CSS stands for (Cascading style sheets). CSS defines how HTML elements should be displayed on screen by applying styles like colors, fonts, layout, ensuring a consistent look across different platforms.

Why we need CSS ?

Making websites look attractive and easier to maintain by allowing style changes across multiple pages from a single line.

Selectors ::

Basic Selectors :

Basic selectors in CSS are the most fundamental selectors used to target and style HTML elements.

- `*` : Universal selector, selects all elements.
- `element` : Type selector, selects all elements of a specific type (e.g. `p`, `h1`, `div`).
- `.class` : Class selector, selects all elements with a specific class (e.g. `.header`, `.footer`).
- `#id` : ID selector, selects a single element with a specific ID (e.g. `#header`, `#footer`).

Attribute selectors :

An attribute selectors in CSS is used to select HTML elements based on the value of their attributes. It allows you to apply styles to elements dynamically according to their attributes. Examples :

- `[attribute]` : Select elements with specific attribute.

```
[href] {
  color: blue;
}
```

- `[attribute="value"]` : Select elements with a specific attribute and value.

```
[href="https://www.example.com"] {
  color: red;
}
```

- `[attribute~="value"]` : Select elements with an attribute whose value contains a specific word.

```
[class~="header"] {
  font-weight: bold;
}
```

- `[attribute^="value"]` : Select elements with an attribute whose value starts with a specific value.

```
[src^="https://"] {
  border: 1px solid green;
}
```

Pseudo-classes :

- `:hover` : Selects elements when they are hovered over.
- `:active` : Selects elements when they are active (e.g. when a link is clicked).
- `:focus` : Selects elements when they have focus (e.g. when a form input is selected).
- `:first-child` : Selects the first child element of a parent element.
- `:last-child` : Selects the last child element of a parent element.
- `:nth-child(n)` : Selects the nth child element of a parent element.

Pseudo-elements:

- `::before` : Selects the pseudo-element that is inserted before the content of an element, allowing you to add content before the element's actual content.
- `::after` : Selects the pseudo-element that is inserted after the content of an element, allowing you to add content after the element's actual content.
- `::first-letter` : Selects the first letter of an element's content, Enabling special styling for that initial letter.
- `::first-line` : Selects the first line of an element's content, - allowing you to apply styles to just the first line of text.

Box model:

The box model in CSS describes how elements are structured and spaced on a web page. It contains of several layers like

- `Content` : The actual content of the element, where text and images appear.

- `Padding` : The space between the content and the border.
- `Border` : The line that goes around the padding and the content.
- `Margin` : The space between the border and the surrounding elements.

Flexbox :

Flexbox is a CSS layout model that arranges elements in a `container` and in a flexible way. The elements within the container called flex-items. Flexbox is useful for aligning items of different sizes into rows or columns.

parent => flex-container

- `display : flex` : The flex property sets the flexible length on flexible items.
- `flex-direction` : Specifies the direction of the flex items (e.g. `row`, `column`, `row-reverse`, `column-reverse`).
- `flex-wrap` : This property specifies whether the flex items should wrap or not. (e,g. wrap, no-wrap, wrap-reverse,)
- `flex-flow` : The `flex-flow` property is a shorthand property for setting both the `flex-direction` and `flex-wrap` properties.
- `justify-content` : This property is used to align the flex items.(e.g. flex-start. flex-end, center, space-between, space-around, space-evenly).
- `align-items` : This property is used to align the flex items. (e.g: flex-start, flex-end, center, baseline, stretch).
- `align-content` : This property is used to align the flex items..(e.g. flex-start. flex-end, center, space-between, space-around, space-evenly).

Children => flex-items

The direct child elements of a flex container automatically becomes flexible items.

- order : The `order` property can change the order of the flex items.
- flex-grow : The `flex-grow` property specifies how much a flex item will grow relative to rest of the flex- items.
- flex-shrink : The `flex-shrink` property specifies how much a flex item will shrink relative to the rest of the flex items.
- flex-basis : The `flex-basis` property specifies the initial length of a flex item.
- flex : The `flex` property is a shorthand property for the `flex-grow`, `flex-shrink`, and `flex-basis` properties.

CSS Grid :

CSS Grid is a layout system that allows for the creation of complex and responsive layouts. It is useful for creating grid-based layouts.

- `display: grid` : Sets the container to use grid layout.
- `grid-template-columns` : Specifies the number and size of columns in the grid (e.g. `repeat(3, 1fr)`, `100px 200px 300px`).
- `grid-template-rows` : Specifies the number and size of rows in the grid (e.g. `repeat(2, 1fr)`, `100px 200px`).
- `grid-gap` : Specifies the gap between grid cells (e.g. `10px`, `20px 30px`).

Positioning :

Positioning allows for the precise placement of elements on a web page.

- `position: static` : Sets the element to its default position.
- `position: relative` : Sets the element to a relative position, allowing it to be offset from its default position.
- `position: absolute` : Sets the element to an absolute position, allowing it to be placed anywhere on the page.
- `position: fixed` : Sets the element to a fixed position, allowing it to remain in the same position even when the page is scrolled.

Typography

Typography refers to the style and appearance of text on a web page.

- `font-family` : Specifies the font family (e.g. `Arial`, `Helvetica`, `sans-serif`).
- `font-size` : Specifies the font size (e.g. `16px`, `1.5em`, `2rem`).
  `Rem` is useful for creating scalable designs, `em` is useful for creating responsive designs, and `px` is useful for fixed-size elements
- `font-weight` : Specifies the font weight (e.g. `normal`, `bold`, `lighter`, `bolder`).

Color :

There are three types of colors

RGB (Red, Green, Blue)

The RGB color model represents colors by combining red, green and blue light in varying intensities. **RGB** is ideal for defining colors by their light components.

- **Syntax:** `rgb(red, green, blue)`

- **Range:** Each value can range from `0` to `255`.

HEX(Hexdecimal) :

The HEX color format uses a six-digit hexadecimal number to represent colors. Is a compact way to write RGB value The format is:

- **Syntax:** `#RRGGBB`
- **Range:** Each pair (RR, GG, BB) can range from `00` to `FF` (hexadecimal), where `00` represents the lowest intensity and `FF` represents the highest.

HSL(Hue, Saturation, Lightness) :

The HSL color model represents colors using three components: hue, saturation, and lightness. It is often easier for designers to adjust colors, especially when working with hue, saturation, and brightness adjustments.
Syntax: `hsl(hue, saturation, lightness)`
Represents the color itself and is given in degrees on the color wheel (0 to 360).

Responsive Design:

Media Queries :

Media queries are a fundamental part of responsive design in CSS. They allow you to apply different styles based on the characteristics of the user's device, such as screen width, height, resolution, orientation, and more.

- `@media (max-width: 768px)` : Applies styles when the screen width is less than or equal to 768px.
- `@media (min-width: 1024px)` : Applies styles when the screen width is greater than or equal to 1024px.

```
@media (max-width: 768px) {
    body {
        background-color: lightblue;
    }
}
```

Viewport Tag:

The viewport tag allows for the control of the zooming and scaling of web pages on mobile devices.

`

This example ensures the width of the page matches the width of the device, and sets the initial zoom level to 1.

CSS Transitions and Animations :

CSS Transitions :

CSS transitions allow you to smoothly change a property's value over a specified duration. This is useful for creating hover effects changing colors etc.

```
.element {
    transition: property duration timing-function delay;
}
```

CSS Animations :

CSS animations allow you to change the style of an element gradually over time. Unlike CSS transitions (which only move between two states), CSS animations can create more complex effects by defining multiple steps (called **keyframes**) in the animation.
Keyframes :
Keyframes specify what styles the element should have at different points during the animation.

- Use the `@keyframes` rule to define keyframes.
- You can define changes using percentages (like `0%`, `50%`, `100%`) or keywords (`from` and `to`).

```
@keyframes slideAndFade {
  0% {
    transform: translateX(0);
    opacity: 0;
  }
  50% {
    transform: translateX(100px);
    opacity: 1;
  }
  100% {
    transform: translateX(200px);
    opacity: 0;
  }
}

.moving-element {
```

```
  width: 100px;
  height: 100px;
  background-color: red;
  animation: slideAndFade 3s ease-in-out infinite;
}
```

Specificity :

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

- `!important` : Increases the specificity of a CSS rule, allowing it to override other rules.
- `inline styles` : Increases the specificity of a CSS rule, allowing it to override other rules.
- `IDs` : Increases the specificity of a CSS rule, allowing it to override other rules.
- `classes` : Decreases the specificity of a CSS rule, allowing it to be overridden by other rules.

Inline and inline-block and block :

Inline:

Do not start on a new line and only take up as much width as necessary. you cannot set their width or height explicitly. `<span>`, `<a>`, `<strong>`, `<em>`, etc.

Block:

Always start on a new line and take up the full width available by default. You can set width, height, margins, and padding for block elements `<div>`, `<h1>`, `<p>`, `<section>`, etc.

Inline-block :

`inline-block` combines characteristics of both `inline` and `block` elements.
They do not start on a new line. `inline-block` elements allow you to set their width, height, margin, and padding just like `block` elements.