CSS

CSS
Cascading Style Sheet

CSS or Cascading Style Sheets, is like the fashion designer of the internet. Just as a fashion designer chooses how clothes should look, CSS decides how web pages should look.

Sure, let's explain "cascading" in simple terms:

Imagine you have a set of rules for how you want to dress. You have rules for your shirt, pants, shoes, and accessories. These rules might include the color, style, and size for each item.

Now, let's say you're getting ready in the morning, and you have multiple sources giving you these rules:

1. Your personal rulebook.
2. Your friend's suggestions.
3. Your workplace's dress code.

Cascading in CSS is a bit like this scenario. CSS rules can come from different sources, and they might have conflicting instructions for how elements on a webpage should look. When there's a conflict, the "cascading" part comes into play.

Here's how it works:

1. Specificity: Some rules might be more specific or important than others. For example, your personal rulebook is probably the most important because it's all about your style. Similarly, in CSS, more specific rules take precedence over general ones.

2. Order: If two sources give you conflicting instructions, the order matters. If your friend suggests one thing, and then your workplace suggests something else, you might go with what your workplace says because it's the most recent information. In CSS, rules that come later in the code override earlier rules if they conflict.

So, in simple terms, **cascading in CSS means that when there are different style rules for the same element, the web browser follows a set of rules to decide which one to apply**. It's like following a fashion hierarchy where some rules are more important, and the order of instructions matters when you're getting dressed.

Css Syntax:

```
Selector{property1:property1-value;property2:property2-va
lue;property3:property3-value;}
```

```
Selector{
            Property1:property1-value;
            Property2:property2-value;
            Property3:property3-value;
            }
```

Selector: Selectors are used to target HTML elements or tags to which you want to apply styles. Selectors can be based on **element names, classes, IDs, attributes, and more.**

Property: Properties are attributes you want to style, such as color, font-size, margin, padding, etc.
Each property corresponds to a specific aspect of the element's style.

Values: Values are assigned to properties to specify how you want the selected element to appear. Values can be in various formats, including colors, lengths (like pixels or percentages), fonts, and more

Eg:

```
p {
      color: red;
      font-size: 18px;
      margin-top: 10px;
  }
```

Inserting CSS:-

There are 3 ways inserting CSS in HTML file:

- External style sheet
- Internal style sheet/ Embedded style sheet
- Inline style

External Style Sheet:-

To insert an external style sheet into an HTML document, you can use the <link> element in the <head> section of your HTML file.

1. Create your external CSS file (e.g., styles.css) and define the styles you want to apply to your HTML elements.

Styles.css

```
h1 {
    color: blue;
}
p {
    font-size: 16px;
}
```

2. Linking .CSS file to the HTML file

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>My Web Page</title>
    <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
    <h1>Welcome to My Web Page</h1>
    <p>This is a sample paragraph.</p>
</body>
</html>
```

3. Save both your HTML file and your CSS file in the same directory or provide the correct relative or absolute path to your CSS file.

Internal Style Sheet:

You can insert an internal style sheet directly within the **<head> section** of an HTML document **using the <style> element.**

1. Within the <head> section of your HTML document (typically located between the <head> and </head> tags), add a <style> element.

2. Inside the <style> element, write your CSS rules to define the styles you want to apply to your HTML elements.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Internal Style Sheet Example</title>
    <!-- Internal Style Sheet -->
    <style>
        h1 {
            color: blue;
        }

        p {
            font-size: 16px;
        }
    </style>
</head>
<body>
    <h1>Welcome to My Web Page</h1>
    <p>This is a sample paragraph.</p>
</body>
</html>
```

Internal style sheets are useful when you want to apply specific styles to a single HTML document without the need for an external CSS file.

Inline Style Sheet:-

To insert inline CSS directly into an HTML element, you can use the style attribute within the specific HTML tag.

```
<!DOCTYPE html>
<html>
<head>
    <title>Inline CSS Example</title>
</head>
<body>
    <h1 style="color: darkblue;">This is a heading</h1>
    <p style="font-family: Arial, sans-serif;">This is a
paragraph.</p>
</body>
</html>
```

You can include any CSS property-value pairs within the **style** attribute to customize the appearance and behavior of the individual HTML elements. Inline CSS is useful for making quick style adjustments to specific elements.

Multiple style sheet and its priority:

Inline style sheets have the highest priority, order of priority in external and internal style sheets depends on which comes after in the order.

id selector:

In CSS (Cascading Style Sheets), an ID selector is a way to target and **style a specific HTML element based on its unique identifier.** The ID selector is denoted by a pound sign or hash (#) followed by the ID name.

Syntax:

```
#myElement {
  color: white;
  font-size: 16px;
}
```

ID selector in HTML and CSS together:

```
<div id="myElement">This is a div element with an
ID.</div>
```

Rules:

- Name cannot start with a number.
- Id should be unique for the element.
- Name should start with A-Z or a-z.
- Space is not allowed.

id name is case-sensitive.

Class selector or Style class:

In CSS (Cascading Style Sheets), a class selector is a way to target and **style one or more HTML elements** based on a specific class attribute.

There are two types of class selector or style class:

- Universal class selector
- Element specific class selector

1.Universal class selector or style classes:

Class selectors are denoted by a period (.) followed by the class name.

syntax:

css

```
.highlight {
  color: black;
  font-size: 16px;
}
```

class selector in HTML and CSS together:

HTML:

```
<p class="highlight">This is a paragraph with a class.</p>
<h1 class="highlight">This is a heading with a class.</h1>
```

Class selectors can be used for multiple elements on a page. You can apply the same class to different elements to give them a common set of styles.

2.Element specific class selector or style classes:

In element specific class selector, class selector starts with element name followed by (.) operator and class name.

In CSS, you can create class selectors that are specific to a particular type of HTML element.

css

```css
elementName.className {
  /* CSS styles go here */
}
```

element-specific class selector:

HTML:

```html
<p class="special">This is a paragraph with a special class.</p>
<button class="special">This is a button with the same special class.</button>
```

CSS:

```
p.special {
  color: blue;
  font-weight: bold;
}

button.special {
  background-color: yellow;
  border: 2px solid black;
}
```

3.Universal two or more classes on elements:

4.Element specific two or more classes:

5.Applying css on group of elements:

6.Applying same properties to group of class:

Background:

Background property is used to control the background of an element, such as a header, paragraph, or any HTML element.

1. background-color:

background-color property is used to set the background color of an element.

```
selector {
  background-color: color;
}
```

Examples:

```
/* Using a color name */
.container {
  background-color: blue;
}
```

```
/* Using an RGB value */
.paragraph {
  background-color: rgb(255, 0, 0); /* Red */
}

/* Using a hexadecimal value */
.header {
  background-color: #FFA500; /* Orange */
}


/* Using an HSL value */
.button {
  background-color: hsl(120, 100%, 50%); /* Green */
}
```

RGB value:

RGB stands for "Red, Green, Blue."
In the RGB color model, each color is defined by three numerical values, usually in the **range of 0 to 255,** representing the intensity of red, green, and blue.

These values are combined to create a specific color. For example:

(255, 0, 0) represents pure red.
(0, 255, 0) represents pure green.
(0, 0, 255) represents pure blue.
(0, 0, 0) represents black (no color, all colors at minimum intensity).
(255, 255, 255) represents white (full intensity of all colors).

hexadecimal (hex) value:

A hexadecimal (hex) color value in web design is typically represented as a six-character code that combines numbers and letters.
Each pair of characters in the hexadecimal code represents the intensity of the red, green, and blue (RGB) components of a color.
The full form of a hexadecimal color value is:

#RRGGBB

Here's what each part represents:

#: The hash symbol "#" signifies the beginning of a hexadecimal color value in CSS.

RR: The first two characters (RR) represent the intensity of the red component. This part can range from 00 (no red) to FF (full intensity red).

GG: The next two characters (GG) represent the intensity of the green component. This part can range from 00 (no green) to FF (full intensity green).

BB: The last two characters (BB) represent the intensity of the blue component. This part can range from 00 (no blue) to FF (full intensity blue).

For example:

#FF0000 represents pure red (255 red, 0 green, 0 blue).

#00FF00 represents pure green (0 red, 255 green, 0 blue).

#0000FF represents pure blue (0 red, 0 green, 255 blue).

#FFFFFF represents pure white (255 red, 255 green, 255 blue).

#000000 represents pure black (0 red, 0 green, 0 blue).

HSL value:

The HSL color model stands for "Hue, Saturation, and Lightness."
Component of HSL stands for:

Hue (H): Hue values range from 0 to 360 degrees on the color wheel, where 0 is red, 120 is green, and 240 is blue, for example.

Saturation (S): Saturation controls the intensity or vividness of the color. A saturation value of 100% means the color is fully saturated and appears as vibrant as possible, while a value of 0% results in grayscale (no color).

Lightness (L): Lightness determines how light or dark the color is. A lightness value of 0% is black, 50% is normal, and 100% is white. You can use values between 0% and 100% to adjust the brightness of the color.

For example:

hsl(0, 100%, 50%) represents pure red because the hue is 0 degrees (red on the color wheel), saturation is 100% (fully saturated), and lightness is 50% (normal brightness).

hsl(120, 100%, 50%) represents pure green because the hue is 120 degrees (green on the color wheel), saturation is 100%, and lightness is 50%.

hsl(240, 100%, 50%) represents pure blue because the hue is 240 degrees (blue on the color wheel), saturation is 100%, and lightness is 50%.

2.background-image:

In CSS, you can set a background image for an HTML element using the background-image property.

```
selector {
  background-image: url('path/to/your-image.png');
}
```

3.background-repeat:

The background-repeat property in CSS is used to control how a background image is repeated within the content area of an HTML element. It specifies whether the background image should repeat horizontally, vertically, both, or not at all.

The background-repeat property accepts the following values:

1. repeat (default): This value causes the background image to repeat both horizontally and vertically.

```
.selector {
  background-repeat: repeat;
}
```

2. repeat-x: This value repeats the background image only horizontally, creating a tiled effect from left to right.

```
.selector {
  background-repeat: repeat-x;
}
```

3. repeat-y: This value repeats the background image only vertically, creating a tiled effect from top to bottom.

```
.selector {
  background-repeat: repeat-y;
}
```

4. no-repeat: This value prevents the background image from repeating in any direction, effectively displaying it only once.

```
.selector {
  background-repeat: no-repeat;
}
```

4.background-position:

There are various background-position properties left, center, right: Specifies the horizontal position of the background image.
top, center, bottom: Specifies the vertical position of the background image.

1.left top
2.center top
3.right top
4.center left
5.center center
6.center right
7.left bottom
8.center bottom
9.right bottom
10.x%y%
11.xpx
12.ypx

5.background:

Background shorthand where we can use all property in a one line

background: url(nature.jpg) aqua no-repeat right top fixed;

6.background-size:

1. auto:

The default value. The background image is displayed at its original size.

```
.selector {
  background-size: auto;
}
```

2. cover:

The background image is scaled proportionally to **cover the entire content area of the element.** Some **parts of the image may be cropped** if the aspect ratio of the image and the container do not match.

```
.selector {
  background-size: cover;
}
```

3. contain:

The background image is scaled proportionally to fit completely within the content area of the element. This ensures that the entire image is visible, but there may be empty space around it if the aspect ratios don't match.

```
.selector {
  background-size: contain;
}
```

4. length values:

You can specify specific width and height values in pixels (px) or other length units to set a custom size for the background image. For example, background-size: 200px 100px; would set the background **image size to 200 pixels in width and 100 pixels in height.**

```
.selector {
  background-size: 200px 100px;
}
```

5. percentage values:

You can use percentage values to scale the background image based on the width and height of the containing element.

```
.selector {
  background-size: 50% 75%;
}
```

7.background-origin:

The background-origin property in CSS is used to control the positioning of the background image or color relative to the content box of an element. It determines where the background starts rendering within the element's box. The background-origin property accepts the following values:

1 padding-box (default):

The background image or color starts from the padding edge of the element's box. This means that the padding area may cover the background, and it extends to the content area.

```
css
.selector {
  background-origin: padding-box;
}
```

2. border-box:

The background image or color starts from the border edge of the element's box. This means that the border area may cover the background, and it extends to the padding and content areas.

```
css
.selector {
  background-origin: border-box;
}
```

3. content-box:

The background image or color starts from the content edge of the element's box. It is clipped by the padding and border areas, so only the content area shows the background.

```
css
.selector {
  background-origin: content-box;
}
```

Here's an example of how you can use the background-origin property in CSS:

```
css
.box {
  width: 200px;
  height: 100px;
  border: 10px solid #333;
  padding: 20px;
  background-color: lightblue;
  background-image: url('background-image.jpg');
  background-repeat: no-repeat;
  background-origin: border-box;
```

8.background-clip:

Text:

In CSS, you can apply various styles and formatting to **text** within HTML elements using CSS properties.

1.color:

Defines the color of the text characters using color names, hexadecimal values, RGB values, or HSL values.

```
selector {
  color: #FF0000; /* Red */
}
```

2.direction:

In CSS, the direction property is used to **control the text direction** within an HTML element. It is primarily used to switch between **left-to-right (LTR) and right-to-left (RTL)** text directions, which are important for languages and scripts that are written from right to left, such as **Arabic and Hebrew.**

ltr (Left-to-Right): This is the **default value** and is used for languages that are written from left to right. Text is displayed from left to right.

```
selector {
  direction: ltr;
}
```

rtl (Right-to-Left): This value is used for languages that are written from right to left. Text is displayed from right to left.

```
selector {
  direction: rtl;
}
```

3.text-align:

Specifies the horizontal alignment of the text within its container. Common values include **left, right, center, and justify.**

```
selector {
  text-align: center;
}
```

text-align: justify;, the browser adjusts the spacing between words and characters to make both the left and right edges of the text block align with the corresponding edges of the container. It spreads out the words and characters evenly to achieve this effect.

4.letter-spacing:

Adjusts the spacing between characters in text. It can be set using length values or the normal keyword.
Values can be in - px(pixel),cm(centimeter),in(inches),em

```
selector {
  letter-spacing: 2px;
}
```

5.line-height:

Controls the vertical spacing between lines of text. It can be set as a unitless number, a percentage, or a specific length value.
Values can be in - px(pixel),cm(centimeter),in(inches),em.

```
selector {
  line-height: 1.5; /* 150% of the font size */
}
```

6.text-decoration:

Adds decorative elements to text, such as underlines, overlines, or strikes. Common values include **underline, overline, line-through, and none.**

```
selector {
  text-decoration: underline;
}
```

7.text-indent:

Sets the indentation of the first line of text within a block-level element.

```
    selector {
      text-indent: 20px;
    }
```

8.text-shadow:

In CSS, the text-shadow property is used to add a shadow effect to text elements.

```
selector {
  text-shadow: horizontal-offset vertical-offset blur-radius
color;
}
```

horizontal-offset: Specifies the horizontal position of the shadow relative to the text. Negative values move the shadow to the left, and positive values move it to the right.

vertical-offset: Specifies the vertical position of the shadow relative to the text. Negative values move the shadow up, and positive values move it down.

blur-radius: Determines the blur radius of the shadow. A larger value creates a more blurred or diffuse shadow effect.

color: Specifies the color of the text shadow. You can use color names, hexadecimal values, RGB values, or HSL values to define the shadow color.

Example:
```
text-with-shadow {
  text-shadow: 2px 2px 4px red;
}
```

9.text-transform:

Changes the capitalization of text characters. Values include uppercase, lowercase, capitalize, and none.

```
    selector {
      text-transform: uppercase;
    }
```

10.vertical-align:

It specifies how the element should be positioned vertically relative to the surrounding text or other inline elements. Values can be baseline, **sub, super, top, text-top, bottom, text-bottom, middle,** or specified in the form of length and percentage.

11.white-space:

In CSS, the white-space property is used to control how white space (spaces, tabs, and line breaks) within an element's content is handled and displayed.

1. normal (default): This is the default value.

```
.selector {
  white-space: normal;
}
```

2. nowrap: Prevents text from wrapping to the next line. It collapses consecutive white spaces into a single space but does not allow line breaks. Horizontal scrolling may occur if the content exceeds the container's width.

```
.selector {
  white-space: nowrap;
}
```

3. pre: **Preserves all white space characters, including spaces, tabs, and line breaks**. Consecutive white spaces are displayed as-is. Text wraps to the next line when it reaches the container's width.

```
.selector {
  white-space: pre;
}
```

4. pre-line: Preserves line breaks but collapses other white spaces. Consecutive spaces and tabs are collapsed into a single space, while line breaks are preserved.

```
.selector {
  white-space: pre-line;
}
```

5. pre-wrap: Preserves both line breaks and other white spaces. Consecutive spaces and tabs are retained, and line breaks are preserved.

```
.selector {
  white-space: pre-wrap;
}
```

Font:

In CSS, the font property is used to specify various aspects of a font for text elements within an HTML document.

font-family:

Defines the font family for the text. You can list multiple fonts in a fallback order to ensure compatibility.

```
p{
color: rgb(211, 15, 129);
font-family: Arial, sans-serif, “Times New Roman”;
}
```

font-size:

In CSS, the font-size property is used to specify the size of text within an HTML element.
Values can be in pixels (px), ems (em), percentages (%), and more.
Using Keywords:
xx-small, x-small, small, medium, large, x-large, and xx-large are predefined keywords for font sizes.

```
selector {
  font-size: value;
}
```

font-stretch:

In CSS, the font-stretch property is used to control the width or "stretchiness" of characters in a font family.
Using keyword:
normal, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded.

Note: font-family property must be used to work with font-stretch property

```
selector {
  font-stretch: value;
}
```

font-style:

Specifies the style of the font, such as **normal, italic, or oblique.**

```
p{
    color: rgb(211, 15, 129);
    font-style: italic;
    }
```

font-variant:

Specifies whether the font should be displayed as normal or as a small-caps font.

```
    p{
    color: rgb(211, 15, 129);
    font-variant: small-caps;
    }
```

font-weight:

Defines the thickness or boldness of the font, such as **normal, bold, bolder, lighter, or numeric values** like 100, 200, etc.

```
p{
color: rgb(211, 15, 129);
font-weight: bolder;
}
```

Font shorthand:

In CSS, the font shorthand property allows you to set multiple font-related properties in a single declaration. It's a convenient way to specify the font family, style, variant, weight, size, and line height all at once.

```
selector {
  font: [font-style] [font-variant] [font-weight]
[font-size]/[line-height] [font-family];
}
```

Note:
font-size and font-family values are mandatory.

List:

In CSS, the list-style property is used to set the style of list markers (such as bullets or numbers) for HTML list elements (<ul> for unordered lists and <ol> for ordered lists).

list-style-image:

Allows you to specify a custom image as the list marker. You can use url('image-path') to specify the image file.

```
ul{
   list-style-image: url(image.jpg);
}
```

We can also set list-style-image value to none.

list-style-type:

Specifies the type of marker to be used for the list. Common values include:

disc (default): Filled circle (for unordered lists).
circle: Empty circle (for unordered lists).
square: Filled square (for unordered lists).
decimal: Decimal numbers (for ordered lists).
lower-roman: Lowercase Roman numerals (for ordered lists).

upper-roman: Uppercase Roman numerals (for ordered lists).
lower-alpha: Lowercase letters (for ordered lists).
upper-alpha: Uppercase letters (for ordered lists).
none: No marker (to remove the marker).

list-style-position:

Specifies the position of the marker relative to the list item text.
Common values include:
outside (default): The marker is placed outside the content box.
inside: The marker is placed inside the content box, near the text.

Border:

border-style:

In CSS, the border-style property is used to specify the style of the border around an element.
Syntax:

```
selector {
  border-style: value;
}
```

You can replace value with one or more of the following border style values:

1. none: No border. The element will have no visible border.

2. hidden: Similar to none, but reserved for future use.

3. dotted: Creates a border with a series of dots.

4. dashed: Creates a border with a series of dashes.

5. solid: Creates a solid line border.

6. double: Creates a double-line border.

7. groove: Creates a 3D grooved border. The appearance may depend on the colors used.

8. ridge: Creates a 3D ridge-like border. The appearance may depend on the colors used.

9. inset: Creates an inset border, giving the appearance of being pressed in.

10. outset: Creates an outset border, giving the appearance of being raised.

For 4 values

```
selector {
  border-style: top right bottom left;
}
```

For 3 values:

```
selector {
  border-style: top (right left) bottom;
}
```

For 2 values:

```
selector {
  border-style: (top bottom) (right left);
}
```

Example:

```
selector {
  border-style: solid dashed solid dashed;
}
```

border-width:

In CSS, the border-width property is used to specify the width of the border around an element. It determines how thick the border lines should be.

Note: border-style property must be used to work with border-width property.

```
selector {
  border-width: value;
}
```

You can replace value with one or more of the following options:

1. thin: A thin border.

2. medium: A medium border (default).

3. thick: A thick border.

4. <length>: A specific length value, such as px, em, rem, cm, mm, in, etc.

For 4 values

```
selector {
  border-style: top right bottom left;
}
```

For 3 values:

```
selector {
  border-style: top (right left) bottom;
}
```

For 2 values:

```
selector {
  border-style: (top bottom) (right left);
}
```

Example:

```
selector {
  border-style: solid dashed solid 5x;
}
```

border-color:

In CSS, the border-color property is used to set the color of the border around an element.

Note: border-style property must be used to work with border-width property.

For 4 values

```
selector {
  border-style: top right bottom left;
}
```

For 3 values:

```
selector {
  border-style: top (right left) bottom;
}
```

For 2 values:

```
selector {
  border-style: (top bottom) (right left);
}
```

Example:

```
selector {
  border-style: solid dashed solid dashed;
}
```

Border shorthand:

In CSS, the border property is a shorthand property that allows you to set all the individual border properties (such as width, style, and color) in a single declaration.

Syntax:

```
selector {
  border: border-width border-style border-color;
}
```

Example:

```
.selector {
  border: 3px double green;
}
```

Table:

1. border:

Border property that allows you to set all the individual border properties (such as width, style, and color) in a single declaration.

Syntax:

```
table {
  border: border-width border-style border-color;
}
```

Example:

```
Table, th, td {
  border: 3px double green;
}
```

Commonly used properties related to table borders:

2. Border-collapse:

It has two values:

separate(default): Borders are separated, and each cell and row has its own border.

collapse : Borders are collapsed into a single border, and adjacent borders are merged.

```
table {
  border-collapse: collapse; or separate
}
```

3. border-spacing:

We can use this property when **border-collapse is set to separate**, to specify the distance between adjacent table borders.

We can use the border-spacing property in **horizontal and vertical in cm and px.**

```
table {
  border-collapse: separate;
  border-spacing: 60px 20 px
}
```

In the above example 60px is for horizontal spacing and 20 px is for vertical spacing.

4. border-radius:

This property allows you to round the corners of the table, making it visually more appealing.

```
table {
  border-radius: 10px;
}
```

5. caption:

<caption> element is used to add a title or caption to an HTML <table>.
By default it is on the top.

Syntax:

```
        caption{
           caption-side: bottom;
        }

<table>
  <caption>Your Table Caption Goes Here</caption>
  <!-- Table content (thead, tbody, tr, th, td, etc.) goes here
-->
</table>
```

6. empty-cells:

It is used to show the border and background of the cell in the table.
It has two values:
show(default): By default empty-cells value is set to show.
Hide: It is used to hide the border and background of the cell in the table.

```
       table{
          border-collapse: separate;
          empty-cells: hide;
       }
```

7. table-layout:

It allows you to control how a table's layout is computed and specify whether the table should have a fixed layout or an automatic layout.

There are two possible values for the table-layout property:

1. auto (default): This means that columns will expand or contract based on the content's width. This is the default behavior.

2. fixed: In fixed table layout, the width of table columns is determined by the width of the first row of cells. Once the widths are determined, they remain fixed regardless of the content.

8. Width and height:

You can use width and height property to set the width and height of the table, table heading and table data.

Set height of table heading and table data in pixel.

```
table, th ,td{
          border: 2px solid rebeccapurple;
          border-collapse: collapse;
```

```
        }
        table{
            width: 70%;
        }
        th{
            height: 70%;
        }
        td{
            height: 7px;
        }
```

9. padding:

create space between the cell content and the cell borders. You can set padding for specific cells, rows, columns, or the entire table using CSS properties.

10. Color:

Using background-color properties we can set the color of the table.

```
    <style>
        table, th ,td{
            border: 2px solid rebeccapurple;
            border-collapse: collapse;
        }


        td{
            background-color: red;
        }
    </style>
```

Display:

In CSS, the display property is used to control how an HTML element is displayed on a web page.
There are several values for the display property, each serving a specific purpose.

Block-level element:

block-level elements are HTML elements that create block-level boxes when displayed on a web page.
Block-level elements **start on a new line** and **typically extend the full width of their parent container**. They **stack vertically** on top of each other, creating a vertical flow of content.
Examples of block-level elements:
<div>

Headings (<h1>, <h2>, <h3>, <h4>, <h5>, <h6>)
Paragraph (<p>)
List Elements (<ul>, <ol>, <li>)
Blockquotes (<blockquote>)
Forms (<form>, <input>, <textarea>, etc.)
Table Elements (<table>, <tr>, <td>, <th>)

inline elements:

In CSS, inline elements are HTML elements that generate inline-level boxes when displayed on a web page. Unlike block-level elements, inline elements **do not start on a new line and only take up as much width as necessary to contain their content**. They **stack horizontally** next to each other within the same line.
Examples of inline elements:

Text Elements (<span>, <strong>, <em> etc.)
<img> <object> <iframe>
<input>, <button>, <select>
<q>
<mark>
<small>
<sub> and <sup>

Inline block inline-block none:

inline:

- Elements with display: inline; generate an inline-level box.
- Inline-level elements do not start on a new line and only take up as much width as necessary.
- They stack horizontally next to each other within the same line.
- **Can’t set width and height properties.**

block:

- Elements with display: block; create a block-level box, which takes up the full width of its parent container by default.
- Block-level elements start on a new line and stack vertically.
- **we can set width and height properties.**

inline-block:

- Elements with display: inline-block; generate an inline-level block container.
- They behave like inline elements but **can have block-level properties like width and height.**

- Inline-block elements stack horizontally but can have their own dimensions.
- Often used for creating elements like buttons and custom-styled checkboxes.

none:

- Elements with display: none; are completely removed from the document's layout and are not visible on the web page.

visibility:

In CSS, the visibility property is used to control the visibility of an HTML element on a web page. It determines whether an element is visible or hidden, but it doesn't affect the layout of the page.

The visibility property accepts two main values:

1. visibility: visible;:

- This is the default value.
- It makes the element visible, meaning it will be displayed on the web page as usual.

2. visibility: hidden;:

- This **value hides the element without removing it** from the layout. In other words, the element is **still present** in the document structure, but it's **not visible to the user.**
- Hidden elements **take up space in the layout** as if they were visible, which can affect the positioning of surrounding elements.
- Hidden elements are often used when you want to temporarily hide an element.

```
.selector {
  visibility: visible;
}

.selector {
  visibility: hidden;
}
```

Link:

In CSS, there are several types of links, common types of links are:

- Unvisited Links - a:link { color: purple; }

- Visited Links - a:visited { color: blue; }

- Hovered Links - a:hover { color: red; }

- Active Links - a:active { color: orange; }

Note: Order of link should be the same as above mentioned otherwise it won’t work i.e
a:hover link should come after a:link and a:visited
a:active link should come after a:hover .

Focused Links (:focus):
- Focused links are links that have received keyboard focus, typically by using the "Tab" key.
a:focus { outline: 2px solid green; }

1. color:
- The color property sets the color of the text for unvisited links.
- Example: a:link { color: blue; }

2. text-decoration:
- The text-decoration property controls the decoration of text within links, such as underlined or line-throughs.
- Example: a:link { text-decoration: underline; }

3. font-weight:
   - The font-weight property controls the thickness of the text within links.
   - Example: a:link { font-weight: bold; }

4. background-color:
   - The background-color property sets the background color of links.
   - Example: a:link { background-color: yellow; }

5. outline:
   - The outline property sets the outline around links when they are focused.
   - Example: a:focus { outline: 2px solid green; }

Media Queries:

Media queries are a feature of CSS that **allow you to apply styles based on** the characteristics of the user's device, such as **screen size, resolution, orientation, and more.**

They help **make websites responsive**, meaning the layout and design can adapt to different screen sizes and devices.

Syntax:

```
@media not|only media-type and (condition) {
   /* CSS rules here */
}
```

Note: Using media-type is optional, but **it is mandatory when we are using with** not|only

For AND operators **we use and.**
For OR operators **we use comma(,).**

Media Types:

Media types define the category of devices the styles should apply to:

- **all:** Applies to all media types.
- **print:** Applies to printers and print preview mode.
- **screen:** Applies to screens such as desktops, tablets, and mobile devices.
- **speech:** Applies to speech synthesizers e.g screen reader “reads a page aloud” .

Common Media Features and Conditions:

- **width / height:** The width/height of the viewport.

```
@media (min-width: 600px) {
   body {
      background-color: lightblue;
   }
}
```

- **min-width / max-width:** Minimum/maximum width of the viewport.

```
@media (min-width: 600px) {
   body {
      background-color: lightblue;
   }
}
```

- **min-height / max-height:** Minimum/maximum height of the viewport.

```
@media screen and (max-height: 800px) {
   body {
        background-color: green;
        font-size: 14px;
          }
 }
```

- **orientation:** The orientation of the device (landscape or portrait).

```
@media (orientation: landscape) {
    body {
        background-color: lightgreen;
    }
}
```

- **resolution:** The resolution of the device, in DPI or DPCM.

```
@media (min-resolution: 192dpi) {
    body {
        font-size: 34px;
    }
}
```

- **aspect-ratio:** The aspect ratio of the viewport.

```
@media (min-aspect-ratio: 16/9) {
    body {
        background-color: red;
    }
}
```

- **device-width / device-height:** The width/height of the device’s screen.

```
@media (min-device-width: 1200px) {
    body {
        background-color: lightyellow;
    }
}
```

Box Model:

Box properties in CSS are used to define the dimensions, padding, borders, and margins of HTML elements.

The CSS box model consists of the following components:

1. Content: The actual content of the box where text and images appear.
2. Padding: The space between the content and the border.
3. Border: The border around the padding (if any) and content.
4. Margin: The space outside the border.

Main Box Properties

1. Width and Height

width: Sets the width of the content area.

height: Sets the height of the content area.

```
div {
    width: 200px;
    height: 100px;
}
```

2.Padding

Padding adds space inside the element, between the content and the border.

```
div {
    padding: 20px; /* Applies 20px padding on all sides
*/
    /* Individual sides */
    padding-top: 10px;
    padding-right: 15px;
    padding-bottom: 20px;
    padding-left: 25px;
}
```

3.Border

Borders are lines that surround the padding and content. You can set the width, style, and color of borders.

```
div {
    border: 2px solid black; /* Solid border with 2px
width */
    /* Individual sides */
    border-top: 3px dashed red;
    border-right: 4px dotted blue;
    border-bottom: 5px double green;
    border-left: 6px groove purple;
}
```

4.Margin

Margins add space outside the border, creating distance between the element and other elements.

```
div {
    margin: 20px; /* Applies 20px margin on all sides */
    /* Individual sides */
    margin-top: 10px;
    margin-right: 15px;
    margin-bottom: 20px;
    margin-left: 25px;
}
```

Box Shadow:

The box-shadow property in CSS is used to add shadow effects around an element's frame.

Syntax:

box-shadow: h-offset v-offset blur-radius spread-radius color inset;

- **h-offset**: Required. The horizontal offset of the shadow. Positive values move the shadow to the right, and negative values move it to the left.

- **v-offset**: Required. The vertical offset of the shadow. Positive values move the shadow down, and negative values move it up.

- **blur-radius**: Optional. The blur radius of the shadow. Higher values make the shadow more blurred.

- **spread-radius**: Optional. The spread radius of the shadow. Positive values expand the shadow, and negative values shrink it.

- **color**: Optional. The color of the shadow. If omitted, the shadow will be black.

Note: h-offset and v-offset is mandatory otherwise it won't work.