Chapter 2

HTML BASICS

Introduction to HTML Basics What is HTML?

- **HTML** stands for **HyperText Markup Language**.
- It is the standard language used to **create and design web pages**.
- HTML describes the **structure** of a webpage using a system of **tags**.

Features of HTML

- It uses **tags** to mark different elements (like headings, paragraphs, images).
- HTML is **not case-sensitive** (but conventionally, tags are written in lowercase).
- Browsers like Chrome, Firefox, and Safari **read HTML** to display webpages.
- HTML documents are **saved with a `.html` or `.htm` extension**.

Basic Structure of an HTML Document

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>

    <h1>Welcome to My Webpage</h1>
    <p>This is a paragraph of text.</p>

</body>
</html>
```

Explanation:

- `<!DOCTYPE html>`: Tells the browser that this is an HTML5 document.
- `<html>`: Root tag that wraps all the content.
- `<head>`: Contains meta-information like the title.
- `<title>`: Sets the title of the webpage (shown in browser tab).

- `<body>`: Contains all the content visible on the page (text, images, links).

Common HTML Tags

Tag	Purpose
`<h1>` to `<h6>`	Headings (h1 = largest, h6 = smallest)
`<p>`	Paragraph
` `	Line break
`<hr>`	Horizontal line
`<a>`	Hyperlink (link to another page)
`<img>`	Image
`<ul>`, `<ol>`, `<li>`	Lists (unordered, ordered, list item)
`<b>`, `<i>`, `<u>`	Bold, Italic, Underline text
`<div>`, `<span>`	Grouping elements for styling

Example: Creating a Simple Webpage

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
</head>
<body>

    <h1>Hello, World!</h1>
    <p>This is my first webpage using HTML.</p>
    <a href="https://www.google.com">Visit
Google</a>
</body>
</html>
```

Basic Structure of HTML

1. What is HTML Structure?

The **HTML structure** is the **foundation** of every webpage.
It organizes the content and tells the browser **what to display** and **how**.

2. Basic Skeleton of an HTML Page

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>

    <!-- Page content goes here -->

</body>
</html>
```

3. Parts of an HTML Document

Part	Description
`<!DOCTYPE html>`	Declares the document type and version (HTML5).
`<html>`	The root element that contains all other elements.
`<head>`	Contains information **about the page** (not visible to users).
`<title>`	Sets the title of the page (appears on the browser tab).
`<body>`	Contains all the **content** that is shown on the webpage.

4. Explanation of Each Part

a) `<!DOCTYPE html>`

- Must be at the very **top** of the file.
- Tells the browser that the page uses **HTML5**.

b) `<html> ... </html>`

- Wraps all the HTML content.
- It is the **root** element.

c) `<head> ... </head>`

- Contains meta information like:
  - Title ○ Character set ○ Links to CSS files ○ Other head data (scripts, etc.)

Example:

```
html
CopyEdit
<head>
    <title>My Webpage</title>
</head>
```

d) `<body> ... </body>`

- Everything inside `<body>` is **visible** on the webpage:
  - Headings ○ Paragraphs ○ Images ○ Links ○ Lists, tables, etc.

Example:

```
html
CopyEdit
<body>
    <h1>Welcome!</h1>
    <p>This is my webpage.</p>
</body>
```

5. Diagram Overview

```
css
CopyEdit
Document
 ├── Doctype Declaration
 └── HTML
     ├── Head
     │   └── Title
     └── Body
         ├── Heading
```

```
          ├──
Paragraph
└── Other content
```

Formatting tags

1.Using the HTML Tag

1. What is a `<div>` Tag?

- The `<div>` tag is short for **division**.
- It is a **container** that groups together sections of HTML content.
- It **does not affect** how the content looks by itself — it is used mainly for **layout and styling**.

2. Why Use `<div>`?

- To **organize** the webpage into sections.
- To apply **CSS styles** to a group of elements.
- To **control layout** (for example: header, sidebar, footer).
- Makes the page **easier to manage** and **design**.

3. Basic Example of a `<div>`

```
html
CopyEdit
<div>
    <h1>Welcome to My Website</h1>
    <p>This is the introduction section.</p>
</div>
```

- Here, both the heading and paragraph are inside one `<div>` container.

4. Using `<div>` with CSS

You can use a **class** or **id** to style a `<div>` with CSS.

Example:

```
html
CopyEdit
<div style="background-color: lightblue; padding: 20px;">
    <h2>About Me</h2>
    <p>I am a student learning HTML.</p>
</div>
```

- This `div` has a light blue background and some padding around the content.

5. Example: Multiple `<div>` Sections

```
html
CopyEdit
<div style="background-color: lightgray;">
    <h1>Header Section</h1>
</div>

<div style="background-color: white;">
    <p>Main content goes here.</p>
</div>

<div style="background-color: lightgray;">
    <p>Footer Section</p>
</div>
```

- This separates the page into three different parts: **Header**, **Content**, and **Footer**.

6. Important Points

Point	Description
`<div>` groups content	Acts like a box around elements.
By default, `<div>` is **block-level**	It takes the full width of the page.
Used with **CSS** for layout	Like making columns or colored areas. Can have `id` or `class`
	To style it uniquely with CSS.

2. Absolute Links

1. What is an Absolute Link?

- An **absolute link** (or **absolute URL**) is a **full web address** that points to a specific page or file on the internet.
- It **includes**:

- Protocol (like `http://` or `https://`)
- Domain name (like `www.example.com`)
- Optional path to a specific page or file.

It can be used from **anywhere** because it gives the **complete address**.

2. Example of an Absolute Link

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

- **`<a>`** = anchor tag (used to create a link)
- **`href`** = hyperlink reference (where the link goes)
- Clicking this link will open the **Google homepage**.

3. Structure of an Absolute URL

```
pgsql
CopyEdit
Protocol       Domain Name          Path
  ↓               ↓                   ↓
https://www.example.com/folder/page.html
```

- **Protocol**: The method used to access the resource (http or https).
- **Domain Name**: The website's address.
- **Path**: Specific location of the page or file on the website (optional).

4. Important Points

Feature	Explanation
Starts with `http://` or `https://`	Always begins with the protocol.
Works from any location	Because it gives the complete path. Used to
link to other websites	Useful for linking external pages.

5. Another Example

```
html
CopyEdit
<a href="https://www.bbc.com/news">BBC News</a>
```

- Clicking this will open the **BBC News** website.

6. Difference Between Absolute and Relative Links

Absolute Link	Relative Link
Full web address (with `https://`)	Short path relative to current page
Works from anywhere	Works only from specific locations Example:
`https://www.yahoo.com` Example: `about.html`	

3. Image Element

1. What is the `<img>` Tag?

- The `<img>` tag is used to **display images** on a webpage.
- It is an **empty tag**, meaning it does **not** have a closing tag.

The `<img>` tag uses **attributes** to provide information about the image.

2. Basic Syntax

```
html
CopyEdit
<img src="image_url" alt="description">
```

- **`src`** = source of the image (path or URL).
- **`alt`** = alternative text shown if the image cannot load.

3. Example of an Image

```
html
CopyEdit
<img src="https://example.com/image.jpg"
   alt="Example Image">
```

• **`src`** points to the image location.

- **`alt`** provides a description ("Example Image") for accessibility and in case the image does not display.

4. Important Attributes of `<img>`

Attribute	Purpose
`src`	Specifies the **path** to the image file.
`alt`	Provides **alternative text** if the image fails to load.

`width` Sets the **width** of the image (in pixels or %). `height`
Sets the **height** of the image (in pixels or %).

5. Example with Width and Height

```
html
CopyEdit
<img src="photo.jpg" alt="My Photo" width="300" height="200">
```

- This image will display at **300 pixels wide** and **200 pixels high**.

6. Types of Image Sources

- **Absolute URL** (full website link):

```
html
CopyEdit
<img src="https://www.example.com/images/pic.jpg"
alt="Example Picture">
```

- **Relative URL** (file located in the same folder):

```
html
CopyEdit
<img src="pic.jpg" alt="Local Picture">
```

7. Tips for Using Images

- Always use the `alt` attribute for **better accessibility**.
- Choose appropriate image sizes to make pages load faster.
- Use correct file formats:
  - **JPEG** for photos.
  - **PNG** for images needing transparency. o **GIF** for animations.

4. Paragraph `<p>` Tag

1. What is the `<p>` Tag?

- The `<p>` tag is used to create a **paragraph** of text on a webpage.
- Paragraphs help **organize** content and **make text easier to read**.

A paragraph automatically adds **some space** before and after the text.

2. Basic Syntax

```
html
CopyEdit
<p>This is a paragraph of text.</p>
```

- ** ** = opening tag
- ** ** = closing tag
- Text goes **inside** the tags.

3. Example of a Paragraph

```
html
CopyEdit
<p>HTML is the standard language for creating web pages.</p>
```

- This will display the sentence as a paragraph with space around it.

4. Important Points about

Feature	Explanation
Block-level element	Takes up the full width of the page.
Adds spacing	Automatically adds margin (space) around. Must be closed
	Always close the paragraph with .

5. Using Multiple Paragraphs

```
html
CopyEdit
<p>HTML stands for HyperText Markup Language.</p>

<p>It is used to design the structure of web pages.</p>
```

- Each paragraph will appear **on a new line** with space between them.

6. Adding Line Breaks Inside Paragraphs

If you want to **start a new line** inside a paragraph without ending it, use the
 tag:

```
html
CopyEdit
<p>
This is the first line.<br>
This is the second line within the same paragraph.
</p>
```

- `<br>` = line break (no closing tag needed).

7. Styling Paragraphs (Basic Example)

You can style a paragraph using **CSS** directly:

```
html
CopyEdit
<p
style="col
or: blue;
font-size:
18px;">
This
paragraph
has blue
text and
larger
font.
</p>
```

5. Comments in HTML Documents

1. What is a Comment in HTML?

- A **comment** is a **note** or **message** written inside the HTML code.
- **Browsers ignore comments** — they are **not displayed** on the webpage.
- Comments help **explain** the code or **leave reminders** for yourself or others.

Comments make the HTML easier to **read** and **understand**.

2. Syntax of an HTML Comment

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

- Comments **start** with `<!--`
- Comments **end** with `-->`
- Anything between `<!--` and `-->` is **ignored** by the browser.

3. Example of an HTML Comment

```
html
```

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

<!-- This paragraph is for future use
<p>This is a hidden paragraph.</p>
-->
```

- Only the first paragraph will be displayed.
- The second paragraph is **inside a comment** and will not appear on the page.

4. Why Use Comments?

Purpose	Example
Explain parts of the code	`<!-- This section displays the main menu -->`
Temporarily disable code	Hide code without deleting it.

Purpose	Example
Organize large files	Add section titles to make code cleaner.

5. Important Points

Point	Detail
Not shown to users	Only visible in the HTML source code.
Good for teamwork	Helps others understand your work.
Must not be nested	You cannot place one comment inside another.

6. Example: Organizing a Webpage with Comments

```
html
CopyEdit
<!-- Header Section -->
<h1>Welcome to My
Website</h1>

<!-- Main Content Section -->
<p>This is where the main content goes.</p>

<!-- Footer Section -->
<p>Contact us at info@example.com</p>
```

HTML TABLES

1 . BASIC TABLE TAGS

1. What is an HTML Table?

- An HTML **table** is used to **display data** in **rows** and **columns**.
- Tables organize information neatly.

Tables are made up of **rows** (`<tr>`) and **cells** (`<td>` or `<th>`).

2. Main Table Tags

Tag	Meaning
`<table>`	Starts and ends the table.
`<tr>`	Table Row — groups cells in a row.
`<td>`	Table Data — a normal cell.
`<th>`	Table Header — a cell with bold heading text.

3. Basic Structure of a Table

```
html
CopyEdit
<table>
    <tr>
        <th>Heading 1</th>
        <th>Heading 2</th>
    </tr>
    <tr>
        <td>Data 1</td>
        <td>Data 2</td>
```

```
    </tr>
    <tr>
        <td>Data 3</td>
        <td>Data 4</td>
    </tr>
</table>
```

Explanation:

- `<table>`: Starts the table.
- `<tr>`: A row.
- `<th>`: A header cell (bold and centered by default).
- `<td>`: A normal data cell.

4. Example Table Output

Heading 1	Heading 2
Data 1	Data 2
Data 3	Data 4

5. Adding Borders to a Table

```
html
CopyEdit
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Ali</td>
        <td>18</td>
    </tr>
</table>
```

- The `border="1"` attribute adds a simple border to the table and cells.

6. Important Points

Point	Details
`<table>`	Must wrap all table content.
`<tr>` (Table Row)	Groups cells horizontally.
`<td>` (Table Data)	Creates a standard cell.
`<th>` (Table Header)	Creates a header cell (bold text).
Border	Can be added using `border="1"` inside `<table>`.

2. Creating a Table in HTML

1. What is a Table?

- A **table** is used to **arrange data** into **rows** and **columns**.
- Helps to **organize information** clearly on a webpage.

2. Basic Tags for Tables

Tag	Purpose
`<table>`	Starts and ends the table.
`<tr>`	Table row — groups together table cells.
`<th>`	Table header — creates a **bold** heading cell.
`<td>`	Table data — creates a regular cell for data.

3. Basic Structure of a Table

```
html
CopyEdit
<table>
    <tr>
        <th>Header 1</th>
        <th>Header 2</th>
    </tr>
    <tr>
        <td>Data 1</td>
        <td>Data 2</td>
    </tr>
    <tr>
        <td>Data 3</td>
```

```
        <td>Data 4</td>
    </tr>
</table>
```

4. Explanation

- `<table>`: Starts the table.
- `<tr>`: Each table row.
- `<th>`: Header cells (usually bold and centered).
- `<td>`: Normal data cells.

5. Example Table Output

Header 1	**Header 2**
Data 1	Data 2
Data 3	Data 4

6. Adding a Border to the Table

```
html
CopyEdit
<table
border="1"
>
    <tr>
        <th>Subject</th>
        <th>Marks</th>
    </tr>
    <tr>
        <td>Maths</td>
        <td>95</td>
    </tr>
</table>
```

- The `border="1"` attribute adds a **simple black border** around the table and cells.

7. Some Extra Table Attributes

Attribute	**Purpose**
`border`	Adds a border to the table. `cellpadding` Adds space **inside** the cell. `cellspacing` Adds space **between** cells.

Example:

```
html
CopyEdit
<table border="1" cellpadding="5" cellspacing="3">
    ...
</table>
```

3. Table Attributes

Definition:
In a database, a **table** stores data in rows and columns.
Each **column** in a table is called an **attribute**.
An **attribute** represents a single type of data or a field.

Key Points:

- **Table**: A collection of related data organized in rows and columns.
- **Row (Record)**: A single complete set of fields (data entries).
- **Column (Attribute/Field)**: A characteristic or property of the data stored (e.g., Name, Age, Email).

Examples:

StudentID	**Name**	**Age**	**Grade**
001	Sarah Smith	16	A

StudentID	**Name**	**Age**	**Grade**
002	John Doe	17	B

- **Here, `StudentID`, `Name`, `Age`, and `Grade` are attributes**.

More about Attributes:

- **Attribute Name**: Should be unique and meaningful (e.g., "StudentID" instead of just "ID").
- **Data Type**: Each attribute has a specific type, like:
    - **Text/String** (for names, addresses) ○ **Number** (for ages, quantities)
    - **Date/Time** (for birthdates, registration dates) ○ **Boolean** (for true/false values)

Primary Key:

- An attribute (or combination of attributes) that uniquely identifies each record in a table.
- Example: `StudentID` is often a **primary key**.

Foreign Key:

- An attribute in one table that links to the **primary key** in another table to establish a relationship between them.

HTML LIST

In HTML, lists are used to group related items together in a specific order or without any order. There are three main types of lists:

1. Ordered List (ol)

An **ordered list** is a list where the items are numbered or ordered by default.

Syntax:

```
html
CopyEdit
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

- `<ol>`: Used to define an ordered list.
- `<li>`: Defines a list item.
- By default, the browser will number the list items, but you can customize the numbering style using the `type` attribute.

Attributes:

- `type`: Specifies the numbering style (e.g., `1` (default), `A`, `a`, `I`, `i`).

```
html
CopyEdit
<ol type="A">
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```

2. Unordered List (ul)

An **unordered list** is a list where the items are marked with bullet points, not numbers.

Syntax:

```
html
CopyEdit
<ul>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ul>
```

- **`<ul>`**: Defines an unordered list.
- **`<li>`**: Defines a list item.

3. Description List (dl)

A **description list** is used for defining terms and their descriptions.

Syntax:

```
html
CopyEdit
<dl>
  <dt>HTML</dt>
  <dd>A markup language for creating web pages.</dd>

  <dt>CSS</dt>
  <dd>A style sheet language used to describe the
presentation of a web page.</dd> </dl>
```

- **`<dl>`**: Defines a description list.
- **`<dt>`**: Defines a term (definition term).
- **`<dd>`**: Defines the description for the term.

Common List Attributes

- **`type`**: For `<ol>`, it determines the numbering style (e.g., `1`, `A`, `I`).
- **`start`**: For `<ol>`, it specifies the starting number for the list.

Example:

```
html
CopyEdit
<ol start="5">
  <li>Item 5</li>
  <li>Item 6</li>
</ol>
```

- **`compact`** (deprecated): For `<ul>` or `<ol>`, used to reduce the spacing between items.

Nested Lists

You can nest lists inside other lists to create sub-lists.

Example:

```
html
CopyEdit
<ol>
  <li>Item 1
    <ul>
      <li>Sub-item 1</li>
      <li>Sub-item 2</li>
    </ul>
  </li>
  <li>Item 2</li>
</ol>
```

Practical Example of All List Types:

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

  <h2>Ordered List</h2>
  <ol type="A">
    <li>Apple</li>
    <li>Banana</li>
    <li>Cherry</li>
  </ol>

  <h2>Unordered List</h2>
  <ul>
    <li>First item</li>
    <li>Second item</li>
```

```
    <li>Third item</li>
  </ul>

  <h2>Description List</h2>
  <dl>
    <dt>HTML</dt>
    <dd>HyperText Markup Language</dd>
    <dt>CSS</dt>
    <dd>Cascading Style Sheets</dd>
  </dl>

</body>
</html>
```

HTML Forms

HTML forms are used to collect user input on a web page. A form can contain various types of input elements like text fields, checkboxes, radio buttons, submit buttons, and more.

1. Form Tag (`<form>`)

The `<form>` element is used to define a form. It is the container for all form controls (inputs, buttons, etc.). **Syntax:**

```
html
CopyEdit
<form action="submit_form.php" method="post">
  <!-- Form elements go here -->
</form>
```

- **`action`**: Specifies the URL where the form data will be sent when submitted.
- **`method`**: Defines the HTTP method for sending form data:
  - `GET`: Sends data in the URL (not secure).
  - `POST`: Sends data in the HTTP request body (secure, commonly used).

2. Form Input Elements

Form input elements are used to collect data from the user.

a. Text Field (<input type="text">)

Used for single-line text input.

Syntax:

```
html
CopyEdit
<label for="name">Name:</label>
<input type="text" id="name"
name="name">
```

- **`id`**: A unique identifier for the input element.
- **`name`**: The name attribute is sent with the form data when the form is submitted.

b. Password Field (<input type="password">)
Used for entering a password. The input is masked.

Syntax:

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

c. Radio Button (<input type="radio">)
Used for selecting one option from a group of choices.

Syntax:

```
html
CopyEdit
<label for="male">Male</label>
<input type="radio" id="male" name="gender" value="male">

<label for="female">Female</label>
<input type="radio" id="female" name="gender" value="female">
```

- **`name`**: All radio buttons in the same group should have the same `name` so only one can be selected at a time.

d. Checkbox (<input type="checkbox">)
Used for selecting multiple options.

Syntax:

```
html
```

```
CopyEdit
<label for="subscribe">Subscribe to newsletter</label>
<input type="checkbox" id="subscribe" name="subscribe"
value="yes">
```

e. Submit Button (<input type="submit">)
Used to submit the form.

Syntax:

```
html
CopyEdit
<input type="submit" value="Submit">
```

- **`value`**: Text that appears on the submit button.

f. Text Area (**`<textarea>`**)

Used for multi-line text input.

Syntax:

```
html
CopyEdit
<label for="message">Message:</label>
<textarea id="message" name="message" rows="4"
cols="50"></textarea>
```

- **`rows`**: Number of visible lines.
- **`cols`**: Width of the text area in characters.

g. Select Dropdown (**`<select>`**)

Used to create a dropdown list.

Syntax:

```
html
CopyEdit
<label for="country">Select Country:</label>
<select id="country" name="country">
  <option value="usa">USA</option>
  <option value="uk">UK</option>
```

```
  <option value="india">India</option>
</select>
```

- **`<option>`**: Defines individual items in the dropdown list.

3. Form Attributes

- **`action`**: Specifies where the form data should be sent.
- **`method`**: Defines how the form data will be sent (GET or POST).
- **`target`**: Specifies where to display the response from the server (e.g., `_blank` for a new window).
- **`enctype`**: Defines how to encode the form data (important for file uploads).

4. Form Labels (`<label>`)

The `<label>` tag is used to define a label for an input element. It improves accessibility, as it associates a label with a form control.

Syntax:

```
html
CopyEdit
<label for="email">Email:</label>
<input type="email" id="email" name="email">
```

- **`for`**: Associates the label with the input element by matching the `id` of the input.

5. Form Validation

You can add simple client-side validation to ensure the user enters valid data before submitting the form.

Example:

```
html
CopyEdit
<form action="/submit" method="POST">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email"
required>
  <label for="age">Age:</label>
  <input type="number" id="age" name="age" min="18" required>

  <input type="submit" value="Submit">
</form>
```

- **`required`**: Makes the input field mandatory.
- **`min`**: Specifies the minimum value for a number input (e.g., age must be at least 18).
- **`type="email"`**: Validates that the input is in email format. **6. File Input**

(`<input type="file">`)

Used to allow users to upload files.

Syntax:

```
html
CopyEdit
<label for="file">Choose a file:</label>
<input type="file" id="file" name="file">
```

- **`multiple`**: Allows multiple files to be selected.

```
html
CopyEdit
<input type="file" name="files" multiple>
```

Example of a Complete HTML Form:

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

  <h2>Contact Us</h2>

  <form action="/submit_form" method="POST">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required><br><br>
```

```
    <label for="email">Email:</label>
    <input type="email" id="email" name="email"
required><br><br>

    <label for="gender">Gender:</label>
    <input type="radio" id="male" name="gender"
value="male"> Male     <input type="radio" id="female"
name="gender" value="female">
Female<br><br>

    <label for="message">Message:</label><br>
<textarea id="message" name="message"
rows="4" cols="50"></textarea><br><br>

    <input type="submit" value="Submit">
  </form>

</body>
</html>
```

HTML Multimedia

HTML provides several tags to embed multimedia content such as images, audio, video, and interactive media into web pages. Below is an overview of the most commonly used multimedia elements in HTML:

1. Embedding Images (`<img>`)

Images can be embedded in web pages using the `<img>` tag.

Syntax:

```
html
CopyEdit
<img src="image.jpg" alt="Description of Image" width="500"
   height="300">
```

- **`src`**: Specifies the path to the image.
- **`alt`**: Provides alternative text for the image if it cannot be displayed (important for accessibility).
- **`width`** and **`height`**: Define the size of the image (optional).

Example:

```
html
CopyEdit
<img src="logo.png" alt="Company Logo" width="100"
height="100">
```

2. Embedding Audio (`<audio>`)

The `<audio>` tag is used to embed audio files like MP3, WAV, etc.

Syntax:

```
html
CopyEdit
<audio controls>
  <source src="audio.mp3" type="audio/mp3">
  Your browser does not support the audio element.
</audio>
```

- **`controls`**: Displays the audio controls (play, pause, volume).
- **`src`**: Specifies the path to the audio file.
- **`type`**: Specifies the type of the audio file (useful for cross-browser compatibility).

Example:

```
html
CopyEdit
<audio controls>
  <source src="song.mp3" type="audio/mp3">
  Your browser does not support the audio
element. </audio>
```

You can add multiple formats for better compatibility:

```
html
CopyEdit
<audio controls>
  <source src="song.mp3" type="audio/mp3">
  <source src="song.ogg" type="audio/ogg">
  Your browser does not support the audio element.
</audio>
```

3. Embedding Video (`<video>`)

The `<video>` tag is used to embed video files like MP4, WebM, or Ogg.

Syntax:

```
html
CopyEdit
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg"
type="video/ogg">   Your browser does not
support the video element.
</video>
```

- **`controls`**: Displays the video controls (play, pause, volume).
- **`width`** and **`height`**: Define the size of the video player.
- **`src`**: Specifies the path to the video file.
- **`type`**: Specifies the type of the video file.

Example:

```
html
CopyEdit
<video width="600" height="400" controls>
  <source src="nature.mp4" type="video/mp4">
  <source src="nature.ogg"
type="video/ogg">   Your browser does not
support the video element. </video>
```

4. Embedding YouTube Videos (`<iframe>`)

To embed a YouTube video, the `<iframe>` tag is used. This is an example of embedding an external webpage (like a video).

Syntax:

```
html
CopyEdit
<iframe width="560" height="315"
src="https://www.youtube.com/embed/videoID" frameborder="0"
allowfullscreen></iframe>
```

- **`src`**: URL of the YouTube video (in the embed format).
- **`frameborder`**: Controls the border of the iframe (usually set to `0` for no border).
- **`allowfullscreen`**: Allows the video to be viewed in full-screen mode.

Example:

```
html
CopyEdit
<iframe width="560" height="315"
src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0"
allowfullscreen></iframe>
```

5. Embedding Flash Content (`<object>` or `<embed>`)

Flash content can be embedded using either the `<object>` or `<embed>` tag. However, Flash is now outdated and no longer supported by most browsers. HTML5 recommends using modern media formats like video and audio instead.

Syntax using `<object>`:

```
html
CopyEdit
<object data="flashfile.swf" width="600"

height="400"></object>
```

Syntax using `<embed>`:

```
html
CopyEdit
<embed src="flashfile.swf" width="600" height="400">
```

6. Embedding SVG Images (`<svg>`)

SVG (Scalable Vector Graphics) images can be embedded directly into HTML pages for interactive and scalable graphics.

Syntax:

```
html
CopyEdit
```

- **`<svg>`**: Defines an SVG element.
- **`<circle>`**: Draws a circle within the SVG element.

7. Embedding External Media (`<object>`)

The `<object>` tag can be used to embed a wide range of media types, including images, videos, Flash, and more.

Syntax:

```
html
CopyEdit
<object data="movie.mp4" type="video/mp4" width="500"
height="300">
  Your browser does not support this object.
</object>
```

8. Embedding PDF Files (`<object>` or `<iframe>`)

To embed a PDF file in HTML, use the `<object>` or `<iframe>` tag.

Syntax using `<object>`:

```
html
CopyEdit
<object data="file.pdf" type="application/pdf" width="600"
height="400">
  <p>Your browser does not support PDFs.</p>
</object>
```

Syntax using `<iframe>`:

```
html
CopyEdit
<iframe src="file.pdf" width="600" height="400"></iframe>
```

9. Embedding Other Media (Interactive Content)

You can also embed other types of interactive content like maps, games, and other third-party widgets using `<iframe>`.

Example (Google Map):

```
html
CopyEdit
```

```
<iframe src="https://www.google.com/maps/embed?pb=...your map details..." width="600" height="450"></iframe>
```

HTML Layout

1. HTML Layout Using Tables

While using **tables** for layout is **not recommended** in modern web design (since it is intended for displaying tabular data), understanding how to use tables for layout is still important for O-Level studies.

Tables are structured using rows and columns, and can be styled to create a layout for a webpage. Below is a guide on how to use HTML tables for layout purposes.

1. Basic Table Structure

A table in HTML consists of the following tags:

- **<table>**: Defines the table.
- **<tr>**: Defines a table row.
- **<td>**: Defines a table cell (data).
- **<th>**: Defines a table header cell (optional).

Example of a Simple Table:

```
html
CopyEdit
<table border="1">
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
  <tr>
    <td>Data 3</td>
    <td>Data 4</td>
  </tr>
</table>
```

2. Creating Layouts Using Tables

Using tables for layout involves creating a table structure where each cell (<td>) acts as a "section" or "block" of the page. You can divide the table into different sections like headers, main content, sidebars, and footers.

a. Basic Layout with Tables

Let's create a simple layout with a header, main content area, sidebar, and footer using tables.

Example:

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Layout Using Tables</title>
</head>
<body>

  <table border="1" width="100%">
    <!-- Header Row -->
    <tr>
      <td colspan="3" align="center"><h1>Website
Header</h1></td>
    </tr>

    <!-- Navigation Row -->
    <tr>
      <td colspan="3" align="center">
        <nav>
          <a href="#home">Home</a> |
          <a
href="#about">About</a> |
<a href="#contact">Contact</a>
        </nav>
      </td>
    </tr>

    <!-- Content Row -->
    <tr>
```

```
      <td width="20%" valign="top">
        <!-- Sidebar -->
        <h3>Sidebar</h3>
        <p>Links or ads go here.</p>
      </td>
      <td width="60%" valign="top">
        <!-- Main Content -->
        <h2>Main Content</h2>
        <p>This is the main content area.</p>
      </td>
      <td width="20%" valign="top">
        <!-- Additional Sidebar or Ads -->
        <h3>Ads</h3>
        <p>Advertisement goes here.</p>
      </td>
    </tr>

    <!-- Footer Row -->
    <tr>
      <td colspan="3" align="center"><footer>Footer Content © 
2025</footer></td>
    </tr>
  </table>

</body>
</html>
```

Explanation:

- The **<table>** tag defines the overall table structure.
- The **<tr>** tag defines each row in the table.
- The **<td>** tags define each individual cell within the row. The `colspan="3"` attribute allows a single cell to span across multiple columns (e.g., the header and footer cells).
- **width="20%"** is used to define the width of columns (sidebar and ads), and **width="60%"** is used for the main content area.
- **valign="top"** ensures that the content inside each table cell is aligned at the top.

3. Nested Tables

You can also create more complex layouts by nesting tables inside cells. This allows for more flexibility and control over the layout, though it can become messy and harder to maintain.

Example of Nested Tables:

```
html
CopyEdit
<table border="1" width="100%">
  <!-- Outer Table -->
  <tr>
    <td colspan="2" align="center"><h1>Website
Header</h1></td>   </tr>

  <tr>
    <td width="20%" valign="top">
      <!-- Sidebar Table -->
      <table border="1" width="100%">
        <tr>
          <td><h3>Sidebar</h3></td>
        </tr>
        <tr>
          <td>Link 1</td>
        </tr>
        <tr>
          <td>Link 2</td>
        </tr>
      </table>
    </td>

    <td width="80%" valign="top">
      <h2>Main Content</h2>
      <p>This is the main content area.</p>
    </td>
  </tr>

  <tr>
    <td colspan="2" align="center"><footer>Footer Content ©
2025</footer></td>
  </tr>
</table>
```

In this example:

- The sidebar is created using a nested table inside the left column.
- The outer table defines the main structure, while the inner table creates the sidebar's content.

4. Table Layout Best Practices (for O-Level)

Although using tables for layout was common in older web designs, here are some **best practices** for using tables for layout purposes:

- **Avoid Complex Nested Tables**: Deeply nested tables can be hard to manage and may make the webpage difficult to maintain or scale.
- **Use Table Elements Properly**: Tables should be used to display tabular data, not for layout. The `<table>` tag should be used for content like reports, data, etc.
- **Use `colspan` and `rowspan`**: The `colspan` and `rowspan` attributes are helpful for merging cells across rows and columns to create a more complex structure in the table.
- **Avoid Using Fixed Widths**: Rather than defining fixed pixel widths, consider using percentages
  (e.g., `width="20%"`) for responsive layouts, though it's better to use **CSS Flexbox** or **Grid** in modern web design.

5. Disadvantages of Using Tables for Layout

- **Not Semantically Correct**: Tables should be used to present data, not for layout. Using them for layout can make your website harder to understand for search engines and assistive technologies (e.g., screen readers).
- **Poor Accessibility**: A layout based on tables may be difficult for users with disabilities to navigate, especially if it is not properly structured.
- **Difficult to Maintain**: Complex table-based layouts are harder to modify, especially when making the page responsive for different devices.

2. DIV-Based Layout

In modern web design, **div-based layouts** are far more common and flexible than using tables. The **`<div>`** element is a block-level element used to group content and apply styles to sections of a webpage. Div-based layouts allow for better control over the positioning and design of a webpage.

Why Use Div-Based Layouts?

- **More Flexible**: The `<div>` tag is very versatile and can be styled using CSS to create any layout.
- **Easier to Maintain**: It's easier to manage and modify compared to table-based layouts.
- **Responsive Design**: Div-based layouts can be made responsive to adapt to different screen sizes using CSS techniques like Flexbox or Grid.

1. Basic Structure of a Div-Based Layout

Here's a basic example showing the structure of a webpage using `<div>` tags:

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Div-Based Layout</title>
<style>
    /* Basic styles for layout
*/     body {
      font-family: Arial, sans-serif;
    }

    /* Layout
Styling */
.container {
width: 80%;
margin: 0 auto;
overflow: hidden;
    }

    .header, .footer {
background-color: #333;
color: white;       text-
align: center;
padding: 10px 0;
    }

    .nav {
      background-color:
#f4f4f4;       text-align:
center;       padding: 10px
0;
    }

   .content {
      display: flex;
margin-top: 20px;
    }
```

```
    .main-content {
flex: 3;
padding: 20px;
    }

    .sidebar {
flex: 1;
padding: 20px;
      background-color: #f4f4f4;
    }
  </style>
</head>
<body>

  <div class="container">
    <!-- Header Section -->
    <div class="header">
<h1>Website Header</h1>
    </div>

    <!-- Navigation Section -->
    <div class="nav">
      <a href="#home">Home</a> |
      <a href="#about">About</a> |
      <a href="#contact">Contact</a>
    </div>

    <!-- Main Content Area -->
    <div class="content">
      <div class="main-content">
        <h2>Main Content</h2>
        <p>This is the main content of the page. Here you can
add text, images, and other media.</p>
      </div>

      <!-- Sidebar Section -->
      <div class="sidebar">
        <h3>Sidebar</h3>
        <p>This is the sidebar. You can add links, ads, or
other content here.</p>       </div>
    </div>

    <!-- Footer Section -->
    <div class="footer">
```

```
      <p>Footer Content © 2025</p>
    </div>
  </div>

</body>
</html>
```

Explanation:

- **`.container`**: This `<div>` acts as the wrapper for the entire page content. It's centered and has a set width.
- **`.header`**: The header section contains the page title or logo and is styled with a background color.
- **`.nav`**: This is the navigation bar with links (styled with basic padding and text alignment).
- **`.content`**: The main content area, which uses **CSS Flexbox** to divide it into two sections: the main content area and the sidebar.
- **`.main-content`**: This div takes up more space (flex: 3) and contains the primary content.
- **`.sidebar`**: This div takes up less space (flex: 1) and can contain secondary content like links or advertisements.
- **`.footer`**: The footer section with copyright information.

2. Using Flexbox for Layout

Flexbox is a CSS layout module that makes it easier to design flexible and responsive layouts. You can use it to create a two-column or three-column layout with a **`<div>`**-based structure.

Example: Two-Column Layout Using Flexbox

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Two-Column Layout with Flexbox</title>
  <style>
    /* Basic Layout Styling
*/     body {
      font-family: Arial, sans-serif;
    }
```

```
    .container {
display: flex;       flex-
direction: row;       justify-
content: space-between;
margin: 20px;
    }

    .main-content {
width: 70%;
padding: 20px;
      background-color: #f4f4f4;
    }

    .sidebar {
width: 25%;
padding: 20px;
      background-color: #e2e2e2;
   }

    .footer {
text-align: center;
padding: 10px;
background-color: #333;
color: white;
margin-top: 20px;
    }
  </style>
</head>
<body>

  <div class="container">
    <!-- Main Content -->
    <div class="main-content">
      <h2>Main Content Area</h2>
      <p>This is the main content. It takes up 70% of the
width.</p>
    </div>

    <!-- Sidebar -->
    <div class="sidebar">
      <h3>Sidebar</h3>
      <p>This sidebar takes up 25% of the width.</p>
    </div>
  </div>
```

```
  <!-- Footer -->
  <div class="footer">
    <p>Footer Content © 2025</p>
  </div>

</body>
</html>
```

Explanation:

- **`display: flex`**: This is used on the `.container` to make the two child elements (main content and sidebar) flex items.
- **`flex-direction: row`**: This arranges the items horizontally (side by side).
- **`justify-content: space-between`**: This ensures the two columns (main content and sidebar) are spaced out evenly.
- **`width` for `.main-content` and `.sidebar`**: Specifies how much space each section should occupy.

3. Responsive Div Layout Using Media Queries

To ensure the layout adjusts for different screen sizes (mobile, tablet, desktop), you can use **CSS media queries**.

Example: Responsive Div Layout

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Responsive Div Layout</title>
  <style>
body {
      font-family: Arial, sans-serif;
    }

    .container {
display: flex;       flex-
direction: row;       justify-
content: space-between;
margin: 20px;
```

```
    }

    .main-content {
width: 70%;
padding: 20px;
      background-color: #f4f4f4;
    }

    .sidebar {
width: 25%;
padding: 20px;
      background-color: #e2e2e2;
    }

    .footer {
text-align: center;
padding: 10px;
background-color: #333;
color: white;
margin-top: 20px;
    }

    /* Media Query for Mobile */
    @media (max-width: 768px) {
      .container {
        flex-direction: column;
      }

      .main-content, .sidebar {
width: 100%;
      }
    }
  </style>
</head>
<body>

  <div class="container">
    <div class="main-content">
      <h2>Main Content</h2>
      <p>This content area will stack above the sidebar on
smaller screens.</p>
   </div>

    <div class="sidebar">
```

```
      <h3>Sidebar</h3>
      <p>This sidebar will move below the content on smaller
screens.</p>
    </div>
  </div>

  <div class="footer">
    <p>Footer Content © 2025</p>
  </div>

</body>
</html>
```

Explanation:

- **Media Query (`@media`)**: This CSS rule checks the screen width. If the screen is **768px or less**, the layout changes:
    - The container switches to a **column** layout instead of row layout (using `flexdirection: column`).
    - The `.main-content` and `.sidebar` sections take up **100%** width to stack vertically.

4. Benefits of Div-Based Layouts

- **Flexibility**: You can easily manipulate the layout with CSS, adding margins, padding, and adjusting widths.
- **Clean HTML Structure**: The `<div>` tag is semantically neutral and can be styled to any design requirement.
- **Easier Maintenance**: Changes in layout (e.g., making a column wider or repositioning elements) are easier to manage without touching the HTML.

HTML `<iframe>`

An **`<iframe>`** (inline frame) is an HTML element that allows you to embed another HTML document within the current page. Essentially, it acts as a window within a webpage that can display another webpage or media from a different source.

The `<iframe>` element is commonly used for embedding content like:

- Videos (e.g., YouTube)
- Maps (e.g., Google Maps)
- External websites or web applications
- Documents (e.g., PDFs)

1. Basic Structure of an `<iframe>`

The basic syntax of an `<iframe>` is as follows:

```
html
CopyEdit
<iframe src="URL"></iframe>
```

- **`src`**: Specifies the URL of the content to be displayed within the iframe.
- **`<iframe>`** is an inline element, meaning it can be placed directly within the HTML code wherever you want the embedded content to appear.

Example:

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

In this example:

- The `src="https://www.example.com"` specifies the external webpage to embed.
- The `width="600"` and `height="400"` specify the size of the iframe.

2. Common Attributes of `<iframe>`

The `<iframe>` tag has several attributes that allow you to control the appearance, behavior, and security of the embedded content:

a. `width` ***and*** `height`

These attributes control the size of the iframe (in pixels or percentages).

```
html
CopyEdit
<iframe src="https://www.example.com" width="800"
height="600"></iframe>
```

b. **`src`**

This attribute specifies the URL of the page you want to embed in the iframe.

```
html
CopyEdit
<iframe src="https://www.youtube.com" width="800"
height="600"></iframe>
```

c. **`frameborder`**

This attribute defines whether the iframe should have a border around it. The value can be `0` (no border) or `1` (with a border).

```
html
CopyEdit
<iframe src="https://www.example.com" width="600" height="400"
frameborder="0"></iframe>
```

d. `allowfullscreen`

This attribute allows the embedded content (e.g., a video) to be displayed in fullscreen mode.

```
html
CopyEdit
<iframe  src="https://www.youtube.com"  width="800"
height="600" allowfullscreen></iframe>
```

e. **`name`**

This attribute assigns a name to the iframe, which can be used to target the iframe when linking from other pages.

```
html
CopyEdit
<iframe  src="https://www.example.com"  name="iframe1"
width="600" height="400"></iframe>
```

f. **`sandbox`**

The `sandbox` attribute provides extra security by limiting the actions that the embedded content can perform. It can take several values:

- `allow-forms`: Allows form submissions.
- `allow-scripts`: Allows the execution of scripts.
- `allow-same-origin`: Allows the iframe content to access its own domain.
- `allow-popups`: Allows pop-ups.

Example of using `sandbox`:

```
html
CopyEdit
<iframe  src="https://www.example.com"  width="600"
height="400"       sandbox="allow-forms       allow-
scripts"></iframe>
```

g. **`loading`**

The `loading` attribute controls whether the iframe should be lazily loaded. This improves page load times by deferring the loading of non-essential content until the user scrolls near it.

- `loading="lazy"`: Delays the loading of the iframe until it is near the viewport.

- `loading="eager"`: Loads the iframe immediately when the page loads.

```
html
CopyEdit
<iframe src="https://www.example.com" width="600" height="400"
loading="lazy"></iframe>
```

3. Example of Embedding a YouTube Video

One of the most common uses of an `<iframe>` is embedding a YouTube video. YouTube provides an **embed** link for each video, which includes the iframe code.

Example:

```
html
CopyEdit
<iframe width="560" height="315"
src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0"
allow="accelerometer; autoplay; encrypted-media; gyroscope;
picture-inpicture" allowfullscreen></iframe>
```

Explanation:

- **`src`**: The URL to the YouTube video in embed format.
- **`allowfullscreen`**: Allows the video to be played in fullscreen mode.
- **`allow`**: Specifies which features are allowed to be accessed by the iframe, such as autoplay, gyroscope, etc.

4. Security Considerations

When embedding content from external sources, it's important to consider **security**. Here are a few things to keep in mind:

a. Cross-Origin Resource Sharing (CORS)

By default, content within an iframe cannot access data on the parent page unless they are from the same origin (same domain). This is a security measure to prevent malicious scripts from stealing sensitive data.

b. **`sandbox`** Attribute

Using the `sandbox` attribute with appropriate restrictions (e.g., `sandbox="allow-scripts"`) can help improve security by preventing embedded content from performing potentially harmful actions, such as running JavaScript or submitting forms.

5. Responsive `<iframe>`

To make an iframe responsive, it's common to use CSS to ensure that the iframe adjusts its size according to the screen size (for mobile devices, for example). One technique is to use **CSS padding** to maintain the aspect ratio of the iframe.

Example of a Responsive YouTube Video*:*

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <style>
    .video-container {
position: relative;
      padding-bottom: 56.25%; /* 16:9 aspect
ratio */       height: 0;       overflow:
hidden;       max-width: 100%;
    }

    .video-container
iframe {       position:
absolute;       top: 0;
left: 0;       width:
100%;       height:
100%;
    }
  </style>
  <title>Responsive Iframe</title>
</head>
<body>

  <div class="video-container">
    <iframe src="https://www.youtube.com/embed/dQw4w9WgXcQ"
frameborder="0" allow="accelerometer; autoplay; encrypted-
media; gyroscope; picture-inpicture" allowfullscreen></iframe>
  </div>

</body>
</html>
```

This method ensures that the iframe adjusts to the size of the parent container while maintaining a 16:9 aspect ratio.

6. Disadvantages of Using `<iframe>`

- **Performance**: Loading an external website or media within an iframe can slow down your page, especially if the iframe content is complex.
- **SEO Issues**: Search engines may not index content within an iframe as easily as content in the parent page.
- **Security Risks**: If you're embedding content from untrusted sources, you need to be cautious, as it could contain malicious scripts.

CHAPTER 3

CASCADING STYLE SHEET

Introduction to CSS

Cascading Style Sheets (CSS) is a style sheet language used to control the presentation of web pages, including layout, colors, fonts, and spacing. It is one of the core technologies of the web, alongside HTML and JavaScript. While **HTML** provides the structure of a web page, **CSS** is used to define the visual style of that structure.

1. What is CSS?

CSS is used to style HTML elements by applying rules to them. These rules define how the elements should be displayed, including properties like:

- **Colors**
- **Fonts**
- **Spacing**
- **Positioning**
- **Layout** of the web page

2. CSS Syntax

A CSS rule consists of two main parts:

- **Selector**: The HTML element(s) that the style will be applied to.
- **Declaration Block**: One or more declarations enclosed in curly braces `{}`. Each declaration contains a **property** and a **value**.

Example:

```
css CopyEdit p
{   color:
blue;   font-
size: 16px; }
```

Explanation:

- **Selector**: `p` (applies to all `<p>` elements)
- **Declaration Block**: `{ color: blue; font-size: 16px; }`
    - **Property**: `color`, `font-size`
    - **Value**: `blue`, `16px`

3. Ways to Apply CSS

There are three main ways to apply CSS to a webpage:

a. Inline CSS

This is applied directly within an HTML element using the `style` attribute.

```
html
CopyEdit
<p style="color: red; font-size: 18px;">This is an inline CSS
example.</p>
```

- **Advantages**: Quick and easy for small changes.
- **Disadvantages**: Difficult to maintain for large websites.

b. Internal CSS

This is written inside the `<style>` tag in the `<head>` section of the HTML document.

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Internal CSS Example</title>
  <style>
body {
      background-color: lightblue;
font-family: Arial, sans-serif;
    }
h1 {
      color: green;
    }
  </style>
</head>
<body>
  <h1>Welcome to the Webpage</h1>
  <p>This page uses internal CSS.</p>
</body>
</html>
```

- **Advantages**: Better for styling multiple elements on a single page.
- **Disadvantages**: Only applies to the current document.

c. External CSS

This method uses an external CSS file linked to the HTML document. The external CSS file has a `.css` extension.

```
html
CopyEdit
<!-- Link to external CSS file -->
<link rel="stylesheet" href="styles.css">

<!-- HTML content -->
<h1>Hello, World!</h1>
<p>This page is styled using external CSS.</p>
```

The **styles.css** file:

```
css CopyEdit h1
{   color:
blue;   font-
size: 24px;
}
```

- **Advantages**: Reusable across multiple HTML pages, making maintenance easier.
- **Disadvantages**: Requires an additional HTTP request to load the CSS file.

4. CSS Selectors

CSS selectors define which HTML elements a rule will be applied to. There are different types of selectors:

a. Universal Selector (`*`)

The universal selector applies to all elements on the page.

```
css
CopyEdit
* {
  color: black;
}
```

b. Type Selector

A type selector targets all elements of a specific type (e.g., all `<h1>` elements).

```
css
CopyEdit
h1 {
color:
red;
}
```

c. Class Selector (`.`)

A class selector applies styles to any element with a specific class attribute. It is prefixed by a period `.`.

```
css CopyEdit
.myclass {
font-size:
18px; }
```

In HTML:

```
html
CopyEdit
<p class="myclass">This is a paragraph with class
"myclass".</p>
```

d. ID Selector (`#`)

An ID selector targets a unique element with a specific `id` attribute. It is prefixed by a hash `#`.

```
css
CopyEdit
#myid {
color:
blue; }
```

In HTML:

```
html
CopyEdit
<p id="myid">This is a paragraph with id "myid".</p>
```

e. Attribute Selector

You can also select elements based on their attributes.

```
css CopyEdit
input[type="text"] {
background-color:
lightgray;
}
```

5. CSS Properties

CSS has a wide variety of properties to control the appearance of elements. Here are some common ones:

a. Text Properties

- `color`: Sets the text color.
- `font-size`: Sets the font size.
- `font-family`: Defines the font.
- `text-align`: Aligns text (left, center, right).

```
css CopyEdit p {
color: black;
font-size: 16px;
text-align:
center;
}
```

b. Box Model Properties

The CSS box model consists of content, padding, border, and margin.

- **`padding`**: Space inside the element, around the content.
- **`border`**: The border surrounding the element.
- **`margin`**: Space outside the element.

```
css CopyEdit div {
padding: 20px;
border: 1px solid
black;   margin:
10px;
}
```

c. Background Properties

You can change the background color, image, and position of elements.

```
css
CopyEdit
body {
  background-color: lightblue;
}
div
{
  background-image: url('background.jpg');
background-repeat: no-repeat;   background-
position: center;
}
```

d. Layout Properties

- `width` and `height`: Set the width and height of elements.
- `display`: Defines the display behavior of an element (`block`, `inline`, `flex`, etc.).

```
css
CopyEdit
div {
width:
50%;
height:
300px;
  background-color: lightgreen;
display: block;
}
```

6. CSS Box Model

The **CSS Box Model** represents the structure of a web element. It consists of:

1. **Content**: The actual content of the element (e.g., text or images).
2. **Padding**: Space between the content and the border.
3. **Border**: Surrounds the padding (if defined).
4. **Margin**: The outermost space around the element.

```
css CopyEdit div {
width: 200px;
padding: 20px;
border: 5px solid
black;   margin:
10px; }
```

This will create a box with content inside, 20px of padding, a 5px border, and a 10px margin around it.

7. CSS Colors

CSS allows you to specify colors in several formats:

- **Named colors**: `red`, `blue`, `green`
- **Hexadecimal**: `#FF5733` (RGB values in base 16)
- **RGB**: `rgb(255, 87, 51)`
- **RGBA** (RGB with alpha transparency): `rgba(255, 87, 51, 0.5)` • **HSL** (Hue, Saturation, Lightness): `hsl(9, 100%, 60%)`

```
css
CopyEdit
h1 {
  color: #FF5733; /* Hexadecimal color */
}
```

8. CSS Positioning

Positioning defines how elements are placed on a web page. There are several positioning types:

a. Static (default):

The element is positioned according to the normal document flow.

```
css
CopyEdit
div {
  position: static;
}
```

b. Relative:

The element is positioned relative to its normal position.

```
css
CopyEdit
div {
  position:
relative;   top:
20px;   left:
30px;
}
```

c. Absolute:

The element is positioned relative to the nearest positioned ancestor.

```
css
CopyEdit
div {
  position:
absolute;   top:
50px;   right:
30px;
}
```

d. Fixed:

The element is positioned relative to the browser window and remains fixed when scrolling.

```
css CopyEdit
div {
position:
fixed;   top:
0;   left: 0; }
```

9. Responsive Web Design

CSS is a key part of making web pages responsive (adjusting to different screen sizes). This is done using **media queries**.

Example:

```
css
CopyEdit
@media screen and (max-width:
600px) {   body {
    background-color: lightblue;
```

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

This CSS rule applies when the screen width is 600px or less, changing the background color and font size.

Basic Border Property

The `border` property is a shorthand for setting all the border-related properties in one go: `borderwidth`, `border-style`, and `border-color`.

Syntax:

```
css
CopyEdit
element {
  border: [border-width] [border-style] [border-color];
}
```

- **`border-width`**: Specifies the thickness of the border (e.g., `1px`, `5px`, `thin`).
- **`border-style`**: Specifies the style of the border (e.g., `solid`, `dashed`, `dotted`).
- **`border-color`**: Specifies the color of the border (e.g., `red`, `#000000`, `rgb(255, 0, 0)`).

Example:

```
css
CopyEdit
div {
  border: 2px solid
black; }
```

In this example:

- **`2px`**: The border width is 2 pixels.
- **`solid`**: The border style is solid (a continuous line).
- **`black`**: The border color is black.

2. Individual Border Properties

You can also set each border property separately for more control. These include:

a. `border-width`

Specifies the width (thickness) of the border.

```
css
CopyEdit
div {
  border-width: 5px; /* Sets all borders to
5px */ }
```

You can also set individual borders:

```
css CopyEdit div {
border-top-width: 3px;
border-right-width:
4px;   border-bottom-
width: 5px;   border-
left-width: 6px;
}
```

b. `border-style`

Specifies the style of the border. Common values include:

- **`solid`**: A solid line.
- **`dashed`**: A dashed line.
- **`dotted`**: A dotted line.
- **`double`**: Two solid lines.
- **`groove`**: A 3D grooved border.
- **`ridge`**: A 3D ridged border.
- **`inset`**: A border that makes the element look embedded.
- **`outset`**: A border that makes the element look raised.

Example:

```
css
CopyEdit
div {
  border-style: dashed;
}
```

c. `border-color`

Specifies the color of the border.

```
css CopyEdit div
{   border-
color: red;
}
```

You can also specify different colors for each border:

```
css
CopyEdit
div {
  border-top-color: blue;   border-
right-color: green;   border-bottom-
color: yellow;   border-left-color:
purple;
}
```

3. Shorthand for Borders

You can combine all individual border properties (`border-width`, `border-style`, `border-color`) into a single shorthand declaration:

```
css
CopyEdit
div {
  border: 3px solid red;  /* 3px width, solid style,
red color */ }
```

Alternatively, you can specify the width, style, and color for each border separately:

```
css
CopyEdit
div {
  border-top: 3px dashed blue;
border-right: 5px solid green;
border-bottom: 2px dotted yellow;
border-left: 4px solid red;
}
```

4. Border Radius (Rounded Corners)

The **`border-radius`** property allows you to create rounded corners on an element. You can apply it to all four corners at once or individually.

Syntax for all corners:

```
css
CopyEdit
div {
  border-radius: 10px; /* All corners are rounded with 10px
radius */ }
```

Syntax for individual corners:

```
css
CopyEdit div {
  border-top-left-radius: 10px;
  border-top-right-radius:
15px;   border-bottom-left-
radius: 20px;   border-bottom-
right-radius: 25px; }
```

You can also use percentages to make the border radius relative to the element's size:

```
css
CopyEdit
div {
  border-radius: 50%; /* Makes a perfect circle (if width =
height) */ }
```

5. Border Box Model

When applying borders, the total width and height of an element will increase because the border takes up space. To prevent this, you can use the **`box-sizing`** property with the value **`border-box`**. This makes the element's width and height include the border and padding, so the total size of the element doesn't grow unexpectedly.

```
css
CopyEdit
div {
  box-sizing: border-
box;   width: 200px;
padding: 10px;
border: 5px solid
black; }
```

In this case, the element's total width remains 200px, including the padding and border.

6. CSS Border Shorthand for All Properties

You can use shorthand properties to define all border properties more efficiently:

- **`border`**: Shorthand for `border-width`, `border-style`, and `border-color`.
- **`border-top`**, **`border-right`**, **`border-bottom`**, **`border-left`**: Shorthand for individual borders.

Example:

```
css
CopyEdit
div {
  border: 4px dashed green; /* Combines border-width,
border-style, and border-color */ }
```

You can use the shorthand for individual borders:

```
css
CopyEdit
div {
  border-top: 3px solid red;
border-left: 2px dotted blue;
}
```

7. CSS Border Summary

Property	Description
`border`	Shorthand for `border-width`, `border-style`, and `border-color`.
`border-width`	Sets the thickness of the border.
`border-style`	Sets the style of the border (solid, dashed, dotted, etc.).
`border-color`	Sets the color of the border.
`border-radius`	Rounds the corners of the element.

8. Practical Example: Borders on a Box

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <style>     .box {
width: 200px;
height: 200px;
border: 5px solid black;
border-radius: 15px;
margin: 20px;
      background-color: lightblue;
    }
  </style>
  <title>CSS Borders</title>
</head>
<body>

  <div class="box"></div>

</body>
</html>
```

In this example:

- The `div` element has a `5px` solid black border.
- The corners of the box are rounded with a radius of `15px`.
- The background color is light blue, and there's a `20px` margin around the box.

Block Properties in CSS

In CSS, **block-level elements** are elements that occupy the full width available, meaning they create a new "block" on the page and typically stack vertically, one after another. These elements usually extend the entire width of their parent container, and their default behavior is to force a line break before and after the element.

Common Block-Level Elements:

- `<div>`
- `<p>`

- <h1>, <h2>, <h3>, etc.
- <section>
- <article>
- <header>
- <footer>

1. Block-Level Element Behavior

By default, block-level elements:

- Occupy the full width of their parent container.
- Have a new line before and after them (meaning they do not sit beside other elements).
- Can have width, height, padding, margins, and borders applied.

Example:

```
html
CopyEdit
<p>This is a block-level element.</p>
<p>This is another block-level element.</p>
```

Here, both <p> elements are block-level and will appear one below the other.

2. Changing Block-Level Elements to Inline Elements

You can change the default behavior of block-level elements by using the **display** property. This property determines how an element is displayed on the page.

- **display: block** (default for block-level elements)
- **display: inline** (removes the block behavior, and the element behaves like an inline element)
- **display: inline-block** (behaves like inline but allows width and height)

Example:

```
css
CopyEdit
p {
  display:
inline; }
```

This will make the <p> element behave like an inline element, meaning it will no longer take up the full width, and it will flow horizontally with other inline elements (e.g., text, , etc.).

3. Width and Height for Block Elements

Block-level elements can have the **`width`** and **`height`** properties applied to them, which are not typically applicable to inline elements.

Example:

```
css CopyEdit
div {
width: 50%;
height:
200px;
  background-color:
lightblue; }
```

In this case, the `<div>` will take up 50% of its parent container's width and will have a fixed height of 200px.

4. Margins and Padding for Block Elements

Block-level elements support **`margin`** and **`padding`** properties, which allow you to control the space outside and inside the element, respectively.

a. Margin

- **Margin** is the space outside the element, between the element's border and other surrounding elements.

```
css
CopyEdit
div {
  margin: 20px; /* Adds 20px space around all sides of the
element */ }
```

You can also specify individual margins for each side:

```
css CopyEdit
div {   margin-
top: 10px;
margin-right:
15px;   margin-
bottom: 20px;
margin-left:
25px;
}
```

b. Padding

- **Padding** is the space inside the element, between the content and the border.

```
css
CopyEdit
div {
  padding: 20px; /* Adds 20px padding inside all sides of the
element */ }
```

Like margins, padding can also be set individually for each side:

```
css
CopyEdit
div {
  padding-top:
10px;   padding-
right: 15px;
padding-bottom:
20px;   padding-
left: 25px; }
```

5. Text Alignment in Block Elements

You can control the alignment of text and inline content inside block-level elements using the **`textalign`** property.

Example:

```
css
CopyEdit
div {
  text-align: center; /* Centers the text inside
the div */ }
```

Other common values for **`text-align`**:

- **`left`** (default)
- **`right`**
- **`center`**
- **`justify`** (spreads the text to fit the width of the container)

6. Block Formatting Context (BFC)

When an element is a block-level element, it can create a **Block Formatting Context (BFC)**, which is useful for handling the layout and overflow of elements. A BFC is a part of the

rendering area in which block elements behave in certain predictable ways, such as preventing floating elements from affecting their layout.

BFC is created by:

- Adding **`overflow: hidden`** or **`overflow: auto`** to a block-level element.
- Using **`display: flow-root`**.
- Floating an element inside a block-level element.

7. Floating Block-Level Elements

When a block-level element is floated, it will no longer take up its full width. Instead, it will move to the left or right of its container, and other content will flow around it.

Example:

```
css
CopyEdit
div {
float:
left;
width: 50%;
}
```

This will make the `<div>` float to the left, and any content will wrap around it (e.g., text or other elements).

Clearing Floats:

After using `float`, you might need to clear the floated elements so that subsequent elements return to the normal flow.

```
css
CopyEdit
.clearfix::afte
r {   content:
"";   clear:
both;
display: block;
}
```

This technique is often used in the **clearfix hack** to prevent layout issues caused by floating elements.

8. Overflow Property in Block Elements

The **`overflow`** property controls what happens when content overflows a block-level element's box.

- **`overflow: visible`** (default): Content overflows the box and is visible outside the box.
- **`overflow: hidden`**: Content that overflows the box is clipped and not visible.
- **`overflow: scroll`**: Scrollbars are added to the element, regardless of whether the content overflows.
- **`overflow: auto`**: Scrollbars are added only when content overflows the element.

Example:

```
css CopyEdit
div {   width:
200px;
height: 150px;
overflow:
auto; }
```

9. Block Element Example: Creating a Simple Layout

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <style>
body {
      font-family: Arial, sans-serif;
    }

    .container {
width: 80%;
margin: 0 auto;
    }

    .header {
      background-color:
lightblue;       padding:
```

```
20px;       text-align:
center;
    }

    .content {
display: flex;
margin-top: 20px;
    }

    .sidebar {
width: 25%;
      background-color: lightgray;
padding: 10px;
    }

    .main {
width: 75%;
padding: 10px;
}

    .footer {
      background-color:
lightblue;       padding:
20px;       text-align:
center;       margin-top:
20px;
    }
  </style>
  <title>Block Element Layout</title>
</head>
<body>

  <div class="container">
    <div class="header">
<h1>Website Header</h1>
    </div>

    <div class="content">
      <div class="sidebar">
        <h2>Sidebar</h2>
        <p>This is a sidebar.</p>
      </div>
      <div class="main">
        <h2>Main Content</h2>
```

```
        <p>This is the main content area.</p>
      </div>
    </div>

    <div class="footer">
      <p>Website Footer</p>
    </div>
  </div>

</body>
</html>
```

In this example:

- The `.container` class uses block-level elements to structure the layout.
- The `.header`, `.content`, and `.footer` are block-level elements, with `.content` using flexbox for layout.
- The `.sidebar` and `.main` sections are block-level elements within the `.content`.

CSS Box Properties

In CSS, every HTML element is treated like a **box**.
This idea is called the **CSS Box Model**.

The Box Model has **four main areas**:

1. **Content** – The actual text or image inside the box.
2. **Padding** – Space between the content and the border.
3. **Border** – A line around the padding and content.
4. **Margin** – Space outside the border, separating the box from other elements.

1. Box Model Structure

```
plaintext
CopyEdit
| Margin |
| Border |
| Padding |
| Content |
```

2. Properties of the CSS Box

a) `width` and `height`

- Define the size of the **content area**.

Example:

```
css CopyEdit
div {
width: 300px;
height:
150px;
}
```

b) `padding`

- Controls the space **inside** the box, around the content.
- It increases the size of the box without changing the content size.

Example:

```
css CopyEdit
div {
padding:
20px; }
```

(20px space between content and border)
You can set padding individually:

```
css CopyEdit div {
padding-top: 10px;
padding-right:
20px;   padding-
bottom: 15px;
padding-left: 5px;
}
```

c) `border`

- Creates a border **around the padding and content**.

- You can set border width, style, and color.

Example:

```
css
CopyEdit
div {
  border: 2px solid
black; }
```

(2px thick solid black border)

d) `margin`

- Controls the space **outside** the border.
- It separates the box from other boxes (elements).

Example:

```
css CopyEdit
div {
margin:
30px; }
```

(30px space around the outside of the box)

You can set margins individually too:

```
css CopyEdit
div {   margin-
top: 10px;
margin-right:
20px;   margin-
bottom: 15px;
margin-left:
5px; }
```

3. Box-Sizing Property

By default, `width` and `height` only apply to the content area.
If you want the padding and border **included** in the total width and height, use:

```
css
CopyEdit
div {
  box-sizing: border-
box; }
```

This makes layout easier and more predictable.

4. Overflow Property

Sometimes content is bigger than the box. The `overflow` property controls what happens:

- `visible` (default) – content spills out.
- `hidden` – extra content is cut off.
- `scroll` – scrollbars are added.
- `auto` – scrollbars appear only when needed.

Example:

```
css CopyEdit
div {   width:
200px;
height: 100px;
overflow:
auto; }
```

5. Quick Example of Full Box Model

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
<style> .box
{   width:
300px;
height:
150px;
padding:
20px;
border: 5px
solid blue;
margin: 30px;
box-sizing:
border-box;
```

```
}
</style>
</head>
<body>

<div class="box">
This is inside the
box!
</div>

</body>
</html>
```

- `300px` total width (including padding and border).
- `150px` total height (including padding and border).

CSS Positioning Properties

1. Types of Positioning

Value	Meaning
`static`	Default positioning (normal document flow)
`relative`	Moves element relative to its normal position
`absolute`	Moves element relative to its nearest positioned ancestor
`fixed`	Positions element relative to the browser window
`sticky`	Acts like relative until a point, then acts fixed

2. static Position (Default)

- This is the **default**.
- Elements are placed according to the normal document flow.
- **Top, bottom, left, right** properties **do not** affect it.

Example:

```
css
CopyEdit
div {
  position:
static; }
```

3. relative Position

- The element is **positioned relative to where it normally would be**.
- You can move it using `top`, `bottom`, `left`, `right`.

Example:

```
css
CopyEdit
div {
  position:
relative;   top:
20px;   left:
30px; }
```

This moves the div **20px down** and **30px to the right** from its normal spot.

4. absolute Position

- The element is **removed from the normal flow**.
- It is positioned **relative to the nearest positioned ancestor** (the closest parent with `position: relative/absolute/fixed`).
- If no positioned ancestor, it positions **relative to `<html>`** (the page itself).

Example:

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

This places the element **50px from the top** and **100px from the left** of the page or ancestor.

5. fixed Position

- The element is **fixed** to the **browser window**.
- It **stays in the same place** even if the page is scrolled.

Example:

```
css

CopyEdit div {
position:
fixed;   top:
0;   right: 0;
}
```

This will stick the element to the **top-right corner** of the browser window.

Common for:

- Sticky headers
- "Back to Top" buttons

6. sticky Position

- Combines **relative** and **fixed**.
- Acts **relative** until it reaches a scroll position, then it **sticks** like fixed.

Example:

```
css CopyEdit
div {
position:
sticky;   top:
0; }
```

7. Controlling Position with `top`, `bottom`, `left`, `right`

You can control how far the element moves using:

- `top`
- `bottom`
- `left`
- `right`

Example:

```
css CopyEdit div
{   position:
absolute;   top:
20px;   left:
50px; }
```

Visual Summary Table

Position Type	Relative to	In normal flow?	Moves with scroll?
`static`	Normal flow	Yes	Yes
`relative`	Its own normal position	Yes	Yes
`absolute`	Nearest positioned ancestor or `<html>`	No	No
`fixed`	Browser window	No	No
`sticky`	Based on scroll position	Yes (until sticky)	Yes (then stuck)

Example (Using Different Positions)

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
<style> .static-box
{   position:
static;
background:
lightgray;
}

.relative-box {
position: relative;
top: 20px;   left:
30px;   background:
lightblue;
}

.absolute-box {
position: absolute;
```

```
top: 100px;   left:
150px;   background:
lightgreen;
}

.fixed-box {
position: fixed;
top: 0;   right:
0;   background:
orange;
}
</style>
</head>
<body>

<div class="static-box">Static Position</div>
<div class="relative-box">Relative Position</div>
<div class="absolute-box">Absolute Position</div>
<div class="fixed-box">Fixed Position</div>

<p>Scroll down the page to see the fixed element stay in the
top-right!</p>
</body>
</html
>
```

CSS LISTS

In HTML, lists are used to display items one after another.
There are two main types of lists:

- **Ordered List** (`<ol>`) — items are numbered (1, 2, 3, ...) • **Unordered List** (`<ul>`) — items are bulleted (•, ○, etc.)

1. List Properties in CSS

Property	Purpose
`list-style-type`	Changes bullet or number style
`list-style-position`	Changes where bullets/numbers

appear `list-style-image`	Uses an image instead of a
bullet `list-style`	Shorthand to combine all three
properties	

2. list-style-type

- Defines the type of bullet or numbering.

For unordered lists (`<ul>`):

- `disc` (default, •)
- `circle` (○)
- `square` (■)
- `none` (no bullet)

For ordered lists (`<ol>`):

- `decimal` (1, 2, 3, ...)
- `lower-roman` (i, ii, iii, ...)
- `upper-roman` (I, II, III, ...)
- `lower-alpha` (a, b, c, ...)
- `upper-alpha` (A, B, C, ...)

Example:

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

3. list-style-position

- Sets whether bullets/numbers appear **inside** or **outside** the list item box.

Value	Meaning
`inside`	Bullet is inside the text block
`outside` (default)	Bullet is outside the text block

Example:

```
css
CopyEdit
ul {
  list-style-position: inside;
}
```

4. list-style-image

- Replaces the default bullet with an **image**.

Example:

```
css
CopyEdit
ul {
  list-style-image:
url('bullet.png'); }
```

If the image is too big or small, adjust it using background properties or by resizing the image itself.

5. list-style (Shorthand)

You can combine `list-style-type`, `list-style-position`, and `list-style-image` into one line.

Example:

```
css
CopyEdit
ul {
  list-style: square
inside; }
```

This means:

- Bullet type: **square**
- Bullet position: **inside**

6. Example: Styling Lists

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: circle;   list-
style-position: inside;
}
ol
{
  list-style-type: lower-alpha;   list-
style-position: outside;
}
</style>
</head>
<body>

<h2>Unordered List</h2>
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>

<h2>Ordered List</h2>
<ol>
  <li>Introduction</li>
  <li>Basics</li>
  <li>Advanced Topics</li>
</ol>

</body>
</html>
```

CSS TABLES

CSS Table Border

Tables in HTML are used to display data in rows and columns.
CSS can be used to **add**, **style**, and **control** borders around tables, rows, and cells.

1. Adding Borders to a Table

You can add a simple border to the whole table, rows, and cells.

Example:

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

- `table` = the table itself
- `th` = table header cells
- `td` = table data cells
- `1px solid black` = border thickness, style, and color

2. Collapsing Borders

By default, table borders are **separate** (double borders may appear between cells).
To **collapse** the borders into a single line:

Example:

```
css
CopyEdit
table {
  border-collapse: collapse;
}
```

Property	Meaning
`border-collapse: separate;`	(default) borders are separate
`border-collapse: collapse;`	merges borders together

3. Border Spacing

If borders are **separate**, you can control the space between them with `border-spacing`.

Example:

```
css
CopyEdit
table {
  border-spacing:
10px; }
```

This will create 10px space between cells.

Note: `border-spacing` only works when `border-collapse: separate`.

4. Different Borders for Table Elements

You can style borders differently for the **table**, **headers**, and **cells**.

Example:

```
css
CopyEdit
table {
  border: 2px solid blue;
}
th
{
  border: 2px dashed green;
}
td
{
  border: 1px dotted
red; }
```

- Table: Solid blue border
- Header cells: Dashed green border
- Data cells: Dotted red border

5. Full Example: Styled Table Borders

```
html
CopyEdit
<!DOCTYPE
html>
<html>
<head>
<style>
table {
width:
50%;
  border-collapse: collapse;
}  th,
td {
  border: 2px solid
black;   padding: 10px;
text-align: center;
}
th
{
  background-color: lightgray;
}
</style>
</head>
<body>

<h2>Sample Table with Borders</h2>

<table>
  <tr>
    <th>Subject</th>
    <th>Marks</th>
  </tr>
  <tr>
    <td>Math</td>
    <td>90</td>
  </tr>
  <tr>
    <td>English</td>
    <td>85</td>
  </tr>
</table>

</body>
```

```
</html>
```

TABLE BORDER RADIUS

Border-radius is a CSS property used to **make table corners rounded**.

Normally, table borders are sharp (square corners).
Using `border-radius`, you can create **smooth and rounded** corners for the table and its cells.

1. Basic Border-Radius Syntax

```
css
CopyEdit
selector {
  border-radius:
value; }
```

- `value` can be in **pixels (px)**, **percentages (%)**, or **em**.
- Example values: `5px`, `10px`, `50%`

---

2. Applying Border-Radius to a Table

Example:

```
css
CopyEdit
table {
  border: 2px solid
black;   border-
radius: 10px; }
```

This gives the entire table rounded corners with a radius of **10px**.

3. Applying Border-Radius to Table Cells (Optional)

You can also add `border-radius` to individual **`<th>`** or **`<td>`** cells if needed.

Example:

```
css
CopyEdit
td {
  border: 1px solid
gray;   border-
radius: 8px;
padding: 10px; }
```

Each table cell will now have rounded corners!

4. Important Note

When you use `border-collapse: collapse` on a table (to merge borders), **border-radius on the `<table>` won't always work properly** in all browsers.

Solution:
Instead of applying `border-radius` only on the table, apply it on the **first** and **last** table cells (`td`/`th`) for better control.
Example:

```
css
CopyEdit
table {
  border-collapse:
separate;   border-
spacing: 0;   border:
2px solid black;
border-radius: 10px;
}  td:first-child {
border-top-left-radius:
10px;   border-bottom-left-
radius: 10px;
}  td:last-child {   border-
top-right-radius: 10px;
border-bottom-right-radius:
10px;
}
```

5. Full Example: Table with Rounded Borders

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

```
<head> <style> table {
width: 60%;   border:
2px solid black;
border-collapse:
separate;   border-
spacing: 0;   border-
radius: 15px;
}  th, td {
border: 1px solid
gray;   padding:
10px;   text-align:
center;
}

/* Optional: rounded corners on first and last cells */
th:first-child {
  border-top-left-radius: 15px;
} th:last-
child {
  border-top-right-radius: 15px;
} td:first-child:last-child
{   border-bottom-left-
radius: 15px;
}
td:last-child:last-child {
  border-bottom-right-radius: 15px;
}
</style>
</head>
<body>

<h2>Table with Rounded Borders</h2>

<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Ali</td>
    <td>16</td>
  </tr>
  <tr>
    <td>Sara</td>
```

```
    <td>17</td>
  </tr>
</table>

</body>
</html>
```

CSS COLOR IN TABLE

In HTML tables, you can use **CSS** to change **text color**, **background color**, and **border color** to make tables more attractive and easier to read.

1. Changing Text Color in a Table

Use the `color` property to change the **text color**.

Example:

```
css
CopyEdit
table {
color:
blue; }
```

This will change all text inside the table to blue.

You can also style only headers or cells separately:

```
css
CopyEdit
th {
  color: darkgreen;
}  td {
color:
darkred;
}
```

2. Changing Background Color in a Table

Use the `background-color` property to change the **background color**.

Example:

```
css
CopyEdit
table {
  background-color:
lightyellow; }
```

This sets a background color for the entire table.

You can target specific parts:

```
css
CopyEdit
th {
  background-color: lightblue;
}
td
{
  background-color:
lightgray; }
```

3. Changing Border Color

Use the `border` property with a color.

Example:

```
css CopyEdit table,
th, td {   border:
2px solid green; }
```

This gives a green border around the table and cells.

4. Example: Coloring a Table

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head> <style> table {
width: 60%;   border-
collapse: collapse;
background-color:
#f9f9f9;
}  th, td {   border:
1px solid black;
```

```
padding: 10px;
text-align: center;
}
th
{
  background-color: #4CAF50; /* green
*/   color: white; /* white text */
}
td
{
  background-color: #f2f2f2; /* light gray
*/   color: black; /* black text */
}
</style>
</head>
<body>

<h2>Colored Table Example</h2>

<table>
  <tr>
    <th>Subject</th>
    <th>Marks</th>
  </tr>
  <tr>
    <td>Math</td>
    <td>95</td>
  </tr>
  <tr>
    <td>Science</td>
    <td>89</td>
  </tr>
</table>

</body>
</html>
```

5. Adding Hover Effect (Bonus)

You can change the color when the mouse hovers over a row!

Example:

```
css
CopyEdit
tr:hover {
  background-color:
lightblue; }
```

COLLAPSE TABLE BORDER

When you create an HTML table with borders, by default, the **borders of cells and table** are **separate** — this causes **double borders** between cells.

To make the table look **neater** with **single borders**, we use the CSS property **`border-collapse`**.

1. What is `border-collapse`?

- **`border-collapse`** is a CSS property used to **merge** the borders of table cells into a **single line** instead of showing two separate borders.

2. Syntax

```
css
CopyEdit
table {
  border-collapse:
value; }
```

- `value` can be:
  - `separate` (default) — borders are separate
  - `collapse` — borders are merged

3. Example without `border-collapse`

```
html
CopyEdit
<table style="border: 2px solid black;">
  <tr>
    <td style="border: 2px solid red;">Cell 1</td>
    <td style="border: 2px solid blue;">Cell
2</td>   </tr>
</table>
```

Result: You will see **double borders** between the cells.

4. Example with `border-collapse: collapse`

```
html
CopyEdit
<table style="border: 2px solid black; border-collapse:
collapse;">
  <tr>
    <td style="border: 2px solid red;">Cell 1</td>
    <td style="border: 2px solid blue;">Cell 2</td>
  </tr>
</table>
```

Result: Borders between cells **merge into a single line**, making the table look **clean and sharp**.

5. Full Example: Clean Collapsed Table

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
<style>
table {
width:
50%;
  border-collapse: collapse;
}  th,
td {
  border: 2px solid
black;   padding: 10px;
text-align: center;
}
</style>
</head>
<body>

<h2>Collapsed Border Table</h2>

<table>
  <tr>
    <th>Subject</th>
    <th>Marks</th>
  </tr>
  <tr>
    <td>English</td>
```

```
    <td>80</td>
  </tr>
  <tr>
    <td>Math</td>
    <td>90</td>
  </tr>
</table>

</body>
</html>
```

TABLE STRIPED

A **striped table** is a table where **alternate rows have different background colors**. This makes the table **easier to read**, especially when it has many rows.

You can create striped tables using **CSS**.

1. Why Use Striped Tables?

- Improves **readability**.
- Makes **long tables** look **organized**.
- Helps the user easily **follow** the data row by row.

2. How to Make a Striped Table (Basic)

You use the CSS **`:nth-child()`** selector to style even or odd rows differently.

Example: Color Every Even Row

```
css CopyEdit tr:nth-
child(even) {
background-color:
#f2f2f2; }
```

This colors **every even row** (2nd, 4th, 6th, etc.) with light gray.

Example: Color Every Odd Row

```
css CopyEdit tr:nth-
child(odd) {
background-color:
#e6f7ff; }
```

This colors **every odd row** (1st, 3rd, 5th, etc.) with light blue.

3. Full Example: Striped Table

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head> <style> table {
width: 60%;   border-
collapse: collapse;
}  th, td {   border:
1px solid black;
padding: 10px;
text-align: center;
}

/* Stripe every even row */
tr:nth-child(even) {
background-color: #f2f2f2;
}

/*   Optional:   Style   the
header */ th {  background-
color:  #4CAF50;     color:
white; }
</style>
</head>
<body>

<h2>Striped Table Example</h2>

<table>
  <tr>
    <th>Subject</th>
    <th>Marks</th>
  </tr>
  <tr>
    <td>English</td>
    <td>80</td>
  </tr>
  <tr>
    <td>Math</td>
    <td>90</td>
```

```
  </tr>
  <tr>
    <td>Science</td>
    <td>85</td>
  </tr>
  <tr>
    <td>History</td>
    <td>78</td>
  </tr>
</table>

</body>
</html
>
```

PADDING IN TABLE

Padding is the space **inside** a table cell, between the **cell content** (text or image) and the **cell border**.

Without padding, the text inside a table looks cramped.
Adding padding makes the table look **neat**, **spacious**, and **easy to read**.

1. What is Padding?

- **Padding** adds **space inside** the element.
- It **pushes** the content **away from the borders**.

Syntax:

```
css CopyEdit
selector {
padding:
value; }
```

- `value` is usually given in pixels (`px`).

2. Adding Padding to Table Cells

You apply padding to `<th>` (table headings) and `<td>` (table data) elements.

Example:

```
css CopyEdit
th, td {
```

```
padding:
10px; }
```

This adds **10px** of space inside every header and data cell.

3. Full Example: Table with Padding

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
<style>
table {
width:
60%;
  border-collapse: collapse;
}  th, td {   border:
1px solid black;
padding: 12px;
text-align: center;
}
</style>
</head>
<body>

<h2>Table with Padding Example</h2>

<table>
  <tr>
    <th>Subject</th>
    <th>Marks</th>
  </tr>
  <tr>
    <td>Math</td>
    <td>90</td>
  </tr>
  <tr>
    <td>Science</td>
    <td>85</td>
  </tr>
</table>

</body>
```

```
</html>
```

The content in each cell now has **space around it**, making the table look clean!

4. Padding on Specific Sides

You can also set **different padding** for each side:

CSS Property	What it Does
`padding-top`	Space at the top
`padding-right`	Space on the right
`padding-bottom`	Space at the bottom
`padding-left`	Space on the left

Example:

```
css
CopyEdit
td {
  padding-top: 10px;
padding-bottom: 5px;
}
```

This gives **10px** padding on top and **5px** padding at the bottom.

BORDER IMAGE IN TABLE

Normally, we use a **color** or a **solid line** for table borders.
But with **CSS border-image**, you can use an **image** as the border of a table (or any element).

This makes your table look **more decorative and stylish**!

1. What is border-image?

- `border-image` is a **CSS property** that allows an **image** to be used as the **border** of an element, like a table.
- The image is **stretched**, **repeated**, or **sliced** to fit the borders.

2. Basic Syntax

```
css
CopyEdit
selector {
  border: width style color; /* normal border
required */   border-image-source: url('image-
path');   border-image-slice: number; }
```

OR shorthand:

```
css
CopyEdit
selector {
  border: width style color;
  border-image: url('image-path') slice
stretch; }
```

- `border-image-source`: sets the image URL.
- `border-image-slice`: how the image is divided into parts.
- `border-image-repeat`: (optional) repeat or stretch the image.

3. Simple Example: Table with Border Image

```
html
CopyEdit
<!DOCTYPE html>
<html>
<head>
<style>
table {
width:
60%;
  border: 10px solid transparent; /* Important: needs a border
first */   border-image:
url('https://via.placeholder.com/20x20') 30 stretch;   border-
collapse: collapse;
}  th,
td {
```

```
  border: 1px solid
black;   padding: 10px;
text-align: center;
}
</style>
</head>
<body>

<h2>Table with Border Image</h2>

<table>
  <tr>
    <th>Subject</th>
    <th>Marks</th>
  </tr>
  <tr>
    <td>English</td>
    <td>85</td>
  </tr>
  <tr>
    <td>Math</td>
    <td>90</td>
  </tr>
</table>

</body>
</html>
```

Now the **outer border** of the table is decorated using an image!

4. Important Points to Remember

- Always set a **normal border** first (even if transparent).
- `border-image-slice` controls how much of the image is used.
- `border-collapse: collapse;` keeps cell borders clean.
- Border image works **only on the full table**, not individual cells directly.