CSS

CSS stands for Cascading Style Sheet.

Used to give styling to the web pages which is structured by the HTML language.

There are three ways in CSS to Style the Web pages

1. Inline CSS
2. Internal CSS
3. External CSS
4. Import.

1. **Inline CSS:** Inline CSS code is written in the opening tag of the HTML element by using style attribute.
   Syntax:

2. **Internal CSS:** Internal CSS code is written in the head section of an HTML element using <style> tag.
   Syntax:
   ```
   <head>
       <style>
           tag_name{
               property: value;
           }
       <style>
   </head>
   ```

3. **External CSS:** External CSS styling can be done by creating a external CSS file with file_name.css extention and providing the link between that CSS file to HTML file by <link> tag in the head section of HTML element.
   Syntax:
   ```
   <head>
       <link rel="stylesheet" href=" file_name.css">
   </head>
   ```

4. **@import**: This Styling can be done by linking one CSS file to another CSS file.
   Syntax: @import url(file_name.css)

Import should be written in very first line of the CSS file which has to be linked.

NOTE: The first priority is inline CSS

But the priorities of the Internal and External CSS varies accordingly depending on the declared position

if, <link rel="stylesheet href="index.css">
<style> </style>
Here, the first priority is for Internal CSS.

if, <style> </style>
<link rel="stylesheet href="index.css">
Here, the first priority is for External CSS.

SELECTOR

Selectors are used to select the particular HTML element and to style them.

There are five types of Selectors

1. Simple Selectors
2. Combinator Selectors
3. Attribute Selectors
4. Pseudo Element Selectors.
5. Pseudo Classes Selectors.

Simple Selectors

Simple selectors style the HTML element in *five* ways:

1. Id Selector.
2. Class Selector.
3. Tagname.
4. Groupname.
5. Universe.

Id selectors:

Id selector targets only individual element in the HTML document.

Prefix symbol used for id selector is # (hash)

Example:

```
#demo{
    color: red;
    background: yellow;
}
```

Class Selector

Class selector targets multiple elements in the HTML document.

Prefix symbol used for id selector is . (dot)

Example:

```
.test{
    color: red;
    background: yellow;
}
```

Tagname Selector

Tagname selector targets the HTML elements by tag name.

There is no symbol used in tagname selectors.

Example:

```
h1{
    color: magenta;
    background: green;
}
h2{
    color: red;
    background: black;
}
h3{
    color: yellow;
    background: orange;
}
```

Grouping Selector

Groupname selector targets a group of HTML elements by tagname separated by commas.

Example:

```
p, div, h4{
    color: magenta;
    background: green;
}
```

Universal Selector

Universal Selectors targets every HMTL element.
The symbol of Universal Selector is *
There is no selector is declared in css.

Example:

```
*{
    color: yellow;
    background: magenta;
}
```

<u>Combinator Selector</u>

Combinator Selectors are used to style the combination of HTML elements.

There are *<u>four</u>* Combinator Selectors:

1. Descendent Selectors
2. Child Selectors
3. Adjacent Sibling Selectors
4. General Sibling Selectors

1. Descendent Selectors: This Selector targets both direct and indirect children of a parent tag. The symbol is (space)

Syntax: parent_tag child_tag{property: value;}

2. Child Selectors: This Selector targets the only the direct children of the parent tag. The symbol is > (greater than)

Syntax: parent_tag > child_tag{property: value;}

3. Adjacent Selectors: This Selector targets the element which is the first sibling of the targeted HTML tag or element.

(or)

This Selector targets the element which is exactly adjacent to the targeted HTML tag or element.

The symbol is + (plus)

Syntax: parent_tag + target_tag{property: value;}

4. General Sibling Selectors: This Selector targets all the siblings of the target HTML tag or element.

(or)

This Selector targets all the adjacent tags of the target HTML tag or element. The symbol is ~ (tilde)

Syntax: parent_tag ~ target_tag{property: value;}

Attribute Selector: Attribute Selectors are used to style the HTML element by targeting the respective attributes and with their values.

Types of Attribute Selector:

1. **tag_name[attribute_name]:** Represents elements with an attribute name of ***attribute_name***.
2. **tag_name [attribute_name= "value"]:** Represents element with an attribute name of ***attribute_name*** whose value is exactly ***value***.

<u>Pseudo Element Selectors.</u>

Pseudo Elements targets the content of the HTML element by the following ways:

1. First Letter
2. First Line
3. Before
4. After
5. Selection
6. Marker

Each pseudo element selectors is declared with the double colon **: :**

First Letter: The first letter styles the very first letter of the content in the targeted HTML element.

Example:
```
p : : first-letter{
    font-size: 30px;
    color: black;
}
```

First Line: The first line styles the very first line of the content in the targeted HTML element.

Example:
```
p : : first-line{
    background-color: black;
    color: white;
}
```

Before: The before pseudo element is used to place the specific content before the targeted HTML element.

Example:
```
p : : before{
        content: “&phone”;
        background-color: black;
        color: white;
    }
```

After: The after pseudo element is used to place the specific content after the targeted HTML element.

Example:
```
p : : after{
        content: “Thank You”;
        color: blue;
    }
```

Selection: The selection pseudo element is used to style the content when the cursor is dragged on the content (when the content is selected) on the targeted HTML element.

Also copying the text content from the user can also be disabled, by using the property user-select: none.

Example:
```
p : : selection{
        background-color: pink;
        color: magenta;
    }
```

Marker: Marker pseudo element is used to style the lists in the HTML document.

Marker is only used to style the list type not the content.

Declaration of selector is not mandatory here.

Background color will not work for Marker.

Example:
```
li : : marker{
        color: red;
        font-size: 30px;
    }

    (or)

: : marker{
```

```
        color: red;
        font-size: 30px;
    }
```

Pseudo Classes

1. Dynamic Pseudo Classes.
2. Structural Pseudo Classes.

Dynamic Pseudo Classes

- Link
- Visited
- Hover
- Active
- Focus

Structural Pseudo Classes

- First-child
- Last-child
- Nth-child
- First-of-type
- Last-of-type

Dynamic Pseudo Classes

Pseudo Class Selector specifies the special state of the content.

The Pseudo Class Selector are:

1. Link
2. Visited
3. Hover
4. Active
5. Focus
    - Each dynamic pseudo class selector is declared with the single colon **:**
    - Link, Visited only applicable for anchor tag.
    - Link is used to style the unvisited link
    - Link doesn't style the already visited link.
    - Active applies the style at the time of click event.
    - Hover is applicable for all the HTML elements.

- Focus styles when the element is focused.
- Focus selector is applicable for <a> tag and also for the elements which takes user input that is for <input> and <textarea> tag.
- While using all the Link, Visited, Hover and Active selectors. Hover should be declared after Link and Visited. Along with that the Active should be declared after Hover, because to make the elements effective.

Example:

```
a : link {
        color: aqua;
        background-color : red;
         }

a : visited {
        color : green;
           }

a : hover {
        color : yellow;
          }

a : active {
         color : chocolate;
          }

input : focus{
         color : red;
         background-color : yellow;
        }
```

Structural Pseudo Classes: The Structural Pseudo Class selectors is used to target the combination of the HTML elements and to style them.

1. First-child
2. Last-child
3. Nth-child
4. First-of-type
5. Last-of-type

First-child: The First-child pseudo class selector targets the first child element of the targeted parent.

Example:
```
div p : first-child{
        color: red;
        background-color: yellow;
}
```

Last-child: The Last-child pseudo class selector targets the last child element of the targeted parent.

Example:
```
div p : last-child{
        color: red;
        background-color: yellow;
}
```

Nth-child: The nth-child( ) pseudo class selector targets the elements according to the argument passed.

Example:
```
div p : nth-child(2){
        color: red;
        background-color: yellow;
}
```

First-of-type: The first-of-type pseudo class selectors targets the first element of the input type.
The input type should be the first type of all the input types.
It also behaves as first-child selector.
It is applicable for text, submit, reset, button.
It is not applicable for radio, checkbox, textarea,.. so on.

Example:
```
input : first-of-type{
        color: red;
        background-color: yellow;
        }
```

Last-of-type: The Last-of-type pseudo class selectors targets the last element

of the input.
The input type should be the last type of all the input types.
It also behaves as last-child selector.
It is applicable for text, submit, reset, button.
It is not applicable for radio, checkbox, textarea,.. so on.

Example:
```
input : last-of-type{
        color: red;
        background-color: yellow;
        }
```

Nth-of-type: The nth-of-type pseudo class selectors targets all the element of the input type.
It also behaves as nth-child selector.
It is applicable for text, submit, reset, button.
It is not applicable for radio, checkbox, textarea,.. so on.

Example:
```
input : nth-of-type(n){
        color: red;
        background-color: yellow;
```

BOX Model

Box Model is essentially a box that wraps around every HTML element. Box Model is used to design and layout.

The CSS Box Model consists of Margin, Border, Padding and the actual Content.

Content: *It is the content of the HTML element.*
Padding: *It is the space after the content.*
Border: *It is the Border of the HTML element.*
Margin: *It is the space after the Border of an HTML element.*
Padding and Margin allow up to four values.

Syntax:
```
padding:  10px ;
padding:  10px   20px;
padding:  10px   20px   30px;
padding:  10px   20px   30px   40px
```

Text Properties

Formatting text properties:

color- changes the text color
Example - color: red;

text-align- Horizontal alignment of the text (here we have right, left, center, justify).
Example- text-align: center;

text-transform- converts the text-letter format(UPPERCASE, lowercase, Capitalize).
Example- text-transform: capitalize;

text-shadow- it allows 4 values they are x-direction y-direction opacity and color
Example- text-shadow: 2px 2px 5px red;

text-decoration- underline, overline, line-through
Example- text-decoration: none;

letter-spacing- to get the space between letters of a text we use it
Example- letter-spacing:2px;

word-spacing- to get the space between words of a text we use it
Example- word-spacing:5px;

text-indent- it'll be saying from where the text should start.
Example- text-indent:2px;

Background Styling Properties

background-color: The background-color property sets the background color of an element.
Syntax: background-color: color|transparent|initial|inherit;

background-image: The background-image property sets one or more background images for an element.
Syntax: background-image: url|none|initial|inherit;

background-position: The background-position property sets the starting position of a background image.
Syntax: background-position: value;

background-repeat: The background-repeat property sets if/how a background image will be repeated. By default, a background-image is repeated both vertically and horizontally.
Syntax: background-repeat: repeat|repeat-x|repeat-y|no-repeat|initial|inherit;

background-size: The background-size property specifies the size of the background images.
Syntax: background-size: auto|length|cover|contain|initial|inherit;

CSS Units:

CSS has several different units for expressing a length. Many CSS properties take "length" values, such as margin, width, padding, height etc.

Absolute Lengths:

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Absolute units are mm, cm, in, px-pixels(1/96 of 1 in), pt-points(1/72 of 1 in), pc- picas (12 pt).

Relative length:

units specify a length relative to another length property. Relative length units scale better between different rendering medium.

Relative units are em, rem, vh, vw, % .

Why ? *{Padding:0, margin:0 , box-sizing: border-box;}

Targeting the universal selector (*) and setting properties like padding: 0, margin: 0, and box-sizing: border-box is a common practice in web development, and it serves several purposes:

Resetting Default Styles: Browsers apply default styles to various HTML elements, such as adding default margins and paddings to elements like body, div, ul, li, etc. By targeting the universal selector, you can reset these default styles to ensure a consistent starting point for styling your webpage.

Consistency: Setting padding: 0 and margin: 0 to all elements eliminates any unexpected spacing or layout issues caused by inconsistent default styles across different browsers. This helps in achieving a consistent look and feel for your website.

Box Sizing: The box-sizing: border-box property ensures that an element's padding and border are included in its total width and height. This can be more intuitive for layout purposes and helps avoid unexpected layout issues. Without this setting, padding and borders would add to the element's specified width and height, potentially causing layout problems.

Flex:

display: flex; is a CSS property that is used to create a flexible container for a set of items (usually referred to as flex items) in a web page layout. When you apply display: flex; to an HTML element, it turns that element into a flex container, and its child elements become flex items. This allows you to control the layout and alignment of these items within the container using various flex properties.

Flex Container: The element to which you apply display: flex; becomes a flex container. It can contain one or more flex items. The container is responsible for defining how the items should be arranged and distributed within the available space.

Flex Items: The child elements of a flex container are referred to as flex items. These items can be text, images, divs, or any other HTML elements. Flex items are aligned and distributed within the flex container according to the rules defined by flex properties.

Main Axis and Cross Axis: Flex layouts are based on two axes. The main axis is the primary axis along which the flex items are aligned, and the cross axis is perpendicular to the main axis. You can control the direction of the main axis using the flex-direction property.

flex-direction: This property determines the direction of the main axis within the flex container. Common values include row (default), row-reverse, column, and column-reverse. These values control whether flex items are arranged horizontally or vertically and in which order.

justify-content: This property controls the alignment of flex items along the main axis. It specifies how the extra space in the container (if any) should be distributed among the items. Common values include flex-start, flex-end, centre, space-between, and space-around.

align-items: This property controls the alignment of flex items along the cross axis. It specifies how items should be aligned vertically within the container. Common values include flex-start, flex-end, centre.

flex-basis: This property sets the initial size of a flex item before any available space is distributed. It can be specified as a fixed size, a percentage, or auto.

flex-wrap: wrap, nowrap, wrap-reverse.

Gap: In the context of Flexbox layouts, the gap property controls the spacing between flex items within a flex container.

Example:

```
.flex-container {
    Display: flex;
    gap: 10px;  }
```