Bootstrap Navigation Bar provides a responsive, customizable, and pre-styled navigation component for web applications. It incorporates features like branding, navigation links, dropdowns, and responsiveness, enabling developers to create effective and visually appealing navigation menus effortlessly.

Navbar: Bootstrap provides various types of navigation bars:

Navbar Type	
Basic Navbar	
Inverted Navbar	
Coloured Navigation Bar	
	Navbar with collapsible menu for space efficiency.

A navigation bar is a navigation header that is placed at the top of the page:

Basic Navbar

With Bootstrap, a navigation bar can extend or collapse, depending on the screen size.

A standard navigation bar is created with the .navbar class, followed by a responsive collapsing class: .navbar-expand-xxl|xl|lg|md|sm (stacks the navbar vertically on xxlarge, extra large, large, medium or small screens).

To add links inside the navbar, use either an <ul> element (or a) with class="navbar-nav". Then add <li> elements with a .nav-item class followed by an <a> element with a .nav-link class:

```
<nav class="navbar navbar-expand-sm bg-light">
  <div class="container-fluid">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="#">Link 1</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link 2</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link 3</a>
      </li>
    </ul>
  </div>
</nav>
```

Browser Output:

Link 1 Link 2 Link 3

Vertical Navbar

Remove the .navbar-expand-* class to create a navigation bar that will always be vertical:

```
<nav class="navbar bg-light">
 <div class="container-fluid">
  <ul class="navbar-nav">
   <li class="nav-item">
    <a class="nav-link" href="#">Link 1</a>
   </li>
   <li class="nav-item">
    <a class="nav-link" href="#">Link 2</a>
   </li>
   <li class="nav-item">
    <a class="nav-link" href="#">Link 3</a>
   </li>
  </ul>
 </div>
</nav>
```

Browser Output:

Link 1
Link 2
Link 3

Centered Navbar

Add the .justify-content-center class to center the navigation bar:

```
<nav class="navbar navbar-expand-sm bg-light justify-content-center">
 <ul class="navbar-nav">
   <li class="nav-item">
     <a class="nav-link" href="#">Link 1</a>
   </li>
   <li class="nav-item">
     <a class="nav-link" href="#">Link 2</a>
   </li>
   <li class="nav-item">
     <a class="nav-link" href="#">Link 3</a>
   </li>
 </ul>
</nav>
```

Browser Output:

Colored Navbar

Use any of the .bg-color classes to change the background color of the navbar (.bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light)

Add a white text color to all links in the navbar with the .navbar-dark class, or use the .navbar-light class to add a black text color.

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
 <div class="container-fluid">
  <ul class="navbar-nav">
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
  </ul>
 </div>
</nav>
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">
 <div class="container-fluid">
  <ul class="navbar-nav">
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
    <li class="nav-item"> <a class="nav-link active" href="#">Active</a> </li>
  </ul>
 </div>
</nav>
```

Browser Output:

Fixed Navigation Bar

The navigation bar can also be fixed at the top or at the bottom of the page.

A fixed navigation bar stays visible in a fixed position (top or bottom) independent of the page scroll.

- The .fixed-top class makes the navigation bar fixed at the top:
- The .fixed-bottom class to make the navbar stay at the bottom of the page:

Example for Fixed Top

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark fixed-top">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">Fixed top</a>
 </div>
</nav>
```

Example for Fixed Bottom

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark fixed-bottom">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">Fixed bottom</a>
 </div>
</nav>
```

Sticky Navbar

The .sticky-top class to make the navbar fixed/stay at the top of the page when you scroll past it.

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark sticky-top">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">Sticky top</a>
 </div>
</nav>
```

Example for bootstrap navigation bar

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Example For Bootstrap Navbar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Logo</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#collapsibleNavbar">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="collapsibleNavbar">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
```

```
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-
toggle="dropdown">Dropdown</a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="#">Link</a></li>
            <li><a class="dropdown-item" href="#">Another link</a></li>
            <li><a class="dropdown-item" href="#">A third link</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
<div class="container-fluid mt-3">
  <h3>Navbar With Dropdown</h3>
  <p>This example adds a dropdown menu in the navbar.</p>
</div>
</body>
</html>
```

Browser Output:

Reference for Frontend Programming

1. MDN Web Docs
   - Comprehensive resource for HTML, CSS, and JavaScript documentation and tutorials.
   - Great for learning web standards and best practices.

2. W3Schools
   - Offers tutorials and references on web development languages.
   - Interactive coding examples and exercises.

3. CSS-Tricks
   - Articles, tutorials, and tips about CSS.
   - Includes guides on modern web development practices and techniques.

4. Bootstrap Official Documentation
   - Comprehensive guide to using Bootstrap.
   - Covers all components, utilities, and layout options.
   - Includes examples and customization options.

5. Bootstrap on W3Schools
   - Tutorials on using Bootstrap with examples.
   - Covers the basics to advanced features of Bootstrap.

6. MDN Web Docs: Bootstrap
   - Introduction to Bootstrap within the MDN learning area.
   - Explains how to integrate and use Bootstrap in projects.

7. https://www.javatpoint.com/html-tutorial

8. https://www.javatpoint.com/css-tutorial

9. https://www.javatpoint.com/bootstrap-tutorial

10. https://www.geeksforgeeks.org/html-tutorial/

11. https://www.geeksforgeeks.org/css-tutorial/