UNIT – III

CSS(cascading style sheet)

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

What does CSS do

- You can add new looks to your old HTML documents.
- You can completely change the look of your website with only a few changes in CSS code.

Why use CSS

These are the three major benefits of CSS:

1) Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

2) Saves a lot of time

CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

Types of CSS

CSS can be added to HTML documents in 3 ways:

- **Inline** - by using the `style` attribute inside HTML elements
- **Internal** - by using a `<style>` element in the `<head>` section
- **External** - by using a `<link>` element to link to an external CSS file

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the `style` attribute of an HTML element.

The following example sets the text color of the `<h1>` element to blue, and the text color of the `<p>` element to red:

Example

```
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

```
<html>
<head>
<style>
body{background-color: powderblue;}
h1{color: blue;}
p{color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the <head> section of each HTML page:

Example

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

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Note: With an external style sheet, you can change the look of an entire web site, by changing one file!

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is what the "styles.css" file looks like:

"styles.css":

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

Common Tasks with CSS

Setting colors and backgrounds (as previously described) are among the most common tasks performed by CSS. Other common tasks include setting fonts and white space around elements. This section guides you through the most commonly used properties in CSS.

Common Tasks: Fonts

Let's start with fonts. If you have used desktop-publishing applications in the past, you should be able to read this little style sheet:

```
h1 { font: 36pt serif }
```

The font property is a shorthand property that simultaneously sets several other properties. By using it, you can shorten your style sheets and set values on all the properties it replaces. If you choose to use the expanded version, you would have to set all these to replace the previous example:

```
H1 {  font-size: 36pt;  font-family: serif;  font-style: normal;  font-weight: normal;  font-variant: normal;  line-height: normal; }
```

Sometimes, you only want to set some of these. For example, you may want to emphasize the list items by setting the font weight to bold and the font style to `italic`. You can emphasize all list items by setting the declarations on the their ancestor element:

```
ul {  font-style: italic;  font-weight: bold; }
```

Common Tasks: Margins

Setting space around elements is a basic tool in typography. The headline above this paragraph has space above it and (slightly less) space below it. This paragraph, as printed in this book, has space on the left and (slightly less) on the right. CSS can be used to express how much space should exist around different kinds of elements.

Font Metrics Used By CSS

- Ascent & Descent (of varying kinds)
- Underline & Strikethrough Position & Thickness.
- Superscript & Subscript Position & Size.
- Cap Height, Ex Height.
- Baselines (All of Them)

Create a webpage to add above CSS

HTML Part.

```
<html>
    <head>
    <link rel="stylesheet" href="style1.css">
      </head>
    <body>
<p>Let's start with fonts. If you have used
    desktop-publishing applications in the past,
     you should be able to read this little style sheet:</p>
    </body>
</html>
```

CSS Part.

```css
p{
font: 30pt serif;
font-family: serif;
font-weight: bold;
font-style: italic;
color: blue;
margin: 30px;
padding: 5px;
}
```

CSS Length Units

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as `width`, `margin`, `padding`, `font-size`, etc.

Length is a number followed by a length unit, such as `10px`, `2em`, etc.

Positions in CSS

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

The position Property

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

Example:

```
<html>
    <head>
    <style>
      div{
        height: 250px;
        width: 260px;
        border: 1px solid black;
        position: relative;
        bottom: -200px;
        left: 200px;
      }
    </style>
      </head>
    <body>
<p>Let's start with fonts. If you have used
    desktop-publishing applications in the past,
     you should be able to read this little style sheet:</p>

     <div>div1</div>
     <p>Let's start with fonts. If you have used
      desktop-publishing applications in the past,
       you should be able to read this little style sheet:</p>
       <p>Let's start with fonts. If you have used
        desktop-publishing applications in the past,
         you should be able to read this little style sheet:</p>

    </body>
</html>
```

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

```
<html>
    <head>
    <style>
      div{
        height: 250px;
        width: 260px;
        border: 1px solid black;
        position: absolute;
        left: 200px;
        top: 200px;
      }
    </style>
      </head>
    <body>
<p>Let's start with fonts. If you have used
    desktop-publishing applications in the past,
     you should be able to read this little style sheet:</p>

     <div>div1</div>
     <p>Let's start with fonts. If you have used
      desktop-publishing applications in the past,
       you should be able to read this little style sheet:</p>
       <p>Let's start with fonts. If you have used
        desktop-publishing applications in the past,
         you should be able to read this little style sheet:</p>

    </body>
</html>
```

position:fixed;

html:

```
 <h1>External CSS</h1>
      <div>division tag</div>
      <pre>sahyog college thane
         west pincode:400601
```

```
        </pre>
```

CSS:

```
body{
    background-color: aquamarine;
}
h1{
    font-size: 30px;
    font-family: serif;
    font-weight: bold;
}
p{
    font-size: 25px;
    color: brown;
}
div{
    height: 300px;
    width: 300px;
    border: 1px solid black;
    position: fixed;
    top: 30px;
    left: 40px;
}
```

position:sticky;

```
 <h1>External CSS</h1>
      <div>division tag</div>
      <pre>sahyog college thane
         west pincode:400601
        </pre>
```

CSS:

```
body{
    background-color: aquamarine;
}
h1{
    font-size: 30px;
    font-family: serif;
```

```
    font-weight: bold;
}
p{
    font-size: 25px;
    color: brown;
}
div{
    height: 300px;
    width: 300px;
    border: 1px solid black;
    position: sticky;
    top: 30px;
    left: 40px;
}
```

Keywords as values

Many CSS properties have specific keywords which may be used as values; in addition, the keywords initial and inherit may be used as the value of any CSS property. The keywords relevant to a specific CSS property are defined in its property page.

inherit keyword:

```
<html>
    <head>
    <style>
      div{
        height: 250px;
        width: 260px;
        border: 1px solid black;
        color: brown;
      }
      p{
        color: inherit;
      }
    </style>
      </head>
    <body>
      <div>
        <h1>heading</h1>
<p>Let's start with fonts. If you have used
```

```
    desktop-publishing applications in the past,
     you should be able to read this little style sheet:</p>
    </div>
    </body>
</html>
```

initial keyword:

```
<html>
    <head>
    <style>
      div{
        height: 250px;
        width: 260px;
        border: 1px solid black;

        color: brown;
      }
      p{
        color: initial;
      }

    </style>
      </head>
    <body>
      <div>
        <h1>heading</h1>
<p>Let's start with fonts. If you have used
    desktop-publishing applications in the past,
     you should be able to read this little style sheet:</p>
    </div>
    </body>
</html>
```

Font-size as keyword:

```
<html>
    <head>
    <style>
      div{
        height: 250px;
        width: 260px;
        border: 1px solid black;
```

```
    font-size: 30px;
      }
      p{
        font-size: initial;
      }
    </style>
      </head>
    <body>
      <div>
        <h1>heading</h1>
<p>Let's start with fonts. If you have used
    desktop-publishing applications in the past,
     you should be able to read this little style sheet:</p>
    </div>
    </body>
</html>
```

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

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, write a hash (#) character, followed by the id of the element.

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, write a hash (#) character, followed by the id of the element.

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

```
<h1 id="center">Red and center-aligned heading</h1>

<p id="center">Red and center-aligned paragraph.</p>
```

Note: An id name cannot start with a number!

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

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

```
<h1 class="center">Red and center-aligned heading</h1>

<p class="center">Red and center-aligned paragraph.</p>
```

Note: A class name cannot start with a number!

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

```
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1>Hello world!</h1>
<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Layer tag CSS

The CSS layers refer to applying the z-index property to elements that overlap with each other. The z-index property is used along with the position property to create an effect of layers. You can specify which

element should come on top and which element should come at bottom.

Create a webpage to display layer as per user requirement.

html

```
<html>
  <head>
    <title>layer css</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="a"></div>
    <div class="b"></div>
    <div class="c"></div>
    <div class="d"></div>
  </body>
</html>
```

Css

```
div{
    height: 150px;
    width: 500px;
    border: 1px solid black;
}
.a{
    background-color: brown;
    position: relative;

}
.b{
    background-color: gray;
    position: relative;
    left: 60px;
    top: -50px;
    z-index: 2;
}
.c{
    background-color: skyblue;
    position: relative;
    left: 120px;
    top: -100px;
```

```
    z-index: 3;
}
.d{
    background-color: rosybrown;
    position: relative;
    left: 180px;
    top: -200px;
    z-index: 4;
}
```

Css Tags

The **w3-tag** class creates a rectangular tag (label or sign). The default color is black:

```
  <head>
    <title>layer css</title>
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  </head>
  <body>
 <p>Hey <span class="w3-tag">Siri</span></p>
  </body>
```

The **w3-red** class creates a rectangular tag (label or sign). The default color is red:

```
<head>
    <title>layer css</title>
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  </head>
  <body>
 <p>Hey <span class="w3-red">Siri</span></p>
  </body>
```

Tag Sizes

By default, a tag will inherit the size of its container.

The **w3-*size*** classes (w3-tiny, w3-small, w3-large, w3-xlarge, w3-xxlarge, w3-xxxlarge, w3-jumbo) can be used to modify the size of a tag:

```
<head>
    <title>layer css</title>
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  </head>
  <body>
 <span class="w3-tiny w3-jumbo w3-tag w3-blue"> Hey Siri</span>
 <span class="w3-xlarge w3-blue w3-tag"> Hey Siri</span>
  </body>
```

Letter Tags

```
<head>
    <title>layer css</title>
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  </head>
  <body>
    <span class="w3-tag w3-xlarge">A</span>
    <span class="w3-tag w3-xlarge">U</span>
    <span class="w3-tag w3-xlarge">G</span>
    <span class="w3-tag w3-xlarge">U</span>
    <span class="w3-tag w3-xlarge">S</span>
    <span class="w3-tag w3-xlarge">T</span>
  </body>
```

Spinning Tags

The **w3-spin** class can be used to let a sign spin 360 degrees:

```
<head>
    <title>layer css</title>
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  </head>
  <body>
    <span class="w3-tag w3-orange w3-spin w3-large" style= "margin-top:100px">
      STOP
      </span>
  </body>
```

HTML 4 Drawbacks/Limitations

- **Audio support-** You can't add Add audio to a web page with a single tag in HTML4.
- Using javascript for animation, drawing and other feature was the toughest task.
- **Video Support-** Unable to add Video with a single tag in the website.
- **External Plugins** – Requires external plugins to run flash, media controls.
- Very few input controls are available. E.g. form inputs.
- **2-3D Supports-** Does not support 2D and 3D animations.
  This problem facing web development is matters to the developer. Audio and viodo support is very poor. Due to solve the direct play video, audio, etc **X HTML** was the next version but interchangeability problems, it took a backseat and **HTML 5** would be the next standard for Web site development.

HTML 4.01 is the most widely used version throughout the **year 2000**. Where it was introduced official standard in **December 1999**. Since HTML version 1 to 4, there is many improvements and eventually its happening.

HTML5 introduction

HTML5 tutorial provides details of all 40+ HTML tags including audio, video, header, footer, data, datalist, article etc. This HTML tutorial is designed for beginners and professionals.

HTML5 is a next version of HTML. Here, you will get some brand new features which will make HTML much easier. These new introducing features make your website layout clearer to both website designers and users. There are some elements like <header>, <footer>, <nav>, <audio>, <video>, <details> and <article> that define the layout of a website.

Features /Advantages of HTML5

It is enriched with advance features which makes it easy and interactive for designer/developer and users.

It allows you to play a video and audio file.

It allows you to draw on a canvas.

It facilitate you to design better forms and build web applications that work offline.

It provides you advance features for that you would normally have to write JavaScript to do.

The most important reason to use HTML 5 is, we believe it is not going anywhere. It will be here to serve for a long time according to W3C recommendation.

HTML 5 Deprecated (removed) Tags:

Complete list of deprecated tags are given below.

TAGS	*DESCRIPTIONS*	*ALTERNATIVE TAGS*
<basefont>	To Specify a basefont.	font style sheets.
<font>	It specifies font text, size and color.	font-family, font-size, color.
** **	It specifies a centered Text.	text-align:center.
<strike>	It specifies a strike-through text.	text-decoration.
<big>	Defines big text.	Use CSS properties or Heading tags.
<dir>	It specifies a directory list.	ul tag.
<isindex>	It specifies a single-line input field.	form tag.
<applet>	It specifies an applet.	object tag.
<acronym>	An acronym is defined using this tag.	Use abbr.

TAGS	DESCRIPTIONS	ALTERNATIVE TAGS
<noframe>	It is used to define a noframe section.	Either use an iframe and CSS instead or use server-side include to generate complete pages with the various invariant parts merged in.
<xmp>	Renders text between the start and end tags without interpreting the HTML in between and using a monospaced font.	Use pre and code instead.
<noembed>	A tag makes it easy to supply alternative content that tells users what they are missing.	Use object instead of embed when fallback is necessary.
<plaintext>	It is used to render all text in the document exactly as it was typed in.	Use the “text/plain” MIME type instead.
<frameset>	It specifies a set of frames.	Either use iframe and CSS instead or use server-side include to generate complete pages with the various invariant parts merged in.
<frame>	It specifies a frame.	Either use iframe and CSS instead or use server-side include to generate complete pages with the various invariant parts merged in.
<u>	It specifies an underlined text.	text-decoration.

TAGS	DESCRIPTIONS	ALTERNATIVE TAGS
<tt>	Defines teletype text.	Use CSS properties.
<s>	Specify a strike-through text.	text-decoration.

HTML Deprecated Attributes: Several tag attributes are also removed. Following is the table having removed attributed and their corresponding impacted tags (elements) ie. elements from which those attributes have been removed permanently.

REMOVED ATTRIBUTES	TAGS IMPACTED	ALTERNATIVES
rev	a, link	Use the rel attribute.
longdesc	img, iframe	Use a regular element to link to the description.
version	Html	Unnecessary, not required to mention.
charset	a, link	Use an HTTP Content-Type header on the linked resource instead.
name	a, img	Use the id attribute instead.
nohref	Area	Omitting the href attribute is sufficient. The nohref attribute is unnecessary, not required to mention.
usemap	Input	Use img instead of input for image maps.
target	Link	Unnecessary, not required to mention.

REMOVED ATTRIBUTES	*TAGS IMPACTED*	*ALTERNATIVES*
scheme	Meta	Use only one scheme per field, or make the scheme declaration part of the value.

Overview of html5 document

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
</body>
</html>
```

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `<span>` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

Semantic Elements in HTML

Many web sites contain HTML code like: to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

Html5 form / web 2.0 form

Web Forms 2.0 is an extension to the forms features found in HTML4. Form elements and attributes in HTML5 provide a greater degree of semantic mark-up than HTML4 and free us from a great deal of tedious scripting and styling that was required in HTML4.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>
</head>
<body>
  <form action="">
    <label for="">Enter Your Name</label><br>
    <input type="text" placeholder="Enter Name"><br>
    <label for="">Enter Your Email</label><br>
    <input type="text" placeholder="Enter Email-Id"><br>
    <label for="">Enter Your Contact</label><br>
    <input type="tel" placeholder="Enter Contact" required="required"
pattern="[0-9]{10}"><br>
    <label for="">Enter Your Address</label><br>
   <textarea name="" id="" cols="22" rows="2"
placeholder="Address"></textarea><br>
```

```
   <label for="">Enter Your DOB</label><br>
   <input type="date"><br>
   <label for="">Enter Your Gender</label><br>
   <input type="radio" name="a">Male
   <input type="radio" name="a">Female <br>
   <input type="file"><br>
   <input type="submit" name="" value="Register Now">
  </form>
</body>
</html>
```

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

The HTML <svg> Element

The HTML <svg> element is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Create a webpage to display circle.

```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-
width="4" fill="yellow" />
</svg>

</body>
</html>
```

Create a webpage to display rectangle.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <svg height="300px" width="300px">
        <rect rx="40" x="10" y="10" stroke="red" fill="green" style="stroke-
width: 4px; height:200px;width:200px;">
    </svg>
</body>
</html>
```

Create a webpage to display ellipse.

```
<html>
    <head></head>
    <body>
   <svg height="300px">
    <ellipse cx="100" cy="100" rx="80" ry="60" stroke="black" fill="blue" />
        <ellipse cx="90" cy="100" rx="70" ry="50" stroke="black" fill="pink" />
          <ellipse cx="80" cy="100" rx="60" ry="40" stroke="black" fill="green"/>
                <ellipse cx="70" cy="100" rx="50" ry="30" stroke="black"
fill="yellow" />
                <ellipse cx="60" cy="100" rx="40" ry="20" stroke="black"
fill="brown" />
   </svg>
    </body>
</html>
```

Create a webpage to display ellipse.

```
<html>
    <head></head>
    <body>
   <svg>
<line x1="0" y1="0" x2="200" y2="0" stroke="blue" fill="green" stroke-
width="5px"/>
   </svg>
    </body>
</html>
```

Html audio tag

Definition and Usage

The <audio> tag is used to embed sound content in a document, such as music or other audio streams.

The <audio> tag contains one or more <source> tags with different audio sources. The browser will choose the first source it supports.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

There are three supported audio formats in HTML: MP3, WAV, and OGG.

Example:

```
<audio controls autoplay loop>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

Html Video tag

Definition and Usage

The <video> tag is used to embed video content in a document, such as a movie clip or other video streams.

The <video> tag contains one or more <source> tags with different video sources. The browser will choose the first source it supports.

The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

There are three supported video formats in HTML: MP4, WebM, and OGG.

Example:

```
<html>
    <head></head>
    <body>
  <video width="400" controls autoplay loop poster="myProject/backg.jpg">
<source src="myProject/videos/inkemInkem.mp4">
Your browser does not support the video tag.
  </video>
    </body>
</html>
```