SYLLABUS

CSS: Overview of CSS, Backgrounds and Color Gradients in CSS, Fonts and Text Styles, Creating Boxes and Columns Using CSS, Displaying, Positioning, and Floating an Element, List Styles, Table Layouts. **Dynamic HTML:** Overview of JavaScript, JavaScript Functions, Events, Image Maps, and Animations.

Overview of CSS

- Cascading Style Sheets (CSS) is used to define styles and layouts of web page written in HTML and XHTML.
- CSS is a text file and extension with .css
- CSS maintained a web document by separating its style information, such as font, colors etc.

Understand the syntax of CSS:

- CSS syntax can be defined as a **rule** consists of a selector, property, and its value.
- The selector points to the HTML element where CSS style is to be applied.
- property: value pair that is defined for the specific selector to apply the style.
- Syntax:

```
selector {
        Property: value;
        Property: value;
                .
                .
                .
        Property: value;
}
```

- Example:

```
body{
        margin : 0;
        background : green;
}
```

Exploring CSS selector:

- CSS selectors is a pattern that used to "find" (or select) the HTML elements to apply style rule.
- Selector can be used as condition to determine the elements that match with the selector.
- We have 9 different selectors as follows
  1. The universal selector
  2. The type selector
  3. The class selector
  4. The id selector
  5. The child selector
  6. The descendant selector
  7. The adjacent sibling selector
  8. The attribute selector
  9. The query selector

1. The universal selector:

- The universal selector selects all the elements that are present in an HTML document.
- It is represented by asterisk (*) sign.
- Syntax:

```
*{      property : value;  }
```

- Example:

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

2. The type selector:

- The type selector matches all the elements specified in a list with given value to determine the elements to which the CSS rules are to be applied.
- Syntax:

```
Element_list{          property : value;  }
```

- Example:

```
h1, h2, p, table {     font-family : sans-serif;   }
```

3. The class selector:

- The class selector allows you to apply CSS rules to the elements that carry a class attribute whose value matches class attribute.
- The class selector represented by the period (.) sign.
- Example: HTML code

```
<h1 class="intro"> Bapatla Engineering College</h1>
<h2 class="intro"> Autonomous</h2>
```

- Syntax: applying the CSS rule to all the elements

```
.class_value{    property : value;   }
```

- Example:

```
.intro{  Font-family: sans-serif ; }
```

- Syntax: applying the CSS rule to specific the elements

```
Element_name . class_value{  property : value;  }
```

- Example:

```
h1.intro {   Font-family: sans-serif; }
```

4. <u>The id selector:</u>

- The id selector allows you to apply CSS rules to the element that carry a id attribute whose value matches id attribute.
- The value of the id attribute is unique within the document.
- Therefore the selector is applied only to the content of one element.
- The id selector represented by the hash (#) sign.
- Example: HTML code

```
<h1 id="header"> Bapatla Engineering College</h1>
```

- < Syntax: applying the CSS rule

```
#id_value{       property : value;   }
```

- Example:

```
#header {  Font-family: sans-serif ; }
```

5. <u>The child selector:</u>

- A child selector matches when an element is the immediate child of another element.
- A child selector is made up of two or more selectors separated by greater than (>) sign.
- Syntax:

```
Parent_Element > child_element{ property : value; }
```

- Example:

```
div > p {  background-color: yellow; }
```

6. <u>The descendant selector:</u>

- The descendant selector matches all elements that are descendants of a specified element.
- A descendant element is an element that is nested inside another element.
- The id selector represented by the space ( ) sign.
- Syntax:

```
Element   Descendant_element{ property : value; }
```

- Example:

```
div   p {  background-color: yellow; }
```

7. <u>The adjacent sibling selector:</u>

- The adjacent sibling selector matches all elements that are adjacent sibling of a specified element.
- Sibling elements must have the same parent element.
- The adjacent sibling selector represented by the plus (+) sign.
- Example: HTML code

```
<h1> Bapatla Engineering College</h1>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
```

- < Syntax: applying the CSS rule

```
Element + adjacent_sibling{    property : value;   }
```

- Example:

```
h1 + p {  Font-family: sans-serif ; }
```

Here, paragraph 1 is adjacent sibling of h1

8. <u>The attribute selector:</u>

The attribute selector selects elements on the basis of some specific attribute or attributes values.

Name	Synatx	Match	Example
Hyphen selector	[attribute |= value]	Matches if the element has an attribute with a value followed by a hyphen.	[lang |= en]{ background-color:red; }
Existence selector	[attribute]	Matches if the element has a specific attribute.	a[title]{ color : green; }
Equality selector	[attribute = value]	Matches if the element has an attribute with a specific value.	[lang =" en"]{ background-color:red; }
Space selector	[attribute ~= value]	Matches if the element has an attribute with a space separated items that match with value.	a[title ~= web]{ background-color:green; }

9. <u>The query selector:</u>

- The querySelector() and querySelectorAll() methods are accept the CSS selectors as parameters and return the matching element node in the document tree.
- The querySelector() method helps in querying the entire document or a specific element of the document.
- If multiple elements are available, then return the first matching element.
- If no elements are available, then return null
- The querySelectorAll() method returns all the available elements as a single static collections of elements called ***staticNodeList***.

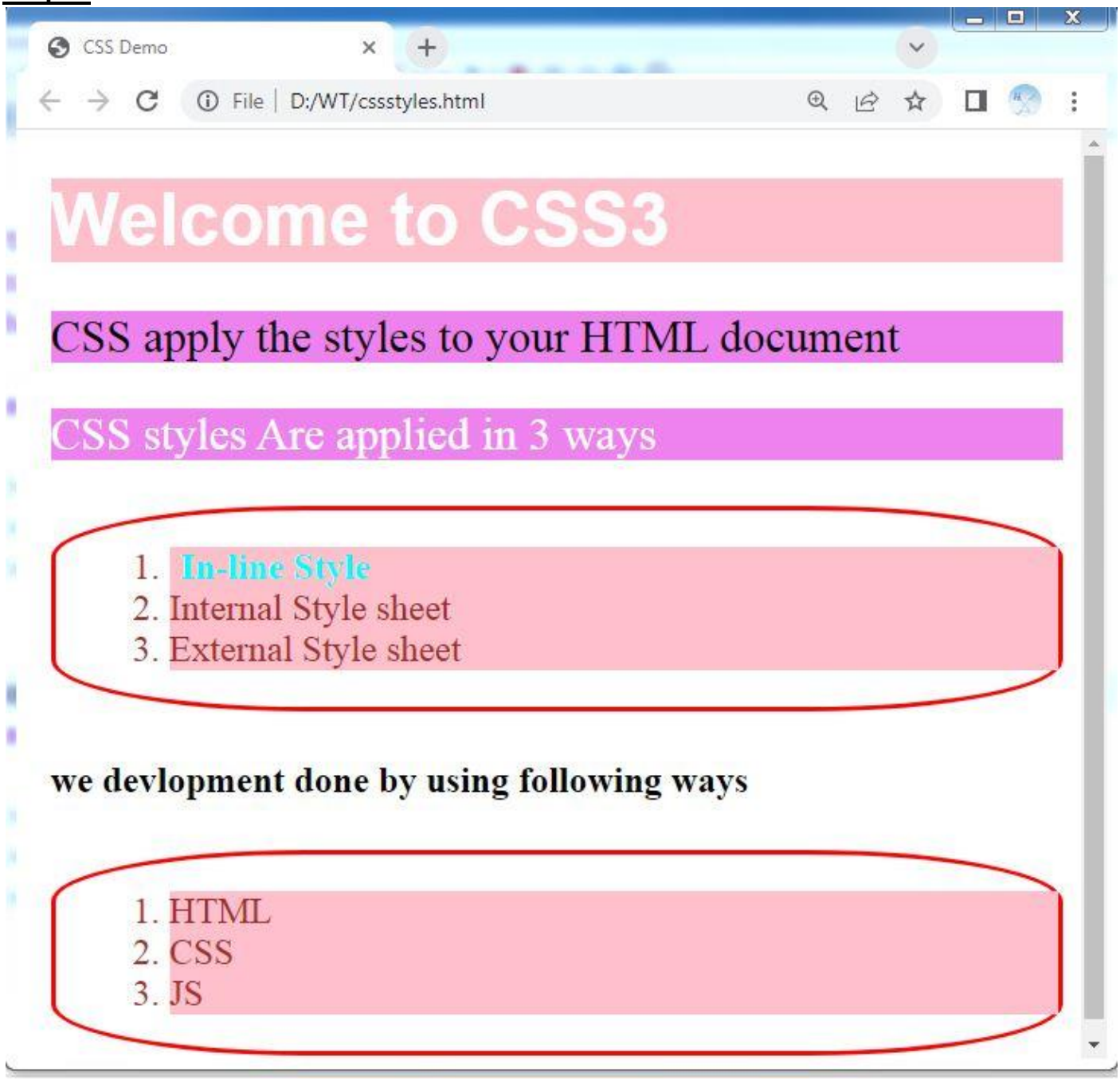
<u>Example:</u> Design a web page that demonstrate the selectors in CSS

```
<!DOCTYPE html>
<html>
	<head>
		<title>CSS Demo</title>
		<style type="text/css">
			/*universal selector*/
			*{
				margin-left:5px;
			}
			/*type selector*/
			h1{
				background-color:brown;
			}
			/*type group selector*/
			h1, li
			{
				font-family:sans-serif;
				color:brown;
				background-color:pink;
			}
```

```
                /*type selector*/
                p{
                        font-size:20px;
                        background-color:violet;
                }
                /*class selector*/
                .theory{color:white;}

                /*id selector*/
                #list1{
                        border:2px solid red;
                }
                #list2{
                        border:2px solid red;
                }

                /*child selector*/
                ol>li{
                        font-family:serif;
                }

                /*descedant selector*/
                ol b{color:cyan}

                /*adjacent selector*/
                p + h1{border:5px dotted green;}

                /*attribute selector*/
                [id]{
                        border-radius:25%;
                }

        </style>
</head>
<body>
        <h1 class="theory">Welcome to CSS3</h1>
        <p class="intro">CSS apply the styles to your HTML document</p>
        <p class="theory">CSS styles Are applied in 3 ways</p>

        <div id="list1">
                <ol>    <!- -here b is descendant for ol- - >
                        <li><b>In-line Style</b></li>
                        <li>Internal Style sheet</li>
                        <li>External Style sheet</li>
                </ol>
        </div >
        <h4>we devlopment done by using following ways</h1>
```

```
            <div id="list2">
                <ol>
                    <li>HTML</li>
                    <li>CSS</li>
                    <li>JS</li>
                </ol>
            </div>
        </body>
</html>
```

Output:



Inserting CSS in an HTML Document

A CSS style sheet can be linked to an HTML documents in 3 ways:

1. The ***Inline*** style sheet
2. The ***Internal*** style sheet
3. The ***External*** style sheet

1. The ***Inline*** style sheet:

- The inline style properties are written in a single line separated by semicolon.
- The properties are placed by using the **style attribute** inside HTML elements.
- Example:

```
<p style="background-color:red; border:1px solid black;">
```

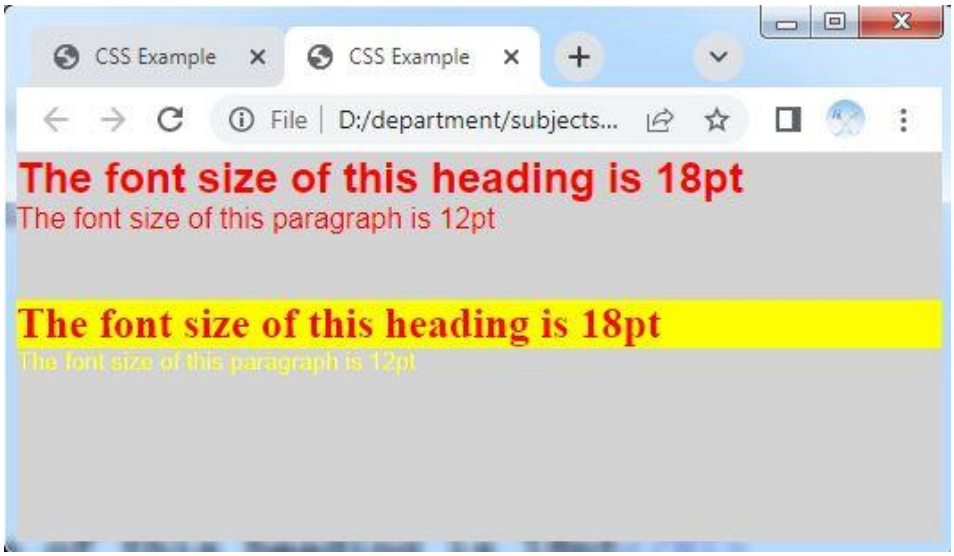
- Advantages:
  - ✓ Provides highest precedence over internal and external style sheets.
  - ✓ Provides an easy and quick approach to add style sheet in a web page.
- Disadvantages:
  - ✓ Makes a document difficult to maintain and increase download time.
  - ✓ Does not allow to style pseudo-code.

Example: Design a web page that demonstrate the CSS inline style sheet

Source code:

```
<!DOCTYPE HTML>
    <HEAD>
    </HEAD>
    <BODY style="color:red; background-color:lightgrey; font-family: sans-serif;">
        <H1 style="font-size:18pt" >The font size of this heading is 18pt</H1>
        <P style=" font-size:12pt ">The font size of this paragraph is 12pt</P><BR/><BR/>
        <H1 style="color:#ff0000;background-color:#ffff00;font-family: Ariel; font-size:18pt ">
            The font size of this heading is 18pt
        </H1>
        <P style="color:yellow; font-size:10pt">The font size of this paragraph is 12pt</P>
    </BODY>
</HTML>
```

Output:



2. The ***Internal*** style sheet:

- The internal style sheet is written within the HEAD element of the HTML document.
- The properties are placed by using the **STYLE element** inside HEAD elements.
- STYLE element having both the starting and ending tags.
- Attributes are

Attribute	Value	Description
media	media_query	Specifies what media/device the media resource is optimized for
type	text/css	Specifies the media type of the <style> tag

- **Syntax:**

```
<head>
   <STYLE type="text/css">
            SELECTOR { CSS properties; }
   </STYLE>
</head>
```

- **Example:**

```
<head>
   <STYLE type="text/css">
            p { background-color:red; }
   </STYLE>
</head>
```

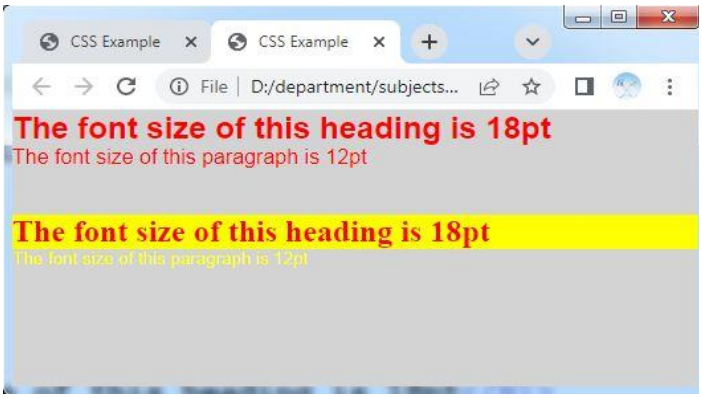
- Advantages:
  - ✓ Affects only the page in which they are placed.
  - ✓ Allows you to change the styles of the same HTML file in which you are working.
- Disadvantages:
  - ✓ Affects only the page in which they are applied.
  - ✓ Increase the page load time.

Example: Design a web page that demonstrate the CSS internal style sheet

Source code:

```
<!DOCTYPE HTML>
    <HEAD>
        <TITLE>CSS Example </TITLE>
        <STYLE>
            *{margin: 0;}
            body   {
              color:red;
              background-color:lightgrey;
              font-family: sans-serif;}
            h1{font-size:18pt}
            p{font-size:12pt}
        </STYLE>
    </HEAD>
    <BODY>
        <H1>The font size of this heading is 18pt</H1>
        <P>The font size of this paragraph is 12pt</P><BR/><BR/>
        <H1 style="color:#ff0000;background-color:#ffff00;font-family: Ariel;">
            The font size of this heading is 18pt </H1>
        <P style="color:yellow; font-size:10pt">The font size of this paragraph is 12pt</P>
    </BODY>  </HTML>
```

Output:



3. The ***External*** style sheet
   - The external style sheet is written outside of the HTML document.
   - The external style sheet was saved into a text file with **.css** extension.
   - The properties of external style sheet were linked to the HTML document intwo ways:

a) Linking:

- LINK element is used to link the external style sheet to HTML document.
- The <link> element is an empty element
- Example:

<LINK type="text/css" href="styles.css" rel="stylesheet">

- Attributes are

Attribute	Value	Description
type	*media_type*	Specifies the MIME type of the linked document
href	*URL*	Specifies the location of the linked document
rel	alternate author dns-prefetch help icon license next pingback preconnect prefetch preload prerender prev search **stylesheet**	Required. Specifies the relationship between the current document and the linked document

Example: Design a web page that demonstrate the CSS external style sheet using linking

Source code: sample.html

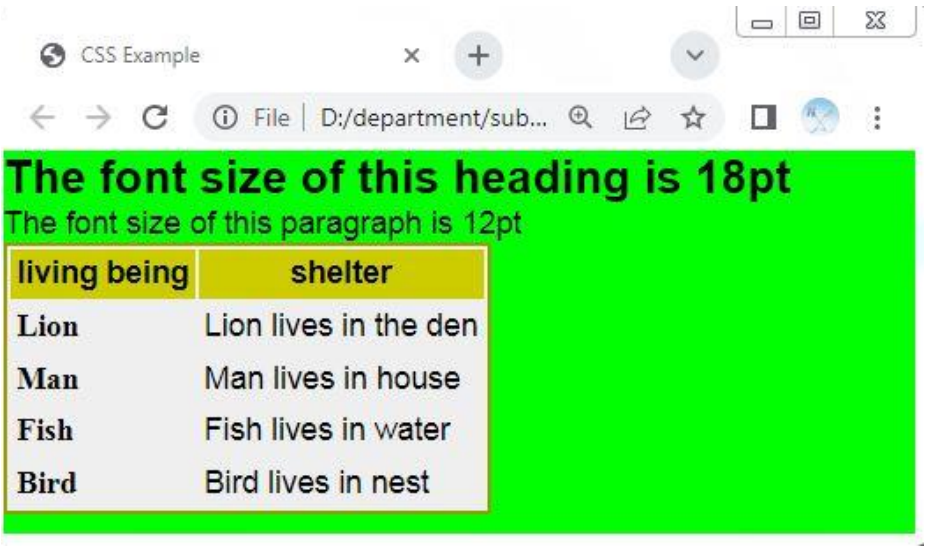
```
<!DOCTYPE HTML>
	<HEAD>
		<TITLE>CSS Example </TITLE>
		<LINK  rel="stylesheet"   type="text/css"   href="example.css"/>
	</HEAD>
	<BODY>
		<H1>The font size of this heading is 18pt</H1>
		<P>The font size of this paragraph is 12pt</P>
		<TABLE>
			<TR>
				<TH>living being</TH>
				<TH>shelter</TH>
			</TR>
			<TR>
				<TD class="code">Lion</TD>
				<TD>Lion lives in the den</TD>
			</TR>
			<TR>
				<TD class="code">Man</TD>
```

```
                <TD>Man lives in house</TD>
            </TR>
            <TR>
                <TD class="code">Fish</TD>
                <TD>Fish lives in water</TD>
            </TR>
            <TR>
                <TD class="code">Bird</TD>
                <TD>Bird lives in nest</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

example.css

```
/* style sheet for sample.html document */
*{margin: 0;}
body
{
    color:#000000;
    background-color:#00ff00;
    font-family: sans-serif;
}

h1{font-size:18pt}

p{font-size:12pt}

table
{
    background-color:#efefef;
    border-style:solid;
    border-width:2px;
    border-color:#999900;
}

th
{
    background-color:#cccc00;
    font-weight:bold;
    padding:3px;
}

td{padding:3px}

td.code
{
    font-family: serif;
    font-weight:bold;
}
```

Output:



b) Importing:

- It allows you to accessing the style rules from another style sheet and represented by **@import**.
- Example:

```
<STYLE type="text/css">
        @import   url(styles.css)
        p { background-color:red; }
</STYLE>
```

- Advantages:
  - ✓ Affects multiple pages in which they are placed with single document.
  - ✓ Allows you to easily group your styles in a more efficient way.
- Disadvantages:
  - ✓ Increase the time to apply the entire CSS file to the HTML document
  - ✓ Display the web page only after the entire style sheet is loaded.

Example: Design a web page that demonstrate the CSS external style sheet using import

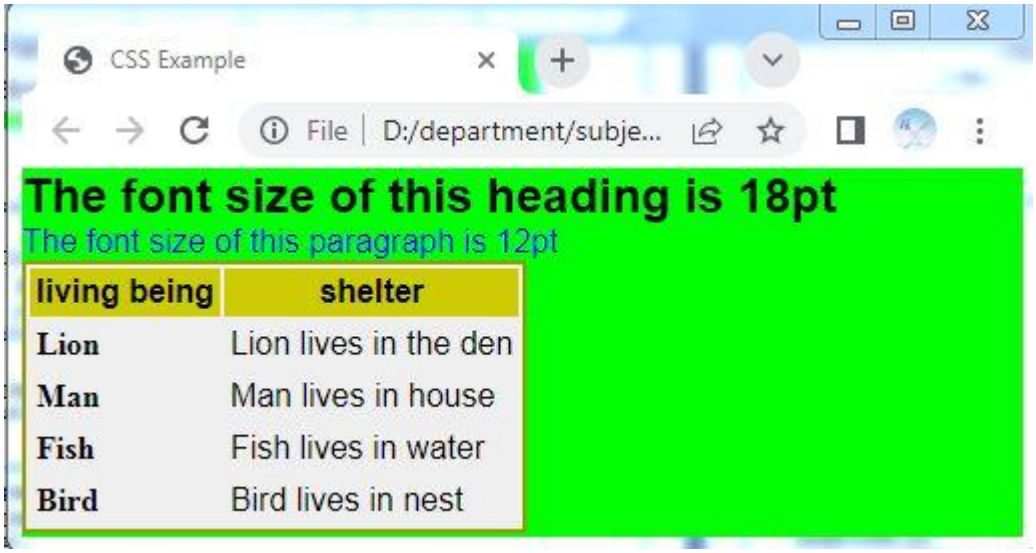
Source code: sample.html

```
<!DOCTYPE HTML>
    <HEAD>
        <TITLE>CSS Example </TITLE>
        <STYLE TYPE="text/css">
            @import url("example.css");
            P {color:blue}
        </STYLE>
    </HEAD>
    <BODY>
        <H1>The font size of this heading is 18pt</H1>
        <P>The font size of this paragraph is 12pt</P>
        <TABLE>
            <TR>
                <TH>living being</TH>
                <TH>shelter</TH>
            </TR>
            <TR>
                <TD class="code">Lion</TD>
                <TD>Lion lives in the den</TD>
            </TR>
            <TR>
                <TD class="code">Man</TD>
                <TD>Man lives in house</TD>
            </TR>
            <TR>
                <TD class="code">Fish</TD>
                <TD>Fish lives in water</TD>
            </TR>
            <TR>
                <TD class="code">Bird</TD>
                <TD>Bird lives in nest</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

example.css

```
/* style sheet for sample.html document */
*{margin: 0;}

body
{
    color:#000000;
    background-color:#00ff00;
    font-family: sans-serif;
}

h1{font-size:18pt}

p{font-size:12pt}

table
{
    background-color:#efefef;
    border-style:solid;
    border-width:2px;
    border-color:#999900;
}

th
{
    background-color:#cccc00;
    font-weight:bold;
    padding:3px;
}

td{padding:3px}

td.code
{
    font-family: serif;
    font-weight:bold;
}
```

Output:

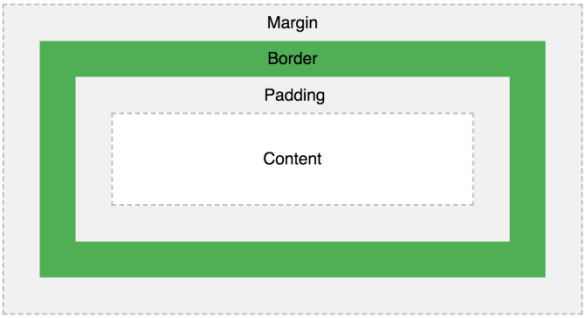


Creating Boxes and Columns using CSS

- Sometimes you need to place the content of your web page in boxes or columns.
- CSS provides various types of layout model
  1. Box model
  2. Line Box Model
  3. Template Layout Model
  4. Multi-Column Model

Exploring the Box Model:

- CSS converts the data of HTML elements in the form of rectangular boxes is called box model.
- The box model allows placing a border around the elements and also provides space between elements.
- It consists of margins, borders, padding, and the actual content areas.
- The image below illustrates the box model.



- **Content Area:** It display the content of HTML document, such as text, images etc.
- **Padding Area:** Area around the content area. It clears an area around the content. The padding is transparent
- **Border Area:** A border that goes around the padding and content
- **Margin Area:** Clears an area outside the border. The margin is transparent
- Box model are 3 types
  1. block-level box: Represents a box to show a paragraph.
  2. line box: Represents a box to show a line text.
  3. inline-level box: Represents a box to show the word of line.

Exploring Padding Properties:

- It specifies the distance between the border of an element and content.
- The value of padding cannot be negative.
- The padding properties are –
  - padding-top: It specifies the distance between the top side border of an element and content.
  - padding-bottom: It specifies the distance between the bottom side border of an element and content.
  - padding-left: It specifies the distance between the left side border of an element and content.
  - padding-right: It specifies the distance between the right side border of an element and content.
  - padding: It specifies the distance between the all side border of an element and content.
- The values are may give in **length (pixel), percentage,** and **auto.**
- If the padding property has only **one value**, then it applied equally on all the four padding.
- If the padding property has only **two values**, then First value applied to top and bottom padding and second value applied to left and right padding.
- If the padding property has only **three values**, then First value applied to top padding, second value applied to left and right padding, and third value applied to bottom padding.
- The padding property has only **four values,** then applied to top, bottom, right, and left padding.

Exploring Border Properties:

- It specifies the distance between magin and the content in the box model.
- Border properties are –
  - ➢ border-style
  - ➢ border-width
  - ➢ border-color
  - ➢ border shorthand
  - ➢ border-radius
  - ➢ border-image
  - ➢ border-shadow

border-style:

- It specifies format of the border, such as solid, dotted etc
- Syntax: border-style : value;
- Example: border-style: solid;
- The possible values are **solid,dotted, dashed, double, groove, ridge, inset, outset, hidden, none**
- It is also possible to set style for four sides, top, bottm, left, right which are
  - ✓ **border-top-style**: set the style of top border of an element
  - ✓ **border-bottom-style** : set the style of bottom border.
  - ✓ **border-left-style** : set the style of left border of an element
  - ✓ **border-right-style** : set the style of right border of an element
  - ✓ **border-style**: set the style of all border of an element.
- The possible property value range from **one to four**.

border-width:

- It specifies style of the border width.
- The width is specifies in pixel.
- Syntax: border-width : value;
- Example: border-width : 2px;
- The possible values are **thin, medium, thick, length,** and **inherit**.
- It is also possible to set style for four sides, top, bottom, left, right which are
  - ➢ **border-top-width**: set the **width** of top border of an element
  - ➢ **border-bottom-width** : set the **width** of bottom border.
  - ➢ **border-left-width** : set the **width** of left border of an element
  - ➢ **border-right-width** : set the **width** of right border of an element
  - ➢ **border-width** : set the **width** of all border of an element.
- The possible property value range from **one to four**.

border-color:

- It specifies style of the border color.
- The style is specifies in pixel.
- Syntax: border-color : value;
- Example: border-color : black;
- The possible values are **color name, RGB, Hex, Inherits,** and **Transparent.**
- It is also possible to set color of different sides –
  - ➢ **border-top-color** : set the **color** of top border of an element

- **border-bottom-color** : set the **color** of bottom border.
- **border-left-color** : set the **color** of left border of an element
- **border-right-color**: set the **color** of right border of an element
- **border-color** : set the style of all border of an element.

- The possible property value range from **one to four**.

border-radius:

- It specifies round corners to a box.
- Syntax: border-radius: value;
- Example: border-radius: 5px;
- The values are may give in **length, percentage.**
- It is also possible to set the different sides –
  - **border-top-left-radius**: set the round corner in the top left direction of border.
  - **border-top-right-radius**: set the round corner in the top right direction of border.
  - **border-bottom-left-radius**: set the round corner in the bottom left direction of border.
  - **border-bottom-right-radius**: set the round corner in the bottom right direction of border.
- The possible property value range from **one to four**.

border-image:

- It inserts image in border edges and corners to a box.
- These properties do not affect the layout of a box and its content.
- It having various properties
  - **border-image-source**: specifies the image instead of style.
    - border-image-source : none | <image>;
  - **border-image-slice**: specifies inward offsets from the top, right, bottom and left edges of an image.
    - border-image-slice : [<number> |<percentage>]{1,4} && fill;
  - **border-image-width**: specifies the width of an image used for the border.
  - **border-image-outset**: specifies of the border image are that can be extended beyond the border box.
  - **border-image-repeat**: specifies the image for the side and middle part of the border image are scaled and tiled.
    - border-image-repeat : [stretch | repeat | round]{1,2} ;
  - **border-image:** shorthand property to set the properties.

border shorthand:

- It specifies all the border related **properties (width, style and color**) into one property.
- Example: border: 2px solid orange;
- It is also possible to set the different sides –
  - **border-top**: set the **width, style and color** of top border of an element
  - **border-bottom**: set the **width, style and color** of bottom border.
  - **border-left**: set the **width, style and color** of left border of an element
  - **border-right**: set the **width, style and color** of right border of an element
  - **border**: set the style of all border of an element.
- The possible property value range from **one to four**.

Exploring Margin Properties

- The blank area around the border of an element is called margin.
- It is used to create extra space around an element.
- It is completely transparent and does not contain any background color.
- The possible values are **auto, length, percentage,** and **inherit**.
- Syntax: p { margin : value;}
- Example: p { margin : 5%;}
- The margin property applies to all sides (top, bottom, left and right) and individual sides using following properties also **margin-top, margin-bottom, margin-left,** and **margin-right**.
- Example: p { margin-left : 15px;
  margin-right : 1%;
  margin-top : auto;
  margin-bottom : inherit;
  border: 2px solid green; }

Multi-Column Model:

➢ It displays your web content in multiple columns.

Property	Description	Example
column-count	Specifies the number of columns an element should be divided into	column-count : <integer>|auto;
column-fill	Specifies how to fill columns	column-fill : balance | auto;
column-gap	Specifies the gap between the columns	column-gap : <length>|normal;
column-rule	A shorthand property for setting all the column-rule-* properties	column-rule : < column-rule-width>||< column-rule-style>||[< color>|transparent];
column-rule-color	Specifies the color of the rule between columns	
column-rule-style	Specifies the style of the rule between columns	
column-rule-width	Specifies the width of the rule between columns	
column-span	Specifies how many columns an element should span across	column-span : 1 | all;
column-width	Specifies a suggested, optimal width for the columns	column-width : <length>|auto;
columns	A shorthand property for setting column-width and column-count	

Marquee

- It effect on content to move from one side to another side.

S.N	Attribute	Description
1	**width**	This specifies the width of the marquee.
2	**height**	This specifies the height of the marquee.
3	**direction**	This specifies the direction in which marquee should scroll. This can be a value like *up, down, left* or *right*.
4	**behavior**	This specifies the type of scrolling of the marquee. This can have a value like *scroll, slide* and *alternate*.
5	**scrolldelay**	This specifies how long to delay between each jump.

Backgrounds and Color Gradients in CSS

Exploring the Backgrounds of a web page:

- Background of web page is the area on which the content of web page(text, image, borders etc) is displayed.
- A web page should have a background the express the motto of the webpage.
- CSS provides various properties to set background of web page as follow
    1. background-color
    2. background-image
    3. background-repeat
    4. background-attachment
    5. background-position
    6. background-clip
    7. background-origin
    8. background-size
    9. background-quantity
    10. background-spacing
    11. background

1. background-color:

- It is used to set the color of the background area on which elements are displayed.
- It can apply to almost any element.
- This property can take three types values: **Color name**, **HEX value**, and **RGB configuration**
- Example

```
h1{background-color : black;}
h1{background-color : #000000;}
```

2. background-image:

- It is used to set an image in the background of an element.
- It can apply to almost any element but better to set for the body element.
- This property can take two values: **url** and **none**
- Example

```
body{  background-image : url("background.jpg");  }
p{  background-image : none;  }
```

3. background-repeat:

- Property allows you to tile the background image along x-axis and y-axis of an element.
- This property is used along with background-image property only.
- This property can take four values
    1. repeat-x: Tiles an image horizontally.
    2. repeat-y: Tiles an image vertically.
    3. Repeat: Tiles an image both horizontally and vertically.
    4. no-repeat: Does not tiles an image.
- Example

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

4. background-attachment:

- This property is used to fix or scroll the background image along with the content of web page.
- This property is used along with background-image only.
- This property can take two values
  1. fixed: the background image does not move with content when the web page is scrolled.
  2. scroll: the background image scrolls along with content when the web page is scrolled.
- Example

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

5. background-position:

- This property is used to set the position of the background image on web page.
- This property is used along with background-image only.
- The position can set in three ways
  1. Represent the position in **pixels**
     - Example

```
body{  background-image : url("background.jpg");
       background-position: 200px 200px ;}
```

  2. Represent the position in **percentage**
     - Example

```
body{  background-image : url("background.jpg");
       background-position: 50%    50% ;}
```

  3. Represent the position by **words**
     - **Words** for x-axis :left, right, and center
     - **Words** for y-axis :top, down, and center
     - Example

```
body{  background-image : url("background.jpg");
       background-position: left   top;}
```

6. background-clip:

- It determines whether background image extends into border or not.
- The position can take three values
  1. border-box: Default value. The background extends behind the border
  2. padding-box: The background extends to the inside edge of the border
  3. content-box: The background extends to the edge of the content box
- Example

```
body{  background-image : url("background.jpg");
       background-clip: border-box;}
```

7. background-origin:

- It determines starting position of the background image in box like shape.
- The position can take three values
  1. border-box: The background image starts from the upper left corner of the padding edge. Default value.
  2. padding-box: The background image starts from the upper left corner of the border
  3. content-box: The background image starts from the upper left corner of the content
- Example

```
body{  background-image : url("background.jpg");
       background-origin: border-box;  }
```

8. background-size:

- It is used to specify size of the image that is used as a background for an element.
- The size can take three values
  1. auto – set the size of its original size
  2. length – set the size in terms of width and height.
  3. percentage – set the size with respect to specified height and width of the area in which the image has displayed.
- Example

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

9. background-quantity:

- It is used to specify the number of time to repeat the image.
- It takes values to display in both horizontal and vertical.
- **The infinite keyword:** Repeats an image infinitely.
- **The integer parameter:** Repeats an image the specified number of times.
- Example

```
body{  background-image : url("background.jpg");
       background-quantity: 5;      }
```

10. background-spacing:

- It is used to specify the distance between the images that are repeated in the background of an element.
- It takes the value to specify the horizontal and vertical space.
- Example

```
body{  background-image : url("background.jpg");
       background-spacing: 20px   30px;
        background-repeat: repeat;  }
```

11. background:

- It is shortcut to specify several background properties at same time place in style sheet.
- It can be used to specify the values for the background-color, background-image, background-repeat, background-attachment, background-position, and background-size properties.
- Example

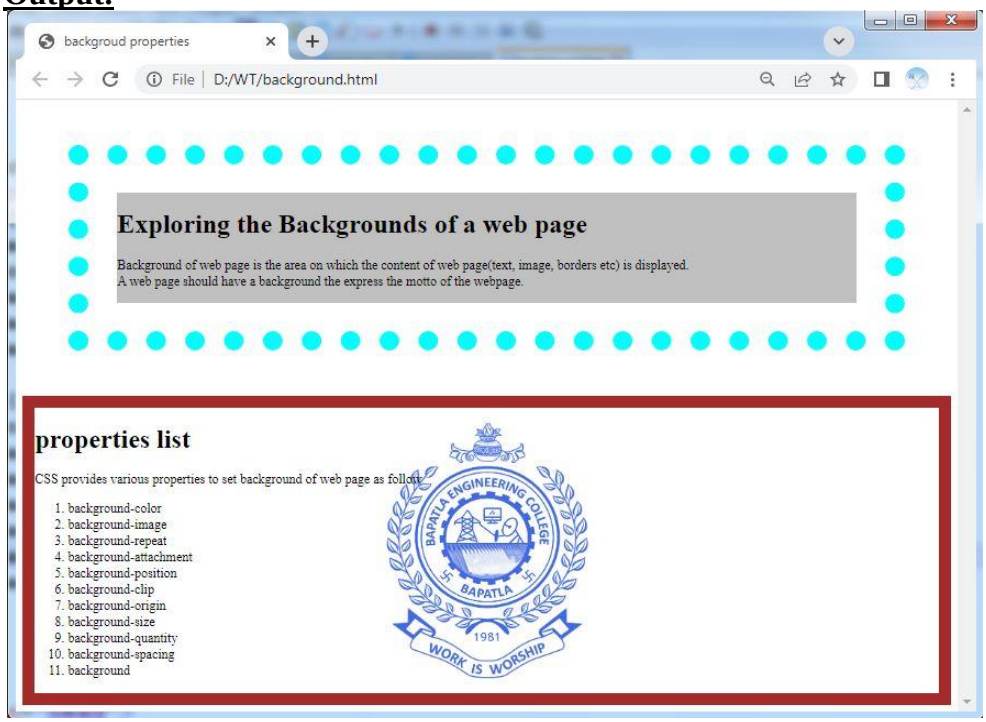
```
body{ background: orange; }
body{ background: url("background.jpg")   repeat   fixed;       }
```

Example: Design a web page that demonstrates the CSS background properties.

```
<!DOCTYPE html>
<html>
        <head>
                <title>backgroud properties</title>
                <style type="text/css">
                        .head{
                                margin:5%;
                                border:25px dotted cyan;
                                padding:3%;
                                background-color:brown;
                                background-clip:content-box;
                        }
                        .list{
                                border:15px solid brown;
```

```
                    background-image:url("images/beclogo.png");
                    background-repeat:no-repeat;
                    background-attachment:scroll;
                    background-size:40%;
                    background-position:center;
                    background-origin:content-box;
                }
            </style>
        </head>
        <body>
            <div class="head">
                <h1>Exploring the Backgrounds of a web page</h1>
                <p>Background of web page is the area on which the content of web page(text, image,
                borders etc) is displayed.<br>
                A web page should have a background the express the motto of the webpage. <br>
                </p>
            </div>
            <div class="list">
                <h1>properties list</h1>
                <p>CSS provides various properties to set background of web page as follow</p>
                <ol>
                    <li>background-color</li>
                    <li>background-image</li>
                    <li>background-repeat</li>
                    <li>background-attachment</li>
                    <li>background-position</li>
                    <li>background-clip</li>
                    <li>background-origin</li>
                    <li>background-size</li>
                    <li>background-quantity</li>
                    <li>background-spacing</li>
                    <li>background</li>
                </ol>
            </div>
        </body>
</html>
```

Output:



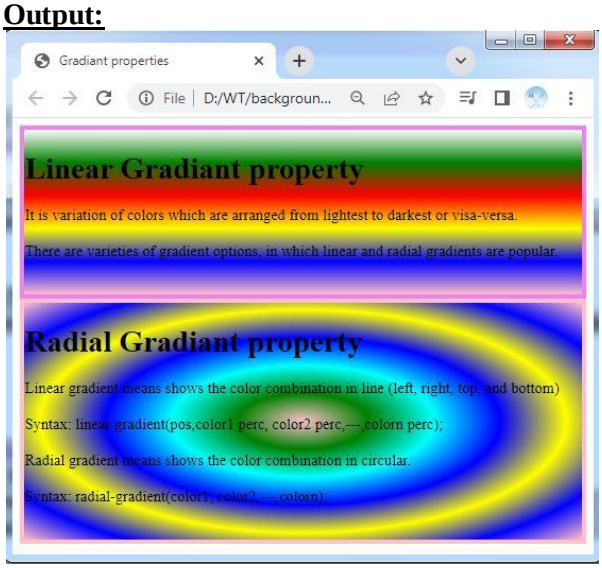
Gradients:

- It is variation of colors which are arranged from lightest to darkest or visa-versa.
- There are varieties of gradient options, in which linear and radial gradients are popular.
- Linear gradient means shows the color combination in line (left, right, top, and bottom)
- Syntax: linear-gradient(pos,color1 perc, color2 perc,---,colorn perc);
- Radial gradient means shows the color combination in circular.
- Syntax: radial-gradient(color1, color2,---,colorn);

Example: Design a web page that demonstrates the CSS gradiant properties.

```
<!DOCTYPE html>
<html>
        <head>
                <title>Gradiant properties</title>
                <style type="text/css">
                        .intro{
                                border:5px solid violet;
                                background-image: linear-gradient(white,green,red, yellow,blue, pink);
                        }
                        .types{
                                border:5px solid pink;
                                background-image:radial-gradient(pink,green,cyan,blue,yellow,blue, pink);
                        }
                </style>
        </head>
        <body>
            <div class="intro">
                <h1>Linear Gradiant property</h1>
                <p>It is variation of colors which are arranged from lightest to darkest or visa-versa. <br>
                 There are varieties of gradient options, in which linear and radial gradients are popular. </p>
            </div>
          <div class="types">
              <h1>Radial Gradiant property</h1>
              <p>Linear gradient means shows the color combination in line (left, right, top, and bottom) <br>
              Syntax:    linear-gradient(pos,color1 perc, color2  perc,---,colorn  perc);<br><br>
              Radial gradient means shows the color combination in circular.<br><br>
              Syntax:    radial-gradient(color1, color2,---,colorn);<br><br>        </p>
        </div>
        </body>
</html>
```

Output:



Fonts and Text Styles

Exploring Fonts:

- Font represents the style and size of the text in web browser.
- It is used to differentiate the content.
- For example, you can easily differentiate main level heading and sub heading based on font size and styles.
- In HTML, you can change the size, style, and family of fonts using various CSS properties as follows
  1. font-family
  2. font-size
  3. font-size-adjust
  4. font-stretch
  5. font-style
  6. font-variant
  7. font-weight
  8. font

1. font-family:

- It is used to specify the name of the font family to apply the specified font style on the text.
- The font families are
  a) **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
  b) **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
  c) **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
  d) **Cursive** fonts imitate human handwriting.
  e) **Fantasy** fonts are decorative/playful fonts.

Font Family	Examples of Font Names
Serif	Times New Roman, Georgia, Garamond
Sans-serif	Arial, Verdana, Helvetica
Monospace	Courier New, Lucida Console, Monaco
Cursive	Brush Script MT, Lucida Handwriting
Fantasy	Copperplate, Papyrus

Note: If you have specified a font family is not installed on your computer, then the web browser displays the text in another font.

2. font-size:

- The font-size property is used to change the size of the text.
- The value of font-size property is specified in pixels.
- Example: p{ font-size : 20px; }
- The value of font-size property is specified in another three ways
  a) **Absolute value:** absolute value refers to the absolute size of the font i.e,. Size was fixed that cannot be changed by used. The absolute values are xx-small, x-small, small, medium, large, x-large, xx-large.
  b) **Relative value:** Relative values are calculated on the basis of the current font values or surrounding element values.
  c) **Percentage value:** Percentage values the size is represented by percentage to increase or decrease the font size.

3. font-size-adjust:

- The font-size-adjust property is used to change the aspect value of the text on a web page.
- Aspect value is the ratio between the font height of lowercase letter and actual height of the font.
- Example: p{ font-size-adjust : 0.5; }

4. font-stretch:

- The font-stretch property is used to change the width of the font.

Value	Description
ultra-condensed	Makes the text as narrow as it gets
extra-condensed	Makes the text narrower than condensed, but not as narrow as ultra-condensed
condensed	Makes the text narrower than semi-condensed, but not as narrow as extra-condensed
semi-condensed	Makes the text narrower than normal, but not as narrow as condensed
normal	Default value. No font stretching
semi-expanded	Makes the text wider than normal, but not as wide as expanded
expanded	Makes the text wider than semi-expanded, but not as wide as extra-expanded
extra-expanded	Makes the text wider than expanded, but not as wide as ultra-expanded
ultra-expanded	Makes the text as wide as it gets
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

5. font-style:

- The font-style property is used to specify the style of the font.
- font-style properties are : normal, italic and oblique
- Example: p{ font-style : italic; }

6. font-variant:

- The font-variant property is used to display the font as normal of small-caps.
- Example: p{ font-variant: small-caps;}

7. **font-weight:**

- The font-weight property is used to specify the weight of the font, such as boldness or thickness.

Value	Description
normal	Defines normal characters. This is default
bold	Defines thick characters
bolder	Defines thicker characters
lighter	Defines lighter characters

- Number values of the font-weight property.

Value	Description
100 200 300 400 500 600 700 800 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold

- Example: p{ font-weight: bold;}

8. **font:**

- The font property is used to specify several font properties at same time place in style sheet.
- Example: p{ font: bold italic 30px verdana;}

Web Font:

- It allows you to write text in fonts that are not existed in your system.
- It is introduced in CSS3.
- Syntax:

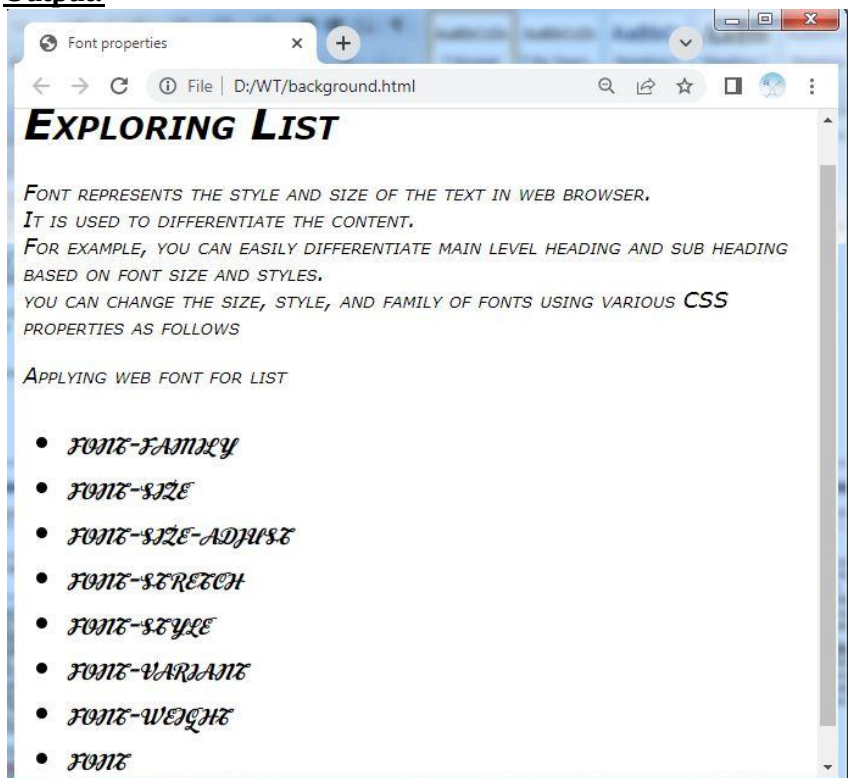
```
@font-face{
        font-family : <name>
        src : <source>
}
```

Example: Design a web page that demonstrates the CSS font properties.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Font properties</title>
        <style type="text/css">
            @font-face{
                font-family :"webfont";
                src : url("OleoScriptSwashCaps-Regular.ttf");
            }
            .intro{
                font-family:verdana;
                font-size:20px;
                font-size-adjust:0.5;
```

```
                font-stretch:expanded;
                font-style:italic;
                font-variant: small-caps;
                font-weight:lighter;
            }
            #list{
                font-family:webfont;
                font-size:30px;
            }
        </style>
    </head>
    <body>
      <div class="intro">
        <h1>Exploring List</h1>
        <p>
            Font represents the style and size of the text in web browser.<br>It is used to
            differentiate the content.<br>For example, you can easily differentiate main level
            heading and sub heading based on font size and styles.<br>you can change the size,
            style, and family of fonts using various CSS properties as follows<br>
        </p>
            <p>Applying web font for list</p>
            <ul id="list">
                <li>font-family</li>
                <li>font-size</li>
                <li>font-size-adjust</li>
                <li>font-stretch</li>
                <li>font-style</li>
                <li>font-variant</li>
                <li>font-weight</li>
                <li>font</li>
            </ul>
        </div>
    </body>
</html>
```

Output:



Exploring Texts:

- It is used to customize the appearance of the text in web page.
- It allows you to apply and modify the style already applied on the text.
- Example color, direction, underline etc.
- CSS text properties are as follow

S.N	Properties	Description
1.	color	Specifies the color of a text. Value can be color name, HEX value, GRB etc
2.	direction	Specifies the text direction. Values are ltr | rtl | inherits.
3.	letter-spacing	Specifies the space between characters.
4.	line-break	Defines a set of line breaking rules to be used with text.
5.	line-height	Specifies the height of line
6.	text-indent	Creates the paragraph look. It applicable for only block type elements, such as headings, paragraph, and div.
7.	text-align	Set the horizontal alignment of the text. Values are left | right | center | justify
8.	text-align-last	Specifies the alignment of the last line of a text. Possible values are left | right | center | justify | start | end | size
9.	text-decoration	Performs various actions on the text. Values are underline | overline | line-trough |blink | none
10.	text-decoration-line	Defines the decoration for the lines of an element. Values are underline | overline | line-trough |blink | none
11.	text-decoration-color	Defines the color for the text decoration lines.
12.	text-decoration-style	Defines the style for the text decoration lines.Values are solid | double | wavy
13.	text-decoration-skip	Specifies the element content to skip the text decoration. Values are spaces | images | ink | all | none
14.	text-justify	Specifies the method of justifying the text. Values are auto | inter-word | inter-character
15.	text-shadow	Specifies a list of shadow effects to be applied on the text. Values are h-shadow | v-shadow | blur-radius | color
16.	text-transformation	Transform the character of the text. Values are capitalize | lowercase | uppercase| none
17.	text-outline	Provides an outline of the text.
18.	text-overflow	Specifies the behavior of the text when it overflows the containing element.
19.	text-stroke	Provides the outlining on the text.
20.	text-underline-position	Set the position of underline specified on the text.
21.	text-wrap	Specifies the way of wrapping the text in an element.
22.	white-space-collapsing	Specifies the way of collapsing white spaces in an element.
23.	white-space	Specifies the handling of white spaces in an element.
24.	word-break	Defines a rule that allows a word to break in to next line.
25.	word-spacing	Defines the minimum and maximum space between words.
26.	word-wrap	Allows long word to break and wrap in the next line.

List Styles

- A list is a series of items belonging to a specific category, such as colors, vehicles, and countries.
- Depending on the order the list are categorized into **order** and **unordered**.
- **order list** are arranged in a sequential or logical order
- **unordered list** there is no specific sequence or order to place items
- The CSS properties to style the lists are given as follows:
  1. **list-style-type**
  2. **list-style-image**
  3. **list-style-position**
  4. **list-style**

1. **list-style-type property:**
   - list-style-type property is responsible for controlling the appearance and shape of the marker.
   - This property is used to change the default appearance of list-makers in HTML.
   - Syntax:

     list-style-tyle: <glyph>| <algorithmic> | <symbolic>
   - **<glyph>:** Includes value that insert their corresponding symbols for an unordered list.
     The <glyph> parameters are disk, circle, square, and none
     Example: **list-style-tyle: circle** **list-style-tyle: disk** **list-style-tyle: square**

     ○ one ○ two ○ three

     • one • two • three

     ▪ one ▪ two ▪ three
   - **<algorithmic>:** Includes value that display their corresponding symbols for an ordered list.
     The <algorithmic> parameters are decimal, decimal-leading-zero, lower-alpha, lower-latin, lower-roman, lower-greek, upper-alpha, upper-roman, upper-latin
     Example: **list-style-tyle: lower-roman**

     i. Bubble Sort
     ii. Merge Sort
     iii. Insertion Sort
   - <symbolic>: Includes the values, asterisks and footnotes, where asterisks shows symbols in and increasing order (*, **, …) and footnotes shows symbol.
   - Example **list-style-tyle: asterisks**
     list-style-tyle: footnotes

2. **list-style-image:**
   - It sets an image for the marker instead of the number or a bullet point.
   - You can an image as a list item marker for both ordered and unordered list
   - Syntax: **list- style-image: <url> | none;**
   - Example: **ul{list- style-image: url(arrow.png);}**

3. **list-style-position:**
   - It specifies the position of a list item marker.
   - It having two values, inside and outside.
   - Syntax: **list- style-position: inside | outside;**
   - Example: **ul{list- style-position: inside;}**

4. **list-style:**
   - It is the shorthand property of the above properties.
   - It is used to proved values for **list-style-type, list-style-image, list-style-position** properties.
   - Example: **ul{list- style: outside url(arrow.png);}**

Example: Design a web page that demonstrates the CSS list properties.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Font properties</title>
		<style type="text/css">
			.list{
				column-count: 3;
			}
			#sorting{
				list-style-type:circle;
				list-style-position: outside;
			}
			#fonts{
				list-style-image:url("icon.png");
				list-style-position: inside;
			}
			#week{
				list-style-tyle: decimal;
				list-style-position: inside;
			}
		</style>
	</head>
	<body>
		<h1 align="center">Exploring List</h1>
		<div class="list">
			<h3>Sorting techniques are</h3>
			<ul id="sorting">
				<li>Merge sort</li>
				<li>Insertion sort</li>
				<li>Radix sort</li>
				<li>Quick sort</li>
				<li>bubble sort</li>
				<li>heep sort</li>
				<li>selection sort</li>
				<li>heep sort</li>
				<li>selection sort</li>
			</ul>
			<h3>font properties are</h3>
			<ul id="fonts">
				<li>font-family</li>
				<li>font-size</li>
				<li>font-size-adjust</li>
				<li>font-stretch</li>
				<li>font-style</li>
				<li>font-variant</li>
				<li>font-weight</li>
			</ul>
```

```
                <h3>Day of weeks</h3>
                <ol id="week">
                        <li>Sunday</li>
                        <li>Monday</li>
                        <li>Tuesday</li>
                        <li>Wednesday</li>
                        <li>Thursday</li>
                        <li>Friday</li>
                        <li>Saturday</li>
                </ol>
        </div>
    </body>
</html>
```

Output:

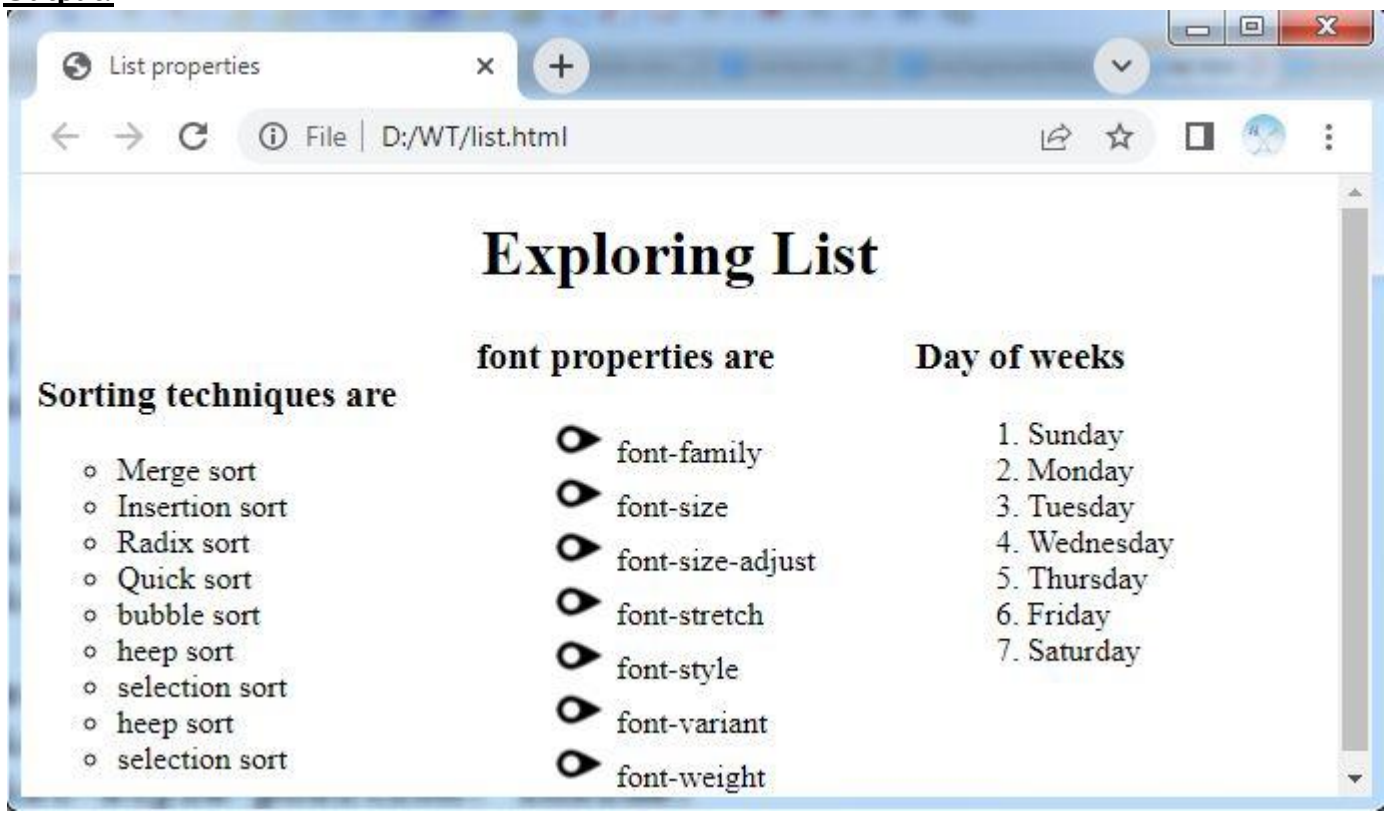


Table layouts

You can customize the appearance and layout of a table using CSS in HTML.

table-layout property:

- The table-layout property specifies the way in which a table should displayed in a web browser.
- It allows flexibility in positioning the tables.
- It is also decrease the loading time of the table, allowing the main content to appear before the graphics.
- The possible values are

S.N	value	Description
1	auto	Web browser automatically resizes a table.
2	fixed	Web browser does not resize a table.

- Example: table { table-layout : auto; }

caption-side property:

- a table caption is short description abou the table.
- caption-side property used to specify the position of the table caption.
- The possible values are

S.N	value	Description
1	top	defines a caption on the top of the table
2	bottom	defines a caption at the bottom of the table
3	inherit	the value is inherits from the containing element.

- Example: table { caption-side : bottom; }

border-collapse property:

- This property allows defining the way in which a border should be displayed around a table cell.
- The possible values are

S.N	value	Description
1	collapse	Sets a common border to all cell of a table.
2	separate	Sets a separate border for each cell of a table.
3	inherit	Inherits the value from the parent element.

- Example: table { border-collapse : separate; }

border-spacing property:

It allows specifying the amount of space between the borders of adjacent cells.

- The possible values are

S.N	value	Description
1	length	Specified the horizontal and vertical spacing between borders.
2	inherit	Inherits the value from the parent element.

- Example: table { border-spacing : 10px 50px; }

empty-cells property:

- The cell does not contain any content is called empty cell.
- empty-cell property is used to define the border of an empty cell.
- The possible values are

S.N	value	Description
1	show	Display the border around the empty cell.
2	hide	Hide the border of an empty cell.
3	inherit	Inherits the value from the parent element.

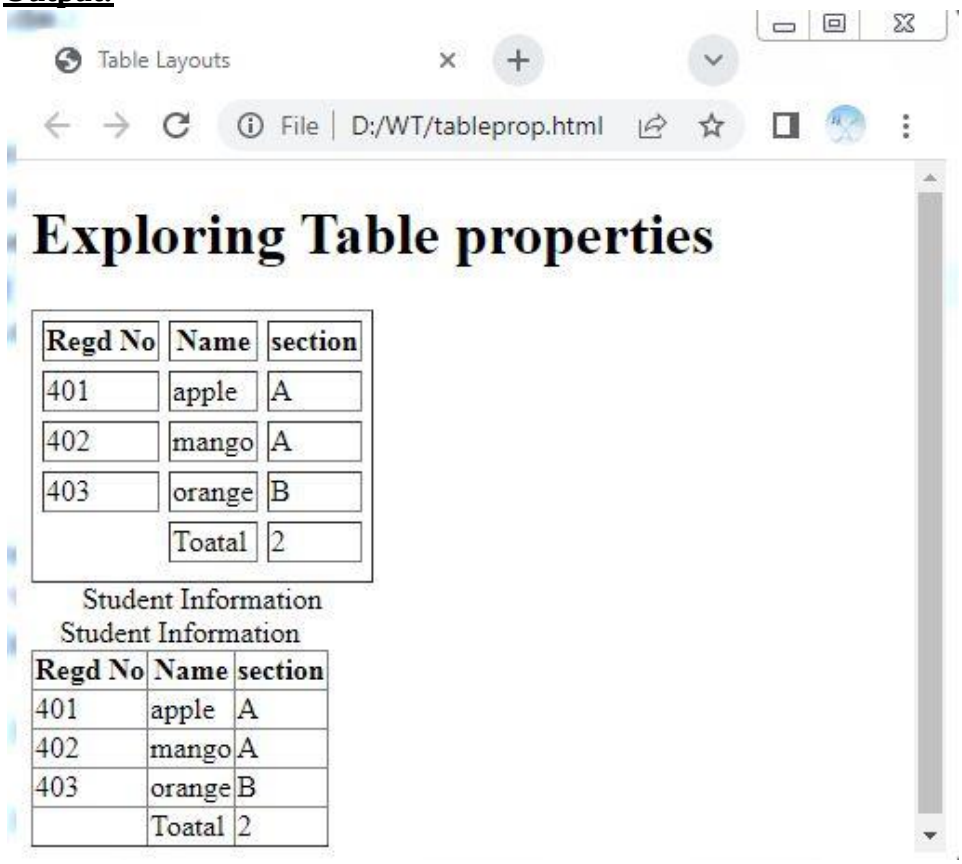
- Example: table { empty-cells : hide; }

Example: Design a web page that demonstrate the table layous

```
<!DOCTYPE html>
<html>
	<head>
		<title>Table Layouts</title>
		<STYLE type="text/css">
			#table1{
				table-layout:auto;
				caption-side:bottom;
				border-collapse:separate;
				border-spacing : 5px   5px;
				empty-cells:hide;
			}
			#table2{
				table-layout:auto;
				caption-side:top;
				border-collapse:collapse;
				border-spacing : 5px   5px;
				empty-cells:hide;
			}
		</style>
	</head>
	<body>
		<h1>Exploring Table properties</h1>
		<table border="1" id="table1">
			<caption>Student Information</caption>
			<tr >
				<th>Regd No</th>
				<th>Name</th>
				<th>section</th>
			</tr>
			<tr>
				<td >401</td>
				<td>apple</td>
				<td>A</td>
			</tr>
			<tr>
				<td >402</td>
				<td>mango</td>
				<td>A</td>
			</tr>
			<tr>
				<td >403</td>
				<td>orange</td>
				<td >B</td>
			</tr>
			<tr>
				<td> </td>
				<td>Toatal </td>
				<td>2</td>
			<tr>
		</table>
		<table border="1" id="table2">
```

```
                <caption>Student Information</caption>
                <tr >
                        <th>Regd No</th>
                        <th>Name</th>
                        <th>section</th>
                </tr>
                <tr>
                        <td >401</td>
                        <td>apple</td>
                        <td>A</td>
                </tr>
                <tr>
                        <td >402</td>
                        <td>mango</td>
                        <td>A</td>
                </tr>
                <tr>
                        <td >403</td>
                        <td>orange</td>
                        <td >B</td>
                </tr>
                <tr>
                        <td> </td>
                        <td>Toatal </td>
                        <td>2</td>
                <tr>
            </table>
        </body>
</html>
```

Output:



Displaying, Positioning, and Floating an Element

Displaying of an element using CSS:

- CSS allow controlling the displaying of an HTML element by using the display and visibility properties.

display property:

- The display property is used to display an element in specific manner.
- It generates a particular type of box for an element.
- Syntax: **display : [value]**
- Example: **h1{display : block}**
- The display property having different properties as follows

Value	Description
none	The element is completely removed
inline	Displays an element as an inline element (like <a>). Any height and width properties will have no effect
block	Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
contents	Makes the container disappear, making the child elements children of the element the next level up in the DOM
flex	Displays an element as a block-level flex container
grid	Displays an element as a block-level grid container
inline-block	Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
inline-flex	Displays an element as an inline-level flex container
inline-grid	Displays an element as an inline-level grid container
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a element
run-in	Displays an element as either block or inline, depending on context
table	Let the element behave like a <table> element
table-caption	Let the element behave like a <caption> element
table-column-group	Let the element behave like a <colgroup> element
table-header-group	Let the element behave like a <thead> element
table-footer-group	Let the element behave like a <tfoot> element
table-row-group	Let the element behave like a <tbody> element
table-cell	Let the element behave like a <td> element
table-column	Let the element behave like a <col> element
table-row	Let the element behave like a <tr> element
inherit	Inherits this property from its parent element.

visibility property:

- The visibility property specifies whether an element is visible on a web page or not.
- Syntax: **visibility : [value]**
- Example: **p{visibility : hidden}**
- The visibility property having different properties as follows

Value	Description
visible	Default value. The element is visible
hidden	The element is hidden (but still takes up space)
collapse	Only for table rows (<tr>), row groups (<tbody>), columns (<col>), column groups (<colgroup>). This value removes a row or column, but it does not affect the table layout. The space taken up by the row or column will be available for other content. If collapse is used on other elements, it renders as "hidden"
inherit	Inherits this property from its parent element.

Position of element using CSS:

- CSS controls the position of element with respect to the normal flow of content of a web page.
- Syntax: **position : [value]**
- Example: **p{position : fixed}**
- The position property having different properties as follows

Value	Description
static	Default value. Elements render in order, as they appear in the document flow
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position
sticky	The element is positioned based on the user's scroll position A sticky element toggles 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).
inherit	Inherits this property from its parent element.

- CSS also provides some properties that specifies the offset position of element with respect to the normal flow of content of a web page
- The offset properties are as follow
  - top: offsets an element in the top direction of a web page.
  - bottom: offsets an element in the bottom direction of a web page.
  - left: offsets an element in the left direction of a web page.
  - right: offsets an element in the right direction of a web page.
- Example: **p{ position : fixed; top : 10px; left : 5px }**

Floating of element using CSS:

- The float property specifies whether an element should float to the left, right, or not at all.
- Syntax: **float : [value]**
- Example: **p{float : right}**
- The float property having following properties

Value	Description
none	The element does not float, (will be displayed just where it occurs in the text). This is default
left	The element floats to the left of its container
right	The element floats the right of its container
inherit	Inherits this property from its parent element.

clear property:

- The *clear* prevents an element from being displayed next to floated elements
- You can disable the effect of float property by using clear property.
- Syntax: **clear : [value]**
- Example: **p{clear : right}**
- The cllear property having following properties

Value	Description
none	Default. The element is not pushed below left or right floated elements
left	The element is pushed below left floated elements
right	The element is pushed below right floated elements
both	The element is pushed below both left and right floated elements
inherit	Inherits this property from its parent element.

Dynamic HTML: Overview of JavaScript

- JavaScript is a client and server-side object-based scripting language that is used to make interactive Web pages.
- A scripting language is a lightweight programming language with less complexity.
- JavaScript is the most usually used scripting language to add dynamism and interactivity to Web pages.
- This is because JavaScript, written on the client-side, executes on a client browser, thereby reducing the load on the server.

Why to Learn JavaScript?

There are the three languages, all web developers must know, these are the following:

- HTML - to define the content of web pages
- CSS - to define the layout of web pages
- JavaScript - to program the behavior of web pages

Features of JavaScript:

- Light Weight Scripting language
- Dynamic Typing
- Object-based programming support
- Functional Style
- Platform Independent
- Prototype-based
- Interpreted Language
- Async Processing
- Client-Side Validation
- More control in the browser

Using JavaScript in an HTML document:

- The HTML SCRIPT element (<script> tag) is used to define a client-side script (JavaScript).
- The SCRIPT element either contains script statements, or it points to an external script file.
- Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
- The SCRIPT element contains five attributes as follow

Attribute	Value	Description
src	URL	Specifies the address of a script file.
type	text/javascript text/ecmascript application/javascript application/ecmascript text/vbscript	Specifies the MIME(multipurpose internet mail extension) type
async	true or false	Specifies whether the script should be executes asynchronously or not
charset	Charset	Specifies character encoding used in script
defer	true or false	Specifies whether browser can continues parsing the web page or not

- You can use SCRIPT element in three ways
  1. In the HEAD element
  2. In the BODY element
  3. As an external script file

1. In the HEAD element:

- You can place SCRIPT element inside the HEAD element of an HTML document.
- The script runs when you perform some action, such as click on the link, click on submit button.
- Syntax:

```
<HEAD>
   <SCRIPT type="text/JavaScript" >
         Script code here
   </SCRIPT>
</HEAD>
```

2. In the BODY element:

- You can place SCRIPT element inside the BODY element of an HTML document.
- The script runs when a web page starts loading in a web browser.
- Syntax:

```
<BODY>
   <SCRIPT type="text/javascript" >
         Script code here
   </SCRIPT>
</BODY>
```

3. As an external script file:

- You can store JavaScript code in external file and save that file using the .js extension.
- This external file was linked to HTML document by using scr attribute in SCRIPT element to access the script.
- Syntax:

```
<BODY>
   <SCRIPT src="URL of external file " >
         Script code here
   </SCRIPT>
</BODY>
```

Programming Fundamentals of JavaScript:

- The programming fundamentals of JavaScript include the following:
  - Lexical structure
  - Variables
  - Operators
  - Control flow statements
  - Popup boxes

Exploring Lexical structure of JavaScript:

- Lexical structure of JavaScript provides set of rules to write programs.
- Lexical structure of JavaScript defines rules for following aspect:
  1. Character set
  2. Case sensitivity
  3. White spaces and line breaks
  4. Optional semicolon
  5. Comments
  6. Literals
  7. Identifiers
  8. Reserved words

Exploring popup boxes:

- A popup box is a window that displays message along with OK button.
- A popup box may also contain CANCEL button.
- JavaScript supports three types of popup boxes
  1. The alert box
  2. The confirm box
  3. The prompt box

1. The alert box:

- The alert box generally used to display an alert message while executing JavaScript.
- The alert box is used to display an error messages after you validate a form.
- The alert box contains OK button, which the user has to click to continue with the execution of the code.
- Syntax: alert (“alert message”);
- Example: alert (“please fill all mandatory fields.”);

2. The confirm box:

- The confirm box is used to verify the user activity regarding their operation.
- The confirm box is used to display a messages and return **true** or **false** value.
- The confirm box contains OK and CANCEL buttons.
- If the user clicks on OK button it return true value and if user click on CANCEL button it return false value.
- Syntax: confirm (“confirm message”);
- Example: confirm (“do you want proceed?”);

3. **The prompt box:**
   - The confirm box is used to input a value from the user.
   - The confirm box contains text box and OK and CANCEL buttons.
   - If the user clicks on OK button it return input value otherwise it return null value.

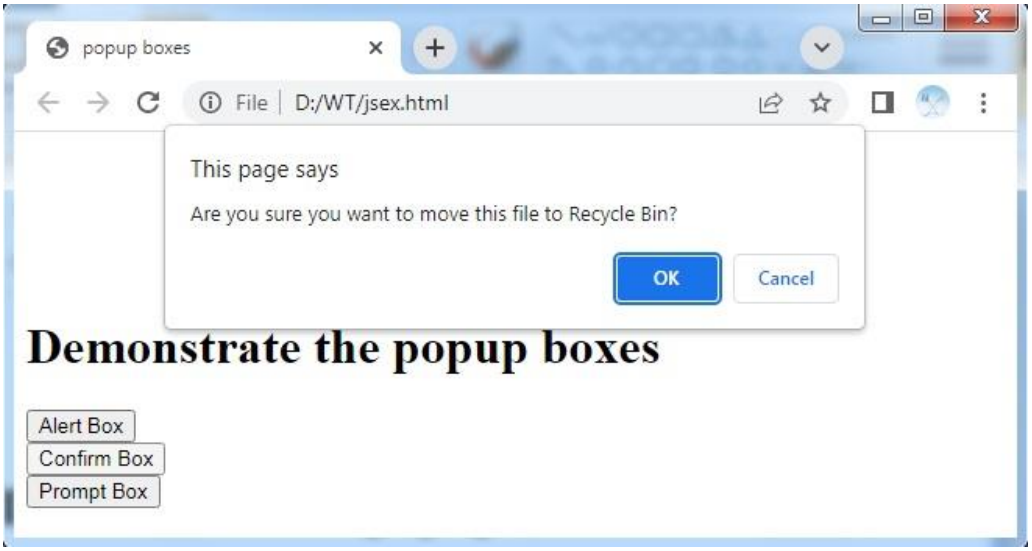
Example: Create a HTML file that demonstrates the popup boxes (alert, confirm, and prompt boxes).

Source code:

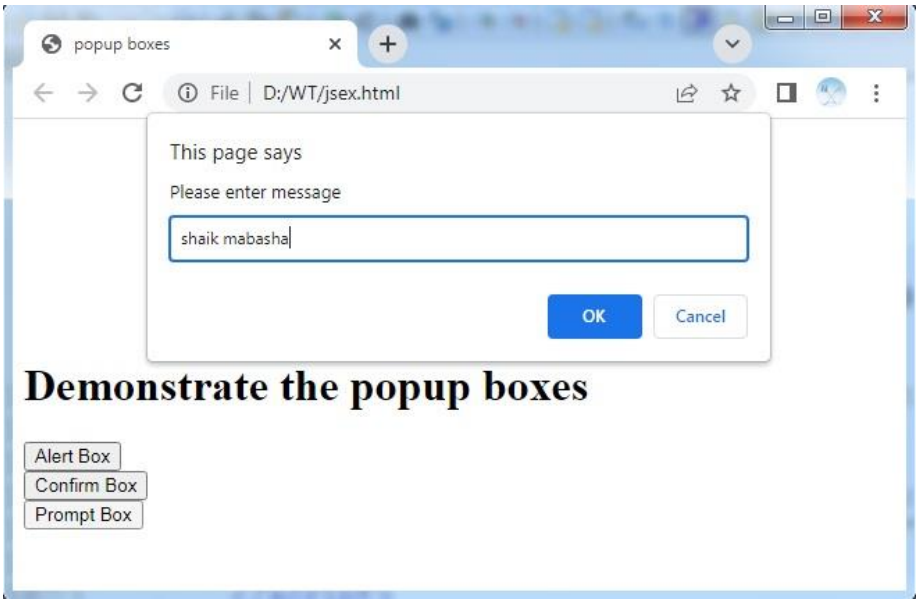
```
<!DOCTYPE  html>
<html>
	<head>
		<title>popup boxes</title>
		<script type="text/javascript">
			function alertmsg(){
				alert("This is alert message");
			}
			function confirmmsg(){
				confirm("Are you sure you want to move this file to Recycle Bin?");
			}
			function promptmsg(){
				var msg = prompt("Please enter message");
				document.getElementById("pmsg").innerHTML="Your message: "+msg;
			}
		</script>
	</head>
	<body>
		<br><br><br><br><br><br><br><br>
		<h1>Demonstrate the popup boxes</h1>
		<button onclick="alertmsg()">Alert Box</button><br>
		<button onclick="confirmmsg()">Confirm Box</button><br>
		<button onclick="promptmsg()">Prompt Box</button><br>
		<p id="pmsg"></p>
	</body>
</html>
```

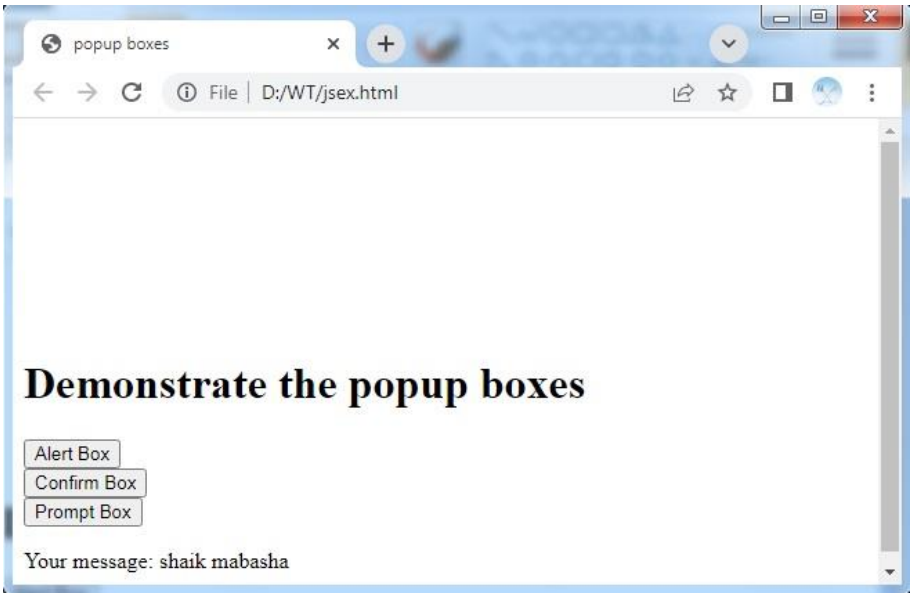
Output 1: popup box

Output 2: confirm box



Output 3: prompt box





JavaScript Functions, Events, Image Maps, and Animations

Exploring Functions:

- A function is a collection of statements that is executed when it is called at some point in a program.
- JavaScript functions are nothing, but a block of code designed to be performing a specific task.
- The JavaScript functions are divided into two categories:
  a. **Function without parameters:** Does not contain parameters.
  b. **Function with parameters:** Contain parameters in parenthesis.

JavaScript provides number of build-in global function; some of them are as follow

Function	Description
alert()	Display the information in message box. This function displays error message when you validate a form.
prompt()	The confirm box contains OK and CANCEL buttons. If the user clicks on OK button it return true value and if user click on CANCEL button it return false value.
confirm()	The confirm box contains OK and CANCEL buttons. If the user clicks on OK button it return true value and if user click on CANCEL button it return false value.
eval()	Evaluate and execute a string and return a result.
isFinite()	Return Boolean value; indicate whether the argument passed to it is finite or infinite.
isNaN()	Determines whether or not a value is an illegal number. NaN stands for Not a Number.
parseInt()	Extract a integer number from beginning of a string.
parseFloat()	Extract a floating pint number from beginning of a string.
Number()	Converts a value of an object into a number.
escape()	Encodes special characters except *,@,-,+ and /.
unescape()	Decodes a string that is encoded by escape() function.

Defining Function in JavaScript

- To define a function in JavaScript, use **function** keyword, followed by the function name, which is followed by the parentheses (contains parameter list).
- Syntax to define a function in JavaScript:

```
function   funName (parameter1, parameter2, parameter3) {
        Code to be executed here
}
```

- Example:

```
<script type="text/javascript">
function  alertmsg()
{
        alert("Hello JavaScript, I am Statements of Function Definition");
}
</script>
```

Calling (Invoking) Function in JavaScript

- To call a function in JavaScript, you have to simply write the name of the function which is going to be called.
- Example:

```
<script type="text/javascript">
        alertmsg()
</script>
```

Calling function with Timer:

- In JavaScript the timer is a very important feature
- It allows us to execute a JavaScript function after a specified period, thereby making it possible to add a new dimension, time, to our website.
- With the help of the timer, we can run a command at specified intervals, run loops repeatedly at a predefined time, and synchronize multiple events in a particular time span.
- There are various methods for using it as in the following:
  1. setTimeout()
  2. clearTimeout()
  3. setInterval()
  4. clearInterval()

1. The setTimeout() method:

- Executes code at a specified interval.
- Syntax: **setTimeout(function, delayTime)**
- Here, **function** parameter specifies the method that the timer calls and the **delayTime** parameter specifies the number of milliseconds to wait before calling the method.

2. The clearTimeout() method:

- Deactivates or cancels the timer that is set using the setTime() method.
- Syntax: **clearTimeout(timer)**
- Here, **timer** is a variable that is created using the setTimeout() method.

3. The setInterval() method:

- Executes a function after a specified time interval.
- Syntax: **setInterval(function, intervalTime)**
- Here, function parameters specify the method to be called; whereas, the intervalTime parameter specifies the time interval between the function calls.

4. The clearInterval() method:

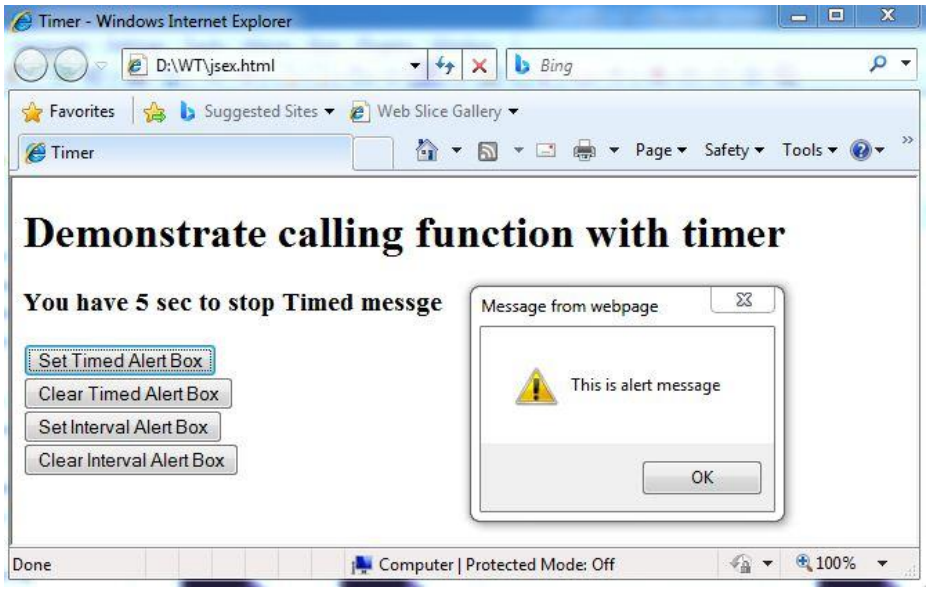
- Deactivates or cancels the timer that is set using the setInterval() method.
- Syntax: **clearInterval(timer)**
- The preceding syntax deactivates the inner timer variable that is created using the setInterval() method.

Example: create a HTML document that demonstrate the calling function with timer in JavaScript

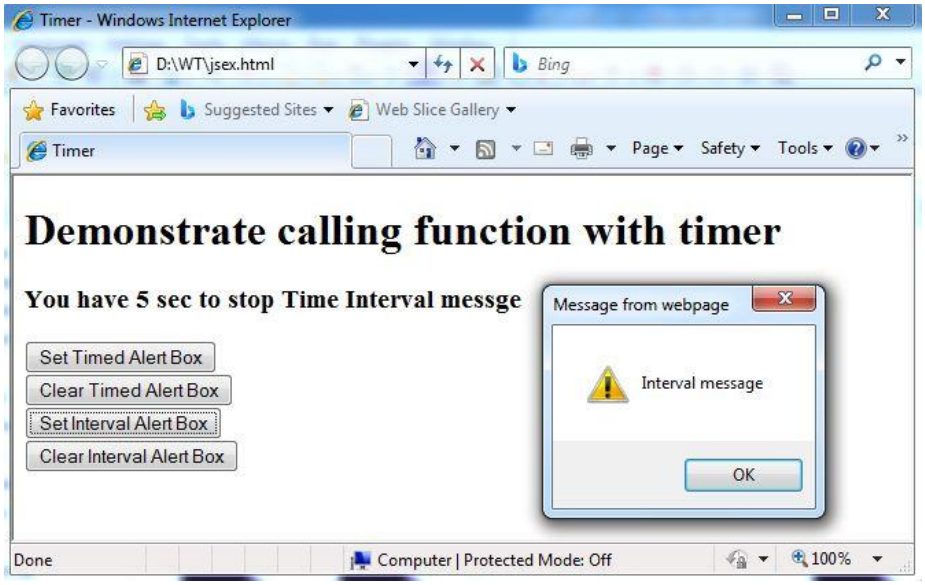
```
<!DOCTYPE html>
<html>
	<head>
		<title>Timer</title>
		<script type="text/javascript">
			var t,i;
			function timedmsg(){
		document.getElementById("msg").innerHTML="You have 5 sec to stop Timed messge";
				t = setTimeout("alert('This is alert message')",5000);
			}
			function cleartimedmsg(){
				clearTimeout(t);
			}
			function intervalmsg(){
	document.getElementById("msg").innerHTML="You have 5 sec to stop Time Interval messge";
				i = setInterval("alert('Interval message')",5000);
			}
			function clearintervalmsg(){
				clearInterval(i);
			}
		</script>
	</head>
```

```
        <body>
                <h1>Demonstrate calling function with timer</h1>
                <h3><p id="msg"></p></h3>
                <button onclick="timedmsg()">Set Timed Alert Box</button><br>
                <button onclick="cleartimedmsg()">Clear Timed Alert Box</button><br>
                <button onclick="intervalmsg()">Set Interval Alert Box</button><br>
                <button onclick="clearintervalmsg()">Clear Interval Alert Box</button><br>
        </body>
</html>
```

Output 1: setTimeout



Output 2: setInterval



Exploring Events:

- Events in JavaScript refer to actions that are detected by a JavaScript program when you perform a particular task.
- For example, onclick event is detected by the program when you click the mouse button.
- Syntax: **onEvent = "code to handle the event"**

Form Events:

- Events triggered by actions inside a HTML form (applies to almost all HTML elements, but is most used in form elements):

Attribute	Description	New in HTML5
onfocus	Fires the moment when the element gets focus	NO
onblur	Fires the moment that the element loses focus	YES
onchange	Fires the moment when the value of the element is changed	YES
oninput	Script to be run when an element gets user input	YES
onselect	Fires after some text has been selected in an element	NO
oninvalid	Script to be run when an element is invalid	YES
onsearch	Fires when the user writes something in a search field (for <input type="search">)	YES
onreset	Fires when the Reset button in a form is clicked	YES
onsubmit	Fires when a form is submitted	NO

Keyboard Events:

Attribute	Description	New in HTML5
onkeydown	Fires when a user is pressing a key	No
onkeypress	Fires when a user presses a key	No
onkeyup	Fires when a user releases a key	No

Mouse Events:

Attribute	Description	New in HTML5
onclick	Fires on a mouse click on the element	No
ondblclick	Fires on a mouse double-click on the element	No
onmousedown	Fires when a mouse button is pressed down on an element	No
onmouseup	Fires when a mouse button is released over an element	No
onmouseover	Fires when the mouse pointer moves over an element	No
onmouseout	Fires when the mouse pointer moves out of an element	No
onmousemove	Fires when the mouse pointer is moving while it is over an element	No
onmousewheel	Deprecated. Use the onwheel attribute instead	Yes
onwheel	Fires when the mouse wheel rolls up or down over an element	Yes
ondrag	Script to be run when an element is dragged	Yes
ondragend	Script to be run at the end of a drag operation	Yes
ondragenter	Script to be run when an element has been dragged to a valid drop target	Yes
ondragleave	Script to be run when an element leaves a valid drop target	Yes
ondragover	Script to be run when an element is being dragged over a valid drop target	Yes
ondragstart	Script to be run at the start of a drag operation	Yes
ondrop	Script to be run when dragged element is being dropped	Yes
onscroll	Script to be run when an element's scrollbar is being scrolled	Yes

Clipboard Events:

Attribute	Description
oncopy	Fires when the user copies the content of an element
oncut	Fires when the user cuts the content of an element
onpaste	Fires when the user pastes some content in an element

Media Events:

- Events triggered by medias like videos, images and audio (applies to all HTML elements, but is most common in media elements, like <audio>, <embed>, <img>, <object>, and <video>).

Attribute	Description	New in HTML5
onabort	Script to be run on abort	Yes
oncanplay	Script to be run when a file is ready to start playing (when it has buffered enough to begin)	Yes
oncanplaythrough	Script to be run when a file can be played all the way to the end without pausing for buffering	Yes
oncuechange	Script to be run when the cue changes in a <track> element	Yes
ondurationchange	Script to be run when the length of the media changes	Yes
onemptied	Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)	Yes
onended	Script to be run when the media has reach the end (a useful event for messages like "thanks for listening")	No
onerror	Script to be run when an error occurs when the file is being loaded	Yes
onloadeddata	Script to be run when media data is loaded	Yes
onloadedmetadata	Script to be run when meta data (like dimensions and duration) are loaded	Yes
onloadstart	Script to be run just as the file begins to load before anything is actually loaded	Yes
onpause	Script to be run when the media is paused either by the user or programmatically	Yes
onplay	Script to be run when the media is ready to start playing	Yes
onplaying	Script to be run when the media actually has started playing	Yes
onprogress	Script to be run when the browser is in the process of getting the media data	Yes
onratechange	Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode)	No
onseeked	Script to be run when the seeking attribute is set to false indicating that seeking has ended	Yes
onseeking	Script to be run when the seeking attribute is set to true indicating that seeking is active	Yes
onstalled	Script to be run when the browser is unable to fetch the media data for whatever reason	Yes
onsuspend	Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason	Yes
ontimeupdate	Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media)	Yes
onvolumechange	Script to be run each time the volume is changed which (includes setting the volume to "mute")	Yes
onwaiting	Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data)	No

Browser (Window) Events:

- Events triggered for the window object (applies to the <body> tag):

Attribute	Description	New in HTML5
onafterprint	Script to be run after the document is printed	Yes
onbeforeprint	Script to be run before the document is printed	Yes
onbeforeunload	Script to be run when the document is about to be unloaded	Yes
onerror	Script to be run when an error occurs	Yes
onhashchange	Script to be run when there has been changes to the anchor part of the a URL	Yes
onload	Fires after the page is finished loading	No
onmessage	Script to be run when the message is triggered	Yes
onoffline	Script to be run when the browser starts to work offline	Yes
ononline	Script to be run when the browser starts to work online	Yes
onpagehide	Script to be run when a user navigates away from a page	Yes
onpageshow	Script to be run when a user navigates to a page	Yes
onpopstate	Script to be run when the window's history changes	Yes
onresize	Fires when the browser window is resized	Yes
onstorage	Script to be run when a Web Storage area is updated	Yes
onunload	Fires once a page has unloaded (or the browser window has been closed)	Yes
onundo	Triggers at the time of performing the undo action in a document.	Yes
onblur	Trigger when a window loses focus	No
onfocus	Trigger when a window gets focus	No
onredo	Triggers at the time of performing the redo action in a document.	Yes

One marks:

1. Define CSS and write its syntax.
2. List the selectors in CSS
3. Write the difference between internal and external style sheet.
4. Write the code to link the external style sheet to HTML document.
5. Define gradient.
6. Differentiate the position:static and position:fixed properties.
7. Differentiate the display:none and visibility:hidden properties.
8. Differentiate static and dynamic HTML.
9. What is JavaScript?
10. List the feature of JavaScript.
11. Write the syntax to insert the JavaScript to HTML document.
12. Is java and JavaScript are same?
13. What are the ways of declaring variables in JavaScript?
14. Why JavaScript is not strongly typed language?
15. Why JavaScript is object based language?
16. List the popup boxes in javascript.
17. How to define the functions in JavaScript and write the syntax.
18. Write the difference between setTimeout() and setInterval() methods.
19. Define event.
20. List some events in JavaScript.

Essay questions:

1. Explain the ways of inserting the CSS to HTML with an example.
2. List and write a shot note of selectors in CSS with an example.
3. Explain different popup windows in JavaScript.
4. Explain different control flow statements in JavaScript.
5. List and explain different mouse events in JavaScript with an example.
6. List and explain different form events in JavaScript with an example.