Major ingredients of the website

HTML
CSS
JS
Foundation
Style
Functionality

Any website is majorly built using

1. HTML
2. CSS
3. Java script

Website and pages can be divided into 2 types majorly:

Static Website	Dynamic Website
The content present on the web page cannot be modified at runtime.	The content present on the web page can be modified at runtime.
There is no interaction with the database.	The website can interact with the database.
It is cheaper to develop a static website as compared to a dynamic website.	A dynamic website requires more cost of development than a static website.
The same content gets loaded every time.	The content that gets loaded may vary as

	per requirements.

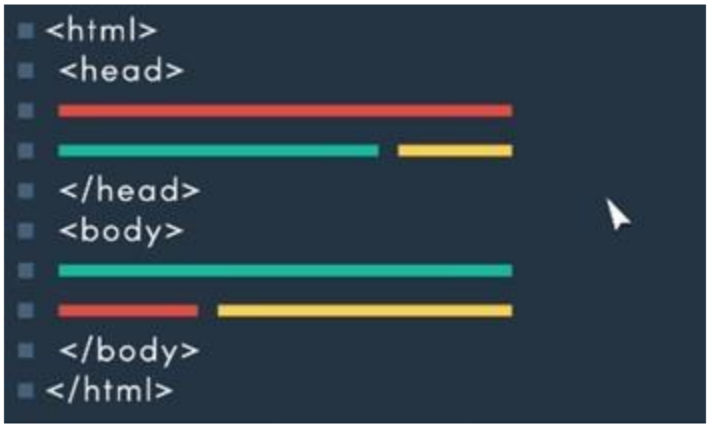
HTML:

- **HTML:** Hyper Text Mark up Language
    - **Hypertext:** Text which contains hyperlinks (or references) to other text.
    - **Mark up Language:** It is a way in which the documents are annotated such that it can be systematically distinguished from the normal text.
- **HTML** is a set of **mark up** symbols intended for displaying content on the Internet.
- **Mark up** tells browsers how to display the content of a webpage, that is, the words and images. Mark ups are commonly known as **tags**.
- The HTML file is broken into small parsing elements called Tokens, which are read by the browser.

<u>Structure of HTML</u>

HTML has majorly below parts:

- HTML (Container): A container that holds the content and the information about the content
- Head: Information about the content
- Body: Content

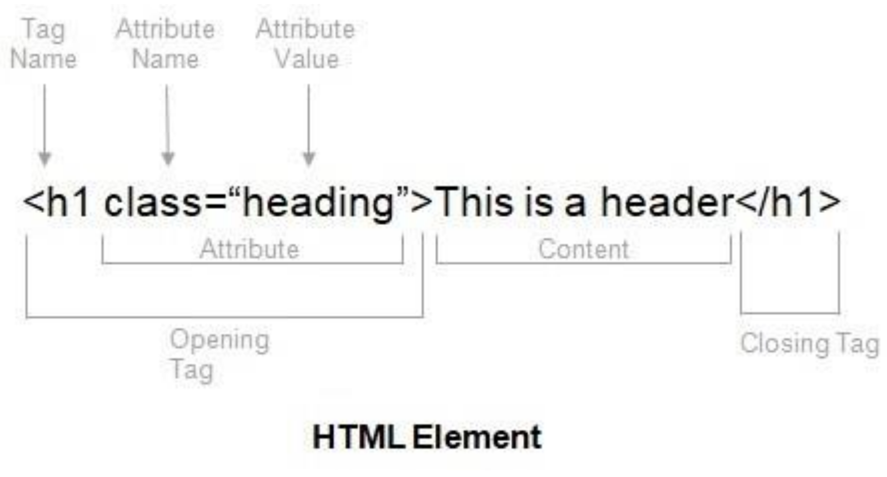


```
<html>
 <head>


 </head>
 <body>


 </body>
</html>
```

Commonly used HTML Tags:

<h1></h1>...<h6></h6>
<p></p>
<a></a>
<span></span>
<div></div>
<select></select>
<table></table>
<img />
<br />
<input />
<hr />

Tag Structure:



Special Characters and Entities:

An HTML entity is basically a text that starts with an ampersand (&) and ends with a semicolon (;).

These entities are often used to display the reserved characters. I.e. the characters which otherwise would be interpreted as simple HTML: code.

Some commonly used characters and their entities are:

Character	Entity	Description
&	&amp;	It symbolizes the beginning of an character entity 'and'.
<	&lt;	It is used to represent the opening tag or less than sign.
>	&gt;	It is used to represent the closing tag or greater than sign.
"	&quot;	It is used to represent the start and end of attribute value.
space	&nbsp;	It is used to represent the space.

Inline vs Block Elements:

You will notice few tags have end tag and few do not.

Few will allow next contents to be plotted beside and few do not.

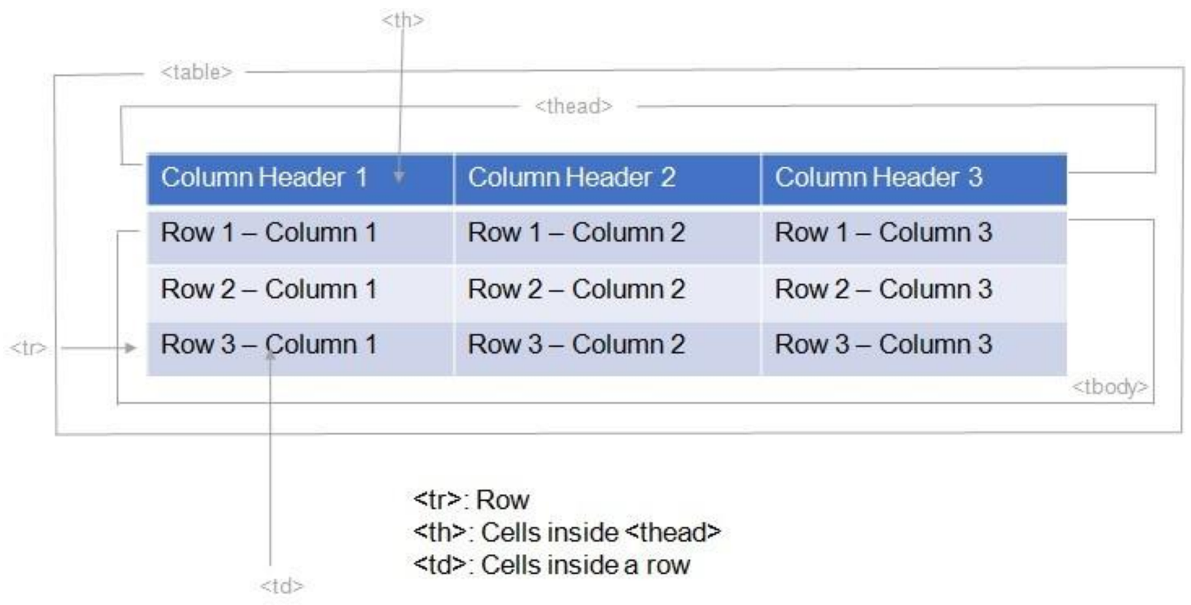
List of inline and block elements / tags:

Inline Elements	Block Elements
<span>	<div>
<b>	<p>
<strong>	<h1> …. <h6>
<i>	<hr>
<em>	<table>
<a>	<ul>
<img>	<ol>
<button> …	<form>
You can see the entire list here. https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements	You can see the entire list here. https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

HTML Tables

Used for presenting data in tabular format:

Table Structure:



<tr>: Row
<th>: Cells inside <thead>
<td>: Cells inside a row

```
<body>

    <table align="right">

      <thead>

        <tr>

          <th>Fruit</th>

          <th>Quantity</th>

        </tr>

      </thead>

      <tbody>

        <tr>

          <td>Apples</td>

          <td>10</td>
        </tr>
         <tr>
            <td>Oranges</td>
            <td>15</td>
         </tr>
       </tbody>
    </table>
</body>
```

Forms

Code Example:

```
<html>
  <head>
    <title>Log In</title>
  </head>
  <body>
    <form action="/action-page.php" method="get" target="_blank">

      <label for="name">Username:</label>
      <input type="text" name="name" id="name" />

      <label for="password">Password:</label>
      <input type="password" name="password" id="password" />

      <input type="checkbox" name="remember" id="remember" />

      <label for="remember">Keep me logged in.</label>

      <input type="submit" value="Login"/>
    </form>
  </body>
</html>
```

Form Tag Attributes and meaning:

action – It defines the action to be performed when the form is submitted.

method – It defines the HTTP method for submitting data.

target – It defines where the submitted results should open.

GET vs POST

GET	POST
Form data IS VISIBLE in the URL.	Form data is NOT VISIBLE in the URL.
Clicking the back button will not re-submit the data; hence, it is harmless.	Clicking the back button will re-submit the data; hence, you need to be cautious while using POST.
It can be bookmarked and cached.	It cannot be bookmarked and cached.
The length of the URL is restricted.	There are no restrictions on the length of the URL.
It is Not Secure.	It is Secure.

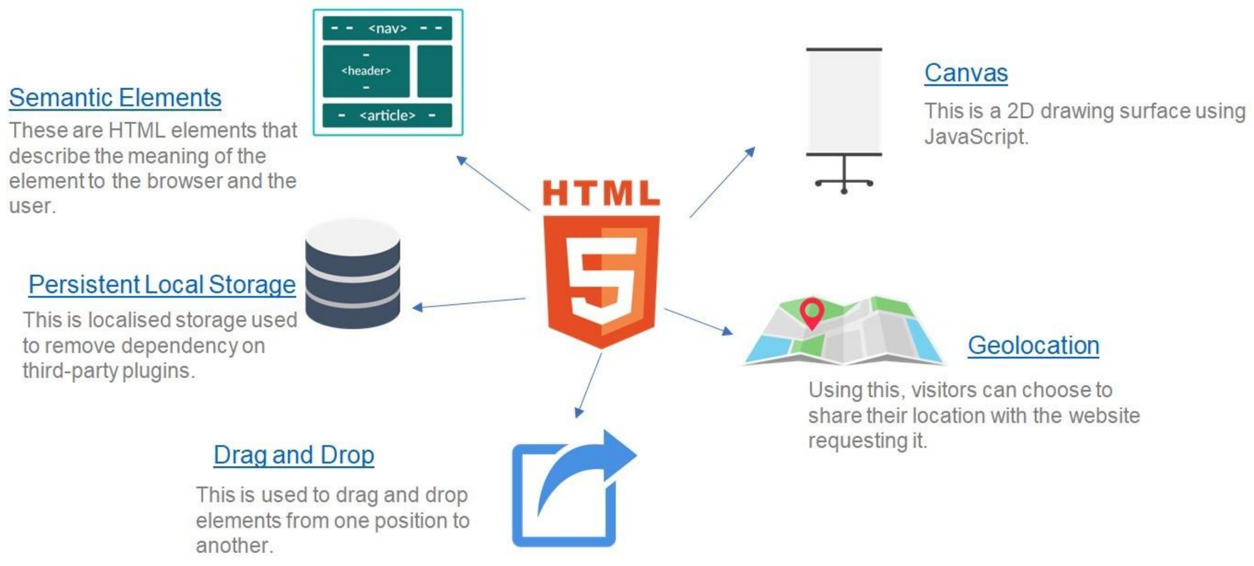
Different input types:

`<input type=“text” />`	Allows users to type text.
`<input type=“email” />`	Allows users to type text in email format, i.e. if the typed text does not contain @ or . as normal email formats do, it will throw up an error message on submitting.
`<input type=“password” />`	Allows users to type text in password format. The text entered would look like this: ••••••
`<input type=“number” />`	Allows users to type characters in number format. Users will not be able to type alphabets or any other special character if the input is defined as type number.
`<input type=“radio” />`	Allows users to include a radio button selection, i.e., choose only one of multiple options provided.
`<input type=“checkbox” />`	Allows users to include a checkbox selection, i.e., choose multiple options provided.
`<input type=“file” />`	Allows users to upload files from their local machines.
`<input type=“range” />`	Allows users to add a range slider between a minimum value and a maximum value.
`<input type=“submit” />`	Allows users to submit a form.

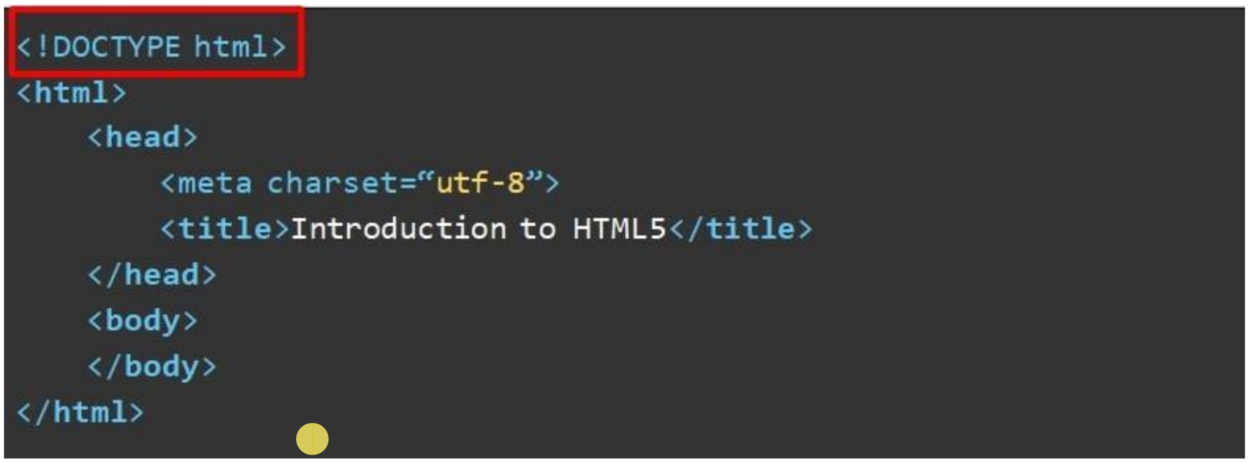
Common Input tag attributes:

`value`	It can be used to specify an initial value for an input field.
`size`	It can used to specify the visible width of the input field box.
`min`	It specifies the minimum value of the input field.
`max`	It specifies the maximum value of the input field.
`maxlength`	It specifies the maximum allowable number of characters.
`readonly`	It is used to ensure that the input field becomes read-only.
`disabled`	It is used to disable the field
`placeholder`	Text written inside the input field when it is empty
`required`	It is used to mark the input fields as compulsorily before submitting the form.

What is new in HTML 5?


Semantic Elements
These are HTML elements that describe the meaning of the element to the browser and the user.
<nav>
<header>
<article>
Persistent Local Storage
This is localised storage used to remove dependency on third-party plugins.
Drag and Drop
This is used to drag and drop elements from one position to another.
HTML 5
Canvas
This is a 2D drawing surface using JavaScript.
Geolocation
Using this, visitors can choose to share their location with the website requesting it.

How do we tell browser that content is HTML 5?



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Introduction to HTML5</title>
    </head>
    <body>
    </body>
</html>
```

Simple HTML 5 Tags Example:

```
<!DOCTYPE html>

<html>
   <head>

      <title>Introduction to HTML5</title>
   </head>
   <body>
      <header role="banner">

            <h1>Introduction to HTML5</h1>

        </header>

        <article>

             <section>

                <p>One article can have multiple sections.</p>

             </section>

        </article>

        <footer>Created by Bonaventure Systems.</footer>
     </body>
  </html>
```

HTML

- HyperText Markup Language
- **Markup Language**
    - Consists of
        - Tag
        - Data/Content
            - Information enclosed by opening and closing tag
            - E.g.
                - <body>hello world!!</body>
                - Where
                    - <body>: opening tag
                    - hello world!!: Data / Content
                    - </body>: closing tag
    - NOT used for programming
- The latest standard is HTML5
    - To add html5 code, start the document with
        - <!DOCTYPE html>
    - DOCTYPE: document type (tag used to start the html document)
- Usage
    - Used for designing web pages
- Case in-sensitive
- To add comment
    - <!-- comment -->

Tag

- Word enclosed by < and > signs
- Also called as an element
- All tags in HTML are pre-defined by W3C
- E.g. <h1>, <p>, <table>
- Types
    - Opening
        - Used to open a data/information
        - E.g. <h1>, <p>, <html>
    - Closing
        - Used to close the data/information
        - E.g. </h1>, </p>, </html>
    - Empty
        - Tag having no data/content
        - Two ways
            - <tag></tag>
            - <tag />
        - E.g. <br/>, <hr/>
    - Root
        - Tag which starts and ends the document
        - Is also called as Document Type or Document Tag or Document Element
        - E.g. <html> is root tag for html document
- **Attribute**
    - Extra information about the tag
    - Attribute always present in

        - name=value format
        - E.g. <meta charset="utf-8">
            - Where
                - Meta: tag
                - Charset: attribute name
                - Utf-8: attribute value
    - A tag may have one or more attributes
    - Every tag has following attributes
        - name: used to create query string
        - id: used to identify an element uniquely
        - style: used to write inline css
        - class:

HTML Structure

- Has two parts
    - Head
        - Contains extra information about the page
        - Tags:
            - title: used to set the title for the tab
            - script: used to add JS code in the page
            - style: used to add CSS code
            - meta: used to add more information about the page
            - link: used to link external documents (files)
            - base: used to set the base url used in the page
    - Body
        - Contains the actual design
        - Categories
            - **Textual**
                - Header: used to add header in page
                    - There 6 levels
                    - Tags: h1 to h6
                    - H1 is the biggest while h6 is the smallest
                - Paragraph (<p>): used to add a para
                - Division(<div>): used to create groups of
                    - Tags
                    - Textual contents
            - **Resources**
                - Images:
                    - Img: used to add image
                        - src: specify the source
                        - alt: used to alternative text (rendered only in case of missing source)
                - Audio
                    - Audio: used to play audio
                        - controls: to render controls
                        - autoplay: to play the file automatically
                        - src: to set the file to play
                - Video
                    - Video: used to play video
                        - controls: to render controls
                        - autoplay: to play the file automatically

            - src: to set the file to play
- **List**
    - Unordered list
        - Does not render the order
        - Syntax:
```
<ul>
        <li>list item1</li>
        <li>list item2</li>
</ul>
```
    - Ordered list
        - Renders the list item order
        - Syntax:
```
<ol>
        <li>list item1</li>
        <li>list item2</li>
</ol>
```
    - Definition list
        - Used to create list of definitions
        - Syntax:
```
<dl>
        <dt>term 1</dt>
        <dd>definition 1</dd>

        <dt>term 2</dt>
        <dd>definition 2</dd>
</dl>
```
- **Table**
    - Used to create tabular representation
    - Divided into 3 sections
        - Header
            - Use <thead> (table header)
        - Body
            - Use <tbody> (table body)
        - Footer
            - Use <tfoot> (table footer)
    - To create row
        - Use <tr>
    - To create column
        - In header
            - Use <th> (table header column)
        - In Body and Footer
            - Use <td> (table data)
    - E.g.
```
<table>
        <thead>
                <tr>
                        <th></th>
                </tr>
        </thead>

        <tbody>
                <tr>
                <td></td>
            </tr>
        </tbody>

        <tfoot>
            <tr>
                <td></td>
            </tr>
        </tfoot>
    </table>
```

- Attributes:
    - border: used to create border
    - colspan: used to merge multiple columns horizontally
    - rowspan: used to merge multiple columns vertically

- **Linking (anchor)**
    - Used to link
        - multiple pages
            - <a href=”page.html”>text</a>
        - multiple sections within same page
            - <a href=”#top”>text</a>
        - multiple pages with multiple sections
            - <a href=”page#section”>text</a>
- **Form**
    - Used to get inputs from user
    - Tags
        - Input: used to get input
            - type
                - text: textual value
                - number: number value
                - email: email value
                - tel: telephone
                - date: date input
                - time: time input
                - radio: create radio button
                - checkbox: multiple selection
                - password: masked characters
                - submit: to submit values
                - reset: to clear the form
                - file: used to upload file
        - textarea: used to get multi-line input
        - select: used to render drop-down

```
<select>
    <option>op1</option>
    <option>op2</option>
</select>
```

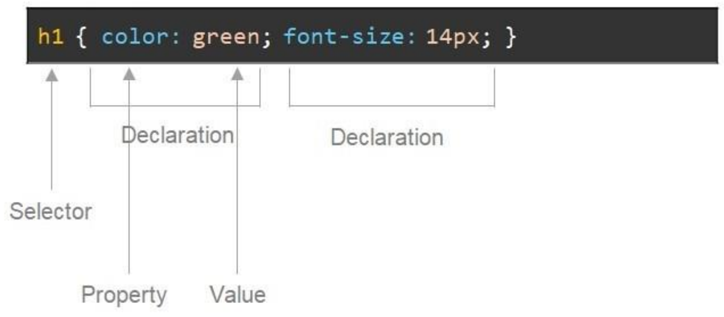
GET vs POST

GET	POST
1. values will be sent using Request head	1. Values will be sent using Request Body
2. Values will be appended on url (visible)	2. Values will be invisible
3. Restriction on maximum size of data to be passed	3. There is no restriction on size of data
4. Only ascii values can be sent using GET	4. Any value of any type (binary) can be sent by using POST
5. URL can be bookmarked with the values	5. URL can be bookmarked without the values
6. Files can NOT sent using GET	6. Files can be using POST

Basics of CSS

- CSS stands for **C**ascading **S**tyle **S**heets.
- It makes an HTML website presentable.
- It adds style to various HTML elements.
- It helps you to define how the elements should look, where they should be placed and whether they should be displayed or not.
- It also helps you to define how elements should look on different devices.

Syntax:



```
h1 { color: green; font-size: 14px; }
```

- A selector can be an element itself such as **p**, **h1** or the **class** or the **id**.
- A property signifies the property of an element, such as the **color** of a text element.
- A value signifies the value that should be assigned to a property, such as **red** for the **color** property.

CSS3

- Cascading Stylesheet Style
- Sequence of styles
    - Browser default CSS (black)
    - External CSS (red)
    - Internal CSS (blue)
    - Inline CSS (brown)

Terminology

- CSS property: pre-defined properties provided by CSS
- **Declaration**:
    - Pair of css property and its value
    - Colon(:) is used to separate the property and its value
    - To terminate a declaration use semi-colon(;)
- **Declaration block**:
    - Collection of declarations
    - Use {} to create a block
- **Selector**:
    - Used to select a type of element(s)
- **Rule/Ruleset**:
    - Pair of selector and a declaration block

Units

- px: pixels
- %: with respect to its parent
- em: emphasis
    - by default the em is considered as 1 (browser default)
- deg: degree
- s: seconds

CSS Types

- Inline CSS
    - Use style attribute of a tag
    - Disadvantages:
        - Needs to be repeated with every tag having same decoration
        - Very difficult to manage (modify)
    - This is discouraged
    - E.g.

```
<p style="color:red">test</p>
```

- Internal CSS
    - Use style tag in Head section
    - Advantage:
        - Simpler than inline
        - Easy to manage
    - Disadvantage:
        - Repeated in multiple pages of a website
    - E.g.

```
<style>
        p {
                color:red;
        }
```

    </style>

- External CSS
    - CSS rules can be written outside the page (in an external file with .css extension)
    - To attach/link external css with a page use link tag in Head section
      `<link rel="stylesheet" href="<css file name>">`
    - Advantages:
        - Single external css file can be used across multiple pages
- Browser Default CSS
    - Available in every browser

Selector Types

- **Type / Element Selector**
    - Used to select similar type of element(s)
    - E.g.
      ```
      p {
          color: red;
      }
      ```
        - only paragraphs will have red color
- **Multiple type/element selector (,)**
    - Used to select multiple type of elements
    - Comma (,) is used to create multiple type selector
    - E.g.
      ```
      p, div {
          color: red;
      }
      ```
        - both paragraph(s) and division(s) will have red color
- **Id Selector (#)**
    - Used to select an element having specified id
    - Hash(#) is used to create an id selector
    - E.g.
      ```
      div#div1 {
          color:red
      }
      ```
        - only div having id div1 will have color red

      ```
      #product1 {
          color:red
      }
      ```
        - any element having id product1 will have color red
- **Class Selector (.)**
    - Used to select element(s) having same class
    - Dot (.) is used to create a class selector
    - E.g.
      ```
      div.div1 {
          color:red
      }
      ```
        - only div having class div1 will have color red

      ```
      .product1 {
          color:red
      }
      ```
        - any element having class product1 will have color red

- **Descendant selector (white-space)**
    - Used when elements have relationships
    - Used to select child elements at any level
    - Space is used to create descendant selector
        - E.g.

```
body p {
        color:red;
}
```

- every element inside body will have red color
- **Child selector (>)**
    - Used when elements have relationships
    - Used to select child elements at first level (direct child element(s))
    - > is used to create descendant selector
        - E.g.

```
body > p {
        color:red;
}
```

- Paragraph(s) declared under body will have red color
- **Universal selector (*)**
    - Used to select All type of elements in a page(s)
    - Use * to create universal selector
    - E.g.

```
* {
        font-family: Arial;
}
```

- all element(s) in the page will have font family set to Arial
- **Attribute selector**
    - Used to select element(s) based on the attribute
    - Use [] to write the criteria
    - E.g.

```
input[type="submit"] {
        Color: red;
}
```

- only input having type = "submit" will have color set to red
- **Pseudo selector**

CSS Box Model

- Every element in html is rendered as a box
- Properties
    - Border
    - Padding: Gap inside/within the border
    - Margin:
        - Gap outside the border
        - Value: auto

CSS Display

- Used to control the display behavior
- Values
    - none: hide the tag
    - Block: new line character will be added at the end of the contents
    - Inline:
        - element(s) will be rendered on the same line

        - width and height will ignored
    - Inline-block
        - element(s) will be rendered on the same line
        - width and height will applied
    - Table
    - Table-cell

CSS Position

- Used to decide the position of the element
- Values
    - Static:
        - default value
        - top, left, right and bottom will be ignored
    - Relative
        - Relative its static/default position
        - top, left, right and bottom will be applied by using its original (static) position
    - Absolute
        - Top, left, right and bottom will be applied by using browser's origin
        - Gets scrolled with page
    - Fixed
        - Top, left, right and bottom will be applied by using browser's origin
        - Never gets scrolled

CSS Float

- Used to decide the position (left and right)
- To clear/cancel the effect of floating use clear property

CSS3 properties

- Shadow:
    - Values:
        - Vertical
        - Horizontal
        - Blur
        - Color
    - Types
        - Text:
            - text-shadow : 2px 2px 5px red;
        - Box
            - box-shadow : 2px 2px 5px red;
- Border radius
    - Used to add rounded corners to any element
    - E.g. border-radius: 10px;
    - Trick:
        - Apply ½ of width to border radius to convert square element into circle shape
- Transform:
    - Used to transform an element
    - Types
        - Rotate: rotate element
            - transform: rotate(45deg);
        - Scale: scale element (zoom)
            - transform: scale(2);
        - Translate: move position

            - transform: translate(10px, 10px);

- Transition:
    - Used to animation (duration in seconds)
    - E.g. transition: all 2s;
- Gradients
    - Used to add multiple colors (blended)
    - Types
        - Linear
            - E.g. background: linear-gradient(red, yellow);
        - Radial
            - E.g. background: radial-gradient(red, yellow);
- Columns
    - Used to distribute the element contents in multiple columns
    - E.g. column-count: 3;
- At (@) rules:
    - Start with @ symbol
    - **Font**:
        - Used to load custom fonts
        - E.g.

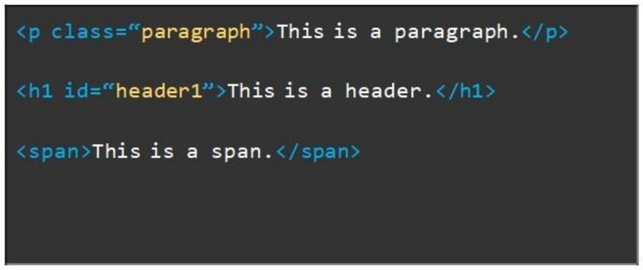
```
@font-face {
        font-family: <family name>;
        src: url('<path>');
}



p {
        font-family: <family name>;
}
```

- - **Media Query**
        - Used to create responsive website
        - A website is having an ability to optimize output according to the device width
            - Desktop
            - Tablet
            - Mobile
        - E.g.

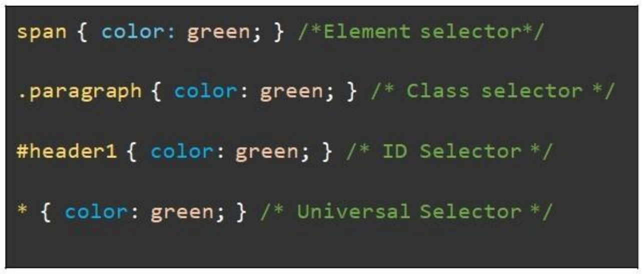
```
@media screen and (max-height:768px) {
        h1 {
                color: red;
        }
}
```

            - will have h1 with color red only on mobile devices

Ways to apply Style:



```
<p class="paragraph">This is a paragraph.</p>

<h1 id="header1">This is a header.</h1>

<span>This is a span.</span>
```

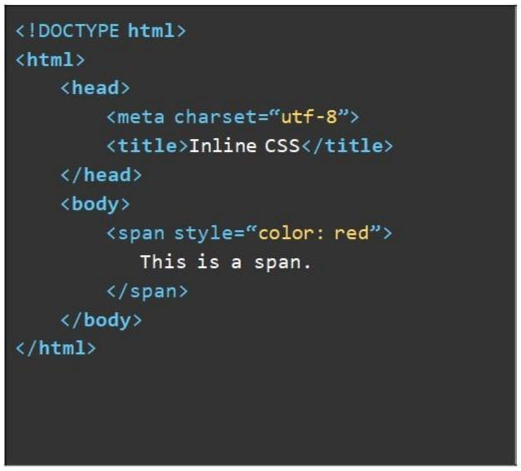


```
span { color: green; } /*Element selector*/

.paragraph { color: green; } /* Class selector */

#header1 { color: green; } /* ID Selector */

* { color: green; } /* Universal Selector */
```

- **Element selectors** only require the element name.
- **Class selectors** require '**.**' before the class name given to an element.
- **ID selectors** require '**#**' before the ID name given to an element.
- **Universal selector** require '*'. It signifies all the elements in a given file.

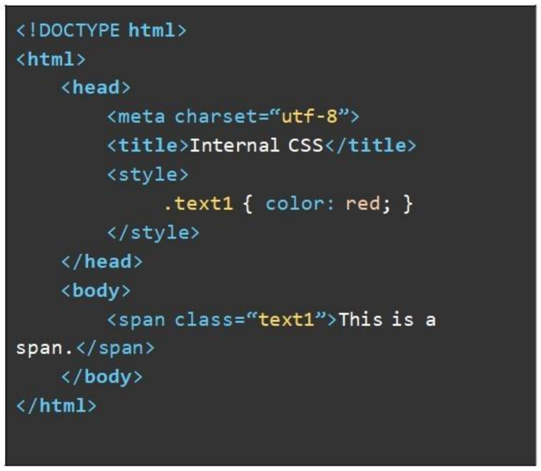
Inline vs Internal vs External:

- Inline stylesheet: This is a CSS that is directly applied to an element inside HTML code.
- Internal stylesheet: This is a CSS that is applied inside a file but not inside an element. It is generally added inside the <style> tag, which is contained within the <head> tag.
- External stylesheet: This is a CSS that is applied from an external file. This external file is referred to inside the HTML code using the <link> tag.
- The following is the order of priority: Inline > Internal > External.



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Inline CSS</title>
    </head>
    <body>
        <span style="color: red">
           This is a span.
        </span>
    </body>
</html>
```

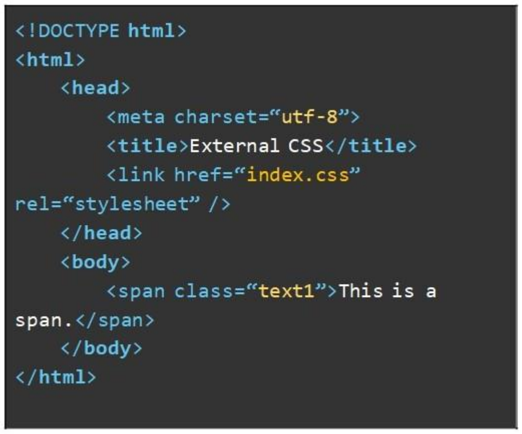
Inline CSS



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Internal CSS</title>
        <style>
             .text1 { color: red; }
        </style>
    </head>
    <body>
        <span class="text1">This is a
span.</span>
    </body>
</html>
```

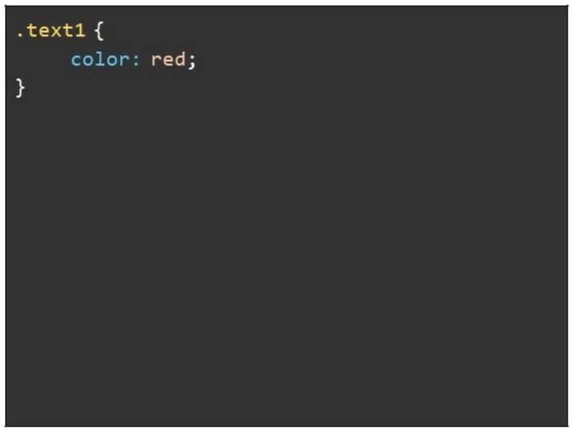
Internal CSS

External CSS:



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>External CSS</title>
        <link href="index.css"
rel="stylesheet" />
    </head>
    <body>
        <span class="text1">This is a
span.</span>
    </body>
</html>
```

index.html

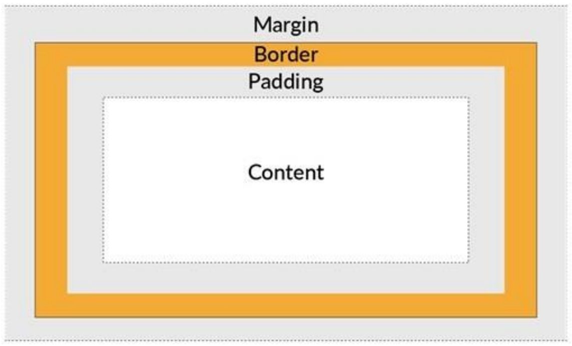


```
.text1 {
     color: red;
}
```

index.css

CSS Box Model:

The CSS box model is a box that surrounds every HTML element. It comprises the following: **content + padding + border + margin.**



- Therefore, the total width of element = Actual width + Left padding + Right padding + Left border + Right border + Left margin + Right margin.
- The total height of element = Actual height + Top padding + Bottom padding + Top border + Bottom border + Top margin + Bottom margin.

What is Bootstrap?

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

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.

Note: Bootstrap 3 and Bootstrap 4 is still supported by the team for critical bugfixes and documentation changes, and it is perfectly safe to continue to use them. However, new features will NOT be added to them.

Why Use Bootstrap?

Advantages of Bootstrap:

- **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap, mobile-first styles are part of the core framework
- **Browser compatibility:** Bootstrap 5 is compatible with all modern browsers (Chrome, Firefox, Edge, Safari, and Opera). **Note** that if you need support for IE11 and down, you must use either BS4 or BS3.

Where to Get Bootstrap 5?

There are two ways to start using Bootstrap 5 on your own web site.

You can:

- Include Bootstrap 5 from a CDN
- Download Bootstrap 5 from getbootstrap.com

One advantage of using the Bootstrap 5 CDN:
Many users already have downloaded Bootstrap 5 from jsDelivr when visiting another site. As a result, it will be loaded from

cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

JavaScript?
Bootstrap 5 uses JavaScript for different components (like modals, tooltips, popovers etc). However, if you just use the CSS part of Bootstrap, you don't need them.

Bootstrap 5 Containers

You learned from the previous chapter that Bootstrap requires a containing element to wrap site contents.

Containers are used to pad the content inside of them, and there are two container classes available:

1. The `.container` class provides a responsive **fixed width container**
2. The `.container-fluid` class provides a **full width container**, spanning the entire width of the viewport

Fixed Container

1. Use the `.container` class to create a responsive, fixed-width container.
2. Note that its width (`max-width`) will change on different screen sizes:

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra Large ≥1200px	XXL ≥1400px
max-width	100%	540px	720px	960px	1140px	1320px

Open the example below and resize the browser window to see that the container width will change at different breakpoints:

```
<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```

Fluid Container

Use the `.container-fluid` class to create a full width container, that will always span the entire width of the screen (`width` is always `100%`):

```
<div class="container-fluid">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
```

</div>

Container Padding

By default, containers have left and right padding, with no top or bottom padding. Therefore, we often use **spacing utilities**, such as extra padding and margins to make them look even better. For example, `.pt-5` means "add a large **top padding**"

<div class="container pt-5"></div>

Container Border and Color

Other utilities, such as borders and colors, are also often used together with containers:

<div class="container p-5 my-5 border"></div>

<div class="container p-5 my-5 bg-dark text-white"></div>

<div class="container p-5 my-5 bg-primary text-white"></div>

Responsive Containers

You can also use the `.container-sm|md|lg|xl` classes to determine when the container should be responsive.

The `max-width` of the container will change on different screen sizes/viewports:

Class	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px	XXL ≥1400px
`.container-sm`	100%	540px	720px	960px	1140px	1320px
`.container-md`	100%	100%	720px	960px	1140px	1320px
`.container-lg`	100%	100%	100%	960px	1140px	1320px
`.container-xl`	100%	100%	100%	100%	1140px	1320px
`.container-xxl`	100%	100%	100%	100%	100%	1320px

Bootstrap 5 Grid System

Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.

If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:

span 1 span 1 span 1 span 1 span 1 span 1 span 1 span 1 span 1 span 1 span 1 span 1

span 4 span 4 span 4

span 4 span 8

span 6 span 6

span 12

The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).

Grid Classes

The Bootstrap 5 grid system has six classes:

- `.col-` (extra small devices - screen width less than 576px)
- `.col-sm-` (small devices - screen width equal to or greater than 576px)
- `.col-md-` (medium devices - screen width equal to or greater than 768px)
- `.col-lg-` (large devices - screen width equal to or greater than 992px)
- `.col-xl-` (xlarge devices - screen width equal to or greater than 1200px)
- `.col-xxl-` (xxlarge devices - screen width equal to or greater than 1400px)

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

Tip: Each class scales up, so if you want to set the same widths for `sm` and `md`, you only need to specify `sm`

Basic Structure of a Bootstrap 5 Grid

The following is a basic structure of a Bootstrap 5 grid:

```
<!-- Control the column width, and how they should appear on different devices -->
<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>

<!-- Or let Bootstrap automatically handle the layout -->
<div class="row">
  <div class="col"></div>
  <div class="col"></div>
  <div class="col"></div>
```

```
</div>
```

First example: create a row (`<div class="row">`). Then, add the desired number of columns (tags with appropriate `.col-*-*` classes). The first star (*) represents the responsiveness: sm, md, lg, xl or xxl, while the second star represents a number, which should add up to 12 for each row.

Second example: instead of adding a number to each `col`, let bootstrap handle the layout, to create equal width columns: two `"col"` elements = 50% width to each col, while three cols = 33.33% width to each col. Four cols = 25% width, etc. You can also use `.col-sm|md|lg|xl|xxl` to make the columns responsive.

Below we have collected some examples of basic Bootstrap 5 grid layouts.

Three Equal Columns

The following example shows how to create three equal-width columns, on all devices and screen widths:

```
<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

Responsive Columns

The following example shows how to create four equal-width columns starting at tablets and scaling to extra large desktops. **On mobile phones or screens that are less than 576px wide, the columns will automatically stack on top of each other**:

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

Two Unequal Responsive Columns

The following example shows how to get two various-width columns starting at tablets and scaling to large extra desktops:

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

Boostarp 5 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`:

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`.

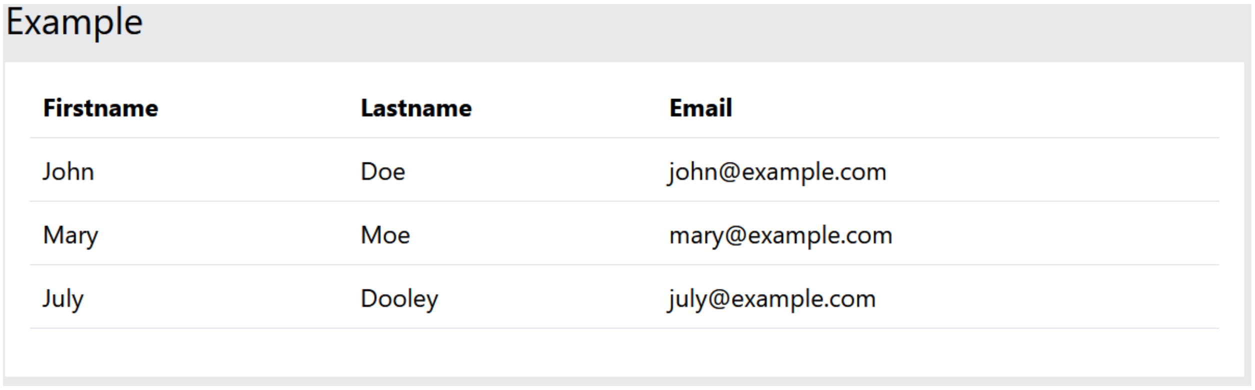
The `.bg-color` classes above does not work well with text, or atleast then you have to specify a proper `.text-color` class to get the right text color for each background.

However, you can use the `.text-bg-color` classes and Bootstrap will automatically handle the appropriate text color for each background color:

Boostarp 5 Tables
Basic Table

A basic Bootstrap 5 table has a light padding and horizontal dividers.

The `.table` class adds basic styling to a table:

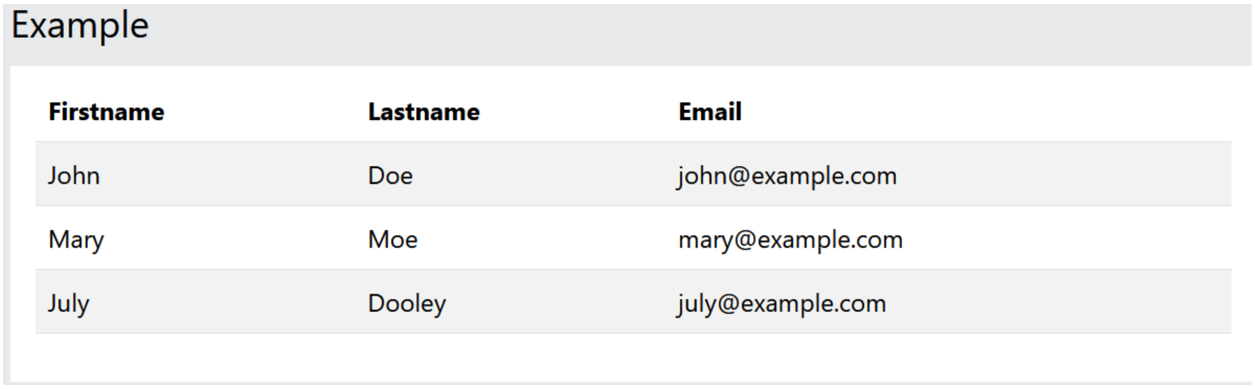


Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Striped Rows

The `.table-striped` class adds zebra-stripes to a table:

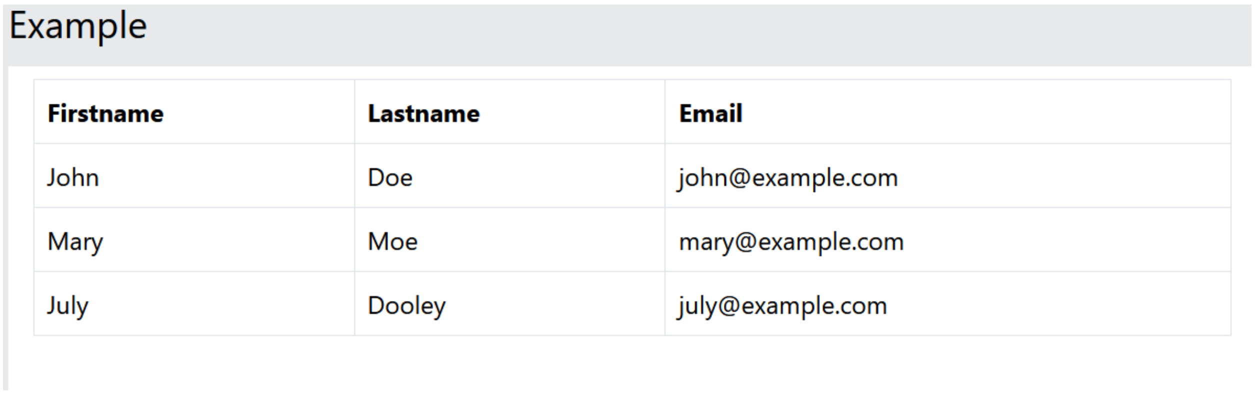


Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Bordered Table

The `.table-bordered` class adds borders on all sides of the table and cells:

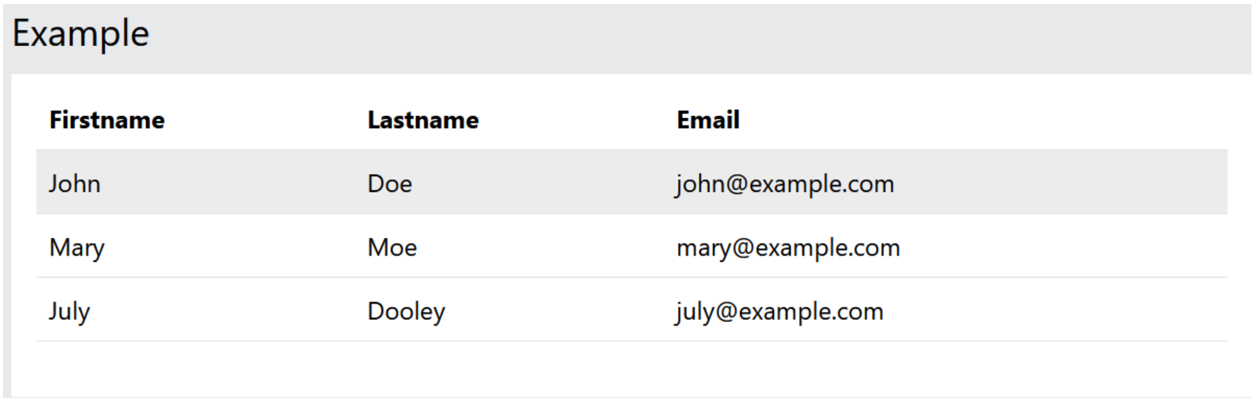


Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Hover Rows

The `.table-hover` class adds a hover effect (grey background color) on table rows:

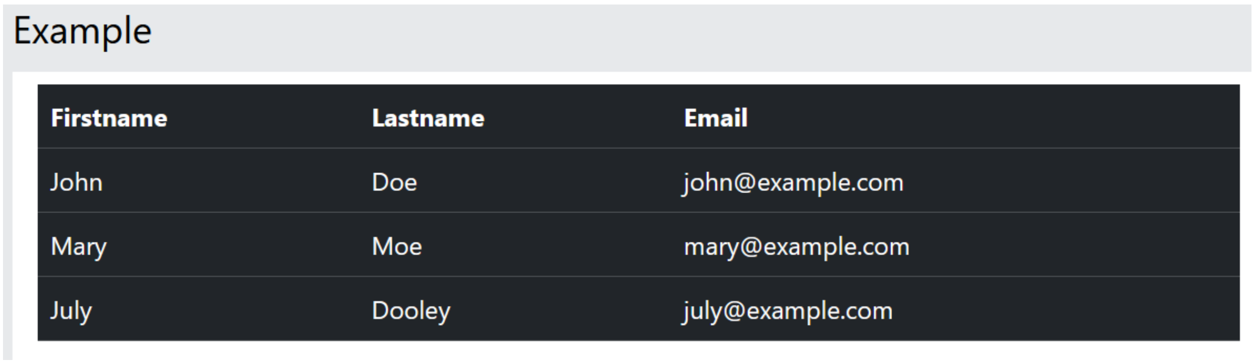


Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Black/Dark Table

The `.table-dark` class adds a black background to the table:

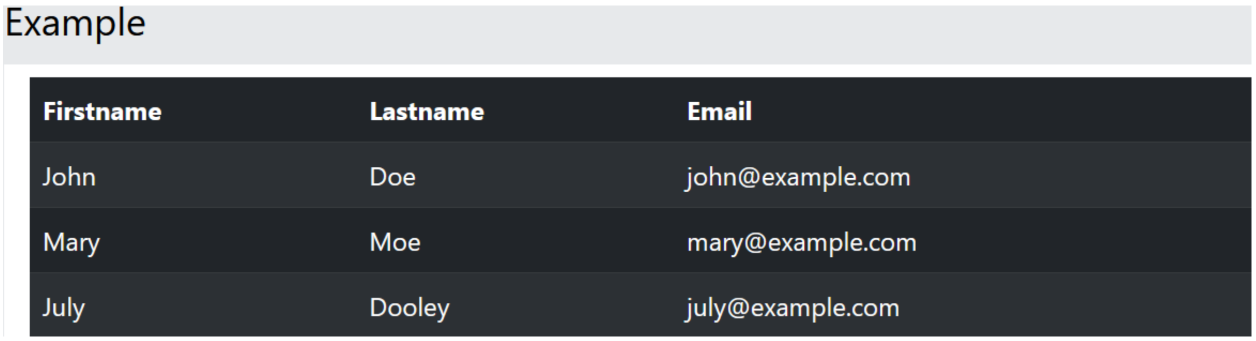


Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Dark Striped Table

Combine `.table-dark` and `.table-striped` to create a dark, striped table:

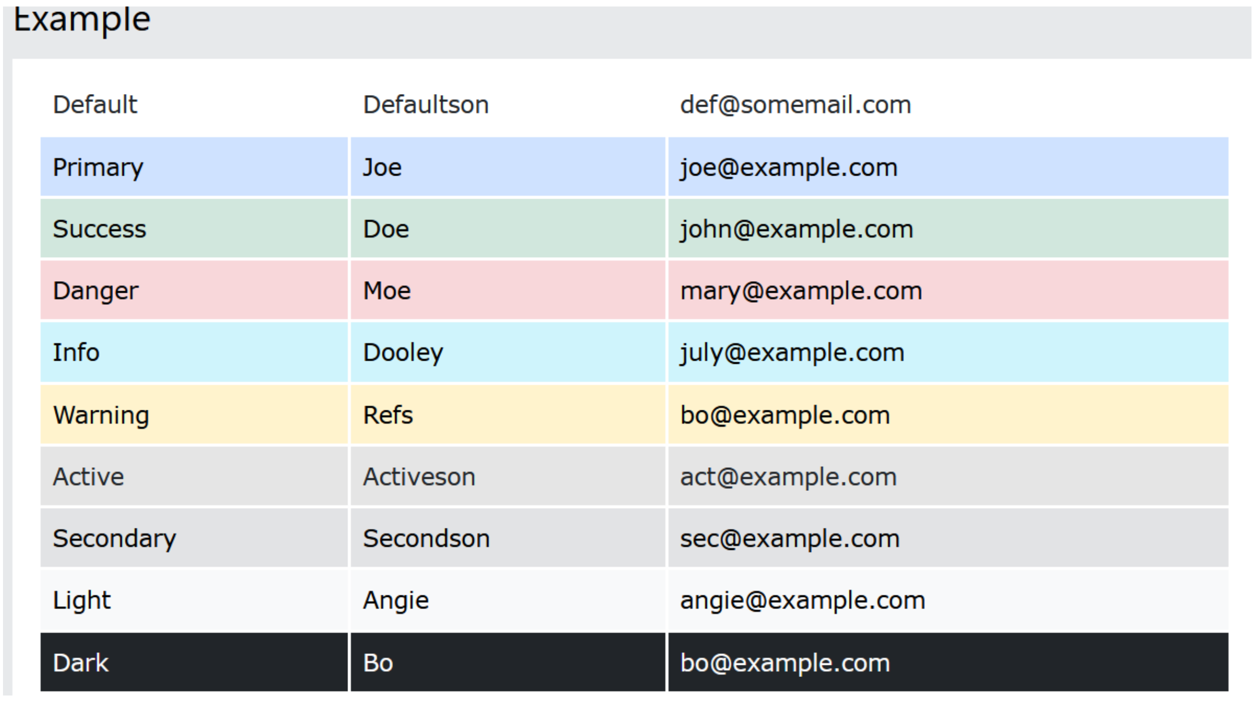


Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Contextual Classes

Contextual classes can be used to color the whole table (`<table>`), the table rows (`<tr>`) or table cells (`<td>`).



Example

Default	Defaultson	def@somemail.com
Primary	Joe	joe@example.com
Success	Doe	john@example.com
Danger	Moe	mary@example.com
Info	Dooley	july@example.com
Warning	Refs	bo@example.com
Active	Activeson	act@example.com
Secondary	Secondson	sec@example.com
Light	Angie	angie@example.com
Dark	Bo	bo@example.com

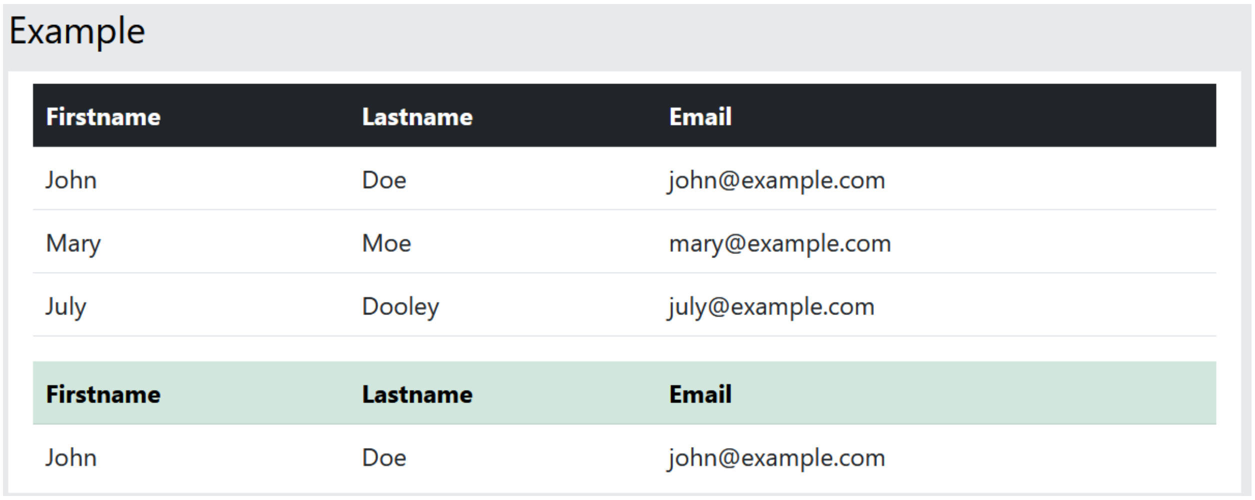
The contextual classes that can be used are:

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

Class	Description
`.table-light`	Light grey table or table row background
`.table-dark`	Dark grey table or table row background

Table Head Colors

You can also use any of the contextual classes to only add a background color to the table header:



Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Firstname	Lastname	Email
John	Doe	john@example.com

Responsive Tables

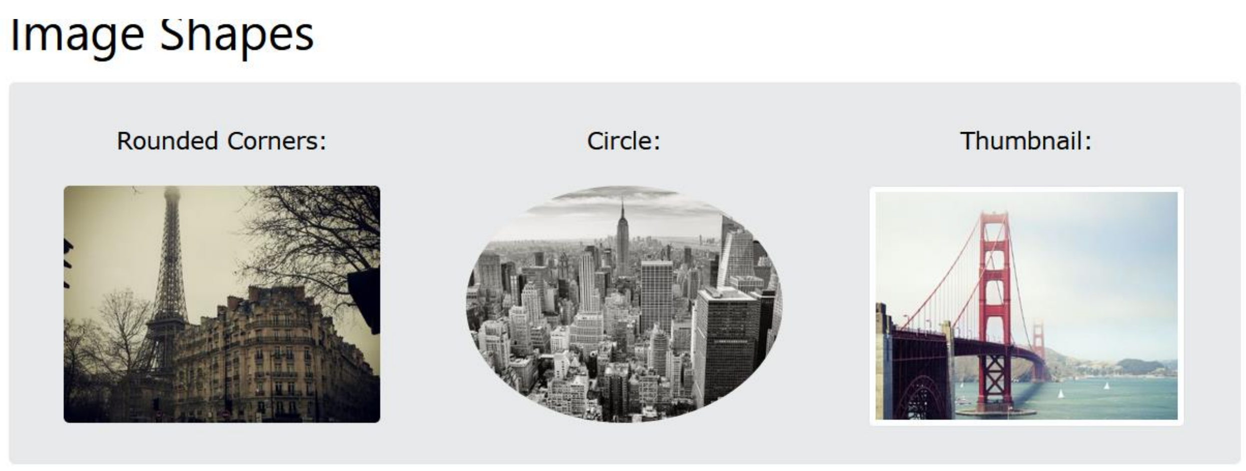
The `.table-responsive` class adds a scrollbar to the table when needed (when it is too big horizontally):

```
<div class="table-responsive">
  <table class="table">
    ...
  </table>
</div>
```

You can also decide when the table should get a scrollbar, depending on the screen width:

Class	Screen width
`.table-responsive-sm`	< 576px
`.table-responsive-md`	< 768px
`.table-responsive-lg`	< 992px
`.table-responsive-xl`	< 1200px
`.table-responsive-xxl`	< 1400px

Boostarp 5 Images



Rounded Corners

The `.rounded` class adds rounded corners to an image:

Circle

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

Thumbnail

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

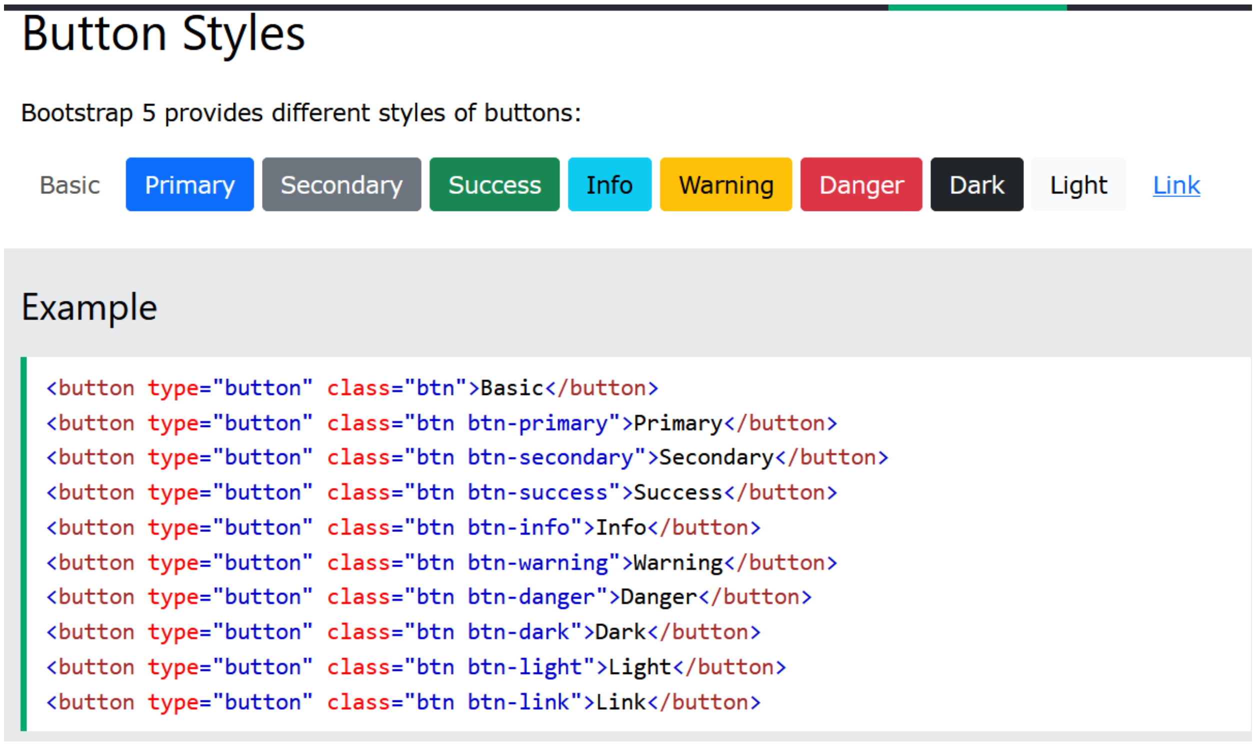
Responsive Images

Images come in all sizes. So do screens. Responsive images automatically adjust to fit the size of the screen.

Create responsive images by adding an `.img-fluid` class to the `<img>` tag. The image will then scale nicely to the parent element.

The `.img-fluid` class applies `max-width: 100%;` and `height: auto;` to the image:

Boostarp 5 buttons



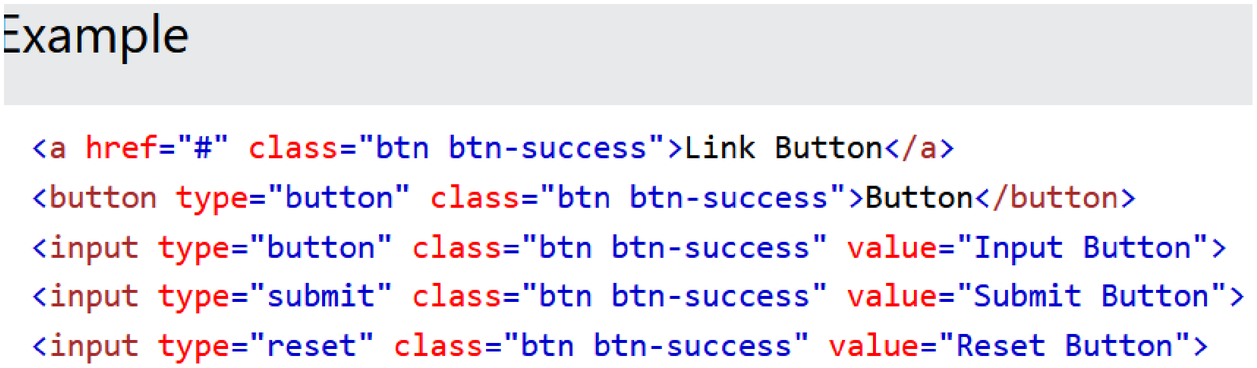
Button Styles

Bootstrap 5 provides different styles of buttons:

Example

```
<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>
```

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



Example

```
<a href="#" class="btn btn-success">Link Button</a>
<button type="button" class="btn btn-success">Button</button>
<input type="button" class="btn btn-success" value="Input Button">
<input type="submit" class="btn btn-success" value="Submit Button">
<input type="reset" class="btn btn-success" value="Reset Button">
```

Button Outline

Bootstrap 5 also provides eight outline/bordered buttons.

Move the mouse over them to see an additional "hover" effect:

Primary Secondary Success Info Warning Danger Dark Light

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-info">Info</button>
<button type="button" class="btn btn-outline-warning">Warning</button>
<button type="button" class="btn btn-outline-danger">Danger</button>
<button type="button" class="btn btn-outline-dark">Dark</button>
<button type="button" class="btn btn-outline-light text-dark">Light</button>
```

Button Sizes

Use the `.btn-lg` class for large buttons or `.btn-sm` class for small buttons:

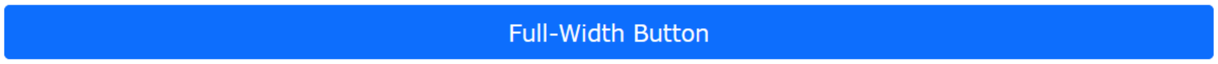
Large Default Small

Example

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

Block Level Buttons

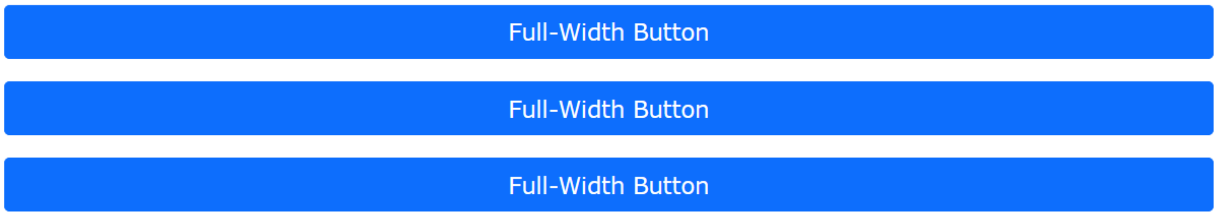
To create a block level button that spans the entire width of the parent element, use the `.d-grid` "helper" class on the parent element:



Example

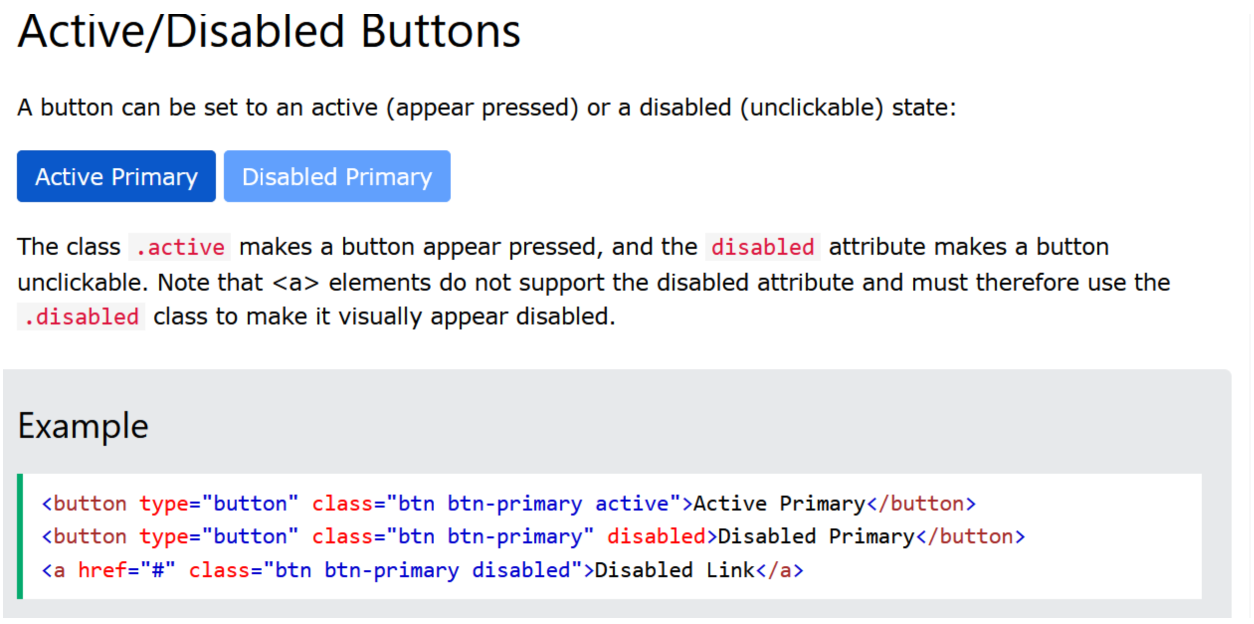
```
<div class="d-grid">
  <button type="button" class="btn btn-primary btn-block">Full-Width Button</button>
</div>
```

If you have many block-level buttons, you can control the space between them with the `.gap-*` class:



Example

```
<div class="d-grid gap-3">
  <button type="button" class="btn btn-primary btn-block">Full-Width Button</button>
  <button type="button" class="btn btn-primary btn-block">Full-Width Button</button>
  <button type="button" class="btn btn-primary btn-block">Full-Width Button</button>
</div>
```



Active/Disabled Buttons

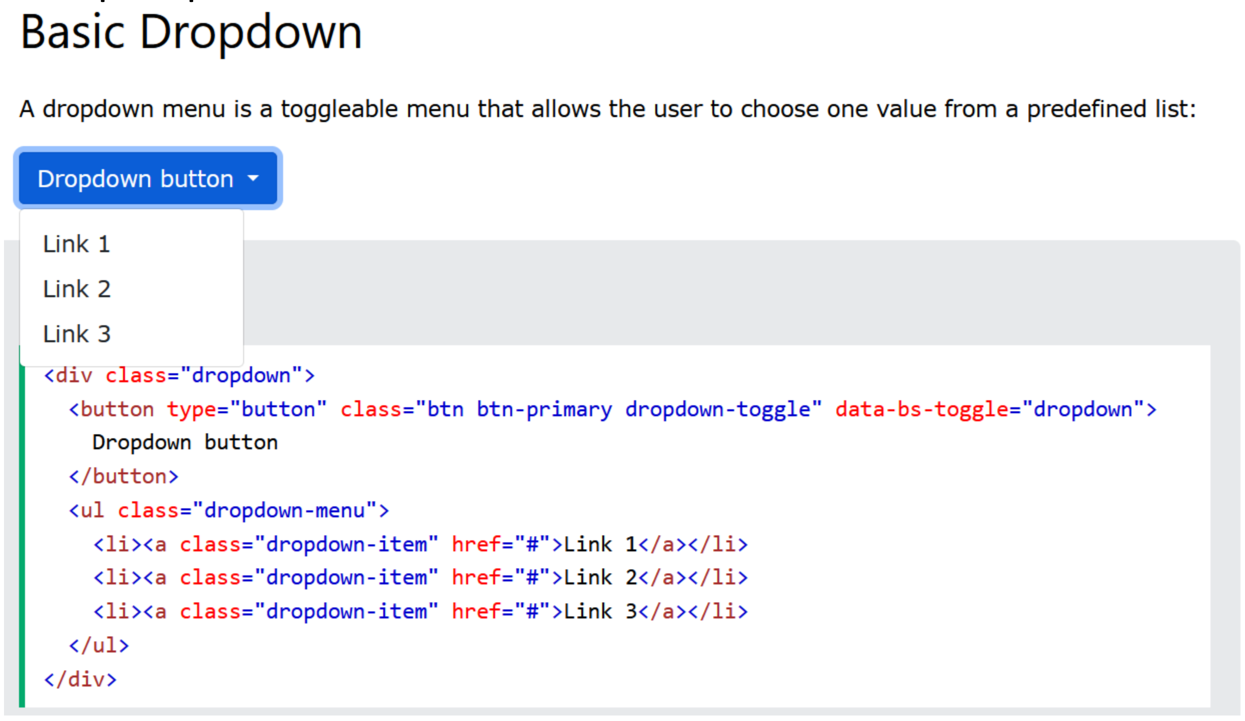
A button can be set to an active (appear pressed) or a disabled (unclickable) state:

The class `.active` makes a button appear pressed, and the `disabled` attribute makes a button unclickable. Note that <a> elements do not support the disabled attribute and must therefore use the `.disabled` class to make it visually appear disabled.

Example

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

Boostarp 5 Dropdowns



Basic Dropdown

A dropdown menu is a toggleable menu that allows the user to choose one value from a predefined list:

```
<div class="dropdown">
  <button type="button" class="btn btn-primary dropdown-toggle" data-bs-toggle="dropdown">
    Dropdown button
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Link 1</a></li>
    <li><a class="dropdown-item" href="#">Link 2</a></li>
    <li><a class="dropdown-item" href="#">Link 3</a></li>
  </ul>
</div>
```

The `.dropdown` class indicates a dropdown menu.

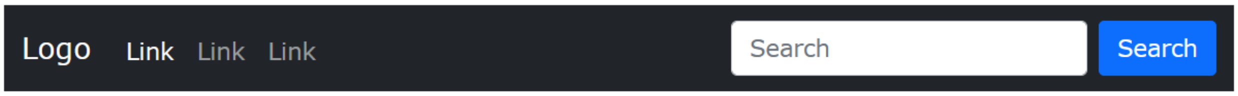
To open the dropdown menu, use a button or a link with a class of `.dropdown-toggle` and the `data-bs-toggle="dropdown"` attribute.

Add the `.dropdown-menu` class to a `<div>` element to actually build the dropdown menu. Then add the `.dropdown-item` class to each element (links or buttons) inside the dropdown menu.

Boostarp 5 Navbar

Navigation Bars

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 `<div>`) with `class="navbar-nav"`. Then add `<li>` elements with a `.nav-item` class followed by an `<a>` element with a `.nav-link` class:

Example

```
<!-- A grey horizontal navbar that becomes vertical on small screens -->
<nav class="navbar navbar-expand-sm bg-light">

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

Vertical Navbar

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

Link 1
Link 2
Link 3

Example

```
<!-- A grey vertical navbar -->
<nav class="navbar bg-light">
  ...
</nav>
```

Centered Navbar

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

Link 1   Link 2   Link 3

Example

```
<nav class="navbar navbar-expand-sm bg-light justify-content-center">
  ...
</nav>
```

Colored Navbar

Active   Link   Link   Disabled

Active   Link   Link   Disabled

Active   Link   Link   Disabled

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`)

Tip: 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.

SQL Tutorial

```
<!-- Grey with black text -->
<nav class="navbar navbar-expand-sm bg-light navbar-light">
  <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" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
  </div>
</nav>

<!-- Black background with white text -->
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">...</nav>
```

Brand / Logo

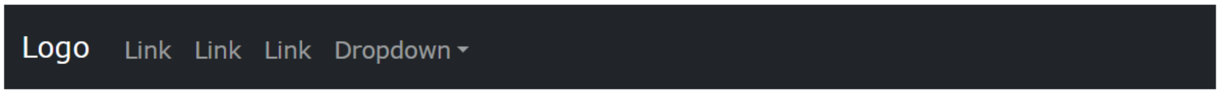
The `.navbar-brand` class is used to highlight the brand/logo/project name of your page:

Logo

Example

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

Navbar With Dropdown



Navbars can also hold dropdown menus:

Example

```
<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>
```

Basics of Java script

- JavaScript (JS) is the programming language of web development. It was initially used to 'make web pages alive'.
- JS programs are called scripts.
- Unlike Java, scripts do not require any compilation to run. JS can execute on a browser, a server and any device that has a 'JavaScript Engine'.
- JS does not provide low-level access to memory or CPU, which is why it is a 'safe' programming language.
- With the help of JS, you can perform the following:
- Adding or modifying HTML content on a page, including, but not restricted to, showing/hiding elements as well as changing the size, style and HTML attributes
- Reacting to user actions such as hovering, mouse clicks, key presses, etc.
- Sending requests to remote servers over the network and saving the data on the client side.

Where to put java script?

- JavaScript (JS) can be placed either internally in an HTML file using the <script> tag, where you can write all the scripts inside the <script> tag, or externally in a JavaScript file (that ends with a '.js' extension), which can be added as a source in the <script> tag.
- The <script> tag can be placed either in the <head> or in the <body> part of the HTML document.
- However, because JS is a render blocker, it is preferable to call these scripts in the <body> tag after all the HTML elements are rendered.

JavaScript

- Interpreted programming language
- E.g. Browser is the best interpreter for JS
- Used for adding programming login in HTML pages
- To make a html page dynamic (DHTML)
- Original name: LiveScript
- Standardized by ECMA (Europian Computers Manufacture's Association)
    - ECMAScript 4
    - ECMAScript 5:
        - Getters/setters
        - Function overloading

    - ECMAScript 6
- To write JS Code
    - Using HTML
        - Use <script> tag to write js code
        - <script> may appear in both head and body sections
    - Using command line (console)
- Case sensitive language
    - firstName and FirstName are different variables
- Types
    - Internal
    - External
- JS DOES NOT support pointers ☺
- JS is loosely typed language
- JS supports dynamic binding

Data Types

- All data types in JS are inferred
- Automatically decided by JS by looking at the variable's **CURRENT** value
- Types
    - Number:
        - used to keep both integer and float values
        - E.g.
            - num = 100;
            - salary = 10.15;
    - String:
        - E.g.
            - firstName = "steve";
            - firstName = 'steve';
    - Boolean:
        - Can have only true or false value
        - E.g.
            - canVote = true;
    - undefined
    - object
        - Everything in JS is an object
        - Functions are also considered objects

Pre-Defined Objects

- console: used to represent the browser console
- window: used to represent the browser GUI
- document:

Pre-Defined Values

- NaN: Not a Number
    - "test" * "test1" = NaN
    - data type: number
- Infinity:
    - Only when a number is divided by zero (0)
- undefined

Function

- function keyword is used to define a function
- function declaration and definitions CAN NOT separated
- Syntax:

```
function <function name>(<param list>) {
    // function body here
}
```

- IMP
  - Function can not supply the param data type
  - There is no change in function signature (prototype) even if the function is returning a value
    - return keyword is used to return a value
  - function **CAN NOT** decide the number of parameters and data type of parameters

```
function function1(n1) {
    // body
}

// n1 is declared as undefined
function1();

// n1 is declared as number (20)
function1(20);

// n1 is declared as string ("test")
function1("test");

// n1 is declared as number (10)
// 20 and 30 will be ignored (*)
function1(10, 20, 30);
```

  - function **CALLER** decides the number of parameters and data type of parameters
  - function can be called before its declaration (definition)
  - may define multiple functions with same name
    - **ONLY THE LAST (bottom most)** definition will be kept and remaining definitions will be ignored
  - Every function receives a hidden parameter called as **arguments**
    - arguments is an array of all the values passed to the function (irrespective of the named parameters)
    - E.g.

```
function add() {
    console.log(arguments);
}

add(10, 20); // arguments = [10, 20]
add(10, 20, 30, 40); // arguments = [10, 20, 30, 40]
```

  - Every function receives a hidden member named **this**
    - In case if the function is called without using an object this reference will refer to Object function

```
function test() {
    console.log(this);
}

test();
```

    - In case if the function is called by using an object then the object becomes this

```
function test() {
      console.log(this);
}

var p = new Object();
p.myTest = test;

// p becomes this reference
p.myTest();

// prints
// p object {…}
```

- Function Alias
  - Another name given to a function
  - Variable of type function
  - E.g.

```
function function1() {}
// function alias
var myfunction1 = function1;
myfunction1();
```

- Anonymous Function
  - Function without a name
  - E.g.

```
var multiply = function(p1, p2) {
            console.log("p1 * p2 = " + (p1 * p2));
};
```

Array

- Collection of objects
- May contain similar or dis-similar objects (values)
  - E.g.

```
var  array = [1, 2, 3, 4]
var  array = [1, "test", 2, true, false, "test2", 3, 4]
```

- to print / process all members in the array

```
for (var index = 0; index < array.length; index++ )
      // code here
}
```

- to manipulate the values
  - push: to add a value at the last position
  - pop: to remove the last element from array
  - splice: to remove an item at an index

Variable Scope

- Local scope:
  - Declare a variable with **var** keyword

    - Function parameters are **ALWAYS** declared as local variables
    - E.g.

```
function function1 () {
        // local
        var num = 100;
}
```

- Global scope:
    - Declare a variable without using **var** keyword
    - Global variables CAN be declared inside a function
    - E.g.

```
// global
salary = 10.15;

function function1 () {
        // global
        num = 100;
}
```

Conversion

- string to number:
    - parseInt()
        - converts string to number in integer format
        - E.g.
            - parseInt(“10.20”); // 10
    - parseFloat()
        - converts string to number in float format
        - E.g.
            - parseInt(“10.20”); // 10.20
- number to string:
    - E.g.
        - “” + 20; // “20” (string)
        - ‘’ + 20; // “20” (string)

JSON

- JavaScript Object Notation
- Types
    - Object
        - Collection of Key-value pairs separated by Comma(,)
        - Use {}
        - E.g.

```
var movie = {
        “title”: ”cars”,
        “price”:  30
}
```

        - Where title and price are keys while cars and 30 are values
        - To access values

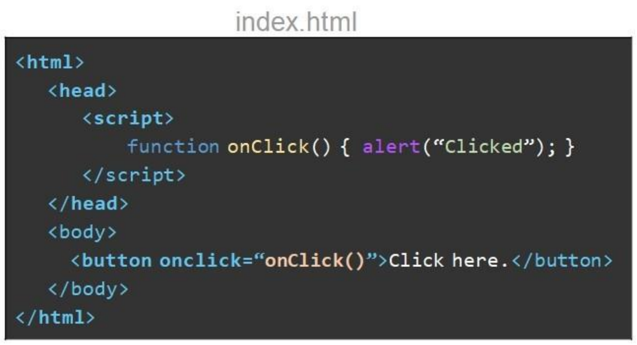
```
console.log(“title: “ + movie.title);
console.log(“price: “ + movie.price);
```

    - Array
        - Collection of objects
        - Use []

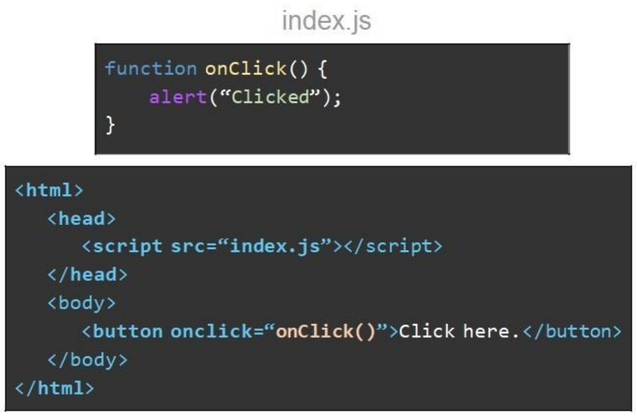
- E.g.
```
var movies = [
        { “title”:”cars”, “price”: 30 },
        { “title”:”moana”, “price”: 40 }
]
```

- To access all objects in array
```
for (var index = 0; index < movies.length; index++) {
        var item = movies[index];
        console.log(“title :” + item.title);
        console.log(“price :” + item.price);
}
```

Internal vs External:



```
<html>
   <head>
      <script>
          function onClick() { alert("Clicked"); }
      </script>
   </head>
   <body>
     <button onclick="onClick()">Click here.</button>
   </body>
</html>
```



```
function onClick() {
    alert("Clicked");
}
```

```
<html>
   <head>
      <script src="index.js"></script>
   </head>
   <body>
      <button onclick="onClick()">Click here.</button>
   </body>
</html>
```

Syntax and Terms:

- JavaScript (JS) programs are a list of programming instructions called **statements** that are executed by web browsers.
- JS statements consist of values, operators, expressions, keywords and comments. They are executed in a waterfall fashion, that is, one by one, in the order in which they are written.
- Generally, a JavaScript (JS) code consists of statements with variables that store values or expressions.

```
var x, y; //Declaring variables

var num1 = 24; //Assigning values to variable

var num2 = 30; //Assigning values to variable

var total = num1 + num2; //Computing expressions and storing result in variable
```

- In a programming language, **variables** are used to store data values. In JS, variables are declared using the **var** keyword. An equal sign ('=') is used to assign value to the variables.
- JS **values** can be strings, numbers, arrays, objects, booleans or expressions. String values need to be written within single or double quotes, for example, 'John Doe' or "John Doe", and number values should not be written within quotes, for example, 2 or 200.45.

- JS operators can be used to compute values; e.g.,

  var sum = (24 + 200) * 50;

- JS **expressions** are a combination of values, variables and operators

  var mul = 10 * 10;

  var x = num1 * 10;

  var name = "John" + " " + "Doe";

- For single-line JS comments, you can use "//", and for multiline comments, you can use "/* */".

- JavaScript (JS) programs are a list of programming instructions called **statements** that are executed by web browsers.
- JS statements consist of values, operators, expressions, keywords and comments. They are executed in a waterfall fashion, that is, one by one, in the order in which they are written.
- JS statements end with a semicolon (';'). Semicolons essentially separate JS statements. It is not mandatory to add a semicolon, but it is highly recommended. Using semicolons, you can add multiple JS statements in one line. For example,

  var name = "John Doe";

  var age = 24;

- JS ignores multiple spaces. However, it is recommended to add white spaces to your code to make it more readable. For example,
- var name = "John";
- A best practice is to avoid having more than 80 characters in one line for better readability. The best place to break a JS code to a new line is immediately after the operator.
- JavaScript (JS) variables are containers for storing data values. In earlier examples, you learnt that variables can store values and expressions.
- All JS variables must be identified with a unique name called **identifier**. The rules for writing an identifier name are as follows:
- Names can contain letters, digits, the dollar sign ('$') and the underscore sign ('_').
- Names should always begin with a letter.
- JavaScript is case-sensitive, i.e., *name* and *Name* are different variables.

- Reserved JS words cannot be used for an identifier.
- Get the details about reserved words from:
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar
- The best practice is to **not** keep a name, such as x, y, z, that does not indicate the meaning or significance of that variable. The identifier name should signify the meaning of the variable; for example, *name* and *age*.

- In JS, you can assign a value to the variable using the ‘=’ operator.

OOP JavaScript

- Everything in JS is an Object, even functions are also considered as objects
- collection of properties and functions/methods
- To create an object/instance
    - Using Object function
        - E.g.

          ```
          var person = new Object();
          person.name = “person1”; // name is a property
          ```

    - Using Constructor Function
    - Using JSON
    - Using class: TypeScript
- Instance / object
    - Collection of properties and methods
        - Where a property is used to store the data (state)
            - To add/access property in an instance
                - Use dot syntax (.)

                  ```
                  var person = new Object();
                  person.name = “steve”;
                  person.address = “USA”;
                  console.log(“Name: “ + person.name);
                  console.log(“Address: “ + person.address);
                  ```

                - Use []

                  ```
                  var car = new Object();
                  car[“model”] = “Nano”;
                  car[“company”] = “Tata”;
                  console.log(“Model: “ + car[“model”];
                  console.log(“Company: “ + car[“company”];
                  ```

        - Where a method is a function alias added inside an object (behavior)

          ```
          function printRecord() {}

          var person = new Object();
          person.name = “person1”;
          person.myPrintRecord = printRecord;
          console.log(“name : “ + person.name);
          person.myPrintRecord();
          ```

Object

- Built-in function
- Using Object JS allows developer to create an instance
- Root function

jQuery

- External JS library (collection of functions)
- Open source and free library
- Large community
- To load jQuery js
  - <script src=”js/jquery.js”></script>
- To use jQuery
  - Use $ to load jQuery functions
  - Syntax:
    - $(<css selector>).<function>()
    - E.g.
      $(‘#div1’).hide();

Introduction to jQuery

- jQuery is a fast, small, and feature-rich JavaScript library.
- It simplifies the process of traversing HTML documents, handling events, animating elements, and AJAX interactions.
- It provides a powerful set of tools for front-end web development.

Getting Started

- To use jQuery, include the jQuery library in your HTML file:

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

- Or download jQuery and include it locally.

Basic Syntax

- The basic syntax of jQuery involves selecting elements and performing actions on them using methods.
- It follows the pattern: **$(selector).action()**

Selectors

- jQuery uses CSS-style selectors to select HTML elements.
- Common selectors include element selectors, class selectors, ID selectors, attribute selectors, etc.

```
// Selecting elements
$("p") // Selects all <p> elements
$(".classname") // Selects all elements with class "classname"
$("#id") // Selects the element with id "id"
$("[attribute]") // Selects all elements with a specified attribute
```

Actions

- **Once elements are selected, various actions can be performed on them using jQuery methods.**

```
// Manipulating elements
$(selector).hide() // Hides the selected element(s)
```

```
$(selector).show() // Shows the selected element(s)
$(selector).addClass("classname") // Adds a class to the selected element(s)
$(selector).removeClass("classname") // Removes a class from the selected element(s)
$(selector).toggle() // Toggles between hide() and show()
$(selector).html(content) // Sets the HTML content of the selected element(s)
$(selector).text(content) // Sets the text content of the selected element(s)
$(selector).attr(attribute, value) // Sets attribute value of the selected element(s)
```

Events

- jQuery simplifies event handling by providing methods to attach event handlers.

```
// Event handling
$(selector).click(function() {
  // Code to be executed when element is clicked
});
--------------------------------------------------------------------------------------
$(selector).mouseenter(function() {
  // Code to be executed when mouse enters element
});
------------------------------------------------------------------------------------
$(selector).mouseleave(function() {
  // Code to be executed when mouse leaves element
});
```

AJAX

- jQuery simplifies AJAX interactions, making it easier to load data from the server without refreshing the page.

```
// AJAX
$.ajax({
  url: "example.json",
  type: "GET",
  data: { id: 1 },
  success: function(response) {
    // Code to be executed on successful AJAX call
  },
  error: function(xhr, status, error) {
    // Code to be executed on AJAX error
  }
});
```

Effects and Animations

- jQuery provides methods for creating animations and effects on elements.

```
// Effects and animations
$(selector).fadeIn() // Fades in the selected element(s)
$(selector).fadeOut() // Fades out the selected element(s)
$(selector).slideUp() // Slides up the selected element(s)
$(selector).slideDown() // Slides down the selected element(s)
$(selector).animate({params}, speed) // Animates the selected element(s)
```