UNIT-3

Cascading Style Sheets (CSS)

CSS stands for Cascading ~~Syle~~ Style Sheets. CSS describes how HTML elements are to be displayed on screen. CSS is the language to style an HTML document. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

CSS Syntax:

Selector { Property: value; Property: value; ...}

(Declaration: Property: value;)

→ The selector points to html element we want to style.
→ The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property and a value separated by a colon.
→ Declaration blocks are surrounded by curly braces.

Example:- In this example <p> elements will be center-aligned, with a red text colour:

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

→ p is a selector in CSS which points to HTML element we are going to style.
→ Color is a property and red is the property value.
→ Similarly text-align is a property and center is the property value.

⊛. Inserting CSS: Inline, Internal, External: [Imp]

There are three ways of inserting a style sheet which are as follows:

1) Inline CSS: An inline style may be used to apply a unique style for a single element. To use inline styles, we add the style attribute to the relevent element. The style attribute can contain any CSS property. An inline style loses many of the advantages of a style sheet so it should be used rarely.

Example:- Inline css can be inserted in the ~~previous co~~ html code ~~we defined~~ as follows:-

```
<!DOCTYPE html>
<html>
    <body>
        <h1 style="color:blue; text-align:center;">This is
        heading </h1>
        <p style="color:red;" This is a paragraph. </p>.
    </body>
</html>
```

2) Internal CSS: An internal style may be used if one single html page has a unique style. The internal style is defined inside the <style> element, inside the head section. Inline and Internal css are used within the html file.

Example:-

```
<!DOCTYPE html>
<html>
    <head>
        <style> h1{ color:blue; text-align:center; }
                P{ color:red; }
        </style>
    </head>

    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

⇒ External CSS: It is the best way of inserting css. All the styles are applied to the external seperate css file having file extension .css and HTML page includes refrence to that external .css file. With an external style sheet, we can change the look of an entire website by changing just one file. External styles are defined within the <link> element, inside the <head> section of HTML page.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="mystyle.css">
    </head>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

The external .css file will be seperatly written externally as:

mystyle.css

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

can be written in single line or multiple lines as our choice

⊛ Class ID and Class Selectors: [Imp]

i.e, CSS selectors

CSS selectors are used to find or select the HTML elements we want to style.

⇒ The CSS element Selector: The element selector selects HTML elements based on the element name.

Example: Here all <p> elements on the page will be center-aligned with a red text colour.

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

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

Example: The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 { text-align:center;
         color:red;
       }
```

Note: A class or id name can not start with number if it is invalid.

iii) **The CSS class Selector:** The class selector uses the class attribute of an HTML element to select a specific element. The class of an element is not unique within the page, since same class name can be used to select multiple elements. Writing same class name to many elements will apply same css to many elements at once. To select an element with a class, we write a period (.) character, followed by the class name.

Example: The CSS rule below will be applied to the HTML element(s) with class="para1":

```
.para1 { text-align:center;
         color:red;
       }
```

iv) **The CSS Universal Selector:** The universal selector (*) selects all HTML elements on the page.

Example:

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

v) **The CSS Grouping Selector:** If we are going to apply same style to many html elements, class, id then instead of writing same code for each of the elements, we can group them in one and apply css to all writing ~~only~~ css only once.

Example:- let we have to apply same css color:red to element p, class para1 and id para2 then it can be done as follows:

```
p, .para1, #para2 { color:red;
}
```

⊛. CSS Colors:

CSS uses the color values to specify the colors. Typically color property is used to set the background or font color on the page. We can use this property for the border-color and other decorative elements. We can specify color values using different methods like Built-in Color Name, Hexadecimal value, RGB value, RGBA Format etc.

i) **Built-in Color Name:-** As the name suggests, the built-in color name is the collection of defined colors that are used by just their names such as red, green, blue and so on.

ii) **Hexadecimal value:-** The hexadecimal is a six-digit representation of the color. It is denoted by # symbol followed by six characters range from 0 to F. The first two digits of the hexadecimal value represent colour value for the red, next two digits represent colour value for green and final two digits represent colour value for blue.

iii) **RGB value:-** The RGB format is the short form for red, green and blue. We specify the color value using the rgb() property. The color value can be any inter value from 0 to 255. It can also be a percentage. All the browsers do not support rgb() property of color, so it is recommended not to use it. It is better to use hexadecimal value.
Syntax: rgb(R,G,B).

iv) **RGBA Format:-** The RGBA format is similar to the RGB format except that RGBA contains A (Alpha) that specifies the elements transparency. It ranges between 0.0 to 1.0.
Syntax: rgba(R,G,B,A).

v) **HSL Format:** HSL is a short form for Hue, Saturation, Lightness. Hue can be defined as a degree on the color wheel ranging between 0 and 360, where 0 is represented by red, 120 by green and 240 by blue. The value of saturation is in percentage. 100% means the color will be fully saturated. Lightness is also a percentage. 0% is black and 100% is white. Syntax: color(H,S,L).

⊛. CSS Backgrounds:-

The CSS background properties are used to add background effects for elements. CSS background can be any color, image or any other element.

1) CSS background-color:

The background-color property specifies the background color of an element.

Example:- The background color of a page is set like this:

```
body {
    background-color: lightblue;
}
```

2) CSS background-image:

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.

Example:- The background image is set like this:

```
body{
    background-image: url("roshan.jpg");
}
```

CSS background-repeat: By default, the background-image property repeats an image both horizontally and vertically.

→ If we want to repeat image vertically then set:

```
background-repeat: repeat-y;
```

→ If we want to repeat image horizontally then set:

```
background-repeat: repeat-x;
```

→ If we do not want to repeat image then set:

```
background-repeat: no-repeat;
```

CSS background-attachment: The background-attachment property specifies whether the background image should scroll or be fixed. If we want scroll then we set its value scroll otherwise we set its value as fixed. By default its value is scroll.

⊛. CSS Borders:-

The CSS border properties allow us to specify the style, width, and color of an element's border.

Example:-
```
p{
    border: 1px solid black;
}
```

The border-style property specifies what kind of border to display. The following values are allowed:

- dotted → defines a dotted border.
- dashed → defines a dashed border.
- solid → defines a solid border.
- double → defines a double border.
- inset → defines a 3D inset border.
- outset → defines a 3D outset border.
- none → defines no border.
- hidden → defines a hidden border.

⊛. CSS Text:-

CSS has a lot of properties for formatting text.

1) **Text Color:** The color property is used to set the color of the text. The color is specified by;
   - → a color name (like "red").
   - → a HEX value (like "#ff0000").
   - → an RGB value - like "rgb(255,0,0)".

   The default text color for a page is defined in the body selector.

Example:
```
body{
    color: blue;
}
h1{
    color: green;
}
```

2) **Text Alignment:** The text-align property is used to set the horizontal alignment of a text. A text can be left or right aligned, centered or justified.

Example:
```
h1{
    text-align: center;
}
```

iii) Text Decoration: The text-decoration property is used to set or remove decorations from text. The value text-decoration: none; is often used to remove underlines from links;

Example:
```
a{
    text-decoration: none;
}
```

iv) Text Transformation: The text-transform property is used to specify uppercase and lowercase letters in text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example:
```
p{
    text-transform: uppercase;
}
```

v) Text Spacing:

→ The text-indent property is used to specify the indentation of the first line of a tex: Example: p{text-indent: 50px}

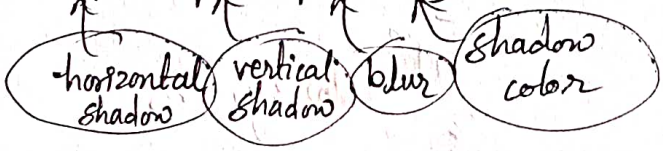
→ The letter-spacing property is used to specify the space between the characters in a text. Example
```
p{
    letter-spacing: 5px;
}
```

→ The line-height property is used to specify space between lines. Example: p{line-height: 0.8;}

→ The word-spacing property is used to specify the space between the words in a text. Example: p{word-spacing: 5px;}

vi) Text Shadow: The text-shadow property adds shadow to text.

Example:
```
p{
    text-shadow: 2px 2px 5px red;
}
```



⊛. CSS Fonts:

Choosing the right font has a huge impact on how the readers experience a website, ~~becau~~ so choosing the right font is important. The font adds value to our text. It is also important to choose the correct color and text size for the font.

In CSS, we use the font-family property to specify the font of a text. The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. In CSS there are five generic font families: Serif, Sans-serif, Monospace, Cursive, and Fantasy.

Example:
```
.p1{
    font-family: "Times New Roman", Times, serif;
}
```

⊛. CSS lists:-

In HTML, there are two main types of lists: unordered lists (<ul>) and ordered lists (<ol>). The CSS list property allow us to:

→ Set different list item markers for ordered lists and unordered lists like circle, square, upper-roman, lower-alpha etc.

Example:
```
ul.a{
    list-style-type: circle;
}
```

→ Set an image as the list item marker.

Example:
```
ul{
    list-style-image: url("sqpurple.gif");
}
```

→ Add background colors to lists and list item.

Example:-
```
ul{
    background: #ff9900;
}
```

⊛. CSS Tables:-

Different CSS properties are used to make table more attractive and readable.

1) Table Borders: To specify table borders in CSS, we use border property. Example:-
```
table, th, td{
    border: 1px solid black;
}
```

2) Table Size:- The table size is defined by the width and height properties. Example:-
```
table { width: 100%; }
th{height: 50px; }
```

iii) Table Alignment: The text-align property sets the horizontal alignment.

Example: td { text-align: center; }

The vertical-align property sets the vertical alignment. like top, bottom. or middle.

Example: td { vertical-align: bottom; }

iv) Table Style:-

→ ~~Table~~ To control the space between the border and content in the table we use padding property on <td> and <th> elements. Example: th, td { padding: 15px }

→ We add the border-bottom property to <td> and <th> for horizontal dividers: Example: th, td {
border-bottom: 1px solid #ddd;
}

→ We use :hover selector on <tr> to highlight table rows on mouse over:

Example: tr:hover { background-color: yellow; }

→ For zebra-striped tables, we use nth-child() selector and add a background-color to all even or odd table rows;-

Example: tr:nth-child(even) { background-color: #f2f2f2; }

→ We can specify text color and background color of <th> elements.

Example: th { background-color: #04AA6D;
color: white;
}

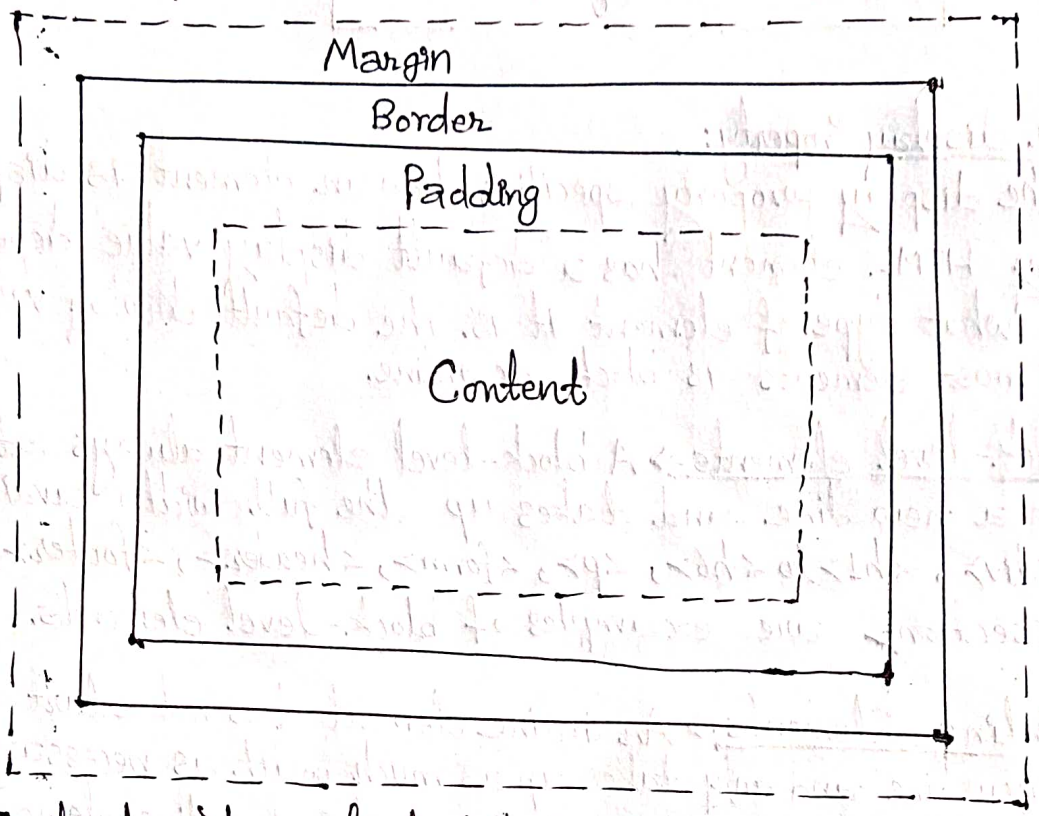
v) Responsive Table:-

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content. For this we add a container element (e.g.) with overflow-x: auto around the <table> element to make it responsive.

Example:
```
<div style="overflow-x: auto;">
  <table>
    ... table content ...
  </table>
</div>
```

⊛ CSS Box Model:

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when taking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding and the actual content. The image below illustrates the box model:



Content → The content of the box, where text and images appear.

Padding → Transparent area or space between the border and content. It is the space inside the border.

Margin → It is a transparent area or space outside the border.

Border → It is an area that goes around padding and content.

Box Layout: The width and height attribute are used to create the box layout in HTML using CSS which is as illustrated in the example below:-

Example:
```
<html>
<head><style>
div{
    background-color: none;
    width: 300px;
    border: 1px solid black;
</style></head>
```

```
</head>
<body>
    <div>This is normal layout</div>
</body>
</html>
```

Output:

This is normal layout

⊛. The display Property:

The display property specifies how an element is displayed. 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.

Block-level elements → A block-level element always starts on a new line and takes up the full width available. , <h1> to <h6>, <p>, <form>, <header>, <footer>, <section> are examples of block-level elements.

Inline Elements → An inline element does not start on a new line and only takes up as much width as necessary. , <a>, <img> are examples of inline elements.

Display: none; display:none; is commonly used with JavaScript to hide and show elements without deleting and recreating them.

⊛. CSS Padding:-

The CSS padding properties are used to generate space around an element's content, inside of an defined borders. With CSS, we have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom and left). So, we have properties: padding-top, padding-right, padding-bottom and padding-left. All padding properties can have length values in px, cm, rem, em etc. OR % value. Negative values are not allowed.

Example:

```
div{
    padding-top: 50px;
    padding-right: 30px;
    padding-bottom: 50px;
    padding-left: 80 px;
}
```

Padding-Shorthand Property: To shorten the code, it is possible to specify all the padding properties in one property. The padding property is the shorthand property for which syntax is as follows: padding: top. right bottom. left;

Example:- Above example can be written in shorthand as;

```
div{
    padding: 50px 30px 50px 80px;
}
```

⊛. CSS Margins:

The CSS margin properties are used to create space around elements, ~~outside~~ outside of any defined borders. With CSS we have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom and left). So we have properties: margin-top, margin-right, margin-bottom and margin-left. All the margin properties can have values in px, cm, %, rem, em etc. Negative values are allowed.

Example:

```
div{
    margin-top: 100px;
    margin-right: 50px;
    margin-bottom: 100px;
    margin-left: 50px;
}
```

Margin-Shorthand Property: To shorten code, it is possible to specify all margin properties in one property using margin shorthand property.

Syntax: margin: top right bottom left;

Example: Above example can be written in shorthand as;

```
div{
    margin: 100px 50px 100px 50px;
}
```

⊛. CSS Positioning:-

The position property specifies the type of positioning method used for an element. There are five different position values: static, relative, fixed, absolute and sticky. Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the ~~property~~ position property is set first. HTML elements are positioned static by default. which is not affected by top, bottom, left, right properties and positioned according to normal flow of page.

position: relative;

An element with position: relative; is positioned relative to its normal position. Setting the top, right, bottom and left properties will cause it to be adjusted away from its normal position.

Example:
```
div {
    position: relative;
    left: 50%;
    top: 50%;
}
```

position: fixed;

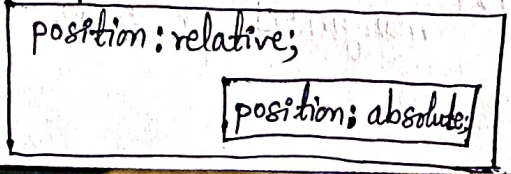
An element with position: fixed; is positioned such that it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

Example:
```
div {
    position: fixed;
    top: 0px;
    left: 0px;
}
```

position: absolute;

An element with position: absolute is positioned relative to the nearest positioned ancestor. If it has no positioned ancestors, it uses document body and moves along with page scrolling. Absolute positioned elements are removed from the normal flow, and can overlap elements.

Example: Consider figure



Here in figure outer div is set as position:relative and inner div is set as position:absolute. Now inner div is positioned with relative to the outer div using CSS properties.

Example:

```
div.relative{ position:relative;
}
div.absolute{ position:absolute;
              top:80px;
              right:0px;
}
```

position:sticky;

An element with position:sticky; is positioned based on the user's scroll position. A sticky element toggles (i.e, changes) between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport, then it "sticks" in place like position:fixed;.

Example: In this example, the element sticks to the top (top:0) when we reach its scroll position.

```
div.sticky{ position:sticky;
            top:0;
}
```

⊛ Box-shadow Property:-

The CSS box-shadow property applies shadow to elements. It is also specified similar to as text-shadow. In its simplest use, we only specify the horizontal shadow and the vertical shadow.

Example:

```
div{ box-shadow:10px 10px 10px grey;
}
```

(10px: horizontal shadow; 10px: vertical shadow; 10px: blur; grey: shadow colour.)

We can also apply shadow to create paper-like cards applying shadow in all directions as follows:-

```
div.card{
        box-shadow:2px 2px 3px #444, -2px -2px 3px #444;
}
```

We can also apply as follows:-

```
box-shadow:0 4px 8px rgba(244,0,0,0.2), 0 6px 20px rgba(244,0,0,0.19);
```

⊛. CSS Text Effects:-

In CSS text effects we have following properties:-

i) CSS text overflow:-

The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled (viewed) to the user.

It can be clipped: This is some long text that

OR it can be rendered as an ellipsis (...);

This is some long text that...

Example:-
```
p{
text-overflow:ellipsis;
}
```

ii) CSS word Wrapping:-

The CSS word-wrap property allows long words to be able to be broken and wrap into the next line. If a word is too long to fit within an area, it expands outside:

e.g. This pargraph contains a very long word.

The word wrap property allows us to force the text to wrap:

e.g. This paragraph contains a very long word.

Example:
```
p{
word-wrap:break-word;
}
```

iii) CSS word Breaking:-

The CSS word-break property specifies line breaking rules.

This paragraph con tains some text.

Example:-
```
p{
word-break :break-all;
}
```

iv) CSS writing Mode:-

The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically. It can have values: horizontal-tb, vertical-rl.

Example:
```
p{
writing-mode:vertical-rl;
}
```

⑧. Responsive Web Design:

Web pages (websites) can be viewed using many different devices: desktops, tablets, and phones. Web page should look good and easy to use regardless of the device. Responsive web design makes our web page look good on all devices. Web pages should not leave out information to fit smaller devices, but rather adopt it's content to fit any device. It is called responsive web design when we use CSS and html to resize, hide, shrink, enlarge or move the content to make it look good on any screen.

Viewport:- The user's visible area of a webpage is called viewport. The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen. Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size. But now we need web pages that changes according to the viewport of device to look good.

HTML5 introduced a method to let web designers take control over the viewport, through `<meta>` tag.

`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

This gives the browser instructions on how to control the page's dimensions and scaling. The "width=device-width" part sets the width of the page to follow the screen-width of the device. The "initial-scale=1.0" part sets the initial zoom level when the page is first loaded by the browser. This viewport through `<meta>` tag is not a perfect method for responsive design but it is a quick fix method and helps ~~alto~~ alot in responsive designs.

Media Queries:- Media query is a CSS technique introduced in CSS3. It uses the `@media` rule to include a block of CSS properties only if a certain condition is true to page a web page responsive.

Example: If browser window is 600px or smaller, the div width will be 360px.

```
@media only screen and (max-width:600px){
    div{
        width:360px;
    }
}
```

There are tons of screens and devices with different heights and widths, so it is hard to create an exact breakpoint for each device. To keep things simple we could target five groups:

This is comment. In CSS comments are written inside /*....*/

```
/*Extra small devices (phones, 600px and down)*/
@media only screen and (max-width: 600px) {...}

/*Small devices (portrait tablets and large phones, 600px and up)*/
@media only screen and (min-width:600px) {...}

/*Medium devices (landscape tablets, 768px and up)*/
@media only screen and (min-width: 768px) {...}

/*Large devices (laptops/desktops, 992px and up)*/
@media only screen and (min-width: 992px) {...}

/*Extra large devices (large laptops and desktops, 1200px and up)*/
@media only screen and (min-width: 1200px) {...}
```

- Media Queries can also be used to change layout of a page depending on the orientation of the browser:-

Example:

```
@media only screen and (orientation:landscape){
    body{
        background-color:lightblue;
    }
}
```

⑧. Introduction to Bootstrap:-

Bootstrap is a free and open-source most popular CSS Framework for developing responsive and mobile-first websites. Bootstrap 5 is the newest version of bootstrap. Bootstrap makes things a whole lot eaiser. Bootstrap can help speed up development times, while maintaining quality and consistency across the site. There are so many bootstrap components likes header, navigation, cards, slider etc which we can use easily and save our time. With the help of these we don't need to spend hours to design any element.

Installation:

Bootstrap can be installed in our web page in two ways:
One by downloading compiled code for bootstrap which includes compiled and minified CSS bundles as well as compiled and minified javascript plugins but doesn't include documentation, source files, or any optional JavaScript dependencies (jQuery and Popper.js).

Another method by which we can install bootstrap on our web page is by using Bootstrap CDN. BootstrapCDN is a link that connects web page with bootstrap. We can get this CDN link from getbootstrap.com and copy and paste that link in the <head> tag of our web page. Using CDN link is better in case of performance. Once we have used CDN link for bootstrap in our web page now we can easily use bootstrap on our web page.