WEEK-6 NOTES

▼ HTML

What is HTML?

- **HTML** stands for **HyperText Markup Language**.
- It is the **standard language** used to **create and structure webpages**.
- HTML is **not a programming language**, it's a **markup language** — it defines **structure and meaning** of content on the web.

Meaning of "HyperText" and "Markup Language"

- **HyperText** → text with links to other documents or resources.
- **Markup Language** → a way to mark up content with tags to tell the browser how to display it.

Why HTML is Used?

- To **structure content** like headings, paragraphs, images, links, lists, tables, etc.
- It forms the **skeleton** of every website (CSS adds style, JS adds functionality).
- Used by **browsers** to render visual content.

Versions of HTML

Version	Year	Description
HTML 1.0	1993	Very basic, limited tags

Version	Year	Description
HTML 2.0	1995	Added forms and basic tables
HTML 3.2	1997	Introduced style and scripting
HTML 4.01	1999	Focused on structure and presentation separation
XHTML	2000	Stricter XML-based syntax
HTML5	**2014**	Latest version, supports multimedia, APIs, semantics

Basic Structure of an HTML Document

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.
0">
  <title>My First Page</title>
</head>
<body>
  <h1>Hello, HTML!</h1>
  <p>This is my first webpage.</p>
</body>
</html>
```

Explanation

- `<!DOCTYPE html>` → declares HTML5 version
- `<html>` → root element
- `<head>` → contains metadata (title, charset, styles, etc.)
- `<body>` → visible content of the webpage

HTML Page Layout

A typical web page layout includes:

- Header

- Navigation bar
- Main content area
- Sidebar
- Footer

Example:

```
<header>Header Section</header>
<nav>Navigation Links</nav>
<main>Main Content</main>
<aside>Sidebar</aside>
<footer>Footer</footer>
```

HTML Elements & Tags

- **Tags**: The building blocks of HTML. Written inside `< >`.
- **Elements**: Start tag + content + end tag.

Example:

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

Types of Elements

- **Block-level**: start on a new line (e.g., `<div>`, `<p>`, `<h1>` – `<h6>`, `<section>`)
- **Inline**: stay in the same line (e.g., `<span>`, `<a>`, `<img>`, `<strong>`)

HTML Attributes

- Provide **extra information** about elements.
- Always written in **name="value"** pairs.

Example:

```
<img src="photo.jpg" alt="Profile Picture" width="200">
```

Common attributes:

- `id`, `class`
- `src`, `href`
- `alt`, `title`
- `style`

Text Formatting Tags

Tag	Description
`<b>`	Bold text
`<strong>`	Important text
`<i>`	Italic text
`<em>`	Emphasized text
`<u>`	Underlined text
`<mark>`	Highlighted text
`<small>`	Smaller text
`<sub>`	Subscript
`<sup>`	Superscript

Headings & Paragraphs

- Headings: `<h1>` to `<h6>` (SEO important!)
- Paragraph: `<p>` tag

Example:

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<p>This is a paragraph explaining something.</p>
```

Links in HTML

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

Attributes:

- `href` → URL
- `target="_blank"` → opens in new tab
- `title` → shows tooltip

Images

```html
<img src="image.jpg" alt="Description" width="300" height="200">
```

Always include `alt` text for accessibility and SEO.

Lists

Ordered List

```html
<ol>
  <li>HTML</li>
  <li>CSS</li>
  <li>JS</li>
</ol>
```

Unordered List

```html
<ul>
  <li>Apple</li>
  <li>Banana</li>
</ul>
```

Description List

```html
<dl>
  <dt>HTML</dt>
  <dd>Markup language for web.</dd>
</dl>
```

Tables

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Nisharg</td>
    <td>21</td>
  </tr>
</table>
```

Tags:

- `<table>` → defines table
- `<tr>` → row
- `<th>` → header cell
- `<td>` → data cell

Forms

Used to **collect user input.**

```
<form action="/submit" method="post">
  <label>Name:</label>
  <input type="text" name="username">
  <input type="submit" value="Send">
</form>
```

Common input types:

- text, email, password, checkbox, radio, file, date, color, etc.

Semantic HTML (Important for SEO)

Semantic tags **give meaning** to content.

Semantic Tag	Purpose
`<header>`	Page header

Semantic Tag	Purpose
`<footer>`	Footer section
`<article>`	Independent content block
`<section>`	Thematic grouping
`<aside>`	Sidebar or secondary content
`<nav>`	Navigation links
`<main>`	Main content area

Multimedia in HTML

Images

`<img>`

Audio

```
<audio controls>
  <source src="song.mp3" type="audio/mpeg">
</audio>
```

Video

```
<video width="400" controls>
  <source src="video.mp4" type="video/mp4">
</video>
```

HTML5 New Features

- Semantic tags (`<header>`, `<section>`, `<footer>`, `<article>`)
- Multimedia tags (`<audio>`, `<video>`)
- Graphics tags (`<canvas>`, `<svg>`)
- Form enhancements (`<email>`, `<date>`, `<range>`)
- Local storage & session storage
- Geolocation API

HTML Entities

Used for special characters:

Symbol	Entity	Output
`<`	`&lt;`	<
`>`	`&gt;`	>
`&`	`&amp;`	&
`©`	`&copy;`	©

Iframes

Used to embed another webpage.

```
<iframe src="https://example.com" width="600" height="400"></iframe>
```

Meta Tags

Inside `<head>` — provide info to browsers & search engines.

```
<meta charset="UTF-8">
<meta name="description" content="Learn HTML from basics to advanced">
<meta name="keywords" content="HTML, web development, frontend">
<meta name="author" content="Nisharg Soni">
```

Responsive Design (Viewport)

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

HTML with CSS and JS

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

```html
<head>
  <link rel="stylesheet" href="style.css">
  <script src="app.js" defer></script>
</head>
<body>
  <h1>Hello World</h1>
</body>
</html>
```

▼ CSS

What is CSS

- CSS stands for **Cascading Style Sheets.**
- It is used to **style** and **layout** HTML elements on a webpage.
- HTML defines the structure, while CSS defines the look and feel of that structure.

Why CSS is Needed

- To make web pages **visually attractive.**
- To separate **content (HTML)** from **design (CSS).**
- To control the **layout** of multiple pages consistently.

Ways to Add CSS

a. Inline CSS

Applied directly to an HTML element using the `style` attribute.

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

b. Internal CSS

Defined inside the `<style>` tag within the `<head>` section.

```
<head>
  <style>
    p {
      color: green;
    }
  </style>
</head>
```

c. External CSS

Defined in a separate `.css` file and linked using the `<link>` tag.

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

CSS Syntax

```
selector {
  property: value;
}
```

Example:

```
h1 {
  color: red;
  font-size: 24px;
}
```

- **Selector** → which HTML element to style
- **Property** → the style attribute (e.g., color, font-size)
- **Value** → the setting for that property

Selectors in CSS

Basic Selectors

```
p { }           /* Element selector */
#heading { }      /* ID selector */
.classname { }    /* Class selector */
```

Grouping Selectors

```
h1, h2, h3 {
  color: navy;
}
```

Universal Selector

```
* {
  margin: 0;
  padding: 0;
}
```

Descendant Selector

```
div p {
  color: gray;
}
```

Attribute Selector

```
input[type="text"] {
  border: 1px solid black;
}
```

Colors in CSS

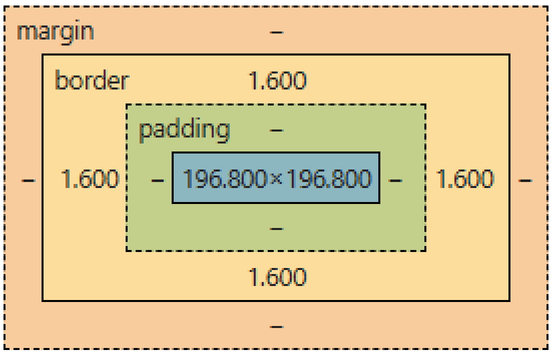
- Named colors: `red`, `blue`, `green`
- Hex: `#ff0000`
- RGB: `rgb(255, 0, 0)`

- RGBA: `rgba(255, 0, 0, 0.5)` (with opacity)
- HSL: `hsl(0, 100%, 50%)`

CSS Comments

```
/* This is a comment */
```

7. CSS Box Model



Every HTML element is a box made up of:

1. **Content** – The actual text or image
2. **Padding** – Space between content and border
3. **Border** – Surrounds the padding
4. **Margin** – Space outside the border

Example:

```
div {
  margin: 10px;
  padding: 15px;
  border: 2px solid black;
}
```

CSS Units

- **Absolute units:** px, cm, mm, in
- **Relative units:** %, em, rem, vh, vw

Example:

```
p {
  font-size: 16px;
  margin: 2em;
}
```

Font Styling

```
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
  font-weight: bold;
  font-style: italic;
  text-align: center;
  text-transform: uppercase;
}
```

Background Properties

```
body {
  background-color: #f0f0f0;
  background-image: url("image.jpg");
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
}
```

Borders and Outlines

```
div {
  border: 2px solid blue;
  border-radius: 10px;
  outline: 2px dotted red;
}
```

Text Styling

```
p {
  color: #333;
  text-align: justify;
  text-decoration: underline;
  letter-spacing: 1px;
  word-spacing: 5px;
  line-height: 1.5;
}
```

Links Styling

```
a:link {
  color: blue;
}
a:visited {
  color: purple;
}
a:hover {
  color: red;
  text-decoration: underline;
}
a:active {
  color: green;
}
```

List Styling

```
ul {
  list-style-type: square;
}
ol {
  list-style-type: upper-roman;
}
```

Display Property

- `block` – takes full width
- `inline` – takes only necessary width
- `inline-block` – behaves like inline but allows height/width
- `none` – hides the element

```
span {
  display: inline-block;
}
```

Positioning in CSS

- `static` – default
- `relative` – positioned relative to itself
- `absolute` – positioned relative to nearest positioned ancestor
- `fixed` – stays fixed when scrolling
- `sticky` – toggles between relative and fixed

Example:

```
div {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

Float and Clear

Used for simple layouts before flexbox/grid existed.

```
img {
  float: right;
}
p {
  clear: both;
}
Overflow Property
```

```
div {
  overflow: auto;
}
```

Values: `visible`, `hidden`, `scroll`, `auto`

Z-Index

Controls the stacking order of positioned elements.

```
div {
  position: absolute;
  z-index: 2;
}
```

Pseudo-Classes and Pseudo-Elements

Pseudo-Classes

```
button:hover {
  background-color: yellow;
}
input:focus {
  border-color: green;
}
```

Pseudo-Elements

```
p::first-letter {
  font-size: 24px;
  color: red;
}
p::after {
  content: " (Read more)";
}
```

CSS Combinators

- **Descendant (space)** → `div p`
- **Child (>)** → `div > p`
- **Adjacent (+)** → `div + p`
- **General sibling (~)** → `div ~ p`

Opacity and Transparency

```
div {
  opacity: 0.8;
}
```

CSS Transitions

Used for smooth effects on property change.

```
button {
  background: blue;
  transition: background 0.5s;
}
button:hover {
  background: green;
}
```

CSS Transformations

```css
div {
  transform: rotate(45deg);
  transform: scale(1.5);
  transform: translate(50px, 20px);
}
```

CSS Variables

```css
:root {
  --main-color: blue;
  --font-size: 18px;
}
h1 {
  color: var(--main-color);
  font-size: var(--font-size);
}
```

Flexbox (Flexible Box Layout)

Introduction

- **Flexbox** is a **one-dimensional layout system** in CSS.
- It helps to align, distribute, and space elements **horizontally or vertically** easily.
- It's perfect for building **responsive layouts** without using floats or complex positioning.

Setting Up Flexbox

To make a container a flex container:

```css
.container {
  display: flex;
}
```

Now all **direct child elements** of `.container` become **flex items.**

Flexbox Main Concepts

Axes in Flexbox

- **Main axis** – Defined by `flex-direction`.
- **Cross axis** – Perpendicular to the main axis.

For example:

- If `flex-direction: row`, main axis → horizontal.
- If `flex-direction: column`, main axis → vertical.

Flex Container Properties

`display`

```
display: flex;
display: inline-flex;
```

- `flex` → block-level flex container
- `inline-flex` → inline-level flex container

`flex-direction`

Defines the direction of main axis.

```
.container {
  flex-direction: row;         /* default */
  flex-direction: row-reverse; /* right to left */
  flex-direction: column;      /* top to bottom */
  flex-direction: column-reverse; /* bottom to top */
}
```

`flex-wrap`

Controls whether flex items wrap onto multiple lines.

```
.container {
  flex-wrap: nowrap;   /* default - all in one line */
  flex-wrap: wrap;     /* wrap to next line */
  flex-wrap: wrap-reverse; /* wrap but reverse order */
}
```

flex-flow

Shorthand for `flex-direction` and `flex-wrap`.

```
.container {
  flex-flow: row wrap;
}
```

justify-content

Aligns items **along the main axis.**

```
.container {
  justify-content: flex-start;    /* default */
  justify-content: flex-end;      /* items at end */
  justify-content: center;        /* items centered */
  justify-content: space-between; /* equal space between */
  justify-content: space-around;  /* equal space around */
  justify-content: space-evenly;  /* equal space between and around */
}
```

Example:

```
.container {
  display: flex;
  justify-content: space-between;
}
```

align-items

Aligns items **along the cross axis.**

```
.container {
  align-items: stretch;    /* default */
  align-items: flex-start; /* top (for row) */
  align-items: flex-end;   /* bottom (for row) */
```

```css
  align-items: center;     /* vertically centered */
  align-items: baseline;   /* align text baselines */
}
```

`align-content`

Used **when items wrap** onto multiple lines (affects the entire line).

```css
.container {
  align-content: stretch;      /* default */
  align-content: flex-start;
  align-content: flex-end;
  align-content: center;
  align-content: space-between;
  align-content: space-around;
}
```

Flex Item Properties

Each child item inside a flex container can also be controlled individually.

`order`

Changes the order of flex items.

```css
.item1 { order: 2; }
.item2 { order: 1; }
.item3 { order: 3; }
```

Default order = 0 (lower values appear first).

`flex-grow`

Defines how much an item grows relative to others.

```css
.item1 { flex-grow: 1; }
.item2 { flex-grow: 2; }
```

If space is available, `.item2` grows twice as fast as `.item1`.

`flex-shrink`

Defines how an item shrinks when space is limited.

```
.item1 { flex-shrink: 1; }
.item2 { flex-shrink: 2; }
```

`.item2` will shrink twice as fast as `.item1` when there’s not enough space.

`flex-basis`

Defines the **initial size** of a flex item before space distribution.

```
.item {
  flex-basis: 200px;
}
```

It acts like setting a default width (or height in column direction).

`flex` (Shorthand)

Shorthand for: `flex-grow flex-shrink flex-basis`

```
.item {
  flex: 1 1 200px;
}
```

Or simpler:

```
.item {
  flex: 1; /* equal size items */
}
```

`align-self`

Overrides `align-items` for a specific item.

```
.item1 {
  align-self: center;
}
.item2 {
  align-self: flex-end;
}
```

Common Flexbox Layout Example

```
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
  height: 200px;
}
.item {
  background: lightblue;
  padding: 20px;
}
```

Flexbox Practical Use Cases

- Navbar alignment
- Centering items both vertically and horizontally
- Equal-height cards or boxes
- Responsive layouts

Centering example:

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}
```

CSS Grid Layout

What is CSS Grid

- **Grid** is a **two-dimensional layout system** in CSS.
- It allows designing layouts with rows and columns easily.
- Perfect for complete page layouts or complex components.

Creating a Grid Container

```
.container {
  display: grid;
}
```

All direct child elements become **grid items.**

Defining Rows and Columns

Using `grid-template-columns` and `grid-template-rows`

```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 100px 100px;
}
```

or use **repeat()** function:

```
grid-template-columns: repeat(3, 1fr);
```

- `fr` → fractional unit (shares available space equally)

Gap Between Grid Cells

```css
.container {
  gap: 10px;            /* row and column gap */
  row-gap: 20px;         /* vertical space */
  column-gap: 30px;       /* horizontal space */
}
```

Placing Grid Items

Using `grid-column` and `grid-row`

```css
.item1 {
  grid-column: 1 / 3;  /* spans from column 1 to 2 */
  grid-row: 1 / 2;
}
```

or use `span`:

```css
.item1 {
  grid-column: span 2; /* span across 2 columns */
}
```

Grid Template Areas

You can define named areas for layout readability.

```css
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto 1fr auto;
}

.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
```

```
.main { grid-area: main; }
.footer { grid-area: footer; }
```

Justifying and Aligning Items

Along Main Axis

```
.container {
  justify-items: start | end | center | stretch;
}
```

Along Cross Axis

```
.container {
  align-items: start | end | center | stretch;
}
```

For entire grid alignment:

```
.container {
  justify-content: center; /* whole grid horizontally */
  align-content: center;   /* whole grid vertically */
}
```

Grid Lines

Grid lines are numbered. You can position items using these lines.

Example:

```
.item1 {
  grid-column: 1 / 4;
  grid-row: 2 / 3;
}
```

What are Media Queries?

Media Queries are a feature in CSS that allow your web page to adapt its layout and style based on the characteristics of the device or viewport that displays it.

In simple terms — **media queries help make your website responsive.**

They allow you to apply CSS styles **conditionally**, depending on things like:

- Screen width and height
- Device orientation
- Resolution
- Color capability
- Print or screen mode, etc.

Basic Syntax

```
@media media-type and (condition) {
  /* CSS rules */
}
```

Example:

```
@media screen and (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}
```

Explanation:

- The `@media` rule defines a media query.
- `screen` specifies the type of media.
- `(max-width: 768px)` is the condition (here, when screen width ≤ 768px).
- CSS inside the block applies **only** when the condition is true.

Common Media Types

Media Type	Description
`all`	Default. Applies to all devices.
`print`	Used when printing a page.
`screen`	Used for computer screens, tablets, smartphones, etc.
`speech`	Used for screen readers that read the page aloud.

Example:

```css
@media print {
  body {
    color: black;
    background: white;
  }
}
```

Media Conditions

Media features describe specific characteristics of the device or browser window.

They are written inside parentheses `()`.

Width and Height

These are the most common features used for responsive design.

Property	Description
`width`	Width of the viewport
`height`	Height of the viewport
`min-width`	Minimum width (used for mobile-first)
`max-width`	Maximum width (used for desktop-first)

Example (Mobile First Approach):

```css
/* Base styles for mobile */
body {
  font-size: 14px;
}

/* For tablets and larger screens */
@media (min-width: 768px) {
  body {
    font-size: 16px;
  }
}

/* For desktop */
@media (min-width: 1024px) {
  body {
    font-size: 18px;
  }
}
```

Orientation

Determines if the device is in portrait or landscape mode.

```css
@media (orientation: portrait) {
  body {
    background: lightcoral;
  }
}

@media (orientation: landscape) {
  body {
    background: lightgreen;
  }
}
```

- `portrait`: Height ≥ width
- `landscape`: Width > height

Aspect Ratio

Specifies the ratio between the width and height of the viewport.

```css
@media (min-aspect-ratio: 16/9) {
  .video {
    width: 100%;
  }
}
```

Resolution

Defines the pixel density of the screen (useful for Retina/HD screens).

```css
@media (min-resolution: 2dppx) {
  img {
    content: url("highres-image.png");
  }
}
```

- `dpi` → dots per inch
- `dppx` → dots per pixel unit

Combining Media Queries

You can combine multiple conditions using logical operators.

1. AND Operator

Applies when **both** conditions are true.

```css
@media screen and (min-width: 768px) and (orientation: landscape) {
  body {
    background-color: lightyellow;
  }
}
```

2. OR Operator (Comma separated)

If **any** condition is true, the CSS applies.

```
@media (max-width: 600px), (orientation: landscape) {
  body {
    background-color: lightpink;
  }
}
```

3. NOT Operator

Used to exclude a certain condition.

```
@media not all and (monochrome) {
  body {
    color: black;
  }
}
```

Common Breakpoints

Device	Breakpoint Range	Example
Small (Mobile)	0px – 480px	`@media (max-width: 480px)`
Medium (Tablet)	481px – 768px	`@media (max-width: 768px)`
Large (Laptop)	769px – 1024px	`@media (max-width: 1024px)`
Extra Large (Desktop)	1025px and up	`@media (min-width: 1025px)`

```
.container {
  display: flex;
  flex-direction: column;
}

/* Tablet */
@media (min-width: 768px) {
  .container {
    flex-direction: row;
  }
}
```

```css
/* Desktop */
@media (min-width: 1024px) {
  .container {
    justify-content: space-between;
  }
}
```

- On mobile → stacked layout
- On tablet → row layout
- On desktop → spaced layout

Media Queries for Print

Used to style documents when users print a webpage.

```css
@media print {
  body {
    font-size: 12pt;
    color: black;
    background: none;
  }
  nav, footer {
    display: none;
  }
}
```

Responsive Images using Media Queries

```css
@media (max-width: 600px) {
  img {
    width: 100%;
  }
}
```

▼ BOOTSTRAP 5

History of Bootstrap

Bootstrap was originally created by **Mark Otto** and **Jacob Thornton**, who were engineers at **Twitter**.

1. The Beginning (2010–2011)

- At Twitter, different developers were using different libraries and design patterns.
- This caused **inconsistency** in how internal tools looked and behaved.
- To solve this, Mark Otto started working on a **unified front-end framework** called **"Twitter Blueprint"**.
- The idea was to create a **standard toolkit** for building user interfaces quickly and consistently.

2. Public Release (2011)

- In **August 2011**, Twitter decided to open-source this framework.
- It was renamed to **Bootstrap** and released on **GitHub**.
- The first version was **Bootstrap 1.0**, which included layouts, forms, buttons, and some basic JavaScript components.

3. Evolution of Bootstrap Versions

Here's how Bootstrap evolved over time:

Version	Year	Key Features
Bootstrap 1	2011	First public release; focused on consistency in UI design.
Bootstrap 2	2012	Introduced a 12-column responsive grid system.
Bootstrap 3	2013	Fully mobile-first design and flat UI style.

Version	Year	Key Features
Bootstrap 4	2018	Rewritten using **Sass**, added **Flexbox**, new cards, utilities, and better customization.
Bootstrap 5	2021	Removed dependency on jQuery, improved grid system, added utilities API, new form controls, and better customization features.

Purpose of Bootstrap

Bootstrap was designed to **make front-end web development faster and easier**, while maintaining **design consistency**.

Let’s break that into key points:

1. Faster Development

Bootstrap provides pre-written **CSS and JavaScript** code for common UI elements (like buttons, navbars, modals, forms, etc.).

You don’t need to write everything from scratch — you can just use Bootstrap classes.

Example:

```html
<button class="btn btn-primary">Click Me</button>
```

This single line creates a modern, styled blue button without writing any CSS.

2. Consistent Design

When multiple developers work on the same project, designs can easily become inconsistent.

Bootstrap ensures a **standardized look** for all components.

Example:

Buttons, forms, alerts, and modals all follow the same visual style by default.

3. Responsive and Mobile-First

Bootstrap’s grid system and utilities are built to automatically adjust layouts based on screen size.

This means your website looks good on **mobile, tablet, and desktop** without extra effort.

Example:

Using Bootstrap’s grid:

```
<div class="row">
  <div class="col-md-6 col-sm-12">Left</div>
  <div class="col-md-6 col-sm-12">Right</div>
</div>
```

This layout automatically stacks on mobile and appears side by side on desktop.

4. Cross-Browser Compatibility

Bootstrap ensures that your website works across **all major browsers** like Chrome, Firefox, Safari, and Edge without needing extra fixes.

5. Easy Customization

Bootstrap can be customized to match your brand by overriding default styles or using **Sass variables** to change colors, fonts, and component shapes.

6. Open Source and Community Support

Bootstrap is **free and open-source.**

It has a large **developer community**, so you can easily find documentation, themes, plugins, and support.

Let’s go step-by-step through **how to include Bootstrap 5** in your project using both **CDN** and **Download methods**, with clear explanations and examples.

Including Bootstrap via CDN

What is CDN?

CDN stands for **Content Delivery Network**.

It means the Bootstrap files (CSS and JS) are stored on fast global servers, and your webpage can load them directly from the internet — you don't need to download anything.

Why Use CDN?

- No need to store files locally.
- Easy to set up.
- Loads faster because browsers often cache CDN resources.
- Always up to date with the latest version.

Steps to Include Bootstrap via CDN

You only need to include two links:

1. **Bootstrap CSS** in the `<head>` section.
2. **Bootstrap JavaScript** before the closing `</body>` tag.

Here's a simple example:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=
1">
  <title>Bootstrap via CDN</title>

  <!-- Bootstrap 5 CSS CDN →
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/boot
strap.min.css" rel="stylesheet">
</head>
<body>
```

```
  <div class="container mt-5">
    <h1 class="text-center text-primary">Hello, Bootstrap 5!</h1>
    <p class="text-center">This page uses Bootstrap via CDN.</p>
    <button class="btn btn-success">Click Me</button>
  </div>

  <!-- Bootstrap 5 JavaScript Bundle (includes Popper.js) →
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/boots
trap.bundle.min.js"></script>
</body>
</html>
```

Explanation:

- `<link>` — Loads the Bootstrap CSS (for styling).
- `<script>` — Loads Bootstrap's JavaScript and **Popper.js** (used for dropdowns, modals, tooltips, etc.).
- `container` — A Bootstrap class that gives padding and centers your content.
- `btn`, `btn-success`, `text-primary` — Predefined Bootstrap classes for buttons and colors.

2. Including Bootstrap via Download (Offline Method)

If you want to **use Bootstrap offline** or need to customize its source files, you can download it.

Step 1: Download Bootstrap

Go to the official Bootstrap website:

https://getbootstrap.com

Click on **"Download"**, then choose:

- **Compiled CSS and JS** (recommended for most users).

  This gives you ready-to-use `bootstrap.min.css` and `bootstrap.bundle.min.js` files.

Step 2: Add Files to Your Project

After downloading, you'll get a folder like this:

```
bootstrap-5.3.3-dist/
│
├── css/
│   ├── bootstrap.css
│   └── bootstrap.min.css
│
└── js/
    ├── bootstrap.bundle.js
    └── bootstrap.bundle.min.js
```

Copy the `css` and `js` folders into your project folder.

Step 3: Link Bootstrap in Your HTML File

Now, include the local Bootstrap files just like you did with CDN — but using your local path.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=
1">
  <title>Bootstrap Offline Example</title>

  <!-- Local Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>

  <div class="container mt-5">
    <h2 class="text-center text-danger">Bootstrap 5 (Offline)</h2>
    <p class="text-center">This page works without internet connection.
</p>
```

```
    <button class="btn btn-primary">Click Me</button>
  </div>

  <!-- Local Bootstrap JS -->
  <script src="js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Containers

What is a Container?

A **container** in Bootstrap is a layout wrapper that holds your page content and centers it properly.

It provides **proper spacing (padding)** on the left and right sides of your content.

Without a container, your elements will touch the screen edges, which doesn't look good.

Types of Containers

Class	Description
`.container`	Fixed-width container — changes size depending on the screen width.
`.container-fluid`	Always takes full width of the screen (100%).
`.container-{breakpoint}`	Fixed width until a certain breakpoint (like `.container-sm`, `.container-md`, etc.).

Example 1: Basic Container

```
<div class="container bg-light border mt-3">
  <h2>Fixed Container</h2>
  <p>This container has a fixed width that changes with screen size.</p>
</div>
```

This container will automatically adjust its width for different screen sizes but will not fill the entire width.

Example 2: Fluid Container

```html
<div class="container-fluid bg-info text-white mt-3">
  <h2>Fluid Container</h2>
  <p>This container always takes the full width of the screen.</p>
</div>
```

Use `.container-fluid` when you want your section (like a header or footer) to span across the entire screen width.

Example 3: Responsive Container Sizes

```html
<div class="container-sm bg-warning mt-3">Small container</div>
<div class="container-md bg-success text-white mt-3">Medium contai
ner</div>
<div class="container-lg bg-danger text-white mt-3">Large container</
div>
```

These containers grow or shrink based on the **breakpoint** (screen size).

The Grid System

The **grid system** is the heart of Bootstrap layout.

It uses **rows and columns** to organize your content in a responsive 12-column structure.

Basic Structure

```html
<div class="container">
  <div class="row">
    <div class="col">Column 1</div>
    <div class="col">Column 2</div>
    <div class="col">Column 3</div>
  </div>
</div>
```

Explanation:

- `.row` → Creates a horizontal group of columns.
- `.col` → Each column automatically divides the available space equally.
- The grid has **12 columns** in total per row.

So if you add:

- 3 columns → each gets 4 spaces (12 ÷ 3)
- 4 columns → each gets 3 spaces (12 ÷ 4)

Example: Fixed Column Widths

You can define exactly how many columns each should take using numbers from 1 to 12.

```
<div class="container">
  <div class="row">
    <div class="col-4 bg-primary text-white">4 columns wide</div>
    <div class="col-8 bg-secondary text-white">8 columns wide</div>
  </div>
</div>
```

Here, one column takes 4 out of 12 parts, and the other takes 8 parts.

Responsive Grid

Bootstrap allows columns to adjust automatically depending on screen size using **breakpoints**.

Breakpoint	Class Prefix	Screen Width
Extra small	`.col-`	< 576px
Small	`.col-sm-`	≥ 576px
Medium	`.col-md-`	≥ 768px
Large	`.col-lg-`	≥ 992px
Extra large	`.col-xl-`	≥ 1200px
Extra extra large	`.col-xxl-`	≥ 1400px

Example: Responsive Columns

```
<div class="container">
  <div class="row">
    <div class="col-sm-12 col-md-6 col-lg-4 bg-danger text-white">Column 1</div>
    <div class="col-sm-12 col-md-6 col-lg-8 bg-success text-white">Column 2</div>
  </div>
</div>
```

Explanation:

- On small screens → both take full width (stacked vertically).
- On medium screens → each takes half width.
- On large screens → 4 and 8 columns respectively.

Responsive Layout

Bootstrap's grid is **mobile-first** — meaning layouts are designed for smaller screens first and then scale up for larger ones.

You can combine container, row, and column classes to make your layout automatically adjust to any device.

Example: Fully Responsive Layout

```
<div class="container mt-4">
  <div class="row">
    <div class="col-12 col-md-4 bg-primary text-white p-3">Left Sidebar</div>
    <div class="col-12 col-md-8 bg-light p-3">Main Content</div>
  </div>
</div>
```

Behavior:

- On mobile → the sidebar and content are stacked.
- On desktop → they appear side by side.

Column Alignment and Ordering

Bootstrap gives you classes to **align and reorder** columns inside rows.

Vertical Alignment

Use **`align-items-*`** on the `.row` or **`align-self-*`** on a specific `.col`.

Class	Effect
`align-items-start`	Aligns columns to the top
`align-items-center`	Vertically centers columns
`align-items-end`	Aligns columns to the bottom

Example:

```html
<div class="container" style="height: 200px;">
  <div class="row align-items-center bg-light" style="height: 100%;">
    <div class="col bg-warning">Top</div>
    <div class="col bg-success text-white">Center</div>
    <div class="col bg-info text-white">Bottom</div>
  </div>
</div>
```

Horizontal Alignment

Use **`justify-content-*`** on the `.row`:

Class	Effect
`justify-content-start`	Left align (default)
`justify-content-center`	Center align
`justify-content-end`	Right align
`justify-content-between`	Space between
`justify-content-around`	Space around
`justify-content-evenly`	Even spacing

Example:

```html
<div class="container mt-3">
  <div class="row justify-content-center">
```

```
    <div class="col-3 bg-primary text-white">Box 1</div>
    <div class="col-3 bg-secondary text-white">Box 2</div>
  </div>
</div>
```

Column Ordering

You can change the **order of columns** using `.order-*` classes.

Class	Description
`.order-1` to `.order-12`	Changes column order numerically
`.order-first`	Moves column to the beginning
`.order-last`	Moves column to the end

Example:

```
<div class="container mt-3">
  <div class="row">
    <div class="col bg-danger text-white order-3">First visually (order 3)
</div>
    <div class="col bg-success text-white order-1">Second visually (ord
er 1)</div>
    <div class="col bg-primary text-white order-2">Third visually (order
2)</div>
  </div>
</div>
```

Bootstrap Typography

Typography in Bootstrap controls how text looks — headings, paragraphs, inline elements, colors, alignment, and spacing.

Bootstrap provides **ready-made classes** to style text quickly without writing CSS.

A. Headings

Bootstrap supports all HTML heading tags from `<h1>` to `<h6>`.

```html
<h1>h1. Bootstrap heading</h1>
<h2>h2. Bootstrap heading</h2>
<h3>h3. Bootstrap heading</h3>
<h4>h4. Bootstrap heading</h4>
<h5>h5. Bootstrap heading</h5>
<h6>h6. Bootstrap heading</h6>
```

You can also apply heading styles to other elements (like `<p>` or `<div>`) using Bootstrap classes:

```html
<p class="h1">This looks like an h1 heading</p>
<p class="h3">This looks like an h3 heading</p>
```

B. Display Headings

Display headings are **larger and bolder** than normal headings.

They are often used for main titles or hero sections.

```html
<h1 class="display-1">Display 1</h1>
<h1 class="display-2">Display 2</h1>
<h1 class="display-3">Display 3</h1>
<h1 class="display-4">Display 4</h1>
<h1 class="display-5">Display 5</h1>
<h1 class="display-6">Display 6</h1>
```

Each class makes the text size slightly smaller as the number increases.

C. Paragraphs

Bootstrap makes paragraph text look nice with classes like:

- `.lead` – makes a paragraph stand out
- `.small` – makes text smaller
- `.fw-bold` – bold text
- `.fst-italic` – italic text

Example:

```html
<p>This is a normal paragraph.</p>
<p class="lead">This is a lead paragraph that stands out more.</p>
<p class="small">This is small text.</p>
<p class="fw-bold">This text is bold.</p>
<p class="fst-italic">This text is italic.</p>
```

D. Inline Text Elements

Bootstrap also styles basic inline elements like bold, italics, etc.

Element	Description	Example
`<mark>`	Highlights text	`<mark>highlighted</mark>`
`<del>`	Deleted text	`<del>deleted</del>`
`<s>`	Strikethrough	`<s>no longer relevant</s>`
`<ins>`	Inserted text	`<ins>new text</ins>`
`<u>`	Underline	`<u>underlined</u>`
`<small>`	Small text	`<small>fine print</small>`
`<strong>`	Bold text	`<strong>important</strong>`
`<em>`	Italic text	`<em>emphasis</em>`

Example:

```html
<p>This is <mark>highlighted</mark>, <del>deleted</del>, and <stron
g>bold</strong> text.</p>
```

E. Text Alignment

You can align text using these classes:

Class	Description
`.text-start`	Left aligned (default)
`.text-center`	Center aligned
`.text-end`	Right aligned

Example:

```html
<p class="text-start">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-end">Right aligned text.</p>
```

These also work **responsively**, for example:

```html
<p class="text-sm-start text-md-center text-lg-end">
  This text aligns left on small screens, center on medium, and right on la
rge screens.
</p>
```

F. Text Colors

Bootstrap provides text color classes:

Class	Color
`.text-primary`	Blue
`.text-secondary`	Gray
`.text-success`	Green
`.text-danger`	Red
`.text-warning`	Yellow
`.text-info`	Light blue
`.text-light`	White (use on dark backgrounds)
`.text-dark`	Black
`.text-body`	Default body color
`.text-muted`	Muted gray text
`.text-white-50`	Half-transparent white

Example:

```html
<p class="text-primary">Primary text</p>
<p class="text-success">Success text</p>
<p class="text-danger">Danger text</p>
<p class="text-muted">Muted text</p>
```

G. Text Utilities

Text Transform

Class	Effect
`.text-lowercase`	All lowercase
`.text-uppercase`	All uppercase
`.text-capitalize`	Capitalizes first letter of each word

Example:

```
<p class="text-uppercase">uppercase text</p>
<p class="text-lowercase">LOWERCASE TEXT</p>
<p class="text-capitalize">capitalize every word</p>
```

Font Weight and Style

Class	Effect
`.fw-bold`	Bold
`.fw-normal`	Normal
`.fw-light`	Light
`.fst-italic`	Italic
`.fst-normal`	Normal (no italic)

Example:

```
<p class="fw-bold">Bold text</p>
<p class="fw-light fst-italic">Light and italic text</p>
```

H. Line Height & Text Truncation

```
<p class="lh-1">Tight line height</p>
<p class="lh-sm">Small line height</p>
<p class="lh-base">Normal line height</p>
<p class="lh-lg">Large line height</p>

<p class="text-truncate" style="width:200px;">
```

```
  This is a long line of text that will be truncated with an ellipsis.
</p>
```

Bootstrap Images

Bootstrap provides classes to make images responsive and styled easily.

a) Responsive Images

Use the `.img-fluid` class to make an image scale nicely to fit the parent container.

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

b) Image Shapes

Bootstrap provides classes to style image corners and shapes.

Class	Description
`.rounded`	Adds rounded corners
`.rounded-circle`	Makes image circular
`.img-thumbnail`	Adds border and padding like a photo frame

Example:

```
<img src="photo.jpg" class="rounded" alt="Rounded">
<img src="photo.jpg" class="rounded-circle" alt="Circle">
<img src="photo.jpg" class="img-thumbnail" alt="Thumbnail">
```

c) Figure Captions

You can add captions under images using `<figure>` and `<figcaption>`.

```
<figure class="text-center">
  <img src="photo.jpg" class="img-fluid rounded" alt="Sample">
  <figcaption class="figure-caption">This is an image caption.</figcapti
```

```
on>
</figure>
```

6. Bootstrap Tables

Bootstrap simplifies table styling with easy classes.

a) Basic Table

```
<table class="table">
  <thead>
    <tr><th>Name</th><th>Age</th><th>City</th></tr>
  </thead>
  <tbody>
    <tr><td>John</td><td>25</td><td>New York</td></tr>
    <tr><td>Amy</td><td>22</td><td>London</td></tr>
  </tbody>
</table>
```

7. Bootstrap Buttons

Bootstrap buttons are styled using `.btn` class and color classes.

a) Button Styles and Colors

Class	Button Type
`.btn-primary`	Blue button
`.btn-secondary`	Gray button
`.btn-success`	Green button
`.btn-danger`	Red button
`.btn-warning`	Yellow button
`.btn-info`	Teal button
`.btn-light`	White button
`.btn-dark`	Black button

Example:

```
<button class="btn btn-primary">Primary</button>
<button class="btn btn-success">Success</button>
```

b) Outline Buttons

Outline buttons have borders and no background until hovered.

```
<button class="btn btn-outline-primary">Outline Primary</button>
<button class="btn btn-outline-danger">Outline Danger</button>
```

c) Button Sizes and Groups

Class	Description
`.btn-lg`	Large button
`.btn-sm`	Small button
`.btn-block` *(v4)*	Full width (in v5 use `w-100`)

Example:

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

Button Group:

Group multiple buttons together with `.btn-group`.

```
<div class="btn-group">
  <button class="btn btn-primary">Left</button>
  <button class="btn btn-primary">Middle</button>
  <button class="btn btn-primary">Right</button>
</div>
```

BootStrap Conponents

1. Alerts

Used to show messages like success, error, or info.

```
<div class="alert alert-success" role="alert">
  Operation successful!
</div>

<div class="alert alert-danger" role="alert">
  Something went wrong.
</div>
```

Use classes like `alert-success`, `alert-danger`, `alert-warning`, `alert-info`.

2. Badges

Used to show small counts or labels beside text.

```
<h4>Messages <span class="badge bg-primary">4</span></h4>
<button class="btn btn-dark">
  Notifications <span class="badge bg-danger">9</span>
</button>
```

Badges use `.badge` and background color classes like `bg-primary`.

3. Cards

Used to display content inside a bordered box.

```
<div class="card" style="width: 18rem;">
  <img src="image.jpg" class="card-img-top" alt="Image">
  <div class="card-body">
    <h5 class="card-title">Card Title</h5>
    <p class="card-text">Some quick text.</p>
    <a href="#" class="btn btn-primary">Read More</a>
  </div>
</div>
```

4. Navbar

Used to create navigation bars.

```html
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">MySite</a>
    <button class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navmenu">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navmenu">
      <ul class="navbar-nav ms-auto">
        <li class="nav-item"><a class="nav-link" href="#">Home</a></li>
        <li class="nav-item"><a class="nav-link" href="#">About</a></li>
      </ul>
    </div>
  </div>
</nav>
```

You can make it light or dark using `navbar-light` *or* `navbar-dark`*.*

5. Pagination

Used for page navigation links.

```html
<nav>
  <ul class="pagination justify-content-center">
    <li class="page-item disabled"><a class="page-link">Previous</a></li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item active"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item"><a class="page-link" href="#">Next</a></li>
  </ul>
</nav>
```

6. Progress Bars

Used to show progress visually.

```html
<div class="progress">
  <div class="progress-bar bg-success" style="width: 60%;">60%</div>
</div>
```

7. Spinners

Used to show loading indicators.

```html
<div class="spinner-border text-primary" role="status"></div>
<div class="spinner-grow text-success" role="status"></div>
```