Unit-5
Cascading Style Sheet (CSS)
(Marks: 10)

CTEVT Diploma in Computer Engineering
Subject : Web Technology I
Prepared By: Er. Abinash Adhikari

Shree Sarwajanik Secondary Technical School

1. Introduction to Cascading Style Sheets (CSS)

- CSS (Cascading Style Sheets) is a style sheet language used to control the appearance and layout of HTML elements on a web page. It separates content (HTML) from design, enabling cleaner and more maintainable code.
- Advantages of CSS:
  - Reusability: Styles can be reused across multiple web pages.
  - Consistency: Ensures consistent design and formatting.
  - Flexibility: Enables easy customization and responsive designs.
  - Separation of Concerns: Keeps structure (HTML) and style (CSS) separate.

2. Basic Syntax

- Selector: Identifies the HTML element to style.
- Declaration Block: Contains property-value pairs inside {}.
  1. Property: Specifies the style to apply (e.g., color, font-size).
  2. Value: Defines the desired appearance.
- Example:

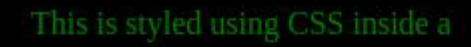
```
selector {
    property: value;
}
```

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

Creating CSS Using <style> Tag

CSS can be written directly within the <style> tag in the <head> section of an HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
      body {
          background-color: black;
           }
        p {
            color: green;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <p>This is styled using CSS inside a <style> tag.</p>
</body>
</html>
```



3. Types of Style Sheets

- There are three types of stylesheet.
  1. Inline Style Sheets
     - Inline CSS applies directly to an HTML element using the style attribute.
  2. Internal/Embedded Style Sheets
     - CSS is defined in the <style> tag within the HTML document.
  3. External Style Sheets
     - CSS is written in a separate file and linked using the <link>

1. Inline Style Sheets

Example:

```
<p style="color: red; font-size: 20px;">This is styled using inline CSS.</p>
```

2. Internal/Embedded Style Sheets

```
<head>
    <style>
        h2 {
            text-align: center;
            color: green;
        }
    </style>
</head>
```

3. External Style Sheets

style.css

```
h3 {
    color: blue;
    font-style: italic;
}
```

index.html

```
<link rel="stylesheet" href="styles.css">
<h3>This is styled using external CSS.</h3>
```

4. Different Styles and Their Attributes

1. Backgrounds and Color Styles and Attributes
2. Fonts and Text Styles and Attributes
3. Margin, Padding and Border Styles and Attributes
4. List Styles and Table Layouts
5. Additional Features - Grouping Style Sheets, Assigning Classes and Span
6. DIV Tag
7. Responsive Web Design

1. Backgrounds

Background styles in CSS control the appearance behind elements. They include properties for color, images, and their placement.

Background Properties:

1. background-color:
   - ➢ Sets the background color of an element.
     Example:

```
div {
  background-color: lightblue;
}
```

2. background-image:
   - ➢ Applies an image as the background.
     Example:

```
div {
  background-image: url('background.jpg');
}
```

3. Background-repeat:
   - ➢ Defines how the background image is repeated.
   - ➢ Values:
     - repeat (default): Repeats the image both horizontally and vertically.
     - no-repeat: No repetition.
     - repeat-x: Repeats horizontally.
     - repeat-y: Repeats vertically.

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

4. background-size:
   - Adjusts the size of the background image.
   - Values:
     i. auto: Default size.
     ii. cover: Scales the image to cover the element.
     iii. contain: Scales the image to fit within the element.

```
div {
  background-size: cover;
}
```

5. background-position:
   - Specifies the starting position of the background image.
   - Values:
     i. top, center, bottom, left, right, or a combination like top left.

```
div {
  background-position: center;
}
```

6. background-attachment:
   - Determines whether the background image scrolls with the page or remains fixed.
   - Values:
     i. scroll: Moves with the content (default).
     ii. fixed: Stays in place.

```
div {
  background-attachment: fixed;
}
```

7. Shorthand Property:
   - Combine multiple background properties in one line.

```
background: [color] [image] [repeat] [attachment] [position];
```

```
div {
  background: lightblue url('bg.jpg') no-repeat fixed center;
}
```

2. Color Styles

CSS offers various ways to define and manipulate colors for elements.

Color Properties:

1. color
   a. Sets the text color of an element.

```
p {
  color: #ff0000; /* red */
}
```

2. opacity
   a. Controls the transparency of an element.
   b. Value: A number between 0 (fully transparent) and 1 (fully opaque).

```
div {
  opacity: 0.5; /* 50% transparent */
}
```

3. background-color
   a. Used for defining the background color of an element.

```
section {
  background-color: rgba(255, 0, 0, 0.5); /* Semi-transparent red */
}
```

4. Gradient Colors
    a. Adds smooth transitions between colors.
    b. Types:
        i. Linear Gradient:

```
div {
  background: linear-gradient(to right, red, yellow);
}
```

        ii. Radial Gradient:

```
div {
  background: radial-gradient(circle, red, yellow);
}
```

** Color Formats **

1. Named Colors:
   a. Simple predefined color names.
2. HEX Colors:
   a. Use hexadecimal notation for colors.
3. RGB and RGBA:
   a. Define colors using red, green, and blue values.
   b. RGBA includes transparency.
4. HSL and HSLA:
   a. Define colors using hue, saturation, and lightness.
   b. HSLA adds transparency.

```
h1 {
  color: blue;
}
```

```
h1 {
  color: #ff5733;
}
```

```
div {
  background-color: rgba(0, 255, 0, 0.3); /* Semi-transparent green */
}
```

```
h1 {
  color: hsl(120, 100%, 50%);
}
```

Fonts and Text Styles in CSS

- Fonts and text styles in CSS are essential for controlling the appearance of textual content on a web page.
- They allow you to define font families, sizes, styles, weights, decorations, and more, ensuring a visually appealing and readable layout.

Fonts and Text Styles in CSS

1. Font Properties	2. Text Style and Attributes	3. Typography Techniques
1. Font Family (font-family) 2. Font Size (font-size) 3. Font Weight (font-weight) 4. Font Style (font-style) 5. Font Variant (font-variant) 6. Shorthand Property (font)	1. Text Color (color) 2. Text Alignment (text-align) 3. Text Decoration (text-decoration) 4. Line Height (line-height) 5. Letter Spacing (letter-spacing) 6. Word Spacing (word-spacing) 7. Text Shadow (text-shadow) 8. Text Transform (text-transform) 9. White Space (white-space)	1. Responsive Typography 2. Web Fonts 3. Fallback Fonts

1.1 Font Family (font-family)

a. Specifies the typeface to be used.

b. You can define multiple font options, separated by commas, as fallbacks in case the first font is unavailable.

c. Syntax:

```
font-family: "Times New Roman", Arial, sans-serif;
```

d. Types of Fonts:

   i. Serif (e.g., Times New Roman, Georgia): Fonts with decorative strokes.

   ii. Sans-serif (e.g., Arial, Helvetica): Clean and modern fonts.

   iii. Monospace (e.g., Courier New, Consolas): Each character has equal width.

1.2 Font Size (font-size)

a. Defines the size of the text.

b. Units:

    i. Absolute: px, pt, etc. (e.g., 16px).

    ii. Relative: %, em, rem, vw, vh (e.g., 1.2em or 100%).

c. Syntax:

```
font-size: 18px;
```

1.3 Font Weight (font-weight)

1. Controls the boldness of the text.
2. Values:
    a. Keywords: normal, bold, lighter, bolder.
    b. Numeric: 100 to 900 (e.g., 400 for normal, 700 for bold).
3. Syntax:

```
font-weight: bold;
```

1.4 Font Style (font-style)

1. Defines the style of the font (e.g., normal, italic).
2. Values:
    a. normal: Default text style.
    b. italic: Slanted text.
    c. oblique: Slanted but less commonly used than italic.
3. Syntax

```
font-style: italic;
```

1.5 Font Variant (font-variant)

1. Adjusts the capitalization of text.
2. Values:
    a. normal: Default.
    b. small-caps: Displays lowercase letters as smaller uppercase letters.
3. Syntax

```
font-variant: small-caps;
```

1.6 Shorthand Property (font)

1. Combines multiple font properties in one line.
2. Syntax:

   font: italic bold 16px "Arial", sans-serif;

2. Text Style and Attributes

1. **Text Color (color)**
   a. Defines the color of the text.
   b. Example
      color: #333333;

2. **Text Alignment (text-align)**
   a. Aligns text horizontally.
   b. Values:
      i. left, right, center, justify.
   c. Example
      text-align: center;

3. **Line Height (line-height)**
   a. Sets the vertical spacing between lines.
   b. Values:
      Numeric (e.g., 1.5), absolute (20px), or relative (120%).
   c. Example
      line-height: 1.6;

4. **Letter Spacing (letter-spacing)**
    a. Controls the space between characters.
    b. Example:

```
letter-spacing: 2px;
```

5. **Word Spacing (word-spacing)**
    a. Adjusts the space between words.
    b. Example

```
word-spacing: 5px;
```

6. **Text Shadow (text-shadow)**
    a. Adds shadow effects to text.
    b. Syntax:

```
text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
```

7. **Text Shadow (text-shadow)**
    a. Adds shadow effects to text.
    b. Syntax
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

8. **Text Transform (text-transform)**
    a. Modifies text capitalization.
    b. Values
        i. none, uppercase, lowercase, capitalize.
    c. Example:
        text-transform: uppercase;

9. **White Space (white-space)**
    a. Controls how whitespace and line breaks are handled.
    b. Values:
        i. normal, nowrap, pre, pre-wrap, pre-line.
    c. Example
        white-space: nowrap;

3. Typography Techniques

1. **Responsive Typography:**
   a. Use relative units like em, rem, or % to make text responsive.
   b. Example
      font-size: 1.2rem;
2. **Web Fonts:**
   a. Import custom fonts from services like Google Fonts.
   b. Example:
      @import
      url('https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap');
      body {
        font-family: 'Roboto', sans-serif;
      }
3. **Fallback Fonts:**
   a. Always specify generic font families as fallbacks.
   b. Example
      font-family: "Open Sans", Arial, sans-serif;

Margin, Padding, and Border in CSS

- Margin, Padding, and Border are part of the CSS Box Model, which describes how elements are structured and spaced in a web page.
- Understanding these properties allows you to control spacing, positioning, and styling around elements.
- Margin
  - The margin is the outermost space of an element, used to create space between the element and its neighbors.
- Padding
  - The padding is the space between the content of an element and its border.
- Border
  - The border wraps around the padding and content, defining the edge of an element.

Margin Properties

1. margin-top:
   a. Sets the top margin.
   b. Example
      margin-top: 20px;

2. margin-right:
   a. Sets the right margin.
   b. Example:
      margin-right: 15px;

3. margin-bottom:
   a. Sets the bottom margin.
   b. Example:
      margin-bottom: 10px;

4. margin-left:
   a. Sets the left margin.
   b. Example:
      margin-left: 25px;

- margin (Shorthand):
  - Combines top, right, bottom, and left margins into a single property.
  - Syntax

    margin: 10px 20px 30px 40px; /* Top, Right, Bottom, Left */
  - Example

    margin: 15px; /* All sides */

    margin: 15px 10px; /* Vertical | Horizontal */

- Auto Margin
  a. auto centers an element horizontally within its container when combined with a specified width.
  b. Example:

```
div {
    margin: 0 auto;
    width: 50%;
}
```

- Negative Margins
  a. Negative values pull the element closer to its neighbors.
  b. Example:

```
margin-top: -10px;
```

Padding

The padding is the space between the content of an element and its border.

Padding Properties:

1. **padding-top:**
   a. Sets the top padding.
   b. Example:
      padding-top: 10px;
2. **padding-right:**
   a. Sets the right padding.
   b. Example
      padding-right: 5px;
3. **padding-bottom:**
   a. Sets the bottom padding.
   b. Example:
      padding-bottom: 20px;
4. **padding-left:**
   a. Sets the left padding.
   b. Example:
      padding-left: 15px;

- **padding (Shorthand):**
  - Combines top, right, bottom, and left paddings.
  - Syntax:
    padding: 10px 20px 30px 40px;
    /* Top, Right, Bottom, Left */
  - Example:
    padding: 10px; /* All sides */
    padding: 10px 15px;
    /* Vertical | Horizontal */
- **Zero Padding**
  - Default padding is 0 unless specified.
  - Example:
    padding: 0;

Border

The border wraps around the padding and content, defining the edge of an element.

1. **border-width:**
   a. Specifies the thickness of the border.
   b. Values:
      i. Keywords: thin, medium, thick.
      ii. Specific units: px, em, etc.
   c. Example
   d. border-width: 5px;
2. **border-style:**
   a. Defines the style of the border.
   b. Values:
      i. none: No border.
      ii. solid: A solid line.
      iii. dashed: A dashed line.
      iv. dotted: A dotted line.
      v. double: Two solid lines.
      vi. groove: A carved effect.
      vii. ridge: A raised effect.
   c. Example
      border-style: dashed;
3. **border-color:**
   a. Sets the color of the border.
   b. Example
      i. border-color: red;
4. **border (Shorthand):**
   a. Combines border width, style, and color in a single property.
   b. Syntax:
      i. border: [width][style][color];
      ii. border: 2px solid #000;
5. **Individual Sides:**
   a. border-top, border-right, border-bottom, border-left.
   b. Example:
      i. border-top: 5px solid blue;

- **Rounded Borders:**
  - Use border-radius to create rounded corners.
  - Example:

```
border-radius: 10px; /* All corners */
border-radius: 10px 20px;
/* Top-left & bottom-right | Top-right & bottom-left */
```

- **Border Images**
  - Use images as borders with the border-image property.
  - Syntax:

```
border-image: url('border.png') 30 round;
```

Combining Margin, Padding, and Border

- The box model determines how these properties interact.
- The total width and height of an element are calculated as:
- Total Width = Content Width + Padding + Border + Margin
- Total Height = Content Height + Padding + Border + Margin

Best Practices

1. Consistent Spacing:
   a. Use margin for spacing between elements and padding for spacing within elements.
2. Use Shorthand:
   a. Combine properties to reduce code redundancy.
   b. Example:

```
margin: 10px 15px 20px 25px;
```

3. Responsive Design:
   a. Use relative units like % or em for margin and padding.
   b. Example:

```
padding: 5%;
```

4. Reset Defaults:
   a. Reset margins and paddings for consistent behavior across browsers.
   b. Example:

```
* {
    margin: 0;
    padding: 0;
}
```

List Styles and Table Layouts

List Styles in CSS

- Lists in CSS allow you to control the appearance of ordered (<ol>), unordered (<ul>), and definition (<dl>) lists.
- CSS provides various properties to customize bullet styles, numbering, and positioning.
- List Style Properties:
  - **list-style-type**
    - Specifies the marker type (bullet or numbering style).
    - Values:
      - For unordered lists: disc (default), circle, square, none.
      - For ordered lists: decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, etc.
    - Example:

      list-style-type: square;

- **list-style-position**
  - Defines the position of the list marker relative to the content.
  - Values:
    - outside (default): Marker is outside the text block.
    - inside: Marker is part of the text block.
  - Example:

```
list-style-position: inside;
```

- **list-style-image**
  - Replaces the default marker with an image.
  - Example:

```
list-style-image: url('custom-bullet.png');
```

List Style Properties(Shorthand)

- list-style
  - Combines list-style-type, list-style-position, and list-style-image into a single declaration.
  - Syntax:

    list-style: [type][position][image];
  - Example:

    list-style: square inside url('bullet.png');

Table Layouts in CSS

- CSS provides properties to style and control the structure of tables (<table>), rows (<tr>), cells (<td>), and headers (<th>).
- You can use these to adjust table spacing, alignment, borders, and layout behavior.
- Table Style Properties:
  - **border-collapse**
    - Defines whether table borders are collapsed into a single border or separated.
    - Values:
      - collapse: Merges adjacent borders.
      - separate (default): Keeps borders separate.
    - Example:

```
border: 1px solid black;
border-collapse: collapse;
```

- **border-spacing**
  - Specifies the space between table borders (applies only when border-collapse: separate).
  - Syntax:
    - border-spacing: [horizontal][vertical];
  - Example:

    border-spacing: 10px 15px;

- **table-layout**
  - Controls the layout algorithm used to render the table.
  - Values:
    - auto (default): Column widths are determined by content.
    - fixed: Column widths are determined by the width property.
  - Shorthand:

    table-layout: fixed;

- **empty-cells**
  - Specifies whether to display or hide empty cells in a table.
  - Values:
    - show (default): Displays empty cells.
    - hide: Hides empty cells.
  - Example:
    - empty-cells: hide;

Table Alignment Properties

- **text-align**
  - Aligns text horizontally within a table cell.
  - Values: left, right, center, justify.
  - Example:

```
text-align: center;
```

- **vertical-align**
  - Aligns content vertically within a table cell.
  - Values: top, middle, bottom, baseline.
  - Example:

```
vertical-align: middle;
```

Table Size and Spacing

1. width and height
    a. Defines the dimensions of the table or its cells.
    b. Example:

```
width: 100%;
height: auto;
```

2. padding
    a. Adds spacing inside table cells.
    b. Example:

```
padding: 10px;
```

3. margin
    a. Adds spacing around the table.
    b. Example:

```
margin: 20px auto;
```

4. border
    a. Specifies the border style for the table or cells.
    b. Example:

```
border: 1px solid #ccc;
```

Additional Features in CSS

- CSS offers advanced capabilities like grouping style sheets, assigning classes, and using spans to control and style web page elements effectively.
- These features allow for modular, reusable, and maintainable code.

1. Grouping Style Sheets

- Grouping in CSS allows you to apply the same styles to multiple selectors, reducing redundancy and improving maintainability.
- Syntax
  - Multiple selectors can be grouped using commas.
  - Example:

```
h1, h2, h3 {
        color: #333;
        font-family: Arial, sans-serif;
}
```

- Benefits
  - Avoids repetition by applying the same style to multiple elements.
  - Makes code more readable and efficient.

Assigning Classes

- Classes in CSS allow you to define styles that can be reused across multiple HTML elements. They are ideal for applying specific styles to a group of elements without affecting others.
  - Declaring a Class
  - Use the . prefix to define a class in CSS.
  - Example:

```
.highlight {
    background-color: yellow;
    font-weight: bold;
}
```

- Applying a Class in HTML
  - Use the class attribute in an HTML element.
  - Example:

```
<p class="highlight">This text is highlighted.</p>
```

- Multiple Classes
  - HTML elements can have multiple classes by separating them with a space.
  - Example:

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

Using span

- The element is an inline container used to apply styles or scripts to a portion of text or content within an element. It does not affect layout by default.
- Syntax
  - Example of styling specific text within a paragraph:

```
<p>This is <span class="highlight">important</span> text.</p>
```

- Benefits of span
  - Allows targeted styling for part of a larger element.
  - Useful for adding interactivity or emphasis to specific text.
- CSS for span
  - Example:

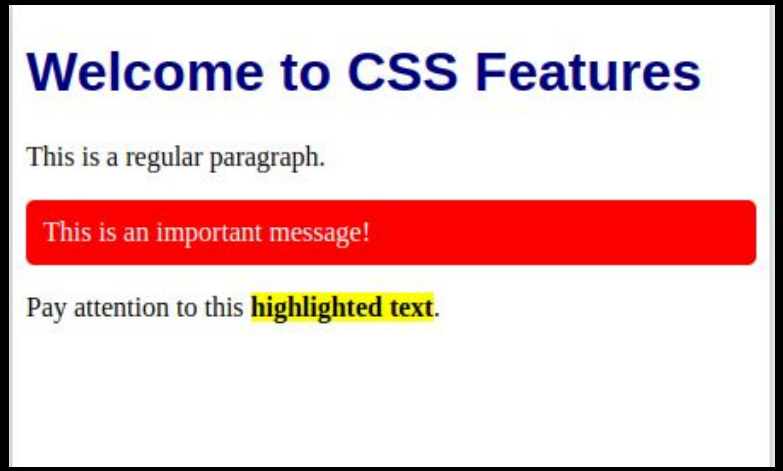
```
.highlight {
    color: red;
    font-style: italic;
}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    /* Grouping styles for headings */
    h1, h2, h3 {
      font-family: 'Verdana', sans-serif;
      color: navy;
    }

    /* Assigning a reusable class */
    .alert {
      color: white;
      background-color: red;
      padding: 10px;
      border-radius: 5px;
    }

    /* Styling a span element */
    .highlight {
      background-color: yellow;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h1>Welcome to CSS Features</h1>
  <p>This is a regular paragraph.</p>
  <p class="alert">This is an important message!</p>
  <p>Pay attention to this <span class="highlight">highlighted text</span>.</p>
</body>
</html>
```

Output:



DIV Tag

- The tag in HTML is a block-level element used as a container for grouping other elements.
- It plays a critical role in structuring web pages and applying styles or scripts to a section of content.
- Features of the Tag
  - Block-Level Element
    - By default, it spans the full width of its parent container and starts on a new line.
  - Semantic Neutrality
    - itself does not convey any semantic meaning but is widely used for layout and styling.
  - Styling and Scripting
    - Commonly paired with CSS classes and JavaScript for advanced customizations.
  - Nesting Support
    - Can contain any other HTML elements, including nested tags.

Usage Examples

- Basic Styling

```
<div style="background-color: lightblue; padding: 10px;">
    <p>This is inside a styled div.</p>
</div>
```

- Grid Layout
  - Divs are essential for creating layouts using CSS grid or Flexbox.

```
<div class="row">
    <div class="column">Column 1</div>
    <div class="column">Column 2</div>
</div>
```

- Dynamic Content with JavaScript
  - Targeted for updates or interactions.

```
<div id="content">
    Content will be updated here.
</div>
```

Responsive Web Design

- Responsive Web Design (RWD) ensures web pages adapt to different screen sizes and devices (e.g., desktops, tablets, smartphones).
- It aims to provide an optimal viewing experience with flexible layouts, images, and CSS.
- Core Principles of RWD
  - Fluid Grid Layouts
  - Flexible Images
  - Media Queries
  - Responsive Typography

Core Principles of RWD

- Fluid Grid Layouts
  - Use percentage-based widths for containers and elements to ensure scalability.
    - .container {
    - width: 100%;
    - }
- Flexible Images
  - Images resize automatically within their container using CSS.
    - img {
    - max-width: 100%;
    - height: auto;
    - }
- Media Queries
  - Apply styles conditionally based on screen size or device type.

```
@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

- Responsive Typography
  - Font sizes scale dynamically using relative units like em, rem, or viewport-based units.
    - body {
      font-size: 1rem;
      /* Root-relative font size */
      }

Responsive Frameworks

- Popular frameworks like Bootstrap, Foundation, or Tailwind CSS simplify creating responsive layouts.

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