**CSS Basics, Selectors, and Text Formatting**

**1. Introduction to CSS**

CSS (**Cascading Style Sheets**) is used to style and format HTML elements to make web pages visually appealing.

**2. Types of CSS**

There are three ways to apply CSS in HTML:

**1. Inline CSS (applied directly to an element using the style attribute)**

<p style="color: blue; font-size: 18px;">This is an inline styled paragraph.</p>

**2. Internal CSS (written inside a <style> tag in the <head> section)**

<style>

p {

color: red;

font-size: 20px;

}

</style>

**3. External CSS (written in a separate .css file and linked using <link>)**

<link rel="stylesheet" href="styles.css">

**3. CSS Syntax**

A CSS rule consists of **selectors**, **properties**, and **values**.

selector {

property: value;

}

Example:

p {

color: green;

font-size: 16px;

}

**4. Connecting CSS to HTML**

There are two main ways to link CSS to an HTML file:

1. **Inline** (Inside an element) → style="color:red;"
2. **Internal** (Inside <head> within <style> tags)
3. **External** (Using <link> to attach a .css file)

**5. CSS Selectors**

CSS selectors help in **targeting specific HTML elements** to apply styles.

**Basic Selectors**

| **Selector** | **Description** | **Example** |
| --- | --- | --- |
| \* | Selects all elements | \* { margin: 0; } |
| element | Selects all instances of a specific tag | p { color: blue; } |
| .class | Selects elements with a specific class | .box { background: yellow; } |
| #id | Selects an element with a specific ID | #header { font-size: 24px; } |

**Combinator Selectors**

| **Selector** | **Description** | **Example** |
| --- | --- | --- |
| descendant (A B) | Selects all B inside A | div p { color: red; } |
| child (A > B) | Selects direct children | div > p { color: blue; } |
| adjacent sibling (A + B) | Selects the next sibling element | h1 + p { font-weight: bold; } |
| general sibling (A ~ B) | Selects all siblings | h1 ~ p { font-style: italic; } |

Example:

div p { color: red; } /\* Targets all <p> inside <div> \*/

div > p { color: blue; } /\* Only direct child <p> inside <div> \*/

h1 + p { font-size: 18px; } /\* First <p> after <h1> \*/

h1 ~ p { text-decoration: underline; } /\* All <p> after <h1> \*/

**Attribute Selectors**

| **Selector** | **Description** | **Example** |
| --- | --- | --- |
| [type="text"] | Targets elements with type="text" | input[type="text"] { border: 1px solid black; } |
| [href^="https"] | Targets links starting with https | a[href^="https"] { color: green; } |
| [data-\*] | Targets custom data- attributes | [data-role="admin"] { background: red; } |

Example:

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

a[href^="https"] { color: green; }

[data-role="admin"] { font-weight: bold; }

**6. Text Formatting in CSS**

**Font Properties**

| **Property** | **Description** | **Example** |
| --- | --- | --- |
| font-family | Specifies font type | font-family: Arial, sans-serif; |
| font-size | Sets font size | font-size: 18px; |
| font-weight | Defines boldness | font-weight: bold; |
| font-style | Specifies italic/normal text | font-style: italic; |

Example:

p {

font-family: 'Arial', sans-serif;

font-size: 16px;

font-weight: bold;

font-style: italic;

}

**Text Properties**

| **Property** | **Description** | **Example** |
| --- | --- | --- |
| text-align | Aligns text | text-align: center; |
| text-decoration | Underline, overline, etc. | text-decoration: underline; |
| text-transform | Changes case | text-transform: uppercase; |
| line-height | Adjusts line spacing | line-height: 1.5; |

Example:

h1 {

text-align: center;

text-decoration: underline;

text-transform: uppercase;

line-height: 1.8;

}

**CSS Box Model, Display Properties, and Layout Design**

**1. Understanding the CSS Box Model**

The **CSS Box Model** is the foundation of web layout. Every element in a webpage is treated as a rectangular box with the following components:

**Box Model Components**

1. **Content**: The actual text or image inside the box.
2. **Padding**: Space between content and the border (inside the box).
3. **Border**: Surrounds the padding and content.
4. **Margin**: Space outside the border that separates elements.

**Example of Box Model in CSS**

.box {

width: 200px;

height: 100px;

padding: 20px;

border: 5px solid black;

margin: 10px;

background-color: lightblue;

}

**Total width = content width + padding + border + margin**  
**Total height = content height + padding + border + margin**

**Box Sizing: content-box vs. border-box**

By default, the **width and height apply only to the content** (content-box).  
Using border-box, width and height include **content, padding, and border**.

**Example:**

.content-box {

width: 200px;

padding: 20px;

border: 5px solid black;

box-sizing: content-box; /\* Default behavior \*/

}

.border-box {

width: 200px;

padding: 20px;

border: 5px solid black;

box-sizing: border-box; /\* Includes padding and border \*/

}

**Use box-sizing: border-box; to make layouts easier!**

**2. CSS Display Properties**

The display property defines how an element behaves in a layout.

**Types of Display Values**

| **Display Type** | **Description** |
| --- | --- |
| block | Takes full width, starts on a new line (e.g., <div>, <p>). |
| inline | Takes only the space needed, does not start a new line (e.g., <span>, <a>). |
| inline-block | Like inline, but allows setting width and height. |
| none | Hides the element completely (removes it from layout). |

**Example:**

.block-example {

display: block;

background-color: lightgreen;

}

.inline-example {

display: inline;

background-color: yellow;

}

.inline-block-example {

display: inline-block;

width: 100px;

height: 50px;

background-color: orange;

}

.none-example {

display: none;

}

**Visibility vs. Display**

| **Property** | **Effect** |
| --- | --- |
| display: none; | Removes the element from the document flow. |
| visibility: hidden; | Hides the element but keeps its space reserved. |

**Example:**

.hidden-element {

visibility: hidden;

}

.none-element {

display: none;

}

**3. Layout Design Techniques**

**Alignment and Spacing**

| **Property** | **Description** |
| --- | --- |
| margin | Controls space **outside** the element. |
| padding | Controls space **inside** the element. |
| text-align | Aligns text (left, center, right). |
| vertical-align | Aligns inline elements vertically. |

**Example:**

.aligned-text {

text-align: center;

vertical-align: middle;

}

.spacing-box {

margin: 20px auto;

padding: 15px;

}

**Overflow Management**

The overflow property controls content **when it exceeds the container size**.

| **Property** | **Effect** |
| --- | --- |
| overflow: visible; | Default. Content overflows the container. |
| overflow: hidden; | Cuts off the overflowed content. |
| overflow: scroll; | Adds a scrollbar (horizontal & vertical). |
| overflow-x, overflow-y | Controls overflow separately for width and height. |

**Example:**

.scroll-box {

width: 200px;

height: 100px;

overflow: scroll;

border: 2px solid black;

}

**CSS Positioning, Pseudo-Classes, and Flexbox**

**1. CSS Positioning**

CSS position controls **how an element is placed** on the webpage.

**Types of Positioning in CSS**

| **Position Type** | **Description** |
| --- | --- |
| static | Default positioning (normal flow). |
| relative | Moves relative to its original position. |
| absolute | Moves relative to the nearest positioned ancestor. |
| fixed | Stays fixed on the screen even when scrolling. |
| sticky | Sticks at a position when scrolling. |

**Examples of CSS Positioning**

**1. Static Position (Default)**

.static-box {

position: static; /\* Default behavior \*/

background-color: lightgray;

padding: 10px;

}

**2. Relative Positioning**

.relative-box {

position: relative;

top: 20px; /\* Moves down 20px \*/

left: 10px; /\* Moves right 10px \*/

background-color: lightblue;

}

**3. Absolute Positioning**

.absolute-box {

position: absolute;

top: 50px;

left: 100px;

background-color: orange;

}

**Requires a positioned ancestor (relative, absolute, or fixed). Otherwise, it moves relative to the <body>.**

**4. Fixed Positioning**

.fixed-box {

position: fixed;

bottom: 0;

width: 100%;

background-color: black;

color: white;

text-align: center;

padding: 10px;

}

**Used for sticky navigation bars and floating buttons.**

**5. Sticky Positioning**

.sticky-box {

position: sticky;

top: 0;

background-color: yellow;

padding: 10px;

}

**Great for sticky headers while scrolling!**

**Use Cases of CSS Positioning**

* **Navigation bars** → fixed
* **Modals (pop-ups)** → absolute
* **Side panels** → fixed or absolute
* **Sticky headers** → sticky

**2. Pseudo-Classes and Pseudo-Elements**

**Pseudo-Classes (:)**

A **pseudo-class** applies styles **based on a specific state** of an element.

| **Pseudo-Class** | **Description** |
| --- | --- |
| :hover | When a user hovers over an element. |
| :focus | When an input field is focused. |
| :active | When an element is clicked. |
| :nth-child(n) | Selects the nth child of a parent. |

**Example of Pseudo-Classes**

.button:hover {

background-color: lightgreen;

}

input:focus {

border: 2px solid blue;

}

a:active {

color: red;

}

p:nth-child(2) {

color: darkblue;

}

**Pseudo-Elements (::)**

A **pseudo-element** allows you to style **specific parts** of an element.

| **Pseudo-Element** | **Description** |
| --- | --- |
| ::before | Inserts content **before** an element. |
| ::after | Inserts content **after** an element. |
| ::placeholder | Styles placeholder text in input fields. |

**Example of Pseudo-Elements**

h1::before {

content: "🔥 ";

}

h1::after {

content: " 🎉";

}

input::placeholder {

color: gray;

font-style: italic;

}

**Great for adding icons, decorative elements, or styling form placeholders.**

**3. Introduction to Flexbox**

Flexbox makes it easy to create **responsive layouts** with flexible elements.

**Flex Container & Flex Items**

1. **Container** → display: flex;
2. **Direction** → flex-direction: row | column
3. **Wrap** → flex-wrap: wrap | nowrap

**Example of Flex Container**

.flex-container {

display: flex;

flex-direction: row;

flex-wrap: wrap;

gap: 10px; /\* Space between items \*/

}

.flex-item {

background-color: lightblue;

padding: 20px;

border: 2px solid black;

}

**Use row for horizontal layout, column for vertical layout.**

**Alignment Techniques in Flexbox**

| **Property** | **Description** |
| --- | --- |
| justify-content | Aligns items horizontally (flex-start, center, space-between, space-around, space-evenly). |
| align-items | Aligns items vertically (flex-start, center, flex-end, stretch). |
| align-content | Aligns multiple flex lines when wrapping (stretch, center, space-between). |

**Example of Justify Content**

.flex-container {

display: flex;

justify-content: center; /\* Center horizontally \*/

}

**Example of Align Items**

.flex-container {

display: flex;

height: 200px;

align-items: center; /\* Center vertically \*/

}