**Date 29 March 2025**

**<head> – The Brain of the Page: -** Think of it like the backstage of a play. It holds important info about the page, but not what users see directly.

|  |
| --- |
| <head>  <title>My First Page</title>  </head> |

**<body> – The Stage for Content: -** This is where all the visible stuff goes. Whatever you want people to see on the screen – like text, images, buttons – goes here.

|  |
| --- |
| <body>  <h1>Welcome to My Website</h1>  <p>This is a paragraph of content.</p>  </body> |

Note: - Html has head and body -head for metadata and body for content.

**Favicon of website**: - A favicon is the small icon you see in the browser tab next to the website title — like a tiny logo for your site.

|  |
| --- |
| <head>  <title>My Website</title>  <link rel="icon" type="image/png" href="favicon.png">  </head> |

**Heading Tags in HTML:** Headings Take Full Width by Default. In HTML, headings like <h1> to <h6> are block-level elements. That means they take up the full width of the page (from left to right). Even if the text is short, it still uses a whole line.

. Headings Are Bold by Default, When you use <h1> to <h6>, the text is automatically bold. You don’t need to use <b> or font-weight: bold.

In HTML, heading tags are used to define headings on a web page. They range from <h1> to <h6>, with <h1> being the most important (largest) and <h6> being the least important (smallest).

|  |
| --- |
| <h1>Hello</h1>  <h2>World</h2> |

These will appear on separate lines, even if you write them side by side.

**Block-Level Elements**: - They start on a new line and take up the full width of their container (like a paragraph). Used for larger sections or blocks of content.

|  |  |
| --- | --- |
| **Tag** | **Purpose** |
| <div> | Generic container |
| <p> | Paragraph |
| <h1> to <h6> | Headings |
| <ul> | Unordered list |
| <ol> | Ordered list |
| <li> | List item (inside <ul> or <ol>) |
| <table> | Table |
| <section>, <article>, <header>, <footer> | Semantic layout tags |

|  |
| --- |
| <p>This is a paragraph.</p>  <div>This is a div block.</div>  <h1>Main Heading</h1> |

**Inline elements: -** They don’t start on a new line, and only take up as much width as needed.

Used for smaller parts of content like words or phrases inside a sentence.

|  |  |
| --- | --- |
| **Tag** | **Purpose** |
| <span> | Generic inline container |
| <a> | Hyperlink |
| <strong> | Bold text |
| <em> | Italic (emphasized) text |
| <img> | Image |
| <br> | Line break (forces a new line) |
| <input>, <label> | Form elements |

|  |
| --- |
| <p>This is a <strong>bold</strong> word and <em>italic</em> text.</p>  <a href="#">This is a link</a> |

**<p> – Paragraph Tag: -** Purpose: Used to define a paragraph of text. Block-level element – it starts on a new line. Adds some space above and below by default.

|  |
| --- |
| <p>This is my first paragraph.</p>  <p>This is another paragraph.</p> |

**<hr> – Horizontal Rule:-** Purpose: Adds a horizontal line across the page. Used to Use to separate sections of content. It’s a self-closing tag (doesn’t need a closing </hr>).

|  |
| --- |
| <p>This is the first section.</p>  <hr>  <p>This is a new section after the horizontal line.</p> |

**Syntax of a comment in HTML:** Explain your code for yourself or others.

Temporarily hide parts of code during testing. Leave reminders or notes (like TODOs).

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

|  |
| --- |
| <!-- This is the main heading -->  <h1>Welcome to My Website</h1>  <!-- This section shows the user bio -->  <p>Hello! I'm a web developer.</p>  <!-- TODO: Add footer later --> |

**Date 01 April 2025**

**<! DOCTYPE> in HTML : -** <! DOCTYPE> is a declaration that tells the web browser what version of HTML the page is written in. This is for HTML5 (the latest version).

HTML tag called as Root tag and Head, Body child tag.

**UTF-8** is a character encoding — it tells the browser how to display text (like letters, numbers, and symbols) on a webpage.

**<b> — Bold for Style: -** Just makes text bold, No extra meaning or importance, Used when you want text to look bold but not for emphasis. <p>this is a <b>bold</b> word. </p>.

**<strong> — Bold for Importance** :- Makes text bold and tells the browser/screen reader that the text is important. Used for emphasis or warning. <p><strong>Warning: < /strong> Do not touch the wire! </p>

**<i> — Italic for Style**: - Just makes the text italic, It’s visual only, no special importance, <p> This is a <i>beautiful</i> painting. </p>

**<em> — Emphasis** :- Makes text italic AND adds emphasis or stress. Tells screen readers: “Read this with more feeling!”, Helps with meaning and tone. <p>I <em>really</em> like this website! </p>

**<br> in HTML** :- <br> stands for "break" — it creates a line break in your text.

To move to the next line without starting a new paragraph, it’s like hitting Enter on your keyboard.

**<u> – Underline (for style): -** Makes text underlined. No extra meaning, just visual styling. Use it for names, headings, or emphasis when you just want a line under the text.

<p>This is a <u>special</u> word. </p>

**<ins> – Inserted Text (underlined with meaning) :-** Shows text that has been added or inserted. Looks underlined, but with meaning — used in edits or document updates.

<p>New price: <ins>₹800</ins></p>

**<del> – Deleted Text (crossed out) :-** Shows text that has been removed or is no longer valid. Often used in editing, pricing changes, or corrections. Text will have a strike-through.

<p>Old price: <del>₹1000</del></p>

**<sup> – Superscript : -** Superscript means small text above the normal line. Math powers: 2<sup>2</sup> (means 2²). <p>Area = a<sup>2</sup></p>

**<sub> – Subscript:-** Subscript means small text below the normal line. Chemical formulas: H<sub>2</sub>O (water). <p>Water formula: H<sub>2</sub>O</p>

**<div> – Block Container :-** <div> stands for "division". It’s a block-level element, which means it takes up the full width of the page. Used to group larger sections of HTML — like a header, sidebar, or content area. Often used with CSS for layout and styling.

|  |
| --- |
| <div>  <h2>My Blog</h2>  <p>This is a blog post.</p>  </div> |

**<span> – Inline Container: -** <span> is an inline element, which means it stays within the line of text. Used to Use to style small pieces of text inside a paragraph or sentence. Also used with CSS for color, font, etc.

|  |
| --- |
| <p>This is <span style="color:red;">important</span> text.</p> |

**Ordered List (<ol>):-** Means the list has a numbered order (1, 2, 3…). Each item is inside an <li> (list item) tag.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Descriptions** | **Example** |
| type | Number style (1, A, a, I, i) | <ol type="A"> → A, B, C... |
| start | Starting number | <ol start="5"> → 5, 6, 7... |
| reversed | Numbers go backward | <ol reversed> → 3, 2, 1 |

|  |
| --- |
| <ol>  <li>Wake up</li>  <li>Brush teeth</li>  <li>Have breakfast</li>  </ol> |

**Unordered List (<ul>):-** Bulleted list (dots, not numbers). Good for random or unordered items.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Descriptions** | **Example** |
| Type (ul) | Type of bullet (not supported in HTML5, better to use CSS) |  |
| Value (li) | Used in <ol> to change the number of a single item |  |

|  |
| --- |
| <ul>  <li>Apples</li>  <li>Bananas</li>  <li>Oranges</li>  </ul> |

**Nested List: -** A nested list means you put one list inside another list. It’s like a list with sub-items (just like folders inside folders).

|  |
| --- |
| <ul>  <li>Fruits  <ul>  <li>Apple</li>  <li>Banana</li>  </ul>  </li>  <li>Vegetables  <ul>  <li>Carrot</li>  <li>Spinach</li>  </ul>  </li>  </ul> |

**Date 02 April 2025**

**Description List: -** A Description List is used to show a list of terms and their descriptions, like a glossary or FAQ

It uses 3 main tags:

|  |  |  |
| --- | --- | --- |
| **Tag** | **Meaning** | **Purpose** |
| <dl> | Description List | Wraps the whole list |
| <dt> | Description Term | The title or item name |
| <dd> | Description Data | The meaning or detail of the term |

**Example**

|  |
| --- |
| <dl>  <dt>HTML</dt>  <dd>HyperText Markup Language - used to structure web pages.</dd>  <dt>CSS</dt>  <dd>Cascading Style Sheets - used to style HTML pages.</dd>  <dt>JavaScript</dt>  <dd>A programming language used to make web pages interactive.</dd>  </dl> |

**Table Tag: -** The <table> tag is used to display data in a tabular format (rows and columns).

* <tr> – (Table Row) It creates a horizontal row in a table, Inside <tr>, we use <td> (table data) or <th> (table header)
* <td> – (Table Data) Used to create normal cells in a row (data content)
* <th> – (Table Header) Used to define heading cells (bold & centered by default),

**Basic Table Structure**

|  |
| --- |
| <table>  <tr>  <th>Heading 1</th>  <th>Heading 2</th>  </tr>  <tr>  <td>Data 1</td>  <td>Data 2</td>  </tr>  </table> |

**Basic Attributes of <table>**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Example** |
| border | Adds a border around the table | border="1" |
| cellpadding | Space inside each cell | cellpadding="10" |
| cellspacing | Space between cells | cellspacing="5" |
| width | Width of the table | width="500" or width="100%" |
| height | Height of the table | height="200" |
| align | Align table on page (left, center, right) | align="center" |
| bgcolor | Background color of the table | bgcolor="lightblue" |

**Example with Basic Attributes:**

|  |
| --- |
| <table border="1" cellpadding="10" cellspacing="5" width="60%" align="center" bgcolor="lightyellow">  <tr>  <th>Name</th>  <th>Age</th>  <th>City</th>  </tr>  <tr>  <td>Amit</td>  <td>30</td>  <td>Mumbai</td>  </tr>  <tr>  <td>Neha</td>  <td>25</td>  <td>Pune</td>  </tr>  </table> |

**Advanced Table Attributes & Tags:**

1. colspan – Merge columns

|  |
| --- |
| <td colspan="2">Merged Column</td> |

2. rowspan – Merge rows

|  |
| --- |
| <td rowspan="2">Merged Row</td> |

**Example with colspan and rowspan:**

|  |
| --- |
| <table border="1" cellpadding="5">  <tr>  <th rowspan="2">Name</th>  <th colspan="2">Marks</th>  </tr>  <tr>  <th>Math</th>  <th>Science</th>  </tr>  <tr>  <td>Ravi</td>  <td>85</td>  <td>90</td>  </tr>  </table> |

3. thead, tbody, tfoot – Structure the table

|  |
| --- |
| <table border=”1”>  <thead>  <tr><th>Name</th><th>Score</th></tr>  </thead>  <tbody>  <tr><td>Raj</td><td>80</td></tr>  <tr><td>Anita</td><td>90</td></tr>  </tbody>  <tfoot>  <tr><td>Total</td><td>170</td></tr>  </tfoot>  </table> |

**Form Tag: -** The <form> tag is used to collect user input like name, email, password, etc

It serves as a container for various form elements like input fields (text, email, password, etc.), text areas, select boxes, radio buttons, checkboxes, and submit buttons.

**Label Tag:** - The <label> tag is used to define labels for input elements like: Textboxes, Radio buttons, Checkboxes, Dropdowns.

It improves form accessibility (especially for screen readers), When you click the label, it focuses or activates the related input box.

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Example |
| for | Connects the label to an input element using the input's id | <label for="name">Name:</label> |

|  |
| --- |
| <form>  <label for="username">Username:</label>  <input type="text" id="username" name="username">  </form> |

* The label is linked to the input with id="username"
* When you click "Username:", it automatically focuses on the input box

**Date 04 April 2025**

**Input Tag** - used to create input fields in a form where users can enter data like text, numbers, passwords, etc. The **<input>** tag is used to create input fields in a form where users can enter data like text, numbers, passwords, etc.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| type | Specifies the kind of input (text, number, password…) | <input type="text"> |
| name | Name of the input (used in form submission) | <input type="text" name="username"> |
| value | Sets a default value inside the field | <input type="text" value="John"> |
| placeholder | Shows hint text inside the input | <input type="text" placeholder="Enter your name"> |
| required | Makes the field mandatory | <input type="email" required> |
| maxlength | Limits number of characters | <input type="text" maxlength="10"> |

**Radio Button** - A radio button lets the user select only one option from a group (like choosing Male or Female, but not both).

It uses the **<input type="radio">** tag**.**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| type="radio" | Tells the browser this is a radio button | <input type="radio"> |
| name | Groups radio buttons together — only one option can be selected | <input type="radio" name="gender"> |
| value | The actual value sent when the form is submitted | <input type="radio" value="Male"> |
| checked | Pre-selects this radio button by default | <input type="radio" checked> |
| id | Gives a unique ID to connect with a <label> | <input type="radio" id="male"> |

**Drop Down**: - A dropdown menu lets the user choose one option from a list.

It is created using the **<select>** tag with multiple **<option>** items inside it.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| <select> | The dropdown menu container | <select name="city"> |
| <option> | Represents a single choice in the dropdown | <option value="india">India</option> |
| name | Name used when form is submitted | <select name="country"> |
| value | Actual data sent when selected | <option value="usa">USA</option> |
| multiple | Allows selecting more than one item | <select multiple> |
| id | Gives a unique ID (used with <label>) | <select id="fruit"> |
| disabled | Disables the dropdown (user can't use it) | <select disabled> |
| selected | Makes this option selected by default | <option selected>India</option> |

**Input Type: File -** This input type lets the user upload files (like images, PDFs, documents) from their device.

Basic syntax - **<input type="file">**, this creates a "Choose File" or "Browse" button to pick a file.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| type="file" | Tells the browser this input is for file upload | <input type="file"> |
| name | Name of the input (used in form submission) | <input type="file" name="resume"> |
| accept | Specifies what type of files are allowed | <input type="file" accept=".jpg, .png, .pdf"> |
| multiple | Allows selecting more than one file | <input type="file" multiple> |
| disabled | Makes the file input unusable (grayed out) | <input type="file" disabled> |
| required | Makes file upload mandatory | <input type="file" required> |

**Input Type: Date -** It creates a date picker so users can easily choose a date (like birthday, appointment, etc.) from a calendar.

Basic syntax - **<input type="date">,** this shows a calendar input where the user can pick a date.

**Input Type: Color -** It creates a color picker that lets users select a color using a visual box or color wheel.

Basic syntax - **<input type="color">,** this shows a small color box that opens a color picker when clicked.

**Input Type: - Checkbox** - It creates a **checkbox** that lets users **select one or more options** (like "I agree", or selecting hobbies).

Basic syntax - **<input type="checkbox">,** this shows a small square box the user can check or uncheck.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| type="checkbox" | Tells the browser this is a checkbox | <input type="checkbox"> |
| name | Name used in form submission used in form | <input type="checkbox" name="hobby"> |
| value | The value sent when checkbox is checked | <input type="checkbox" value="Cricket"> |
| checked | Makes the checkbox selected by default | <input type="checkbox" checked> |
| disabled | Makes at least one checkbox selection mandatory | <input type="checkbox" required> |
| required | Makes the checkbox unusable (grayed out) | <input type="checkbox" disabled> |

**Image Tag:** - The **<img>** tag is used to display images on a web page.

It is a self-closing tag, meaning it doesn’t need a closing tag like **</img>.**

Basic syntax - **<img src="image.jpg" alt="Description of image" width="200" height="150">**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| src | The source (URL or file path) of the image | src="cat.jpg" |
| alt | Alternate text shown if the image can't load (also helps with SEO & screen readers) | alt="A cute cat" |
| width | Width of the image (in pixels or %) | width="200" |
| height | Height of the image (in pixels or %) | height="150" |
| title | Tooltip text when you hover over the image | title="This is a cat" |

**Link Tag:** - The **<a>** tag is used to create links in HTML.

It can link to another webpage, a file, an email, or even a specific section within the same page.

It’s also called an anchor tag. Basic syntax - **<a href="URL">Link Text</a>**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meaning** | **Example** |
| href | The URL or path where the link will go | href="https://google.com" |
| target | Defines where to open the link: same tab or new tab | \_self (default), \_blank (new tab) |
| title | Tooltip text shown on hover | title="Go to Google" |
| download | Tells the browser to download the file instead of opening it | Download  <a href="file.pdf" download>Download PDF</a> |
| rel | Describes the relationship between the current and linked page (used with target="\_blank") | rel="noopener noreferrer" |

**Semantic Elements:** - Semantic elements clearly describe their meaning in a human- and machine-readable way. They help both developers and browsers understand the structure of the page better.

|  |  |
| --- | --- |
| **Tag** | **Meaning** |
| <header> | Top section of the page (like title, logo, nav) |
| <nav> | Navigation links (menus) |
| <main> | Main content of the page |
| <section> | A section of content |
| <article> | Independent content (like a blog post) |
| <aside> | Side content (like ads, tips, or links) |
| <footer> | Bottom of the page (like copyright, contact) |
| <figure> | Used for images with captions |
| <figcaption> | Caption for an image inside <figure> |

**Example: -**

|  |
| --- |
| <header>  <nav>  <a href="#">Home</a>  <a href="#">About</a>  </nav>  </header>  <main>  <article>  <h1>My Blog Post</h1>  <p>This is the content.</p>  </article>  </main>  <footer>  <p>© 2025 My Website</p>  </footer> |

**Non-Semantic Elements**: - Non-semantic elements don’t describe their meaning.

They are generic containers used mainly for layout and styling.

|  |  |
| --- | --- |
| **Tag** | **Use** |
| <div> | A container used to group elements (no meaning)used to |
| <span> | Used to style inline text or parts (no meaning) |

**Example: -**

|  |
| --- |
| <div>  <div>  <a href="#">Home</a>  <a href="#">About</a>  </div>  </div>  <div>  <h1>My Blog Post</h1>  <p>This is the content.</p>  </div>  <div>  <p>© 2025 My Website</p>  </div> |

Conclusion:

* Use semantic tags when possible – they make your HTML clean, accessible, and SEO-friendly.
* Use non-semantic tags like **<div>** or **<span>** when you just need layout/styling without meaning.

**Date 07 April 2025**

**<blockquote> –** Block Quotation Tag:- Purpose: Used to show a long quote from another source. It is a block-level element, so it starts on a new line. Browsers usually indent the text to make it stand out.

|  |
| --- |
| <blockquote>  "The only limit to our realization of tomorrow is our doubts of today."  </blockquote> |

**<q> – Short Inline Quote Tag: -** Purpose: Used for short quotes within a sentence. It is an inline element (doesn’t start a new line). Browsers usually add quotation marks around the quoted text automatically.

Use <q> for small quotes inside text. Use <blockquote> for long quotes or multiple sentences.

|  |
| --- |
| <p>Albert Einstein once said, <q>Imagination is more important than knowledge.</q></p> |

**<cite> – Citation Tag:-** Purpose: Used to reference the title of a creative work, like a book, movie, website, song, painting, etc. It is an inline element. Browsers usually show it in italic style by default.

Use <cite> for names of books, films, songs, articles, artworks, etc. It’s not for quoting text — for that, use <q> or <blockquote>.

|  |
| --- |
| <p>I recently read <cite>The Alchemist</cite> by Paulo Coelho.</p> |

**<audio> Tag — For Sound:-** Used to embed audio files (like music, voice, sound effects) into a web page.

|  |
| --- |
| <audio controls>  <source src="sound.mp3" type="audio/mpeg">  Your browser does not support the audio tag.  </audio> |

|  |  |
| --- | --- |
| **Attribute** | **What is does** |
| Controls | Shows play/pause, volume controls |
| autoplay | Starts playing automatically (⚠️ annoying if misused) |
| loop | Plays the audio repeatedly |
| muted | Starts the audio in muted mode |

**<video> Tag — For Videos:- Used to embed video files (like tutorials, clips, movies) into a web page.**

|  |
| --- |
| <video width="400" controls>  <source src="movie.mp4" type="video/mp4">  Your browser does not support the video tag.  </video> |

|  |  |
| --- | --- |
| **Attribute** | **What is does** |
| Controls | Shows play/pause, volume controls |
| autoplay | Starts playing automatically (⚠️ annoying if misused) |
| loop | Plays the audio repeatedly |
| muted | Starts the audio in muted mode |
| width / height | Sets video size |

**CSS-** CSS stands for Cascading Style Sheets.

It is used to style HTML elements — meaning it controls how things look on a webpage (colors, fonts, layout, spacing, etc.).

**Types of CSS**

There are 3 main types of applying CSS to HTML:

1. **Inline CSS**.
   * CSS is written inside the HTML tag using the style attribute.
   * Best for quick, small changes.

|  |
| --- |
| <p style="color: blue;">This is blue text</p> |

1. **Internal CSS**
   * CSS is written inside a <style> tag in the <head> section of the HTML file
   * Best for styling a single HTML page.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  p {  color: green;  font-size: 18px;  }  </style>  </head>  <body>  <p>This is green text</p>  </body>  </html> |

1. **External CSS**
   * CSS is written in a separate file (e.g., style.css).
   * Linked to the HTML file using a <link> tag.
   * Best for styling multiple pages consistently.

|  |
| --- |
| head>  <link rel="stylesheet" href="style.css">  </head>  <body>  <p>This is styled by external CSS</p>  </body> |

CSS file (style.css):

|  |
| --- |
| p {  color: purple;  font-weight: bold;  } |

**Selector in CSS**: - A selector tells the browser which HTML element(s) you want to style. Think of it like:

👉 “Hey browser, find this element, and apply this style to it.”

**Types of CSS Selectors**

1. **Element Selector**
   * Selects all elements of a specific type (like all <p> or all <h1>).

|  |
| --- |
| CSS  p {  color: blue;  } |

This makes all <p> tags blue.

1. **ID Selector**
   * Selects one element with a specific id.
   * Use # before the id name.

|  |
| --- |
| Css  #title {  font-size: 24px;  } |

Targets an element like:

|  |
| --- |
| <h1 id="title">Hello</h1> |

1. **Class Selector**
   * Selects all elements with a specific class.
   * Use . before the class name.

|  |
| --- |
| css  .text-red {  color: red;  } |

**Targets**

|  |
| --- |
| <p class="text-red">This is red</p>  <span class="text-red">Red too!</span> |

1. **Universal Selector**
   * Selects all elements on the page.
   * Use \* symbol.

|  |
| --- |
| css  \* {  margin: 0;  padding: 0;  } |

Useful for resetting styles.

1. **Group Selector**
   * Selects multiple elements at once, separated by commas.

|  |
| --- |
| css  h1, h2, p {  font-family: Arial;  } |

1. **Descendant Selector**
   * Targets elements inside other elements.

|  |
| --- |
| css  div p {  color: green;  } |

Styles all <p> tags inside a <div>.

**08 April 2025**

**CSS**

1. **padding**
   * What it does: Adds space inside the element, between the content and the border.
   * Syntax: padding: 10px; (can be top/right/bottom/left individually too)

**Example:**

|  |
| --- |
| div {  padding: 20px;  } |

Adds 20px of space inside the div from all sides.

1. **Border**
   * What it does: Adds a border around the element.
   * Syntax: border: 2px solid red;
   * border style - solid, double, dotted, thick , thin

**Example:**

|  |
| --- |
| div {  border: 2px solid blue;  } |

Gives a 2px solid blue border around the div.

1. **margin**
   * What it does: Adds space outside the element (gap between other elements).
   * Syntax: margin: 10px;

**Example:**

|  |
| --- |
| div {  margin: 20px;  }  margin: 40px 40px 40px 40px |

Creates 20px space outside the div

1. **text-decoration**
   * What it does: Adds decoration to text (like underline, line-through).
   * Syntax: text-decoration: underline;

**Example:**

|  |
| --- |
| a {  text-decoration: none;  }  text-decoration: 2px overline dotted tomato; |

Removes the underline from links.

1. **Display**
   * What it does: Controls how an element is displayed (block, inline, flex, etc.).
   * Common values:
     1. **block** – Takes full width,
     2. **inline** – Takes only the content width
     3. Used for flexible layoutsflex – Used for flexible layouts
     4. **Inline-bloack** - (stay in line), but you can control their width/height like block

**Example**:

|  |
| --- |
| div {  display: flex;  }  display: inline-block; |

Makes the div a flex container for layout.

**Common display values:**

|  |  |
| --- | --- |
| **Value** | **What It Does** |
| block | Starts on a new line, takes full width (div, p) |
| inline | Stays in line, takes only content width (span, a) |
| inline-block | Like inline but allows setting width/height |
| none | Hides the element (not visible on the page) |
| flex | Enables flexible layout (used for rows/columns) |
| grid | Creates a grid-based layout |

|  |  |  |
| --- | --- | --- |
| **Display Type** | **Behavior** | **Can set width/height?** |
| block | New line, full width | ✅ Yes |
| inline | Same line, content width | ❌ No |
| inline-block | Same line, but adjustable | ✅ Yes |
| none | Hides element completely | ❌ Not visible |
| flex | Flexible row/column layout | ✅ Yes (container) |
| grid | 2D grid layout | ✅ Yes (container) |

1. **box-shadow**
   * What it does: Adds shadow around the box.
   * Syntax: box-shadow: h-offset v-offset blur color;

Example:

|  |
| --- |
| div {  box-shadow: 5px 5px 10px gray;  } |

Adds a gray shadow to the div.

1. **text-align**
   * What it does: Aligns the text (left, right, center, justify).
   * Syntax: text-align: center;

Example:

|  |
| --- |
| p {  text-align: center;  } |

Centers the text inside the paragraph.

1. **text-shadow**
   * What it does: Adds shadow to text.
   * Syntax: text-shadow: h-offset v-offset blur color;

Example:

|  |
| --- |
| h1 {  text-shadow: 2px 2px 5px gray;  } |

Adds a gray shadow to the heading text.

|  |  |  |
| --- | --- | --- |
| **Property** | **Purpose** | **Example** |
| padding | Inner space | padding: 10px; |
| margin | Outer space | margin: 10px; |
| border | Outline of the element | border: 1px solid black; |
| text-decoration | Decorates text | text-decoration: underline; |
| display | Layout type | display: flex; |
| box-shadow | Shadow around box | box-shadow: 2px 2px 5px; |
| text-align | Align text | text-align: center; |
| text-shadow | Shadow behind text | text-shadow: 1px 1px gray; |

Shecodes.io - color

**09 April 2025**

1. **border-radius**
   * What it does: Rounds the corners of an element.
   * Syntax: border-radius: 10px;

**Example**

|  |
| --- |
| div {  border-radius: 15px;  }  border-radius: 350px 350px 0px 350px; |

Gives the div rounded corners.

0px – square corners (default), 50% – perfect circle (if height = width)

1. **Position**
   * What it does: Controls how an element is placed on the page.
   * Common values:
     + static (default) – Normal flow
     + relative – Move relative to original position
     + absolute – Exact position in the parent
     + fixed – Stays in the same place on scroll
     + sticky – Sticks when scrolling within a container

Example:

|  |
| --- |
| div {  position: absolute;  top: 10px;  left: 20px;  } |

Places the div 10px from the top and 20px from the left.

* 1**. Default position: static (for div and most elements)**
  + No effect of top, left, etc.
  + Element stays in normal flow

Example:

|  |
| --- |
| <style>  div {  position: static;  top: 20px; /\* ❌ No effect \*/  }  </style>  div>This is static position (default)</div> |

* position: relative
  + Element stays in normal flow but moves relative to its original place.
  + You can use top, left, right, bottom.

Example:

|  |
| --- |
| <style>  .relative-box {  position: relative;  top: 20px;  left: 30px;  background-color: lightblue;  padding: 10px;  }  </style>  <div class="relative-box">Relative Position (moved down 20px, right 30px)</div> |

It moves, but the original space it took remains.

* **position: absolute**
  + The element is removed from the normal flow.
  + It's positioned relative to the nearest positioned parent (i.e., the closest parent with position: relative or absolute).
  + Uses top, left, right, bottom.

**Example:**

|  |
| --- |
| <style>  .parent {  position: relative;  height: 150px;  background-color: lightgray;  }  .absolute-box {  position: absolute;  top: 20px;  left: 20px;  background-color: orange;  padding: 10px;  }  </style>  <div class="parent">  <div class="absolute-box">Absolute Position (inside parent)</div>  </div> |

The orange box is placed 20px from the top and left of its parent, not the page.

* **BONUS: If no parent is relative or absolute, absolute positions based on the page (<body>):**

|  |
| --- |
| <style>  .absolute-global {  position: absolute;  top: 50px;  left: 50px;  background-color: pink;  padding: 10px;  }  </style>  <div class="absolute-global">Absolute - positioned from body</div> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Position Type** | **Affects Layout?** | **Moves with top/left?** | **Based on Parent?** |
| static | Normal flow | ❌ No | ❌ |
| relative | Normal flow | ✅ Yes (shifts itself) | ❌ |
| absolute | Removed from flow | ✅ Yes | ✅ (nearest positioned parent) |

1. **z-index**
   * What it does: Controls which element appears on top (higher value = front).
   * Syntax: z-index: 2;

display: inline-block;

|  |
| --- |
| div {  position: absolute;  z-index: 10;  } |

This div will appear above others with lower z-index.

1. **background-position** 
   * What it does: Sets the starting position of a background image.
   * Common values:
     + left top
     + center center
     + right bottom
     + Or in px/percentage: 50% 50%

**Example:**

|  |
| --- |
| div {  background-image: url("image.jpg");  background-position: center center;  } |

Centers the background image in the element.

1. **background-size**
   * What it does: Sets the size of the background image.
   * Common values:
     + cover – Fills the box (may crop)
     + contain – Fits image fully (may leave gaps)
     + auto – Original size
     + 100% 100% – Stretch to fit

**Example:**

|  |
| --- |
| div {  background-size: cover;  }  background-size: 45%; |

Background image covers the full element.

1. **background-repeat**
   * What it does: Controls if the background image repeats.
   * Common values:
     + repeat – Repeats both x and y (default)
     + no-repeat – No repeat
     + repeat-x – Repeats horizontally
     + repeat-y – Repeats vertically

**Example:**

|  |
| --- |
| div {  background-repeat: no-repeat;  } |

1. **background-image** 
   * What it does: Sets an image as the background.
   * Syntax: background-image: url("image.jpg");

**Example:**

|  |
| --- |
| div {  background-image: url("bg.jpg");  } |

Adds a background image to the div.

|  |  |  |
| --- | --- | --- |
| **Property** | **What It Does** | **Common Values/Examples** |
| border-radius | Rounds corners | 10px, 50% (circle) |
| position | Sets element's position | static, relative, absolute, fixed |
| z-index | Layer order (front/back) | z-index: 1;, z-index: 100; |
| background-position | Position of bg image | center center, top left, 50% 50% |
| background-size | Size of background image | cover, contain, 100% 100% |
| background-repeat | Repeats background image | repeat, no-repeat, repeat-x |
| background-image | Adds an image as background | url("image.jpg") |

Note: Default position: static (for div and most elements), No effect of top, left, etc.

Postion by default static for div position relative and absolute – top, bottom, left, right

**10 April 2025**

1. **background-attachment**

The background-attachment property controls how a background image behaves when you scroll the page.

|  |  |
| --- | --- |
| **Value** | **Behavior** |
| scroll | (Default) Background scrolls with the page |
| fixed | Background stays in place (doesn't scroll) — creates a parallax effect |
| local | Background scrolls with the element (rarely used) |

|  |
| --- |
| <style>  .scroll-bg {  height: 400px;  background-image: url('https://via.placeholder.com/800x400');  background-repeat: no-repeat;  background-size: cover;  background-attachment: scroll;  }  </style>  <div class="scroll-bg">  <h2>Scroll with page (default)</h2>  </div> |

1. **overflow**

The overflow property controls what happens when content is too big for the container (div, section, etc.).

|  |  |
| --- | --- |
| **Value** | **Description** |
| visible | Default. Content overflows and is visible |
| hidden | Extra content is cut off (not visible) |
| scroll | Always shows scrollbars (horizontal/vertical) |
| auto | Scrollbars appear only if needed |

|  |
| --- |
| <style>  .box1 {  width: 200px;  height: 100px;  background: lightblue;  overflow: visible; - hidden, scroll, auto  }  </style>  <div class="box1">  This is a very long text that goes outside the box area but it's still visible.  </div> |

**overflow-x and overflow-y:- Control scroll horizontally or vertically:**

|  |
| --- |
| overflow-x: auto; /\* horizontal \*/  overflow-y: hidden; /\* vertical \*/ |

1. **font-family**

The font-family property in CSS sets the typeface (font style) for text — like Arial, Verdana, Georgia, Roboto, etc

|  |
| --- |
| selector {  font-family: font1, font2, generic-family;  }  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; |

Common CSS Properties (you can use inside a class):

|  |  |  |
| --- | --- | --- |
| **Property** | **Example Value** | **What It Does** |
| color | red, #333, rgb(0,0,0) | Text color |
| background-color | blue, #f1f1f1 | Background color |
| font-size | 16px, 1.2em | Size of text |
| padding | 10px | Space inside the box |
| margin | 10px | Space outside the box |
| border | 1px solid black | Border style |
| text-align | left, center | Aligns text |
| width, height | 100px, 50% | Size of element |
| display | block, inline, flex | Layout style |

**Note:** You can use more than one class on the same element (space-separated). Class names should not start with numbers

* **.classname::first-line**
  + Styles only the first line of the text inside the element.

|  |
| --- |
| <style>  .para::first-line {  color: blue;  font-weight: bold;  }  </style>  <p class="para">  This is the first line of this paragraph. This part is long to make sure the line breaks into two.  </p> |

Only the first line gets styled (depends on screen size/width).

* .**classname::first-letter**
  + Styles only the first letter of the text inside the element.

|  |
| --- |
| <style>  .para2::first-letter {  font-size: 30px;  color: red;  }  </style>  <p class="para2">  This paragraph has a big first letter.  </p> |

Great for magazine-style headings or content!

* .**parent > .child**
  + Selects only direct child elements, not nested ones.

|  |
| --- |
| <style>  .container > .item {  color: green;  }  </style>  <div class="container">  <p class="item">This is a direct child</p>  <div>  <p class="item">This is a nested child (won't get styled)</p>  </div>  </div> |

Only the first .item inside .container gets the style.

* **:nth-child(n)**
  + Targets the n-th element inside a parent.

|  |
| --- |
| <style>  .box:nth-child(2) {  background-color: pink;  }  </style>  <div class="box">Box 1</div>  <div class="box">Box 2 (this one is styled)</div>  <div class="box">Box 3</div> |

**:nth-child(odd)** for odd elements, **:nth-child(even)** for even elements

* **:first-child**
  + Selects the first child element inside a parent.

|  |
| --- |
| <style>  .list-item:first-child {  color: orange;  font-weight: bold;  }  </style>  <ul>  <li class="list-item">Item 1 (styled)</li>  <li class="list-item">Item 2</li>  </ul> |

Only the very first .list-item gets styled.

|  |  |
| --- | --- |
| **Selector** | **What It Does** |
| .class::first-line | Styles the first visible line |
| .class::first-letter | Styles the first letter |
| .parent > .child | Styles only direct child elements |
| :nth-child(n) | Styles the n-th child in order |
| :first-child | Styles the first child |

* **> — Child Selector (Direct Child)**
  + Selects elements that are direct children of a parent.

|  |
| --- |
| .parent > .child {  color: red;  }  .container>h6  {  background-color: aqua;  } |

This means: “Select .child elements that are directly inside .parent only.”

|  |
| --- |
| <style>  .box > p {  color: green;  }  </style>  <div class="box">  <p>This is a direct child. ✅</p>  <div>  <p>This is a nested child. ❌</p>  </div>  </div> |

Only the first <p> is styled because it's a direct child.

* **~ — General Sibling Selector**
  + Selects elements that are siblings (same level), appearing after the target element.

|  |
| --- |
| .title ~ p {  color: blue;  } |

This means: “Select all <p> siblings that come after .title.”

|  |
| --- |
| <style>  .heading ~ p {  color: blue;  }  </style>  <h2 class="heading">Heading</h2>  <p>Paragraph 1 ✅</p>  <p>Paragraph 2 ✅</p>  <div>  <p>Nested Paragraph ❌</p>  </div> |

Both <p> elements after .heading are styled because they're siblings.

|  |  |  |
| --- | --- | --- |
| **Selector** | **Called...** | **Selects...** |
| > | Child selector | Direct children only |
| ~ | General sibling selector | All next siblings (same level) that match the selector |

* **Bonus Tip:**
  + If you want to select only the next sibling (not all), you can use:

|  |
| --- |
| .element + .sibling {  /\* Immediate next sibling \*/  } |

Image gallery assignment

11 April 2025

**Attribute selector** – In HTML (and CSS), an attribute selector is used to select elements based on the presence or value of an attribute. This is especially useful in CSS to style elements that have specific attributes.

|  |
| --- |
| Html  <img src="images/picture2.jpg" alt="Photo" title="photo">  <h1 id="h1">Heading</h1>  Css  img[title="photo"]  {  border: 2px solid red;  }  h1[id="h1"]  {  color: purple;  text-shadow: 2px 2px 3px red;  } |

Flexbox:

Flexbox is a layout system in CSS that helps you easily arrange items (like boxes) in rows or columns, align them, space them out, or center them — all without using complex code.

How to Use Flexbox? You need 2 things:

* A container (parent)
* Some items (children inside the container)

|  |
| --- |
| <style>  .container {  display: flex; /\* Activates Flexbox \*/  }  .item {  background: lightblue;  padding: 20px;  margin: 10px;  }  </style>  <div class="container">  <div class="item">Box 1</div>  <div class="item">Box 2</div>  <div class="item">Box 3</div>  </div> |

Important Flexbox Attributes (with meanings)

On the Container (Parent)

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Example |
| display: flex; | Turns on Flexbox | .container { display: flex; } |
| flex-direction | Set direction: row or column | row (default), column, colum reverse, r0w-reverse |
| justify-content | Align items horizontally | center, space-between, space-around, flex-start, flex-end |
| align-items | Align items vertically | center, flex-start, flex-end, stretch |
| flex-wrap | Let items wrap to the next line | wrap, nowrap |
| Flex-wrap-reverse |  |  |

On the Items (Children)

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Example |
| flex | Grows/Shrinks item | flex: 1 makes equal size |
| order | Change item order | order: 2 (higher = comes later) |
| align-self | Individual vertical alignment | align-self: center |

**Flex wrap and it values**

In HTML/CSS, flex-wrap is a property used with Flexbox layouts to control whether the flex items should wrap onto multiple lines or stay on a single line.

🔧 flex-wrap Syntax (used in CSS):

|  |
| --- |
| .container {  display: flex;  flex-wrap: wrap;  } |

Possible Values of flex-wrap:

|  |  |
| --- | --- |
| Value | Description |
| nowrap | (Default) All flex items will be on a single line (no wrapping), even if they overflow the container. |
| wrap | Flex items will wrap onto multiple lines, from top to bottom. |
| wrap-reverse | Flex items will wrap onto multiple lines from bottom to top (reverse direction). |

**flex-direction (CSS Property)**

The flex-direction property specifies how flex items are placed in the flex container, in which direction the container stacks the items.

|  |  |
| --- | --- |
| **Value** | **Description** |
| row (default) | Items are placed horizontally from left to right. |
| row-reverse | Items are placed horizontally, but in reverse (right to left). |
| column | Items are placed vertically, from top to bottom. |
| column-reverse | Items are placed vertically, but in reverse (bottom to top). |

|  |
| --- |
| .container {  display: flex;  flex-direction: row; /\* Try changing to column, row-reverse, etc. \*/  gap: 10px;  border: 2px solid black;  padding: 10px;  } |

**justify-content (CSS Property)**

The justify-content property defines how the browser distributes space between and around content items along the main axis of a flex container (which is defined by flex-direction).

|  |  |
| --- | --- |
| **Value** | **Description** |
| flex-start (default) | Items are aligned to the start of the main axis. |
| flex-end | Items are aligned to the end of the main axis. |
| center | Items are centered along the main axis. |
| space-between | Items are evenly distributed; first at start, last at end. |
| space-around | Items are evenly distributed with equal space around them. |
| space-evenly | Items are evenly distributed with equal space between and around them. |
| start / end / left / right | Logical alignment (useful in writing modes, less common). |

|  |
| --- |
| .container {  display: flex;  justify-content: space-around; /\* Try changing this to see effect \*/  border: 2px solid black;  padding: 10px;  } |

Tips:

* Works along the main axis → horizontal by default (flex-direction: row), vertical if flex-direction: column.
* Combine with align-items to control the cross axis (like vertical alignment when flex-direction is row).

**align-items (CSS Property)**

The align-items property controls how flex items are aligned along the cross axis (perpendicular to the main axis).

* If flex-direction: row, cross axis = vertical
* If flex-direction: column, cross axis = horizontal

|  |  |
| --- | --- |
| **Value** | **Description** |
| stretch (default) | Items stretch to fill the container (if no fixed height/width is set). |
| flex-start | Items align to the start of the cross axis. |
| flex-end | Items align to the end of the cross axis. |
| center | Items are centered along the cross axis. |
| baseline | Items align based on their text baselines. |

|  |
| --- |
| .container {  display: flex;  height: 200px;  align-items: flex-end; /\* Try center, stretch, etc. \*/  border: 2px solid black;  } |

Tips:

* align-items works on all flex items inside a flex container.
* If you want to align a single item differently, use align-self on that specific item.

**flex-grow (CSS Property)**

The flex-grow property defines how much a flex item should grow relative to the rest of the flex items inside a flex container when extra space is available.

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 (default) | Item does not grow; it stays its original size. |
| 1 | Item can grow and will take up a proportional share of the extra space. |
| n (any positive number) | The item will grow n times more than other items with lower values. |

It doesn't use percentages — it’s a unitless proportion.

|  |
| --- |
| .container {  display: flex;  border: 2px solid black;  } |

How It Works:

If you have flex-grow: 1, 2, and 1 — the total grow factor is 4, and space is divided like this:

* Item 1 → 1/4 of extra space
* Item 2 → 2/4 (or 1/2) of extra space
* Item 3 → 1/4 of extra space

**flex-shrink (CSS Property)**

The flex-shrink property defines how much a flex item should shrink relative to the other items in the same flex container, when the container is too small to fit all items.

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | The item will not shrink, even if there's not enough space. |
| 1 (default) | The item will shrink if needed, proportionally with others. |
| n (any positive number) | The item will shrink n times more compared to items with a lower value. |

Like flex-grow, flex-shrink is unitless and works in proportions.

|  |
| --- |
| .container {  display: flex;  width: 300px;  border: 2px solid black;  }  .item1 {  flex-shrink: 1;  background-color: lightpink;  }  .item2 {  flex-shrink: 2;  background-color: lightgreen;  } |

**How It Works:**

If the container can’t hold all items, they shrink based on their flex-shrink values.

Example:

* flex-shrink: 1, 2, 1
* Total shrink factor = 4
* Item 2 will shrink twice as much as item 1 or 3.

**Tips:**

* flex-shrink works with flex-basis and width.
* To prevent shrinking, set flex-shrink: 0

**Grid**

CSS Grid allows you to design web layouts using a grid structure, dividing the page into rows and columns. It’s especially helpful for creating responsive designs and complex layouts.

|  |
| --- |
| .grid-container {  display: grid;  } |

**Common CSS Grid Properties, Attributes & Values:**

**display: grid;** Activates grid layout on the container.

**grid-template-columns**

Defines how many columns and their widths.

|  |
| --- |
| grid-template-columns: 100px 100px 100px; /\* 3 fixed-width columns \*/  grid-template-columns: 1fr 2fr; /\* 2 columns, 1:2 ratio \*/  grid-template-columns: repeat(3, 1fr); /\* 3 equal columns \*/ |

**grid-template-rows**

Defines how many rows and their heights.

|  |
| --- |
| grid-template-rows: 100px 200px; |

**gap (or grid-gap)**

Sets space between rows and columns.

|  |
| --- |
| gap: 10px; /\* gap between both rows & columns \*/  row-gap: 20px; /\* only row spacing \*/  column-gap: 15px; /\* only column spacing \*/ |

**grid-column & grid-row**

Used on child items to define their position and span.

|  |
| --- |
| .item1 {  grid-column: 1 / 3; /\* span from column 1 to 2 (end before 3) \*/  grid-row: 1 / 2;  } |

**justify-items & align-items**

Align items inside grid cells (like text-align, but for blocks).

|  |
| --- |
| justify-items: center; /\* horizontal alignment \*/  align-items: center; /\* vertical alignment \*/ |

**justify-content & align-content**

Align the entire grid inside the container.

|  |
| --- |
| justify-content: space-between;  align-content: center; |

**place-items and place-content**

Shorthand for align-items + justify-items and align-content + justify-content.

|  |
| --- |
| place-items: center; /\* center both horizontally and vertically \*/  place-content: space-between center; |

Pro Tips:

* Use fr units for flexible layouts (1fr, 2fr, etc.).
* Use minmax() for responsive design.
* Combine with media queries for full responsiveness.

**grid-column-start & grid-column-end**

These properties tell where a grid item starts and ends horizontally within a grid container.

**Grid Lines Explained:**

Imagine a grid like this (lines between columns):

|  |
| --- |
| | 1 | 2 | 3 | 4 | |

* The lines between columns are numbered: 1, 2, 3, 4, ...
* You place items between those lines.

|  |
| --- |
| .item {  grid-column-start: 1;  grid-column-end: 3;  } |

This means:

* Start at grid line 1
* End before grid line 3
* So the item spans columns 1 and 2

**Shortcut: grid-column**

Instead of writing both, you can do this:

|  |
| --- |
| .item {  grid-column: 1 / 3;  } |

Or to span columns dynamically:

|  |
| --- |
| .item {  grid-column: span 2;  } |

|  |  |
| --- | --- |
| Property | Purpose |
| grid-column-start | Grid line number to start at |
| grid-column-end | Grid line number to end before |
| grid-column: x / y | Shorthand for start and end |
| grid-column: span n | Span across n columns |

**14 April 2025**

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

is an HTML meta tag that helps control how your webpage appears on mobile devices and responsive screens.

**Explanation**:

* **<meta>:** This tag provides metadata about the HTML document (data about data), which isn’t displayed on the page but is used by browsers and search engines.
* **name="viewport":** This specifies that the meta tag is setting properties for the viewport, which is the visible area of a web page on a screen.
* **content="width=device-width, initial-scale=1.0":**
  + **width=device-width:** This sets the width of the page to match the screen-width of the device (phone, tablet, etc.).
  + **initial-scale=1.0:** This sets the initial zoom level when the page is loaded. A scale of 1.0 means the page is shown at its normal size.

**Why it's important:**

Without this tag, mobile browsers might render your page at a desktop width, making it appear tiny and hard to read. This tag is essential for responsive web design.

**Media**

In HTML and CSS, the media concept is mostly used in the context of responsive design, especially with the <style> tag or when linking external stylesheets using <link>. It's not a tag itself in HTML, but rather an attribute used in the <link> and <style> tags, and in CSS media queries.

1. <link> Tag with media Attribute

|  |
| --- |
| <link rel="stylesheet" href="desktop.css" media="screen and (min-width: 1024px)">  <link rel="stylesheet" href="mobile.css" media="screen and (max-width: 600px)"> |

**Attributes**:

* media: Specifies what media/device the styles are designed for.

**Common Values:**

* all – Default. Suitable for all devices.
* screen – Computer screens, tablets, smartphones.
* print – Styles used when printing the document.
* speech – For screen readers.

You can also combine conditions:

* media="screen and (min-width: 768px)"
* media="only screen and (max-width: 480px)"

2. CSS Media Queries (Inside <style> or CSS file)

|  |
| --- |
| <style>  body {  background-color: white;  }  @media screen and (max-width: 600px) {  body {  background-color: lightblue;  }  }  </style> |

In this example, the background color will change to light blue when the screen width is 600px or less.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Attribute** | **Example Value** | **Use** |
| <link> | media | "screen and (max-width: 600px)" | Apply stylesheet only on small screens |
| <style> | Used with Used with `@media | @media print { ... } | Apply CSS only when printing |

|  |
| --- |
| @media screen and (max-width:1200px)  {  body  {  background-color: aquamarine;  }  }  @media screen and (max-width:600px)  {  body  {  background-color:blueviolet;  }  } |

|  |  |  |
| --- | --- | --- |
| **Property** | **Direction** | **Meaning** |
| max-width | High → Low | Target screens smaller than or equal to a value |
| min-width | Low → High | Target screens larger than or equal to a value |

|  |
| --- |
| <meta name="description" content="this is my desc">  <meta name="keyword" content="html, css, javascript">  <meta name="author" content="ITVendant"> |

**Explanation:**

1. **<meta name="description" content="this is my desc">**
   * Purpose: Describes the summary of your web page.
   * Search Engines like Google may show this text in the search results under the page title.
   * Example use:

|  |
| --- |
| <meta name="description" content="Learn HTML, CSS, and JavaScript from scratch."> |

1. **<meta name="keyword" content="html, css, javascript">**
   * **Purpose:** Lists keywords related to the content of your page.
   * Used to tell search engines what your page is about.
   * Note: Most modern search engines (like Google) no longer use this tag for SEO ranking, but it's okay to include.
2. <meta name="author" content="ITVendant">
   * **Purpose**: Defines the author of the web page.
   * Useful for documentation and may show up in some browser tools or source viewers.

|  |  |  |
| --- | --- | --- |
| **Meta Tag** | **Purpose** | **Example Value** |
| description | Page summary (for SEO/search preview) | "This is a tutorial site." |
| keyword | Keywords related to content | "html, css, javascript" |
| author | Who wrote the page | "ITVendant" |

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="description" content="Learn HTML, CSS, and JavaScript">  <meta name="keyword" content="html, css, javascript">  <meta name="author" content="ITVendant">  <title>My Web Page</title>  </head>  <body>  <h1>Welcome to My Web Page</h1>  </body>  </html> |

**Bootstrap 5**

**Bootstrap 5 is the latest version of Bootstrap, a free and open-source CSS framework used to build responsive and mobile-first websites quickly and easily.**

**It comes with:**

* Ready-made CSS styles
* JavaScript components like modals, sliders, dropdowns
* A powerful grid system for layouts
* Responsive design tools

**What’s new in Bootstrap 5?**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| ❌ No jQuery | Uses only pure JavaScript (lighter & faster) |
| 📱 Improved Grid | Enhanced for better responsive design |
| 🌙 Dark Mode Ready | Easier to implement dark themes |
| 🧩 Utility Classes | More helper classes (e.g., margin, padding, flexbox) |
| 📦 Updated Components | Buttons, Forms, Navbar, Cards, etc. look better |

**How to Add Bootstrap 5 to Your HTML**

Add this in your <head> tag to use Bootstrap 5 via CDN:

|  |
| --- |
| <!-- Bootstrap 5 CSS -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">  <!-- Bootstrap 5 JS (optional, for interactive components) -->  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script> |

Bootstrap Grid System Example

|  |
| --- |
| <div class="container">  <div class="row">  <div class="col-md-6">Column 1</div>  <div class="col-md-6">Column 2</div>  </div>  </div> |

* **container**: Centers and adds padding to your content
* **row**: A horizontal group of columns
* **col-md-6**: Means 6 columns (out of 12) for medium and larger screens

Example: Bootstrap Button

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

* **btn**: Basic button style
* **btn-primary**: Blue color button (you can use btn-success, btn-danger, etc.)

Common Components in Bootstrap 5

|  |  |
| --- | --- |
| **Component** | **Example Class** |
| Button | btn btn-warning |
| Alert | alert alert-success |
| Navbar | navbar navbar-light |
| Card | card card-body |
| Modal | modal (requires JS) |
| Grid Layout | row, col-md-6 |

Example Full Page with Bootstrap 5

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Bootstrap 5 Example</title>  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">  </head>  <body>  <div class="container mt-5">  <h1 class="text-center text-primary">Welcome to Bootstrap 5</h1>  <button class="btn btn-success">Click Here</button>  </div>  </body>  </html> |

**15 april 2025**

**Bootstrap 5 Components Table**

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Meaning / Use** | **Example Code** |
| Alert | Displays feedback messages (success, warning, etc.) | <div class="alert alert-success">Success!</div> |
| Button | Creates clickable buttons | <button class="btn btn-primary">Click Me</button> |
| Card | Displays content in a structured box | <div class="card"><div class="card-body">Card Content</div></div> |
| Navbar | Responsive navigation bar | <nav class="navbar navbar-expand-lg navbar-light bg-light">Brand</nav> |
| Modal | Popup dialog box | <div class="modal">...Modal Content...</div> |
| Carousel | Image/content slider | <div class="carousel slide">...slides...</div> |
| Accordion | Expand/collapse content sections | <div class="accordion"><div class="accordion-item">...</div></div> |
| Dropdown | Toggle-able list menu | <div class="dropdown"><button class="btn dropdown-toggle" data-bs-toggle="dropdown">Menu</button></div> |
| Form | Input fields for user data | <form><input class="form-control" type="text"></form> |
| Badge | Small count/label (e.g., notifications) | <span class="badge bg-danger">5</span> |
| Tooltip | Info box on hover | <button data-bs-toggle="tooltip" title="Info">Hover me</button> |
| Progress | Shows progress (e.g., upload bar) | <div class="progress"><div class="progress-bar" style="width: 50%;"></div></div> |
| Spinner | Loading indicator | <div class="spinner-border" role="status"></div> |
| Table | Styled tables | <table class="table table-striped">...</table> |
| Toast | Small auto-hiding popup messages | <div class="toast">...Toast message...</div> |

Bootstrapmade – template

Footers - <https://frontend-snippets.com/css-snippets/css-footer/responsive-footer-html-css>

1. **line-height**

* Purpose: Controls the amount of space between lines of text.
* CSS Property: line-height
* Values:
  + Numeric (unitless): 1.5 (relative to font size)
  + Length: 20px, 1.2em, etc.
  + Percentage: 150%
* Example:

|  |
| --- |
| p {  line-height: 1.5;  } |

2**. text-indent**

Purpose: Indents the first line of a paragraph.

* CSS Property: text-indent
* Values:
  + Length: 20px, 2em, etc.
  + Percentage: 10% (relative to the containing block width)
* Example:

|  |
| --- |
| p {  text-indent: 30px;  } |

3. **word-spacing** (Seems you meant “word-Specular” but the correct term is word-spacing)

Purpose: Adjusts space between words.

* CSS Property: word-spacing
* Values:
  + Normal: normal
  + Length: 10px, 0.2em, etc.
* Example:

|  |
| --- |
| p {  word-spacing: 5px;  } |

1. **letter-spacing**

Purpose: Adjusts space between letters (also called tracking).

* CSS Property: letter-spacing
* Values:
  + Normal: normal
  + Length: 2px, 0.1em, etc.
* Example:

|  |
| --- |
| p {  letter-spacing: 1px;  } |

Example Combining All:

|  |
| --- |
| <style>  p {  line-height: 1.6;  text-indent: 2em;  word-spacing: 4px;  letter-spacing: 1px;  }  </style>  <p>This is a sample paragraph demonstrating line height, text indent, word spacing, and letter spacing.</p> |

**Client–Server Architecture** is a model used in computer networks where multiple clients (users or devices) request and receive services from a centralized server.

**Basic Concept**

* Client:

A device or application (like a web browser) that sends requests for data or services.

* Server:

A powerful system or application that provides services or resources (like websites, files, databases) in response to those client requests.

**How It Works (Flow)**

* Client sends a request → e.g., open a webpage.
* Server receives the request → processes it.
* Server sends a response → e.g., sends the HTML page.
* Client displays the result → user sees the webpage.

**Real-World Example:**

* You open Google.com in a browser (client).
* The browser sends a request to Google’s web server.
* The server processes the request and sends back the webpage.
* The browser shows the webpage to you.

**Types of Servers in This Architecture:**

* Web Server (e.g., Apache, Nginx)
* Database Server (e.g., MySQL, Oracle)
* File Server
* Mail Server

**Advantages:**

* Centralized control and management.
* Easier to update or back up.
* Scalable (can add more clients).

**Disadvantages:**

* If the server fails, all clients are affected.
* High server load with many clients.