CSS

What is CSS ?

- CSS stands for the cascading style sheet
- CSS, or Cascading Style Sheets, is a language used to style the appearance of content on the web.
- CSS controls how it looks—colors, layouts, fonts, spacing, and more.
- CSS describes how html elements are to be displayed on screen.

What is CSS selectors?

- CSS Selectors are used to select the HTML elements you want to style on a web page.
- They allow you to target specific elements or groups of elements to apply styles like colors, fonts, margins, and more.

- Types of CSS selectors

1. Universal selector

- Targets all elements on the web page

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

2. Type or Element selector

- A element selector targets an HTML element, such as <h1>, <p>, etc.
- This is used when we want to apply similar style to all the <p> tags or <h1> tags in the document.

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

3. Class selector

- A class selector targets an element with a specific value for its class attribute to style it.
- A class in CSS is denoted by "." (period) symbol.

```
.btn{
   background-color: green;
}

<button class=”btn”>submit</button>
```

4. ID selector

- An ID selector targets single element with a particular value for id attribute to style it.
- An id in CSS is denoted by "#" (hash) symbol.
- Same class can be applied to multiple elements, but an id is unique for an element.

```
#para{
   font-size: 20px;
}

<p id=”para”>this is paragraph</p>
```

5. Attribute selector

- An attribute selector targets an element based on a specific attribute or attribute values on an element

```
A[title]{
  Font-size:2em;
}

A[taget=”_self”]{
  Font-size:5em;
}
```

```
<a href="#" title="xyz">click</a>

<a href="#" target="_self">click</a>
```

6. Group selector

- CSS group selector allow us to apply same style to multiple elements at a time.
- Name of elements can be separated by commas.

```
h1, h2 {
  background-color: grey;
}

<h1>java is programming</h1>
<h2>java is Secure</h2>
```

7. Descendant or contextual selector

- Descendant selector is used in css to style all the tags which are child of a particular specified tag.
- A single space between parent element and child element is used to mention as descendant.

```
Div p{
 Color: red;
}

<div>
      <p>
            This is paragraph
      </p>
</div>
```

Ways to specify CSS

- CSS can be specified in three primary ways, each serving different purposes depending on the project's size, organization, and styling needs.

1. Inline CSS

- Styles are added directly to the HTML element using the style attribute.
- **Use Case**: Good for quick, small, or unique styling changes on individual elements.
- Not ideal for larger projects or when styles need to be reused.

```
<h1 style="color: blue; font-size: 24px;">Hello World</h1>
```

2. Internal CSS (Embedded CSS)

- Styles are included within the <style> tag in the <head> section of an HTML document.
- **Use Case:** Useful for styling a single page without linking external files.

```
<html>
<head>
        <style>
              body {
                        font-family: Arial, sans-serif;
                       margin: 0;
                  }
              h1 {
                   color: blue;
                   }
       </style>
</head>
<body>
   <h1>Hello World</h1>
</body>
</html>
```

3. External CSS

- Styles are saved in a separate .css file, which is then linked to HTML documents.
- Use Case: Preferred for larger websites because it allows styles to be applied across multiple pages, making it easier to maintain and update.

- **CSS File (styles.css)**

```
body {
   font-family: Arial, sans-serif;
}
h1 {
   color: blue;
}
```

- **Html document**

```
<html>
      <head>
            <link rel="stylesheet" href="styles.css">
      </head>
<body>
      <h1>Hello World</h1>
</body>
</html>
```

CSS units

- CSS units define measurements for properties like width, height, padding, margin, font-size, and more.
- Type of units

1. Absolute units

- Absolute units are fixed and do not change relative to other elements or screen sizes.
- They're generally used when an exact size is needed.

    1. Pixel –px
    2. Millimeter –mm
    3. Centimeter –cm
    4. Inch –in
    5. Point –pt

2. Relative units

- Relative length units are measured in relation to other elements or viewport of the screen.
- Relative units are great for styling responsive websites because they can be adjusted proportionally based on window size or parent elements.
- These units define lengths relative to other length properties.

    1. Em
        - em unit is relative to the font size of the **current immediate parent** element .
    2. Rem
        - Relative to the root font size (typically set on <html>).
    3. % (Percentage)
        - Relative to the parent element's size.
    4. Vw (Viewport Width)
        - 1vw is 1% of the viewport's width.
    5. Vh (Viewport Height)
        - 1vh is 1% of the viewport's height.

CSS color properties

- CSS offers several ways to define colors:

1. Hex code.

```
P{
      Color: #ff5733;
}
<p>hello world</p>
```

2. RGB (rgb(r, g, b)):

- Defines colors with red, green, and blue components, each between 0 and 255.

```
 P{
      Color: rgb(10, 25, 255);
}
<p>hello world</p>
```

3. RGBA (rgba(r, g, b, a))

- RGB with an alpha (transparency) channel, where **A** is between 0 (transparent) and 1.

```
 P{
      Color: rgba(10, 25, 255, 0.5);
}
<p>hello world</p>
```

4. Named Colors

- CSS includes 140 predefined color names like red, blue, purple, etc.

```
 P{
      Color: red;
}
<p>hello world</p>
```

CSS background properties

- CSS background properties are powerful tools for styling backgrounds of HTML elements.

- The background is a shorthand for the following properties:
  1. background-color
  2. background-image
  3. background-position
  4. background-size
  5. background-repeat
  6. background-attachment

1. background-color
- This property sets a solid color as the background of an element.

```
Body{
Background-color: lightblue;
}
```

2. Background-image
- This property sets an image as the background of an element.

```
Body{
Background-image: url('xyz.jpg');
}
```

3. Background-position
- The background-position property sets the starting position of a background image.
- Values
  a. Keywords (like top, right, bottom, left, and center)
  b. Length values (e.g., 20px 30px)
  c. Percentage values (e.g., 50% 50%)

```
Body{
Background-image: url('xyz.jpg');
Background-position: right top;
}
```

4. Background-size

- The background-size property specifies the size of the background images.
- Values
  a. Auto: Default value. Maintains the original size.
  b. Cover: Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges.
  c. Contain: Resize the background image to make sure the image is fully visible
  d. Specific sizes (e.g., 100px 200px or 100px or 50% 50% or 100%)

```
Div{
        border: 1px solid red;
          width: 800px;
          height: 800px;
          background-image:  url('xyz.jpg');
          background-repeat:  no-repeat;
          background-size:  cover;
    }
```

5. Background-repeat

- The background-repeat property sets how a background image will be repeated.
- Values:
  a. Repeat: Repeats the image both horizontally and vertically.
  1. Repeat -x: Repeats the image horizontally only.
  2. Repeat -y: Repeats the image vertically only.
  3. No -repeat: Doesn't repeat the image.

```
div {
        background-image:  url("abc.png");
        background-repeat:  no-repeat;
}
```

6. Background-attachment

- This property determines if the background scrolls with the page or remains fixed.
- Values:
  a. scroll: The background scrolls with the content.
  b. fixed: The background is fixed in place.

```
div {
     background-image:  url("abc.jpg");
         background-attachment: fixed;
}
```

CSS float and clear

1. Float

- The float property allows an element to "float" to the left or right side of its container. It’s typically used to make text wrap around an image or other elements.
- Values
  a. Left: The element floats to the left of its container, allowing other inline elements to wrap around it on the right.
  b. Right: The element floats to the right of its container, allowing other inline elements to wrap around it on the left.
  c. None: Default value; the element does not float and remains in the normal document flow.
  d. Inherit: Inherits the float property of its parent.

```
<style>
        Img{
                float: left;
        }
</style>

<img src="image.jpg">
<p>This text wraps around the image.</p>
```

2. Clear

- CSS Clear property is used to stop next element to wrap around the adjacent floating elements. Clear can have clear left, clear right or clear both values.

CSS font properties

- The CSS font properties provide control over the appearance of text in a web document.

1. Font-family

- Specifies the typeface(s) to be applied to text.

```
font-family: "Roboto", Arial, sans-serif;
```

2. Font-size

- Controls the size of the font.

```
font-size: 1.2rem;
```

3. Font-weight

- Sets the thickness or boldness of the text
- Values
  a. Keyword values: normal, bold, bolder, lighter
  b. Numeric values: Ranges from 100 (thin) to 900 (extra bold)

```
font-weight: value;
```

4. Font-style

- Specifies whether the text is normal, italicized, or oblique

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

5. Font-variant

- Allows for small-caps text and other stylistic changes.
- Values
  a. normal: Regular text.
  b. small-caps: Transforms lowercase letters to small capital letters.

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

6. Line-height

- Controls the space between lines of text (leading).

```
line-height: 5px;
```

7. Letter spacing

- Defines the spacing between the characters of a block of text.

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

8. word-spacing

- defines the spacing between the words

```
word—spacing: 5px;
```

9. Text-align

- Defines how the text content of the element is horizontally aligned.
- Values
  a. Left
  b. Right
  c. Center
  d. Justify

```
text-align: left;
```

10. Text-decoration

- Defines how the text content of the element is decorated.
- Values
  a. None
  b. Line-through
  c. Over-line
  d. Under-line

```
text-decoration: none;
```

11. Text-indent

- Defines the indentation of the element's first line of text.

```
text-indent:10px;
```

12. Text-transform

- Defines how the text content should be transformed.
- Values
  a. None
  b. Capitalize
  c. Uppercase
  d. Lowercase

```
text-transform: capitalize;
```

13. Text-shadow

- Defines the shadow of the text content.
- Values
- the first is the horizontal offset
- the second is the vertical offset
- The optional third value defines the blur of the shadow.
- You can define a color as the last value.

```
text-shadow: 2px 4px 10px red;
```

CSS border properties

1. Border-width

- Specifies the thickness of the border. You can set specific widths for each side
- **Values**: thin, medium, thick (keywords), or exact values like px, em, %, cm, etc.

```
border-width: 5px;            /* all sides */
border-width: 5px 10px;       /* top and bottom , left and right */
border-width: 5px 10px 15px;  /* top, left and right ,bottom */
border-width: 5px 10px 15px 20px; /* top right bottom left */
```

2. Border-style

- Defines the line style of the border.
- **Values**: none, solid, dotted, dashed, double, groove, ridge, inset, outset, hidden

```
border-style: solid;              /* all sides */
border-style: solid dotted;       /* top and bottom , left and right */
border-style: solid dotted dashed;  /* top, left and right ,bottom */
border-style: solid dotted dashed double; /* top right bottom left */
```

3. Border-color

- Sets the color of the border. You can use color names, hex codes, RGB/RGBA values.

```
border-color: red;                  /* all sides */
border-color: red green;            /* top and bottom, left and right*/
border-color: red green blue;       /* top ,left and right, bottom */
border-color: red green blue yellow;  /* top right bottom left */
```

4. Border

- The border shorthand can set all three primary border properties in one line, like so

```
border: 2px solid #333;
```

5. Border Side Properties

- You can control each side individually with
- border-top, border-right, border-bottom, and border-left.

```
border-top: 2px solid #ff0000;
border-right: 3px dashed blue;
border-bottom: 1px dotted green;
border-left: 4px double black;
```

6. border-radius

- Creates rounded corners. You can apply it uniformly or individually to each corner.
- Values: length or percentage, e.g., 10px, 50%

```
border-radius: 10px;
border-radius: 10px 20px 30px 40px;
border-top-left-radius:25px;
```

7. Box-shadow

- The CSS box-shadow property is used to apply one or more shadows to an element.

- Specify a horizontal and a vertical shadow:

```
box-shadow: 10px 10px;     /*x, y*/
```

- Specify a Color for the Shadow

```
box-shadow: 10px 10px lightblue;
```

- Add a Blur Effect to the Shadow

```
box-shadow: 10px 10px 5px lightblue;
```

- Set the Spread Radius of the Shadow

```
box-shadow: 10px 10px 5px 12px lightblue;
```

- Set the inset Parameter
- The inset parameter changes the shadow from an outer shadow (outset) to an inner shadow.

```
box-shadow: 10px 10px 5px lightblue inset;
```

CSS Margin Properties

1. Margin Shorthand

- The margin shorthand allows you to define all sides in a single property
- margin: [top] [right] [bottom] [left];

- 1 Value: If you specify one value, it applies to all four sides.

```
margin: 20px; /* All sides get 20px */
```

- 2 Values: The first value applies to the top and bottom, the second to the left and right.

```
margin: 20px 40px; /* 20px top/bottom, 40px left/right */
```

- 3 Values: The first value is for the top, the second for left and right, and the third for the bottom.

```
margin: 20px 40px 10px; /* 20px top, 40px left/right,
10px bottom */
```

- 4 Values: Top, right, bottom, and left, respectively.

```
margin: 10px 20px 30px 40px; /* 10px top, 20px right,
30px bottom, 40px left */
```

2. Individual Margin Properties

- CSS also provides individual margin properties for each side:
  a. margin-top: Sets the margin on the top side.
  b. margin-right: Sets the margin on the right side.
  c. margin-bottom: Sets the margin on the bottom side.
  d. margin-left: Sets the margin on the left side.

```
margin-top: 20px;
margin-right: 10px;
margin-bottom: 15px;
margin-left: 5px;
```

3. Using Auto

- The auto value is commonly used for horizontal centering of block-level elements (like divs).
- **Auto Horizontal Centering**: When you set margin-left and margin-right to auto, the element centers within its container.

```
margin: 0 auto;
```

CSS padding properties

- The padding property in CSS is used to create space inside an element, between its content and its border. Understanding how to use padding can help you control the internal spacing of elements.

1. Padding shorthand

- The padding property is a shorthand that sets padding for all four sides of an element (top, right, bottom, and left) in one line.

```
padding: [top] [right] [bottom] [left];
```

- You can specify up to four values

- One value: Applies to all four sides.

```
padding: 10px; /* top, right, bottom, and left all set to 10px */
```

- Two values: The first value applies to the top and bottom, and the second to the left and right.

```
padding: 10px 20px; /* top and bottom = 10px, left and right = 20px */
```

- Three values: The first value applies to the top, the second to the left and right, and the third to the bottom.

```
padding: 10px 20px 30px; /* top = 10px, left/right = 20px, bottom = 30px */
```

- Four values: Sets padding for each side individually, in the order: top, right, bottom, left

```
padding: 10px 20px 30px 40px;    /* top = 10px, right = 20px, bottom = 30px, left = 40px */
```

2. Individual Padding Properties

- CSS also provides individual properties to control each side of padding separately:

a. padding-top
b. padding-right
c. padding-bottom
d. padding-left

```
padding-top: 15px;
padding-right: 20px;
padding-bottom: 10px;
padding-left: 5px;
```

CSS Overflow Properties

- The CSS overflow property controls what happens when content overflows (extends beyond) an element's box.
- This is particularly useful when you have fixed dimensions for elements but unpredictable or variable content.
- Understanding the overflow property helps in managing layouts, creating scrollable areas, and ensuring that overflowing content behaves as expected.

- The overflow property has the following values:
  a. visible - Default. The overflow is not clipped. The content renders outside the element's box
  b. hidden - The overflow is clipped, and the rest of the content will be invisible
  c. scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
  d. auto - Similar to scroll, but it adds scrollbars only when necessary

```
overflow: scroll;
```

CSS Cursor Properties

1. Default Cursor
- Value: auto or default
- Description: The default cursor for the user's operating system.
- Example

```
cursor: auto; /* Default cursor */
```

2. Pointer cursor
- Value: pointer
- Description: Indicates a clickable element, such as a link or button.
- Example

```
cursor: pointer; /* Hand cursor for links */
```

3. Text cursor
- Value: text
- Description: Indicates that text can be selected.
- Example

```
cursor: text; /* I-beam cursor for text selection */
```

4. Move cursor
- Value: move
- Description: Indicates that an element can be moved.
- Example

```
cursor: move; /* Cursor indicating the item can be
dragged */
```

5. Crosshair cursor
- Value: crosshair
- Description: A simple crosshair cursor for precise selection.
- Example

cursor: crosshair; /* Crosshair cursor for a drawing area */

6. Not-Allowed Cursor
- Value: not-allowed
- Description: Indicates that an action is not permitted.
- Example

```
cursor: not-allowed; /* Cursor indicating the action is not allowed */
```

7. Progress cursor
- Value: progress
- Description: Indicates that an operation is ongoing.
- Example

```
cursor: progress; /* Cursor showing ongoing operation */
```

8. Help cursor
- Value: help
- Description: Indicates that help is available.
- Example

```
cursor: help; /* Help cursor for buttons providing assistance */
```

9. Wait cursor
- Value: wait
- Description: Indicates that the user should wait.
- Example

```
cursor: wait; /* Cursor indicating waiting state */
```

10. Zoom-in and Zoom-out Cursor

- Value: zoom-in/out
- Description: Indicates that an element can be zoomed in on.
- Example

```
cursor: zoom-in; /* Cursor indicating zoom-in capability */
```

11. Resize Cursors

- Values:
- n-resize, e-resize, s-resize, w-resize
- ne-resize, nw-resize, se-resize, sw-resize
- Description: Indicates that an edge can be resized in specific directions.
- Example

```
cursor: n-resize; /* Resize from the top */
```

CSS Visibility Property

- Values: visible, hidden, collapse (for table elements only), inherit
- Usage: Controls whether an element is visible or not, without removing it from the document layout.

a. visible: The element is visible.
b. hidden: The element is invisible but still takes up space in the layout.
c. collapse: Only applies to table rows or columns, hiding the element and collapsing its space (acting as hidden in other elements).
d. inherit: Inherits the visibility value from its parent.

```
visibility: hidden;
```

CSS Display Property

- The display property is used to specify how an element is shown on a web page.

- Every HTML element has a default display value, depending on what type of element it is. The default display value for most elements is block or inline.
-
- The display property is used to change the default display behavior of HTML elements.

1. Display : Inline

- Behavior:
- Elements with display: inline only take up as much width as necessary and do not start on a new line.
- Characteristics:
  a. Width and height properties do not apply.
  b. Padding and margins can be applied, but vertical margins (top and bottom) often do not have an effect on surrounding elements.
  c. Text flows around inline elements as they do not interrupt the flow of surrounding content.

display: inline;

2. Display : block

- Behavior:
- Elements with display: block take up the full width of their container, starting on a new line.
- Characteristics:
  a. They occupy the entire width available, pushing other elements to new lines.
  b. Both width and height properties apply.
  c. Padding, margins, and borders are fully respected in all directions.

display: block;

3. Display : inline-block

- Behavior:
- Elements with display: inline-block are similar to inline elements in that they do not force new lines. However, they retain block-level characteristics, such as respect for width and height.
- Characteristics:
  a. Allows setting width and height properties.
  b. Allows padding and margin in all directions.
  c. Stays inline, so multiple inline-block elements can sit next to each other on the same line.

```
display: inline-block;
```

4. Display : none

- Behavior:
- Completely hides the element, removing it from the document layout as if it does not exist.
- Characteristics:
  a. The element is not rendered, so it occupies no space in the layout.
  b. Other elements shift into its place as if it’s not in the DOM.
  c. Often used in JavaScript-based toggling to hide/show elements dynamically.

```
Display : none;
```

CSS Position properties

- CSS provides several position properties to control the layout and positioning of elements.

1. Position : static

- Description:
- The default positioning for all elements. It means the element is positioned according to the normal flow of the document.

- Characteristics:
  a. top, right, bottom, and left properties have no effect.
  b. The element appears in the natural document order, and it doesn't break out of the standard page flow.

- Use Case:
- Used when elements should simply follow the document flow without any special positioning.

```
.static-box {
  position: static;
}

<div class="static-box">Static Positioning</div>
```

2. Position : relative

- Description:
- The element is positioned relative to its original location in the document flow.

- Characteristics:
  a. The element remains in the flow of the document, but it can be offset by top, right, bottom, and left properties.

b. The space it originally occupied is preserved, so other elements won't shift.
c. Useful for slight adjustments while maintaining the document flow.

- Use Case:
- To slightly adjust an element's position without affecting surrounding elements, or to set a positioning context for child elements.

```
.relative-box {
  position: relative;
  top: 10px; /* Moves the element down by 10px */
  left: 20px; /* Moves the element right by 20px */
}

<div class="relative-box">Relative Positioning</div>
```

3. Position : absolute

- Description:
- The element is positioned absolutely with respect to its nearest positioned ancestor (one that has position: relative, absolute, or fixed). If no such ancestor exists, it positions relative to the <html> (root) element.

- Characteristics:
  a. Removed from the document flow, so it doesn't affect surrounding elements.
  b. top, right, bottom, and left properties control its position within the nearest positioned ancestor.
  c. Creates a flexible layout for elements that should overlap or appear in precise locations.

- Use Case:
- Used for elements that need exact positioning within a specific container, such as icons on images, floating buttons, or tooltips.

```
<div class="container">
  <div class="absolute-box">Absolute Positioning</div>
</div>


.container {
  position: relative;
  width: 200px;
  height: 200px;
  background-color: lightgray;
}

.absolute-box {
  position: absolute;
  top: 20px; /* 20px from the top of .container */
  right: 10px; /* 10px from the right of .container */
  background-color: coral;
}
```

4. Position : fixed

- Description:
- The element is positioned relative to the viewport, so it stays in a fixed position even when scrolling.

- Characteristics:
  a. Removed from the document flow.
  b. top, right, bottom, and left properties determine its fixed position relative to the viewport.
  c. The element remains in place as the page scrolls.

- Use Case:
- Often used for fixed headers, footers, floating action buttons, and modal overlays.

```
<div class="fixed-box">Fixed Positioning</div>

.fixed-box {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background-color: darkblue;
  color: white;
  text-align: center;
}
```

5. Position : sticky

- Description:
- The element is positioned relative until it reaches a defined scroll position, after which it "sticks" and behaves as fixed.

- Characteristics:
  a. Combines characteristics of relative and fixed.
  b. top, right, bottom, and left define the "sticking" position.
  c. Stays in the document flow until a specified scroll position is reached, then sticks in place.
  d. Useful for sticky headers and sidebars that should remain visible while scrolling.

- Use Case:
- Ideal for making elements that stick to the viewport after a specific point (e.g., a menu that stays at the top when scrolling).

```
<div class="sticky-box">Sticky Positioning</div>

.sticky-box {
  position: sticky;
  top: 10px; /* Will stick 10px from the top when reached
*/
  background-color: yellow;
}
```

CSS Z-Index property

- The z-index property in CSS controls the stacking order of overlapping elements.
- Elements with higher z-index values appear in front of elements with lower values

- Key Points About z-index

1. Only Works on Positioned Elements:
   a. z-index only affects elements that have position: relative, absolute, fixed, or sticky.
   b. If z-index is applied to a position: static element (the default), it won't have any effect.

2. z-index Values:
   a. Positive Integer: Higher values bring the element closer to the front.
   b. Negative Integer: Lower values push the element further to the back.

```
<div class="box1">Box 1</div>
<div class="box2">Box 2</div>
<div class="box3">Box 3</div>

.box1, .box2, .box3 {
  position: absolute;
  width: 100px;
  height: 100px;
  color: white;
}

.box1 {
  background-color: blue;
  top: 50px;
  left: 50px;
  z-index: 1; /* Behind the other boxes */
}
```

```
.box2 {
  background-color: green;
  top: 80px;
  left: 80px;
  z-index: 2; /* In front of .box1, but behind .box3 */
}

.box3 {
  background-color: red;
  top: 110px;
  left: 110px;
  z-index: 3; /* In front of .box1 and .box2 */
}
```