**Lecture 1: 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;

}

### **Lecture 2: CSS Box Model, Display Properties, Layout Design 1. CSS Box Model – Easy Explanation**

Every HTML element is like a **box**, and this box has 4 parts:

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

**Example:**

box {

margin: 10px;

border: 1px solid black;

padding: 15px;

}

**2. Box Sizing – content-box vs border-box**

* content-box (default):  
  Width/height includes only **content**. Padding & border are **extra**.
* border-box:  
  Width/height includes **content + padding + border**. Better for layout.

**Example:**

box {

box-sizing: border-box;

}

**3. Display Properties**

These define **how elements appear** on the page:

* block: Takes full width. Starts on a **new line**.  
  (e.g., <div>, <p>)
* inline: Stays **in line** with text. No full width.  
  (e.g., <span>)
* inline-block: Like inline, but **can have width & height**.
* none: Hides the element **completely** (removed from layout).

**4. Visibility vs. Display**

* visibility: hidden: Element is **invisible**, but space is still there.
* display: none: Element is **not shown**, and takes **no space**.

**5. Layout Design Techniques**

**a. Alignment & Spacing:**

* margin: Space **outside** the element.
* padding: Space **inside**, around content.
* text-align: Align text (**left, right, center**).
* vertical-align: Align inline/inline-block elements **vertically**.

**b. Overflow:**

Sometimes content is bigger than the box.

* overflow: Controls what happens.
  + hidden – Hides extra.
  + scroll – Adds scrollbars.
  + auto – Adds scroll **only if needed**.
  + visible – Shows extra content.

**6. Hands-on Exercise**

**Task:** Create a simple layout using box model and display properties.

**Example HTML + CSS:**

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

.box {

width: 200px;

padding: 20px;

border: 2px solid blue;

margin: 30px;

display: block;

text-align: center;

box-sizing: border-box;

}

**Lecture 3: CSS Positioning, Pseudo-Classes, and Flexbox**

**1. CSS Positioning**

CSS positioning helps us decide **where an element appears** on the page.

**Types of Positioning:**

1. **static** – This is the default. The element stays in the normal flow of the page.  
   *(No special position is applied.)*
2. **relative** – The element moves **relative to its normal position**.  
   *(Example: top: 10px moves it 10px down.)*
3. **absolute** – The element is placed **relative to the nearest positioned parent**.  
   *(Used for modals or dropdowns.)*
4. **fixed** – The element stays **in the same place on the screen**, even when scrolling.  
   *(Perfect for sticky navbars.)*
5. **sticky** – The element behaves like relative first, then becomes fixed when scrolling.  
   *(Used for headers that stick at the top while scrolling.)*

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

These are **special keywords** that let you style **specific parts or states** of elements.

**Common Pseudo-Classes (:)**

1. :hover – When the mouse is over an element.  
   *(Good for buttons or links.)*
2. :focus – When an input box is active (clicked or typed into).  
   *(Useful for styling forms.)*
3. :active – When the element is being clicked.  
   *(Often used with buttons.)*
4. :nth-child(n) – Selects a specific child element by number.  
   *(Example: 2nd paragraph in a list.)*

**Pseudo-Elements (::)**

1. ::before – Adds content **before** the element.  
   *(Like an icon or decoration.)*
2. ::after – Adds content **after** the element.  
   *(Useful for visual effects.)*
3. ::placeholder – Styles the placeholder text inside input boxes.

**3. Introduction to Flexbox**

Flexbox is a layout model that makes it **easy to design flexible and responsive layouts**.

**Flex Container**

You make a container flexible by using:

display: flex;

You can then use:

* flex-direction: row (default) or column  
  *(Sets direction of items.)*
* flex-wrap: wrap or nowrap  
  *(Wrap items to the next line if needed.)*

**Flex Alignment**

* justify-content: Align items **horizontally**  
  *(start, center, space-between, etc.)*
* align-items: Align items **vertically**  
  *(top, center, bottom alignment)*
* align-content: Align multiple rows if wrapping is used

**4. Practical Activity**

**Build a responsive navigation bar using Flexbox and pseudo-classes.**