**Web Technology CSS (Cascading Stylesheet)**

**2.1 How CSS Fits with HTML Page**

- CSS (Cascading Style Sheets) is used to style HTML elements.
- It controls layout, colors, fonts, and overall appearance.
- CSS can be added to HTML in 3 ways: Inline, Internal, and External.

**2.2 Inline, Internal, and External CSS**

**1. Inline CSS:** Written directly inside HTML tags using the style attribute.

Example:

** This is red text. **

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple CSS Example</title>
</head>
<body>
  <h1 style="color: red; text-align: center;">Hello World!</h1>
  <p style="font-family: Arial; background: yellow; padding: 10px;">
    This is a simple paragraph with inline CSS.
  </p>
  <div style="border: 2px dashed blue; width: 200px; margin: auto;">
    <p style="text-align: center;">A box with a dashed border</p>
  </div>
</body>
</html>
```

**2. Internal CSS:** Defined within <style> tags in the <head> section of HTML.

Example:

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Internal CSS Example</title>
  <style>
    /* Internal CSS starts here */
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      margin: 0;
      padding: 20px;
```

```
    }

    h1 {
      color: #0066cc;
      text-align: center;
      text-decoration: underline;
    }

    .box {
      background-color: white;
      border: 2px solid #0066cc;
      border-radius: 8px;
      padding: 15px;
      margin: 10px auto;
      width: 80%;
      box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    }

    .highlight {
      color: #ff6600;
      font-weight: bold;
    }

    #special {
      font-style: italic;
      background-color: #fffacd;
      padding: 5px;
    }
  </style>
</head>
<body>
  <h1>Welcome to My Webpage</h1>

  <div class="box">
    <p>This is a paragraph inside a styled box.</p>
    <p class="highlight">This text is highlighted!</p>
    <p id="special">This paragraph has a special ID style.</p>
  </div>
</body>
</html>
```

**3. External CSS:** Stored in a separate .css file and linked using <link>.

Example:

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

**2.3 CSS Selectors:**
Used to select HTML elements for styling.
Common selectors:

- Element Selector (p, h1, div)
- Class Selector (.classname)
- ID Selector (#idname)
- Universal Selector (*)
- Grouping Selector (h1, p, .class)

**2.4 CSS Properties for Styling**

- Text: color, font-family, font-size, text-align
- List: list-style-type, list-style-image
- Table: border-collapse, padding, text-align
- Background: background-color, background-image
- Link: a:link, a:visited, a:hover, a:active

**2.5 Pseudo Classes**
Special states of elements:

- :hover (when mouse hovers)
- :focus (when element is selected)
- :active (when clicked)
- :first-line, :first-letter (styling first line/letter)
- :before, :after (insert content before/after element)

## 1. `:hover` (When the mouse hovers over an element)

:hover pseudo-class is applied when the mouse pointer hovers over an element

```
<html>
<head>

  <title>:hover Example</title>
  <style>
    a:hover {
      color: red; /* Change text color to red when hovering over the link */
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <a href="#">Hover over me!</a>
</body>
</html>
```

## 2. `:focus` (When an element is selected, typically with the keyboard or mouse)

:focus pseudo-class is used when an element, like a form input, gains focus (e.g., clicked or tabbed into).

```
<html>
<head>

  <title>:focus Example</title>
  <style>
    input:focus {
      background-color: lightyellow; /* Change background color when input is focused */
      border-color: blue; /* Change border color when focused */
    }
  </style>
</head>
<body>
  <input type="text" placeholder="Click or Tab to Focus">
</body>
</html>
```

## 3. `:active` (When an element is being clicked or activated)

:active pseudo-class is applied when an element is in the process of being clicked, such as during a mouse press.

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

  <title>:active Example</title>
  <style>
    button:active {
      background-color: darkgreen; /* Change button color when clicked */
      color: white;
    }
  </style>
</head>
<body>
  <button>Click me!</button>
</body>
</html>
```

## 4. `:first-line`, `:first-letter` (Styling the first line/letter of an element)

These pseudo-elements allow you to target and style the first line or letter of a block of text.

```
<html>
<head>
  <title>:first-line and :first-letter Example</title>
  <style>
```

```
    p:first-line {
      font-weight: bold; /* Make the first line bold */
      color: darkblue;
    }

    p:first-letter {
      font-size: 2em; /* Make the first letter bigger */
      color: red;
    }
  </style>
</head>
<body>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
</body>
</html>
```

## 5. `:before`, `:after` (Insert content before or after an element)

These pseudo-elements are used to insert content before or after the content of an element.

```
<html>
<head>
  <title>:before and :after Example</title>
  <style>
    p::before {
      content: "Note: "; /* Insert "Note: " before the paragraph */
      font-weight: bold;
      color: green;
    }

    p::after {
      content: " [End of text]"; /* Insert " [End of text]" after the paragraph */
      font-style: italic;
    }
  </style>
</head>
<body>
  <p>This is an example of :before and :after pseudo-elements.</p>
</body>
</html>
```

**2.6 Custom List Numbering using content Property:** The `content` property in CSS allows us to create custom numbering for lists using `::before` pseudo-elements.

## Steps to Implement:

1. Reset the default numbering using `list-style-type: none;`.
2. Use `counter-reset` to create a custom counter.
3. Use `counter-increment` to increase the counter for each `<li>`.
4. Use `::before` to insert custom numbering.

Used with ::before to customize list markers.

```
<html>
<head>

  <title>Custom List Numbering</title>
  <style>
    /* Reset default numbering */
    ol {
      counter-reset: my-counter;
      list-style-type: none;
    }

    /* Increase counter for each list item */
    ol li {
      counter-increment: my-counter;
    }

    /* Insert custom numbering */
    ol li::before {
      content: "Step " counter(my-counter) ": ";
      font-weight: bold;
      color: red;
    }
  </style>
</head>
<body>

  <h2>Steps to Follow: for adding two numbers in c</h2>
  <ol>
    <li>Start</li>
    <li>Enter two User input numbers .</li>
    <li>add x+y</li>
    <li>store add to result variable</li>
    <li>Display result</li>
    <li>End</li>
  </ol>
</body>
</html>
```

#### 2.7 CSS Box Model

The **CSS Box Model** describes how elements are structured and spaced on a webpage. Every HTML element is treated as a rectangular box with the following components:

## Box Model Components:

1. **Content** → The actual text or image inside the element.
2. **Padding** → Space between the content and the border.
3. **Border** → The outer edge surrounding the padding (can be styled).
4. **Margin** → Space between the element and other elements.

Example:

```
<html>
<head>

   <title>CSS Box Model</title>
   <style>
      .box {
         width: 200px;
         height: 100px;
         padding: 20px;
         border: 5px solid blue;
         margin: 30px;
         background-color: lightgray;
      }
   </style>
</head>
<body>

   <div class="box">This is a Box Model Example</div>

</body>
</html>
```

**2.8 Creating Layouts with display, position, and float**

## 1. `display` Property (Controls How Elements are Shown)

The `display` property defines how elements behave in a layout.

| **`display` Type** | **Description** |
|---|---|
| `block` | Takes full width, starts on a new line (e.g., `<div>`, `<p>`). |
| `inline` | Takes width as per content, stays in line (e.g., `<span>`, `<a>`). |
| `inline-block` | Like `inline`, but allows width & height. |
| `flex` | Creates flexible box layouts. |
| `grid` | Creates grid-based layouts. |
| `none` | Hides the element. |

**display:**

- block (takes full width, e.g., div, p)
- inline (takes only needed space, e.g., span, a)
- flex (flexible layout)

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Display Flex Example</title>
  <style>
    .container {
      display: flex;
      justify-content: space-between; /* Space between items */
      background-color: #ddd;
      padding: 20px;
    }

    .box {
      width: 100px;
      height: 100px;
      background-color: lightblue;
      text-align: center;
      line-height: 100px;
      font-weight: bold;
      margin: 5px;
    }
  </style>
</head>
<body>

  <div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
  </div>

</body>
</html>
```

**2. `position` Property (Controls Element Placement):** The `position` property defines how an element is positioned in the document.

| **`position` Type** | **Description** |
|---|---|
| `static` | Default positioning (normal flow). |
| `relative` | Moves relative to its normal position. |
| `absolute` | Positioned relative to the nearest **non-static** parent. |
| `fixed` | Stays in place even when scrolling. |
| `sticky` | Sticks to a position when scrolling. |

**position:**

- static (default)
- relative (adjusts from normal position)
- absolute (relative to nearest positioned parent)
- fixed (stays in place on scroll)

Example:

```
<html>
<head>
  <title>Position Sticky Example</title>
  <style>
    .header {
      position: sticky;
      top: 0;
      background: yellow;
      padding: 15px;
      font-size: 20px;
      text-align: center;
      font-weight: bold;
    }

    .content {
      height: 1500px; /* Extra height to enable scrolling */
      padding: 20px;
    }
  </style>
</head>
<body>

  <div class="header">Sticky Header</div>
  <div class="content">
    <p>Scroll down to see the sticky effect.</p>
    <p>More content...</p>
  </div>
</body>
</html>
```

**3. `float` Property (Align Elements Side-by-Side):** The `float` property allows text and elements to wrap around an element.

| **`float` Type** | **Description** |
|---|---|
| `left` | Aligns element to the left. |
| `right` | Aligns element to the right. |
| `none` | Default, no floating. |

**float:**

- Makes elements float left/right (float: left;).

Example:

```
<html>
<head>
  <title>Float Example</title>
  <style>
    .left {
      float: left;
      width: 50%;
      background: lightblue;
      padding: 20px;
    }

    .right {
      float: right;
      width: 50%;
      background: lightgreen;
      padding: 20px;
    }

    .clearfix::after {
      content: "";
      display: block;
      clear: both;
    }
  </style>
</head>
<body>

  <div class="clearfix">
    <div class="left">Left Column</div>
    <div class="right">Right Column</div>
  </div>

</body>
</html>
```

**2.9 Fixed and Liquid (Fluid) Design:** Website layouts are mainly of two types: **Fixed Design** and **Liquid (Fluid) Design**

**1. Fixed Design (Fixed-Width Layout)**

- Uses a fixed width in `px` (pixels).
- The layout **does not change** when resizing the browser window.
- Ideal for **desktop-focused** designs.
- Example: `width: 1200px;`

Example:

```
<html>
<head>
  <title>Fixed Layout</title>
  <style>
    .container {
      width: 1200px; /* Fixed width */
      margin: 0 auto; /* Center the layout */
      background: lightblue;
      padding: 20px;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>Fixed Width Layout</h2>
    <p>This layout stays the same size even when resizing the browser.</p>
  </div>
</body>
</html>
```

**2. Liquid (Fluid) Design**

- Uses **percentage (%)** instead of pixels for width.
- The layout **adjusts dynamically** when resizing the browser window.
- Ideal for **responsive designs** that adapt to all screen sizes.
- Example: `width: 90%;`

**Example:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Liquid Layout</title>
  <style>
    .container {
      width: 90%; /* Fluid width */
      margin: 0 auto;
      background: lightgreen;
      padding: 20px;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>Liquid (Fluid) Layout</h2>
    <p>This layout resizes dynamically based on screen size.</p>
  </div>
</body>
</html>
```