CSS

2.1 Basic Of CSS

2.1.1 CSS Introduction: -

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External style sheet are stored in CSS file. It is a powerful styling language that allows web designers and developers to define how web pages should look, making them visually appealing and user-friendly.

CSS works by targeting HTML elements and applying specific styles to them. These styles can include properties such as colors, fonts, margins, padding, borders, and more. By separating the content (HTML) from its presentation (CSS), developers can achieve a consistent and organized design throughout a website.

CSS is typically written in separate files called stylesheets, with a .css file extension. These stylesheets can be linked to HTML documents using the <link> tag in the <head> section of the HTML file.

By using CSS, web developers can create visually appealing and well-structured websites, enhance user experience, and maintain consistency across different web pages within the same site. As a result, CSS is an essential tool for modern web design and development.

2.1.2 Different types of Stylesheets: -

The "Cascading" in CSS refers to the way styles are applied and resolved when multiple rules apply to the same element. It follows a specific order of precedence, with styles from different sources like inline styles, internal styles, and external stylesheets cascading down and being applied accordingly.

I. External style sheet: -

An external style sheet is a separate file that contains CSS code used to define the presentation and layout of HTML documents. It allows developers to keep the styles separate from the HTML content, promoting better organization, maintainability, and reusability of styles across multiple web pages.

To use an external style sheet, you create a separate CSS file with a .css file extension and then link it to your HTML document using the <link> element. The <link> element is placed in the <head> section of the HTML document.

- Demo.html

```
<html>
    <head>
        <link rel=”stylesheet” type=”text/css” href=”Test.css”>
    </head>
    <body>
        <p> Hello world </p>
        <p id=”para”> Greetings of the day!!! </p>
    </body>
</html>
```

- Test.css

```
#para{
    text-align: center;
}
P{
    Color : blue;
}
```

II. Internal style sheet: -

An internal style sheet, also known as an embedded style sheet, is a method of adding CSS rules directly within the <head> section of an HTML document. Unlike an external style sheet, the CSS code is placed directly in the HTML file, allowing you to apply specific styles to that particular HTML document.

To use an internal style sheet, you place the CSS code between <style> and </style> tags inside the <head> section of your HTML document. This way, the CSS rules within the <style> block will only affect the elements on that specific page.

- Demo.html

```
<html>
    <head>
        <style type="text/css">
            P{
                color: red;
            }
        </style>
    </head>
    <body>
        <p> Your page's content! </p>
    </body>

</html>
```

III. Inline style: -

Inline CSS refers to the method of applying CSS styles directly within individual HTML elements. Unlike external and internal style sheets, which are used to define styles for multiple elements or an entire HTML document, inline CSS targets specific elements and overrides any external or internal styles that might apply to them.

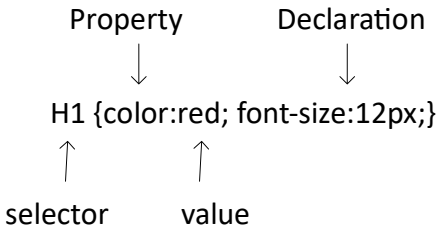
To use inline CSS, you add the style attribute to an HTML element and provide the CSS rules directly as the attribute's value. The format of inline CSS is the same as regular CSS, with property-value pairs separated by semicolons.

- Demo.html

```
<html>
    <head>
        <title> Inline css </title>
    </head>
    <body>
        <h1 style=”color: blue; text-align: center;”> Heading </h1>
        <p style=”background: blue; color : white;”>
            Paragraph is here….
        </p>
    </body>
</html>
```

2.1.3 CSS Syntax: -

A CSS rule has two main parts: a selector, and one or more declarations separated by semicolon. The selector can be HTML element, id or class. Each declaration consists of a property name and a value, separated by a colon. The property is the style attribute you want to change. Each property has a value. Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.




```
p {
        color: red;
        text-align: center;
}
```

p is a selector in CSS. color is a property, and red is the property value text-align is a property, and center is the property value.

2.1.4 CSS Selector: -

In CSS, selectors are patterns used to select the element(s) you want to style. CSS selectors are patterns used to select and style HTML elements on a web page. They allow you to target specific elements based on their attributes, classes, IDs, and relationships with other elements. CSS selectors play a crucial role in applying styles and controlling the layout of web pages.

Selector	Example	Example description
#id	#name	Selects the element with id=”name”
.class	.center	Selects all elements with class=”para”
element.class	p.center	Selects only <p> elements with class=”para”
*	*	Selects all elements
element	div	Selects all elements
element, element,..	div, p	Selects all and <p> elements

- Element selector: -

  The element selector selects HTML elements based on the element name.

```
div {
        text-align: center;
        color: red;
}
```

- Id selector: -

  The id selector uses the id attribute of an HTML element to select a specific element. The id of an element is unique within a page, so the id selector is used to select one unique element! To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#name {
        text-align: center;
        color: red;
}
```

- Class selector: -

  The class selector selects HTML elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the class name.

```
.center {
    text-align: center;
    color: red;
}
```

- Element with class selector: -

  Element with class selector selects only specific HTML elements should be affected by class.

```
p.center {
    text-align: center;
    color: red;
}
```

- Universal selector: -

  The universal selector (*) selects all HTML elements on the page.

```
* {
    text-align: center;
    color: blue;
}
```

- Grouping selector: -

  The grouping selector selects all the HTML elements with the same style definitions. To group selectors, separate each selector with a comma.

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all elements and all <p> elements

element element	div p	Selects all <p> elements inside elements
element>element	div > p	Selects all <p> elements where the parent is a element
element+element	div + p	Selects the first <p> element that is placed immediately after elements
element1~element2	p ~ ul	Selects every element that is preceded by a <p> element
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target="_blank"]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~="flower"]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|="en"]	Selects all elements with a lang attribute value equal to "en" or starting with "en-"

[*attribute*^=*value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[*attribute*$=*value*]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[*attribute**=*value*]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"
:active	a:active	Selects the active link
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
:checked	input:checked	Selects every checked <input> element
:default	input:default	Selects the default <input> element

:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children (including text nodes)
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the input element which has focus

:fullscreen	:fullscreen	Selects the element that is in full-screen mode
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects input elements with a value within a specified range
:indeterminate	input:indeterminate	Selects input elements that are in an indeterminate state
:invalid	input:invalid	Selects all input elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links

::marker	::marker	Selects the markers of list items
:not(*selector*)	:not(p)	Selects every element that is not a <p> element
:nth-child(*n*)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(*n*)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent

:optional	input:optional	Selects input elements with no "required" attribute
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range
::placeholder	input::placeholder	Selects input elements with the "placeholder" attribute specified
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified
:required	input:required	Selects input elements with the "required" attribute specified
:root	:root	Selects the document's root element
::selection	::selection	Selects the portion of an element that is selected by a user

:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all input elements with a valid value
:visited	a:visited	Selects all visited links

2.1.5 CSS Basic property: -

CSS (Cascading Style Sheets) provides a wide range of properties that allow you to control the visual appearance and layout of HTML elements on a web page. Here are some basic CSS properties along with a brief explanation of each:

- **color**: Sets the text color for an element. You can use color names, hexadecimal codes, RGB values, or HSL values.
- **background-color**: Sets the background color of an element.
- **font-family**: Specifies the font for text inside an element. You can provide a list of fonts, and the browser will use the first available font.
- **font-size**: Sets the size of the font. You can use different units like pixels (px), ems (em), or percentages (%).
- **font-weight**: Sets the thickness of the font characters. Values can be normal, bold, bolder, lighter, or numeric values.
- **text-align**: Aligns the text within an element. Values can be left, right, center, or justify.
- **line-height**: Defines the vertical space between lines of text.
- **margin**: Sets the space outside an element. You can specify values for top, right, bottom, and left margins.
- **padding**: Sets the space inside an element, between its content and its border. Similar to margin, you can specify values for top, right, bottom, and left padding.
- **border**: Defines the border around an element. It combines three properties: border-width, border-style, and border-color.
- **width**: Sets the width of an element. You can use various units like pixels (px), percentages (%), or viewport units (vw).
- **height**: Sets the height of an element using similar units as width.
- **display**: Determines how an element is displayed. Common values include block, inline, and inline-block.
- **position**: Specifies the positioning method for an element. Values can be static, relative, absolute, fixed, or sticky.
- **float**: Controls the positioning of an element within its container. It's often used for creating columns or layouts.
- **clear**: Controls whether an element can be next to floating elements or not.
- **text-decoration**: Adds visual effects to text, such as underlining, overlining, and striking through.
- **list-style-type**: Sets the style of bullet points or numbering for lists.
- **box-shadow**: Adds a shadow effect to an element's box.
- **border-radius**: Creates rounded corners for an element's border.