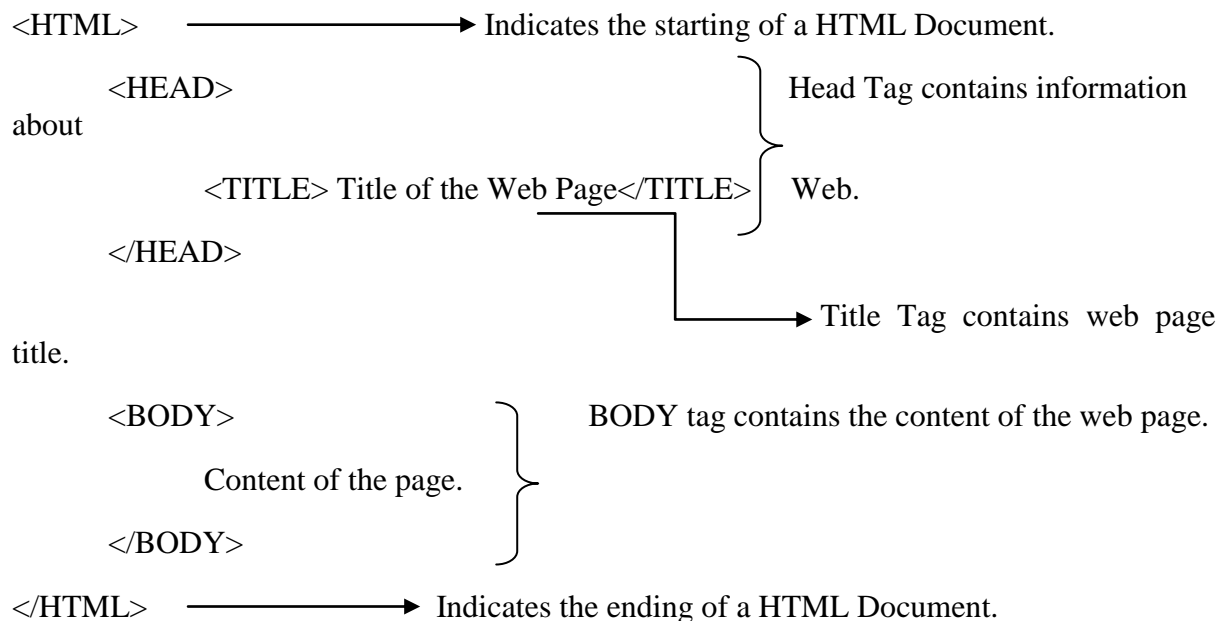


# HTML Basics

1. HTML means Hyper Text Markup Language. HTML is the language of the Internet used to create Web pages of web sites.
2. HTML is a set of tags or commands used to create HTML document.
3. HTML file display images, text, different font etc.
4. HTML has same features of a basic word processing program and it is capable of using graphics.
5. HTML is used to design various homepages and hypertext documents of web pages.
6. HTML document allows information to be presented in a multimedia format with hyperlinks.
7. Hyperlink is used to link between pages.

## STRUCTURE OF HTML DOCUMENT:

1. HTML files are normal text files, usually have the extension of .htm or .html.
2. HTML is not a case sensitive language.
3. HTML instructions divide the text of a document into blocks called elements.
4. An HTML document has two parts.
  - a. Head part
  - b. Body part



**HEAD:** The head element contains title and meta data of a web document.

**BODY:** The body element contains the information that display on a web page.

1. The first tag is `<html>`. This tag tells your browser that this is the start of an HTML document. The last tag is `</html>`. This tag tells your browser that this is the end of the HTML document.
2. The text between the `<head>` tag and `</head>` tag is header information. Header information is not displayed in the browser window.
3. The text between the `<title>` tag and `</title>` tag is the title of your document. The title is displayed in your browser's title bar.

4. The text between the `<body>` tag and `</body>` is the text that will be displayed in your browser.

## TYPES OF TAGS:

There are two types of tags in HTML. They are

1. Paired Tag:  
A Tag which has both opening and closing tag is called as Paired Tag.  
Example: `<b>Welcome</b>`
2. Un-Paired Tag:  
A Tag which doesn't have its companion tag is called as Un-Paired Tag.  
Example: `<br>`, `<hr>`

HTML supports 6 levels of heading tags.

`<h1>` - Level 1 heading

`<h2>` - Level 2 heading

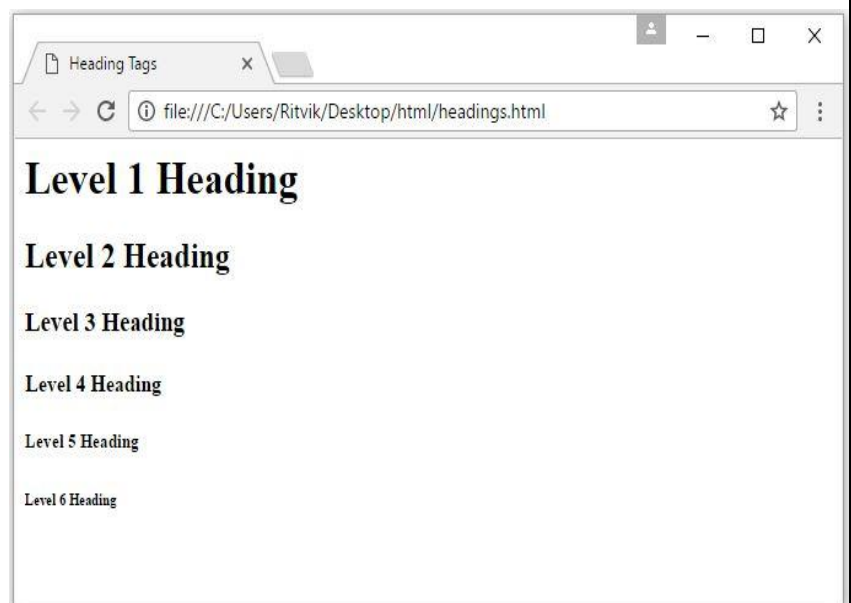
`<h3>` - Level 3 heading

`<h4>` - Level 4 heading

`<h5>` - Level 5 heading

`<h6>` - Level 6 heading

```
<HTML>
<HEAD>
<TITLE> Heading Tags </TITLE>
</HEAD>
<BODY>
<H1> Level 1 Heading</H1>
<H2> Level 2 Heading</H2>
<H3> Level 3 Heading</H3>
<H4> Level 4 Heading</H4>
<H5> Level 5 Heading</H5>
<H6> Level 6 Heading</H6>
</BODY>
</HTML>
```



## TEXT FORMATTING TAGS:

Text formatting elements are used to change the actual presentation styles of text to make the text bold, italic, underlined etc.

These elements are used to format texts of a page and present them effectively on the user's screen. All browsers support these elements.

The following are the important text formatting tags.

<code>&lt;B&gt;</code>	Specifies that the text written within <code>&lt;B&gt; ... &lt;/B&gt;</code> tags should be displayed in boldface.
<code>&lt;I&gt;</code>	Specifies that the text written within <code>&lt;I&gt; ... &lt;/I&gt;</code> tags should be displayed in Italic font style.

<U>	Specifies that the text written within <U> ... </U> tags should be underlined.
<BIG>	Specifies that the text written within <BIG> ... </BIG> tags should be displayed in bigger font size than the current font size.
<SMALL>	Specifies that the text written within <SMALL>...</SMALL> tags should be displayed in smaller font size than the current font size.
<STRIKE>	Specifies that the text written within <STRIKE> ... </STRIKE> tags should be displayed with a horizontal line striking through the text.
<SUB>	Specifies that the text written within <SUB>...</SUB> tags should be displayed as subscript by using a smaller font size.
<SUP>	Specifies that the text written within <SUP>...</SUP> tags should be displayed as superscript by using a smaller font size.

<HTML>

<HEAD>

<TITLE>Text Formatting Tags</TITLE>

</HEAD>

<BODY>

<B>Using Bold Tag</B><BR>

<I>Using Italic Tag </I><BR>

<U>Using Underline Tag</U><BR>

<BIG>Using Big Tag</BIG><BR>

<SMALL>Using Small Tag </SMALL><BR>

<STRIKE>Using STRIKE Tag</STRIKE><BR>

Using of Subscript Tag : X<SUB>1</SUB> , X<SUB>2</SUB><BR>

Using of Superscript Tag : X<SUP>2</SUB> , Y<SUP>3</SUB><BR>

</BODY>

</HTML>



## FONT TAG:

The <font> tag alone doesn't provide any real functionality, but with the help of a few attributes this tag is used to change the style, face, size and color of HTML text elements.

Attributes of font Tag:

1. Size: Changes the size of text. The value must be between 1 and 7.
2. Color: Changes the color of the text.
3. Face: Changes the font type of the text.

Syntax: <FONT SIZE=value COLOR=color FACE="font-names">

Example: <FONT SIZE=3 FACE="Times new Roman" COLOR=red> Hello World</FONT>

## IMAGE TAG:

1. Images are inserted in web documents using the IMG tag and this tag has no closing tag.
2. The <IMG> tag requires the location of the image file.
3. <IMG> tag alone doesn't place an image in the HTML document. It requires another attributes SRC, HEIGHT and WIDTH.

Syntax:

<IMG SRC="URL/Path of the image file" WIDTH=value HEIGHT=value>

Example:

<IMG SRC="adam.jpeg" WIDTH=300 HEIGHT=300>

## ANCHOR TAGS OR LINK TAG:

1. HTML allows linking to other HTML documents as well as images.
2. Hyperlinks appear blue in color by default.
3. When the mouse cursor is placed over it, the standard arrow shaped mouse cursor changes to the shape of a hand.

LINK – changes the default color of a hyperlink to whatever color is specified with this tag.

ALINK – changes the default color of a hyperlink that is activated to whatever color is specified with this tag.

VLINK – changes the default color of a hyperlink that is already visited to whatever color is specified with this tag.

Syntax: <A HREF="target file name"> Clickable Text </A>

Example: <A HREF=https://www.google.com>Click here for Google</A>

## LISTS IN HTML:

1. Lists are the index of items to be appeared in web pages in a specified format.
2. There are three types of lists
  - a. Ordered Lists

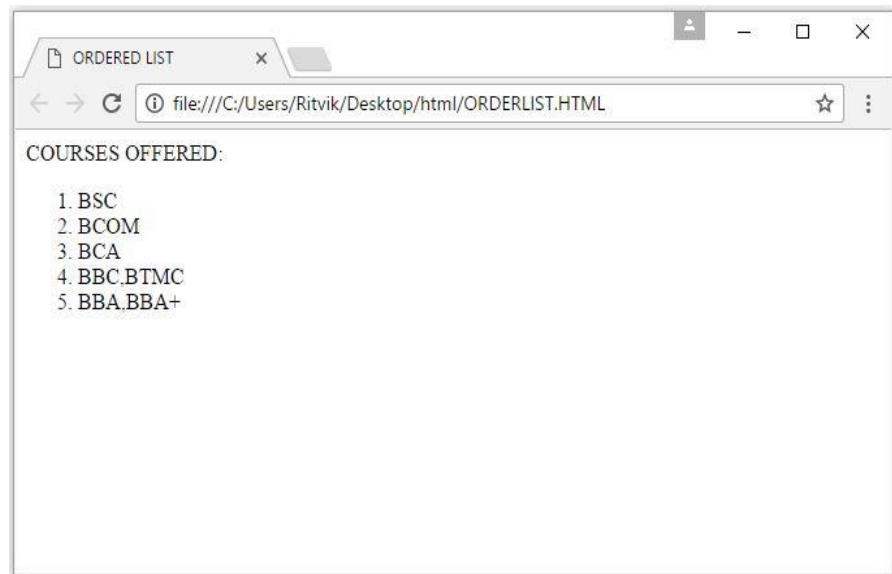
- b. Unordered Lists
- c. Definition Lists

<UL> </UL>	Specifies an Unordered List.
<OL> </OL>	Specifies an Ordered List.
<LI> </LI>	Specifies a list item.
<DL> </DL>	Specifies a Definition List.
<DT> </DT>	Specifies the term in a description list.
<DD> </DD>	Specifies description of term in a description list.

### ORDERED LIST:

1. The ordered list is also known as the Number List, in which each list item has a number in front of it.
2. To create ordered list use <OL> tag.

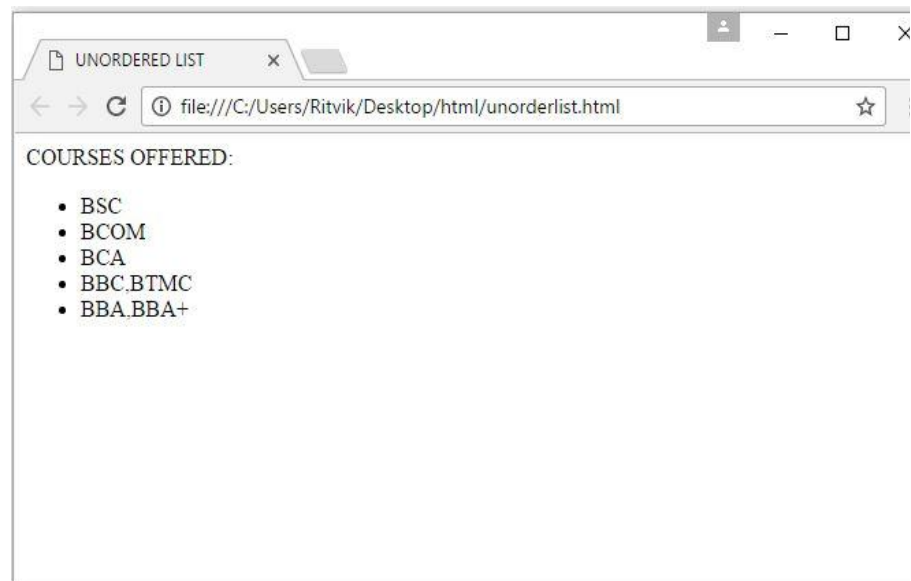
```
<HTML>
<HEAD>
<TITLE>ORDERED
LIST</TITLE>
</HEAD>
<BODY>
COURSES
OFFERED:
<OL>
<LI>BSC
<LI>BCOM
<LI>BCA
<LI>BBC,BTMC
<LI>BBA,BBA+
</OL>
</BODY>
</HTML>
```



### UNORDERED LIST:

1. The unordered list is created using <UL> tag.
2. Each list item is displayed with bullets if we use unordered lists.

```
<HTML>
<HEAD>
<TITLE>UNORDERED
LIST</TITLE>
</HEAD>
<BODY>
COURSES
OFFERED:
<UL>
<LI>BSC
<LI>BCOM
<LI>BCA
<LI>BBC,BTMC
<LI>BBA,BBA+
</UL>
</BODY>
</HTML>
```



## DEFINITION LIST:

1. A definition list consists of name-value groups.
2. Definition lists are intended for groups of terms and definitions, meta data topics and values.

<DT> .... </DT> : A name in a definition list

<DD> .... </DD> : A value in a definition list

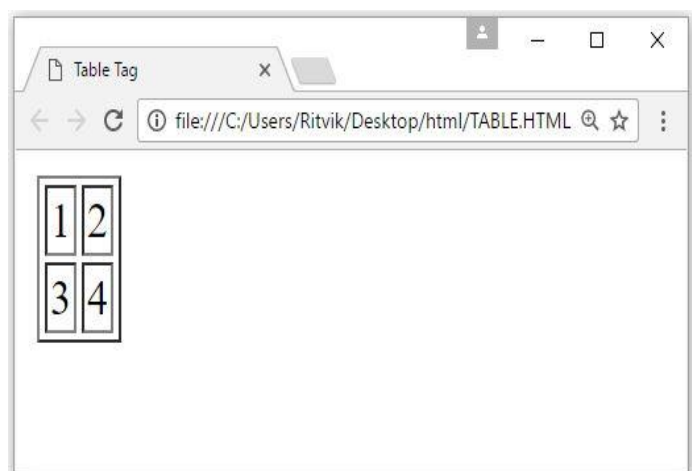
## TABLE TAG:

An HTML table is a rectangular grid of rows and columns on a web page, to which you can enter all kinds of information, including text, numbers, links, and even images.

<TABLE> </TABLE>	Specifies a table
<TR> </TR>	Specifies a row in the table.
<TD> </TD>	Specifies the data in a cell of the table.
<TH> </TH>	Specifies header cell in the table.

1. The BORDER attribute in <TABLE> tag tells the visibility of table border.
2. The ROWSPAN attribute in <TR> tag is used to merge the cells vertically.
3. The COLSPAN attribute in <TR> tag is used to merge the cells horizontally.
4. The CELLPADDING attribute in <TABLE> tag specifies the distance between the cell content and cell borders.
5. The CELLSPACING attribute in <TABLE> tag specifies the distance between the cells in a table.

```
<HTML>
<HEAD>
<TITLE>Table Tag</TITLE>
</HEAD>
<BODY>
<TABLE BORDER=1>
<TR><TD>1</TD><TD>2</TD></TR>
<TR><TD>3</TD><TD>4</TD></TR>
</TABLE>
</BODY>
</HTML>
```



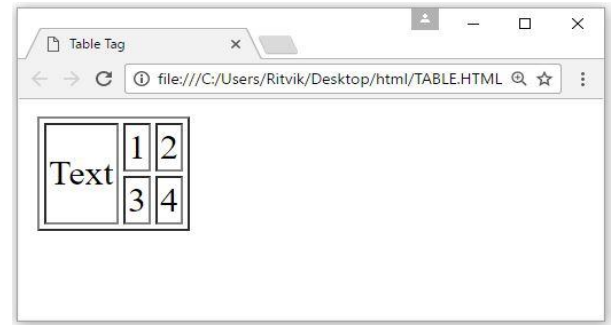
```
<TABLE BORDER=1>
<TR>
<TD COLSPAN=3>Enter Text</TD>
</TR>
<TR>
<TD>1</TD><TD>2</TD><TD>3</TD></TR>
</TABLE>
```



```

<TABLE BORDER=1>
<TR><TD
ROWSPAN=3>Text</TD><TD>1</TD><TD>2
</TD></TR>
<TR><TD>3</TD><TD>4</TD></TR>
</TABLE>

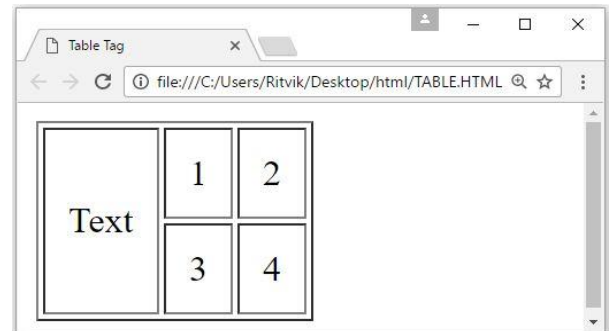
```



```

<TABLE BORDER=1 CELLPADDING=10>
<TR><TD
ROWSPAN=3>Text</TD><TD>1</TD><TD>2
</TD></TR>
<TR><TD>3</TD><TD>4</TD></TR>
</TABLE>

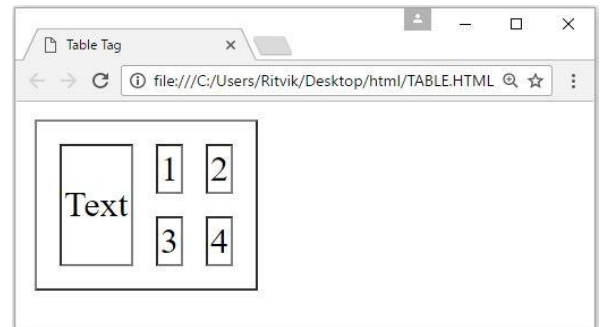
```



```

<TABLE BORDER=1 CELLSPACING=10>
<TR><TD
ROWSPAN=3>Text</TD><TD>1</TD><TD>2
</TD></TR>
<TR><TD>3</TD><TD>4</TD></TR>
</TABLE>

```



## FRAMES IN HTML:

1. Frames divide a browser window into two or more document windows, each displaying a different document, or a different part of the same document.
2. Frames in an HTML document can cause a web page to appear to be divided into multiple, scrollable regions.
3. Each frame can be assigned a name, a source document locator, dimensions, border alignment and decoration, scroll and resize etc.
4. Frames in HTML documents are created and controlled through the structure of three element types:

<FRAMESET> → Define a set of frames

<FRAME> → Define a sub window (a frame)

<NOFRAME> → Defines a noframe section for browsers that do not handle frames

Syntax for FRAMES:

```

<FRAMESET cols="50%,50%">
  <FRAME name="frame1">
  <FRAME name="frame2">
</FRAMESET>

```

## Elements of FRAMESET:

1. ROWS: specifies the number of rows and their height in either pixels, percentages. Default is 100%.
2. COLS: specifies the number of columns and their height in pixels, percentages. Default value is 100%.
3. FRAMEBORDER: determines if there should be 3D borders between the frames which should be either yes(1) or no(0).
4. FRAMESPACING: specifies space between the frames in integer.
5. BORDERCOLOR: specifies the color of border in the frameset in '#rrggbb'.

## Elements of FRAME:

1. NAME: assigns a name to a frame.
2. SRC: define the path or full URL to the HTML page to appear in a named frame.
3. SCROLLING: specifies whether a frame should be scrollable or not. (auto, yes, no)
4. FRAMEBORDER: specifies whether a frame should have a border or not.
5. MARGINWIDTH: specifies the margin, in pixels, between the frame's contents and its left and right margins.
6. MARGINHEIGHT: specifies the margin, in pixels, between the frame's content and its top and bottom margins.
7. NORESIZE: making individual frames non-resizable.
8. TARGET: used to direct the new page to another named frame.
  - \_self: a document is loaded into the same frame.
  - \_parent: a document is loaded into the same frameset window.
  - \_top: a document is loaded into the full area of the browser window and all frames will be removed.
  - \_blank: a document is loaded into a new and completely separate window.

### framedemo.html

```
<html>
<head>
  <title>Frame Demo</title>
</head>
<frameset rows="150,850">
  <frame name="titles" src="titles.html">
<frameset cols="150,850">
  <frame name="links" src="links.html">
  <frame name="destination">
</frameset>
</frameset>
</html>
```

### titles.html

```
<html>
<head>
  <title>Heading Frame</title>
</head>
<body bgcolor=gray>
<center>
<h1>HTML Tags</h1>
</center>
</html>
```

### links.html

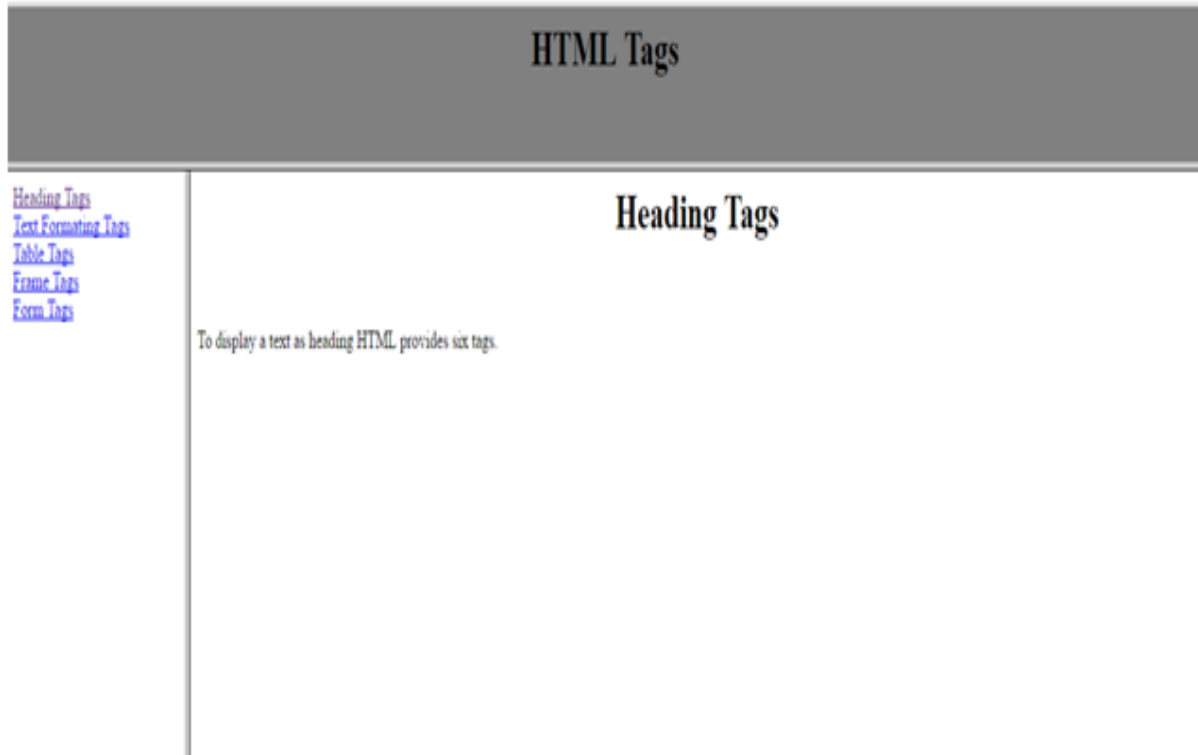
```
<html>
<head><title>Links</title></head>
<body>
<a href="heading.html" target="destination">Heading Tags</a><br>
<a href="text.html" target="destination">Text Formating Tags</a><br>
<a href="table.html" target="destination">Table Tags</a><br>
```



```
<a href="frames.html" target="destination">Frame Tags</a><br>
<a href="form.html" target="destination">Form Tags</a><br>
</body>
</html>
```

**NOFRAMES:** If your browser does not support frames then NOFRAME section will be displayed on the web page.

```
<html>
<head><title>Frames Set with Header</title></head>
<FRAMESET rows="45,*">
<FRAME name="top" scrolling="no" noresize target="main" src="form1.html">
<FRAME name="main" src="form2.html" scrolling="auto">
<NOFRAMES>
<BODY>
<P>This page uses frames, but your browser doesn't support them.<br>
</BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```



#### ADVANTAGES:

1. The most obvious feature of frames is the ability to keep one part of the page static while changing another part.
2. Frames can also help reduce bandwidth and server load, because the same content does not need to be loaded every time a new page is visited.

#### DISADVANTAGES:

1. The browser's back button does not work if a web page is designed using frames.
2. Frames reduce the amount of usable space on the web page.
3. The URL in the address bar will not change when you navigate to different links.
4. It is not possible to bookmark a web page since the URL will not change.
5. Frames create problems with printing.

## FORMS

1. Forms are used to add an element of interactivity to a web site. Usually forms are used to let the user send information back to the server.

`<FORM name="form1" action="URL" method="post/get"> . . . </FORM>`

2. The action attribute specifies the name and location of a CGI script that will be used to process the data.
3. Data can be sent in one of two ways: post or get. Get is used to retrieve information from a server and post is used to send information to a server.
4. Forms are a mechanism that allow the user to type information into fields on a browser screen and submit to a web server.
5. Forms provide an interface for collecting, displaying, and delivering information, and are used as a key component in HTML.
6. The HTML forms are created using a `<form>` and `</form>` dependent tag. All the form controls included in `<input>` elements should be enclosed between the `<form>` and `</form>` tag.
7. A webpage contain more than one HTML forms. But Forms are not nested in HTML it means user is not allowed to implement one form in another form.
8. The basic structure of the form is created using the following code:

`<form action="URL" method="post/get">`

`<input type=control based on user choice>`

`</form>`

### ELEMENTS OF FORM Tag:

1. Elements are the sub tags implemented between the starting and closing of the root element.
2. HTML `<form>` support with many elements which plays a vital role in creating and executing user defined structure for processing the information.
3. The following are the sub element of `<form>` tag which supports with different controls to accept and submit message from html forms.
  - a. `<INPUT>`
  - b. `<SELECT>`
  - c. `<TEXTAREA>`
  - d. `<FIELDSET>`

### INPUT Tag:

1. The primary element of HTML FORM which helps the user to create various controls using its **type** attribute.
2. It is an independent tag, which don't have a corresponding closing tag.

`<input type=`

- |             |           |
|-------------|-----------|
| 1. Text     | 6. Submit |
| 2. Password | 7. Button |
| 3. Checkbox | 8. File   |
| 4. Radio    | 9. Image  |
| 5. Reset    |           |

Name, value, size, maxlength, checked, style, align, type are the common attributes of input element.

### **TEXT BOX:**

- ❖ A rectangular shaped field in which a user can enter text is considered as text box.
- ❖ TYPE and NAME attributes are required for the input tag. The name attribute specifies the name of the parameter that will be assigned as the value that the user enters into the field.
- ❖ The SIZE attribute specifies that the text box is to be how many characters wide and the MAXLENGTH attribute specifies that the most characters a user can enter into the field.

Example:

Enter your Name: `<input type="text" name="T1" size="30" maxlength="50">`

### **PASSWORD:**

- ❖ The password input type creates a single line empty text box where the user can enter text into it.
- ❖ It is very similar to text field.
- ❖ The difference is the display structure of the text is changed into “stars” or “dots”.

Example:

Enter password: `<input type="password" size="25">`

### **CHECKBOX:**

- ❖ A checkbox is represented by square icon, which is used for multiple selections.
- ❖ The user can select or deselect by clicking on it.
- ❖ Checkbox are often used in series, so that a user can easily specify all of their preferences.

Example:

Select Your Qualification:  
`<input type="checkbox">S.S.C`  
`<input type="checkbox">Intermediate`  
`<input type="checkbox">Graduation`  
`<input type="checkbox">Post Graduation`

### **RADIO BUTTONS:**

- ❖ Radio buttons are similar to check box but these are displayed in circular format.
- ❖ These radio buttons can be implemented in 2 ways.
  - a. Which is used for multiple selection as checkbox
  - b. Group selection
- ❖ In group selection all radio buttons have the same name, so that a user can select only one element.

Example: `<input type="radio" name="gender" value="male">Male`  
`<input type="radio" name="gender" value="female">Female`

## BUTTONS:

- ❖ There are two types of buttons. Predefined and user defined where predefined are considered as Action Buttons.
- ❖ User defined button helps the user to create his own control button and fix the action using script code.

a. As the type of input tag

```
<input type="button" value="Login">
```

b. As the element of form tag

```
<button value="Login">
```

- ❖ There are two types of action buttons. They are **SUBMIT** and **RESET**. When the user clicks on submit all the form data will be sent for process.

```
<input type="submit" value="submit">
```

- ❖ The reset button allows the user to clear all the entered data in the form.

```
<input type="reset" value="clear">
```

## FILE:

- ❖ This filed is used to include more than one file while submitting the form.
- ❖ After the form is processed the given files are stored into the disk of the web server.
- ❖ That is the reason file input is often known as “File Upload”.

```
<input type="file">
```

## IMAGE:

- ❖ The image can be used as a submit button or to collect data from the image itself.

```
<input type="image" src="submit.gif" alt="Submit" width="48" height="48">
```

## SELECT:

- ❖ The select tag allows to choose any subset of items from a group by using select tag<SELECT> with corresponding ending tag</SELECT>.
- ❖ The radio button and checkboxes will occupy a lot of screen space where as select tag does not take.
- ❖ The items in a given select tag are usually rendered in the style of a pop-up menu, indicated with in <OPTION></OPTION> tag. The closing tag is optional.

Select your Branch:

```
<select name="branch">
```

```
<option> Bsc
```

```
<option> BCA
```

```
<option> BCom
```

```
<option> BBC
```

```
<option> BBA
```

```
</select>
```

- ❖ You can create a group element so that the multiple blocks for multiple sub items can be created by using <optgroup></optgroup> tag a dependent element.

```
<select name="branch"
```

```

<optgroup label="BSc">
    <option value="mpcs">MPCS
    <option value="mscs">MSCS
    <option value="mecs">MECS
</optgroup>
<optgroup label="BCom">
    <option value="general">General
    <option value="computers">Computers
</optgroup>
</select>

```

## TEXTAREA:

- ❖ The <textarea> tag defines a multi-line text input control. A text area can hold any number of characters, and the text renders in a fixed width font.
- ❖ The size of a text area can be specified by the cols and rows attributes.

```

<textarea name="address" rows="20" cols="80" maxlength="2000" wrap>
Enter Text to be displayed.
</textarea>

```

- ❖ <FILEDSET> and <LEGEND> are used to draw the border for the specified form with a title in order to identify each form separately in a single web page.

```

<FIELDSET>
<LEGEND>Personal Information</LEGEND>
|
|
|
</FIELDSET>

```

```

<html>
<head><title>Registration Form</title></head>
<body>
<u><h1 align=center>Registration Form</h1></u>
<form>
<table align=center><tr><td>
<fieldset><legend>Personal Details</legend>
<table>
<tr><th>First Name</th><td><input type="text" name="fname"></td></tr>
<tr><th>Middle Name</th><td><input type="text" name="mname"></td></tr>
<tr><th>Last Name</th><td><input type="text" name="lname"></td></tr>
<tr><th>User ID</th><td><input type="text" name="uid"></td></tr>
<tr><th>Password</th><td><input type="password" name="pwd"></td></tr>
<tr><th>Confirm Password</th><td><input type="password" name="cnfpwd"></td></tr>
<tr><th>DOB</th><td>
<select><option>Day<option>1<option>2<option>3</select>
<select><option>Month<option>Jan<option>Feb<option>March</select>
<select><option>Year<option>1989<option>1990<option>1991<option>1992</select></td></tr>
<tr><th>Gender</th><td><input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="female">Female </td></tr>

```

```

</table>
</fieldset>
</td></tr>
<tr><td>
<fieldset><legend>Address</legend>
<table>
<tr><th>Present Address</th><th>Permanent Address</th></tr>
<tr><td><textarea rows=5 cols=30></textarea></td><td><textarea rows=5
cols=30></textarea></td></tr>
<tr><td colspan=2 align="center"><input type="button" value="Submit"><input
type="reset"></td></tr>
</table> </fieldset> </td></tr> </table> </form> </body> </html>

```

## Registration Form

Personal Details	
<b>First Name</b>	<input style="width: 90%;" type="text"/>
<b>Middle Name</b>	<input style="width: 90%;" type="text"/>
<b>Last Name</b>	<input style="width: 90%;" type="text"/>
<b>User ID</b>	<input style="width: 90%;" type="text"/>
<b>Password</b>	<input style="width: 90%;" type="password"/>
<b>Confirm Password</b>	<input style="width: 90%;" type="password"/>
<b>DOB</b>	<div style="display: flex; gap: 5px;"> <div style="border: 1px solid #ccc; padding: 2px 5px;">Day ▼</div> <div style="border: 1px solid #ccc; padding: 2px 5px;">Month ▼</div> <div style="border: 1px solid #ccc; padding: 2px 5px;">Year ▼</div> </div>
<b>Gender</b>	<input type="radio"/> Male <input type="radio"/> Female

Address	
<b>Present Address</b> <div style="border: 1px solid #ccc; height: 100px; margin-top: 5px;"></div>	<b>Permanent Address</b> <div style="border: 1px solid #ccc; height: 100px; margin-top: 5px;"></div>
<div style="display: flex; justify-content: center; gap: 20px;"> <div style="border: 1px solid #ccc; padding: 5px 15px; background-color: #f0f0f0;">Submit</div> <div style="border: 1px solid #ccc; padding: 5px 15px; background-color: #f0f0f0;">Reset</div> </div>	

## UNIT –II

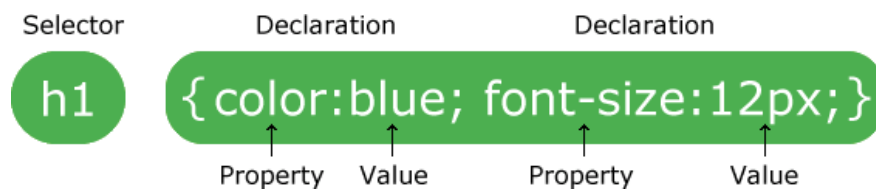
### CASCADING STYLE SHEETS

#### Introduction:

- ❖ CSS stands for Cascading Style Sheets.
- ❖ CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- ❖ CSS saves a lot of work. It can control the layout of multiple web pages all at once.
- ❖ External style sheets are stored in CSS files.
- ❖ CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

#### SYNTAX:

A CSS rule-set consists of a selector and a declaration block:



- ❖ The selector points to the HTML element you want to style.
- ❖ The declaration block contains one or more declarations separated by semicolons.
- ❖ Each declaration includes a CSS property name and a value, separated by a colon.
- ❖ A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

#### Example:

In the following example all <p> elements will be center-aligned, with a red text color:

```
<!DOCTYPE html>
<html>
<head>
<style>      p { color: red; text-align: center; }  </style>
</head>
<body>
<p>Hello World!</p>
<p>Aditya Degree College Rajamahendravaram</p>
</body>
</html>
```

#### Advantages and Disadvantages of CSS:

- ❖ CSS allows you to specify how documents are presented to users by using various style properties for a given HTML element.
- ❖ CSS properties are the key to altering the styling of HTML elements in your web documents.
- ❖ **Separation of Style and Structure:** It allows separating content of an html document from the layout & style of that document.
- ❖ **Search engine friendly:** It helps get your pages found in a search engine. Since your CSS website contains less code and has a simpler structure which allows search engine spiders to pass from your code faster means your web pages can be indexed faster.

- ❖ **Save time:** It flexibility to set the properties of an component. You can write CSS code and then the same code can be applied to the HTML components groups, and also can be reused in different HTML pages.
- ❖ **Easy to Maintain:** It gives an easy mode to update document formatting and manage consistency across various documents. By doing one change to the website's CSS file, all the web pages' elements will be updated automatically.
- ❖ **Pages load faster:** It empowers multiple pages to share formatting, and reduce complexity and redundancy in the structural content. It significantly reduces the file transfer size, which helps to lode page faster.
- ❖ **Superior styles to HTML:** It has more extensive presentation capabilities than HTML, so you can give far superior look to your HTML pages in contrast with the HTML elements and attributes.
- ❖ **Multiple Device Compatibility:** It allows the HTML document to be improved for more than kind of device. Using it the same HTML document can be presented in various review styles for various rendering devices.
- ❖ **Provide Platform independence:** It offer consistent platform independence and also support latest browsers as well.
- ❖ **Global web standards:** It's a better to start using CSS in all the HTML pages to make them compatible for future browsers. There are various CSS properties which help to enhance your web experience.

## TYPES OF CSS

There are the following three types of CSS:

1. Inline CSS.
2. Internal / Embedded CSS.
3. External CSS.

### Inline CSS

- ❖ For Inline CSS every style content is in HTML elements.
- ❖ It is used for a limited section.
- ❖ Whenever our requirements are very small we can use inline CSS.
- ❖ It will affect only single elements.
- ❖ In HTML we require that various HTML tag's views are different so then we use inline Cascading Style Sheets.
- ❖ There are disadvantage of inline Cascading Style Sheets.
- ❖ It must be specified on every HTML tag.
- ❖ There is very much time consumed by that and it is not the best practice for a good programmer and the code will be quite large and very complex.

Examples

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue;">This is a Blue Heading</h1>
</body>
</html>
```

### Internal CSS

- ❖ In internal CSS the style of CSS is specified in the <head> section.
- ❖ This is internal CSS, it affects all the elements in the body section.
- ❖ Internal CSS is used in the condition when we want a style to be used in the complete HTML body.
- ❖ For that we can use style in the head tag.
- ❖ This style performs an action in the entire HTML body.

```
<!DOCTYPE html>
<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

- ❖ In External CSS we create a .css file and use it in our HTML page as per our requirements.
- ❖ Generally external Cascading Style Sheets are used whenever we have many of HTML attributes and we can use them as required.
- ❖ There is no need to rewrite the CSS style again and again in a complete body of HTML that inherits the property of the CSS file.

Example:

styles.css

```

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

```

HTML Program

```

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

```

Defining your own styles

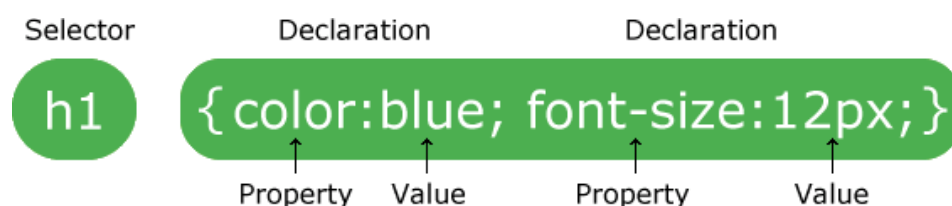
Styles are defined by simple rules. A style can contain as many rules as you want and with processing

### HTML. 1. Cascading styles:

- ❖ Conventionally styles are cascaded (overridden).
- ❖ This means that you do not have to use just a single set of rules inside a document, you can import as many styles as you like.
- ❖ The only difficulty with importing multiple style sheets is that they cascade.
- ❖ This means that the first is overridden by second, the second by third and so on.

### 2. Style Rule:

A style rule has two parts: The selector and a set of declarations. The selector is used to create a link between the rule and the HTML tag.



The declaration part has two parts: a property and a value.

Selectors can be placed into classes so that a tag can be formatted in variety of ways. Declarations must be separated using colons and terminated using semicolon.

### 3. Classes

- A class is defined a set of styles which can be applied as user choice.
- Choices can be applied to a single type of element or may be anonymous and hence applicable to any element.
- If you want to apply a style to a particular element use the following syntax  
Selector.classname{property: value; property: value}  
<selector class="classname">

```
<!DOCTYPE html>
<html>
<head>
<style> p.center { text-align: center; color: red; } </style>
</head>
<body>
<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
</body>
</html>
```

### 4. Anonymous classes

- Sometimes you want to apply a piece of formatting too many different elements within a page but not necessarily to the entire page.
- Instead of defining styles to individual elements we are creating a style class that can be applicable any number of tags in your HTML document.
- Such classes are called anonymous classes. It has the following syntax  
.classname { Property: value; Property: value; }  
<selector1 class="classname"> ..... <selector1>  
<selector2 class="classname"> ..... <selector2>

Example:

```
<!DOCTYPE html>
<html>
<head>
<style> .center { text-align: center; color: red; } </style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
</body>
</html>
```

### Including style sheets

We can include style sheets in our HTML page in two way

- by linking
- by importing

#### By linking

We can link our .css file in the head section of a HTML page using the following syntax

```
<link rel="style sheet" href="URL" type="text/css" media="screen">
```

- The href is hyperlink to your style sheet
- Rel tells to the browser what type of link you're using
- Type tells to the browser what type of document you're including.
- the 'type' statement gives the relevant MIME type.
- HTML Specifies a Variety of ways of using a document a document, including screen viewing, printing and as a presentations.
- Use the 'media' attribute to describe the type of use.

```

<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css" media="screen">
  <style> h1 { color: orange; } </style>
</head>
<body>
  <h1>This is a heading</h1>
  <p>The style of this document is a combination of an external stylesheet, and internal style</p>
</body>
</html>

```

### By importing

```
[<style type="text/css">] <!--@import url(url); --> </style>
```

These lines are both needed if you intended to use more than one style sheet.

The first sheet is included as if it were the only one; any further style sheets have to be imported.

Example:

```

<link rel="stylesheet" type="text/css" href="mystyle.css" media="screen">
<style type="text/css">
  <!--@import url("http://www.smiggins.co./style.css"); --> </style>

```

## Properties and Values

There are number of style properties, including properties that control lists and positing floating elements like images.

Font properties:

- font-family: family name;  
Fonts are identified by giving the name of a specific font.
- font-style: normal | italic | oblique;  
This property provides the style to the font.
- font-weight: normal | bold | bolder | lighter;  
The font-weight property selects the weight or darkness of the font. Values of the property range from 100 to 900 in increments of 100.
- font-size: small | medium | large | smaller | larger;  
This property is used to set the relative or physical size of the font used.

Example:

```

<html>
<head>
<title> Font properties </title>
<style type="text/css">
  h1{ font-family:arial; font-style: italic; }
  P{ Font-weight: bold Font-size:100px; }
</style>
</head>
<body>
  <h1> demo from css properties </h1>
  <p> this paragraph represents font size and font weight</p>
</body>
</html>

```

## Backgrounds and color properties

Color:<value>

Background-color:<value> | transparent

Background-image: URL | none

- The color of any attribute can be changed. Value should be given as hexadecimal values.

- Backgrounds for the whole page or individual element can have their color set from the style sheet.
- Elements can also have transparent backgrounds.
- Instead of a color an image can be used, identified by its URL.

Example:

```
<html>
<head>
<title> Font properties </title>
<style type="text/css">
    h1{ background-color: white; color: black; }
    P{ background-color: black; color: white; }
</style>
</head>
<body>
    <h1> demo from css background and color properties </h1>
    <p> this paragraph represents font size and font weight</p>
</body>
</html>
```

## Text Properties

Text properties are used to effect the presentation, spacing and layout of text. The basic properties includes such as decoration, indentation, word spacing, letter spacing, spacing between lines, horizontal and vertical text alignment.

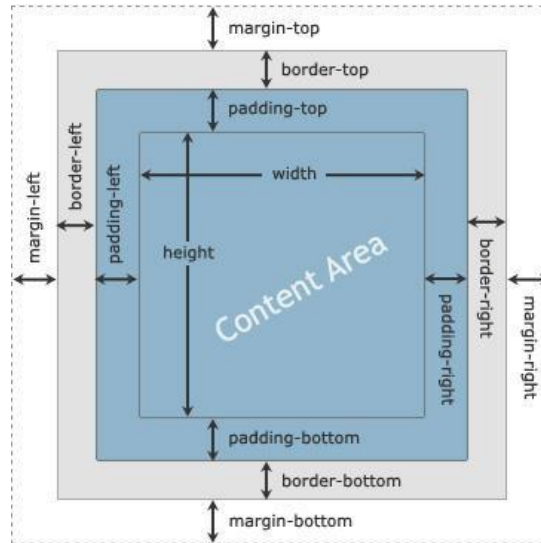
- text-decoration: none | underline | overline| line-through| blink;
- text-transformation: none | uppercase | lowercase| capitalize;
- text-align: left | right | center | justify;
- text-indentation: length | percentage;
- line-height : length | percentage;
- letter-spacing: length | percentage;
- word-spacing: length | percentage;

Example

```
<html>
    <head>
        <title> text properties </title>
        <style type="text / css">
            a{ text-decoration: overline;
                text-transformation: uppercase;
                text-align: left;
                text-indentation: 12px; }
        </style>
    </head>
    <body>
        <a href="D:/CSS Demo/My css.html"> This hypertext has text properties </a>
    </body>
</html>
```

## Border and Margin Properties:

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



- Margin: length | percentage | auto { 1,4 }
- Border-width: thin | thick | medium | length { 1,4 }
- Padding: length | percentage { 1,4 }

Padding - Clears an area around the content. The padding is transparent

Border - A border that goes around the padding and content

Margin - Clears an area outside the border. The margin is transparent

Border-color: value { 1,4 }

Border-style: none | dotted | solid | double | groove Ridge { 1,4 }

This sets the color of the border around the element.

Up to four different colors can be specified.

They are applied to the borders in the same orders as margins.

Each edge of the border can have a different style.

Width: length | percentage | auto Height: length | auto

Example:

```
<!DOCTYPE html>
<html>
<head>
<style> div { width: 320px; padding: 10px; border: 5px solid gray; margin: 0; } </style>
</head>
<body>
  <h2>Calculate the total width:</h2>
  
  <div>The picture above is 350px wide. The total width of this element is also 350px.</div> </body>
</html>
```

## Formatting Blocks of Information

- 🚦 A webpage may consist of more than one block.
- 🚦 We can format all the blocks with single style or separate style for each and every block.
- 🚦 This can be achieved in three ways
  1. Using style class
  2. Using <div> tag
  3. Using <span> tag

## Using class

A class is defined a set of styles which can be applied as user choice.

Choices can be applied to a single type of element or may be anonymous and hence applicable to any element.

If you want to apply a style to a particular element use the following syntax

```
Selector.classname{property: value; property: value}  
<selector class="classname">
```

```
<!DOCTYPE html>  
<html>  
<head>  
<style> p.center { text-align: center; color: red; } </style>  
</head>  
<body>  
<h1 class="center">This heading will not be affected</h1>  
<p class="center">This paragraph will be red and center-aligned.</p>  
</body>  
</html>
```

## Anonymous classes

Sometimes you want to apply a piece of formatting too many different elements within a page but not necessarily to the entire page.

Instead of defining styles to individual elements we are creating a style class that can be applicable any number of tags in your HTML document.

Such classes are called anonymous classes. It has the following syntax

```
.classname { Property: value; Property: value; }  
<selector1 class="classname"> ..... <selector1>  
<selector2 class="classname"> ..... <selector2>
```

Example:

```
<!DOCTYPE html>  
<html>  
<head>  
<style> .center { text-align: center; color: red; } </style>  
</head>  
<body>  
<h1 class="center">Red and center-aligned heading</h1>  
<p class="center">Red and center-aligned paragraph.</p>  
</body>  
</html>
```

## Using <div> tag

An element in an HTML document is either a block element or an inline element.

A block would be something like a paragraph, while an inline might be something like text, a figure or an individual character i.e., part of block.

Each of these can be manipulated separately, rather than applying the formatting to the element itself, a <div> ..... </div> pair of tags are wrapped around the elements.

Any formatting that needs adding is placed inside the <div> tag. The <div> tag has the following syntax:

```
<div class="any element"> <p> ..... </p> <p> ..... </p> <hr> </div>
```

Example:

```
<html>  
<head>  
<title> div tag css demo </title>  
<head>  
  <style type="text / css">  
    .fred { color: white; background-color: 009900; margin: 2px; font-size: 25px; }  
  </style>
```

```

</head>
<body>
  <div class "fred">
    <h3> this is heading inside division </h3>
    <p> The div tag is known as Division tag. The Div tag is used in HTML to make divisions
  of      content in the web page like (text, images, header, footer, navigation bar etc).</p>
  </div>
</body>
</html>

```

### Using <span> tag:

The <span> tag has very similar properties to the <div> tag. If you want to group text without using elements use a <span> tag.

It has the following syntax <span [id="....."][class="....." | style="....."]>..... </span>

What is the difference between <span> and <div>?

Span as an in-line element and a div as a block level element. Span:






1. The span element basically occupies only the amount of space required by the element , it will not take the whole space/width of the web page .
2. Second thing is span doesn't breaks the line , i.e. all the spans will be shown in single line in a web page as shown in the screenshot below.
3. Div: The div element occupies all the width available to it on the web page and every time a new div element is made it breaks the line and starts in a new line.

```

<!DOCTYPE html>
<html>
<head>
<title>HTML span Tag</title>
</head>
<body>
<p>This is a paragraph <span style = "color:#FF0000;"> This is a paragraph</span>This is a
paragraph</p>
<p><span style = "color:#8866ff;"> This is another paragraph</span></p>
</body>
</html>

```

## Layers in HTML

-  The page layout that a browser creates results from layering text and images on the top of each other.
-  This lets web designers use images as the backgrounds of their pages and then place further images and text over them.
-  By extending the idea slightly we can place text items and images on top of each other in multiple layers. Netscape has extended the HTML standard by adding the layer tag.
-  The layer tag is browser specific and it use leads to confusion with the more general idea of layers.
-  The following are the properties and values of layers.

Z-index: n The browser maintains a stack of layers of content.

Position: absolute | relative

The placement of the layer may be absolute or relative. The default is absolute.

Left: n,

Top: n

The location of the division in pixels. We can specify the position of their top-left corner.

Width: n,

Height: m

The size of division in pixels defaults to the document of space needed to display the content of the division.

Example:

```

<!DOCTYPE html>
<html>

```

```
<head>
<title>HTML layer Tag</title>
</head>
<body>
  <div style="z-index:2; left:50px; top:250px; position:absolute; color:red; background-color:white;
    font-size:36pt; border:thin-groove;">
    <p> This is the higher layer </p>
  </div>
  <div style="z-index:1; left:100px; top:255px; position:absolute; color:magenta; background-
    color:green; font-size:46pt; border:thin-groove;">
    <p> some mor text</p>
  </div>
  <div style="z-index:4; left:10px; top:40px; width:150px; position:absolute; color:black; background-
    color:yellow; font-size:18pt;">
    <p> some mor text plcaed in a box that doesn't go right across the screen</p>
  </div>
  <div style=" left:500px; top:300px; width:25px; position:absolute; color:blue; background-
    color:#aeae00; font-size:16pt; font-style:italic;z-index:2; ">
    <p> And in the bottom right corner.....</p>
  </div>
</body>
</html>
```



# JavaScript

## JavaScript:-

JavaScript is a lightweight, interpreted programming language with object oriented capabilities. It is an interpreted language, usually embedded directly onto HTML pages

## Benefits Of Javascript

1. It is widely supported by web browser.
2. It gives easy access to the document object.
3. It can give interesting animations without long download times
4. It can't get a virus infection directly from javascript
5. Javascript allow many page effects
  - a) User page time in/ out
  - b) popups & tooltips
  - c) Embedded audio
  - d) scrolling banners
  - e) print pages

## Advantages of javascript:

1. **An Interpreted Language:** which requires no compilation steps the browser just as it interprets HTML tags
2. **Embedded within HTML:-** Javascript does not requires any special or separate editor for programs written  
It can be written along with HTML tags in notepad
3. **Javascript can react to events:** To execute when something happens, like a user click on HTML buttons
4. **Javascript can read and write HTML elements:** A javascript can read & change the content of HTML elements

**Javascript can be used to validate data:** Using javascript we can validate data for example enter valid email – id, enter valid name, mobile number etc.

## Inserting java script into a HTML Page:

The following are the basic rules to be followed while developing JavaScript code.

1. Java script statements end with semi-colon
2. Java script is case – sensitive
3. Java script support two forms of comments
  - Single – line comment begin with (//)
  - multi line comment begin with (/\*) and end with (\*/)
4. Block of code must be surrounded by a pair of braces ({ })
5. Functions have parameters which are passed inside parenthesis ()

## Including Java Script into HTML:

- Java Script is placed between tags starting with **<script language = “javascript”>** and end with **</script>**
- Java script can be placed in various locations in HTML
  - ↪ Java script in **HEAD** section
  - ↪ Java script in **BODY** section
  - ↪ Java script in both **HEAD** and **BODY** section
  - ↪ Java script in **External File (with .js )**
- Java script placed in the HEAD section of HTML will be executed when called.
- Java script placed in the BODY section of HTML will be execute only when the page is loaded

### Java script in the **HEAD** section:

```
<html>
<head>
<script language="javascript">
    ---- java script code here-----
</script>
</head>
</body> -----</body>
</html>
```

### Java script in the **BODY** section

```
<html>
<head> </head>
<body>
    <script language="javascript">
        ----- java script code here-----
    </script>
</body>
</html>
```

### Link Java script file in HTML

```
<html>
<head>
    <script language="javascript" src="filename.js">
    </script>
</head>
</body> -----</body>
</html>
```

### **VARIABLES in Java Script:-**

- ✓ Variables are used to hold data in memory. It is a name of the memory location.
- ✓ Before using a variable we must declare it in a JavaScript program.
- ✓ Variable are declared with the **var** key word in JavaScript.

**Ex:-**

```
<script language="javascript">
    var rollno;
    var sname;
</scrip>
```

- ✓ Multiple variable declaration      var rollno, sname, avg;
- ✓ **Global Variable** : A global variable will be available in entire program.
- ✓ **Local Variable**: A local variable will be visible only within a function where it is defined.

```
<script language="javascript">
    var rollno=10;    // global variable
    function display()
    {
        var rollno=20;    // local variable
    }
</scrip>
```

### Variable Rules:-

- ❖ Javascript variable name starts with letter(a-z), should not start with number(0-9)
- ❖ Cannot use spaces in between names.
- ❖ An underscore( \_ ) can be used between multiple words.
- ❖ Variable names are case sensitive.
- ❖ Reserved words are not used as a variable name.

**Ex:- some valid variables:**     **var rno;**     **var s\_name;**

### DATA TYPES :-

JavaScript uses 4 data types number, string, Boolean, null

1) **Number:-** It is possible to express both integers and floating point values.

**Ex:- 23, 44, -5.6, 6.7**

2) **String:-** These are collection of characters. Indicated with single or double quoted.

**Ex:- 'murthy'        "sailu"**

3) **String:-** Variables hold the values TRUE and FALSE

**Ex:-        boolean    res=true;**

4) **NULL :-** The null value represents just that – nothing. It does not mean nil or zero.

### STRING MANIPULATIONS:

1. **charAt():** This method returns the character from the specified index. (First character index is 0) (Last character index length-1)

**Ex:-**             **var str = new String("welcome");**

**str.charAt(2);                                Output: 1**

2. **concat():** This method adds two or more Strings and returns a new String.

**Ex: var    str1="welcome to", str2 ="Java";**

**var str3 = str1.concat(str2);**

3. **indexOf():** This method returns the index within the calling String object of the first occurrence of the specified value if not found returns (-1)

**Ex: var    str1 = new String ("This is string one");**

**var    index=str1.indexOf("String");**

4. **toLowerCase():** This method returns the calling String value converted to lower case

Syntax: String.toLowerCase()

**Ex: var    str = new String("WELCOME");**

**str.toLowerCase();**

5. **toUpperCase():** This method returns the calling String value converted to uppercase

Syntax: String.toUpperCase()

**Ex: var    str = new String("welcome");**

**str.toUpperCase();**

6. **substring():** This method is used to take a part of a String

**Syntax: string.substring(start , length];**

**Ex: var    str = "welcome";**

**str.substring(1,2);        Output: el**

7. **length():** This property returns the number of characters in the given string

**Ex: var    str = "welcome";**

**int    len = str.length( );**

8. **replace():-** This method replaces one string with another String. Searches for pattern1, if the search is successful pattern1 is replaced with pattern2.

**Syntax: string.replace(oldstring, newstring);**



```
Ex: var    str = "welcome to java"
      var    res = str.replace("java", "html" );
```

**Output:** welcome to html

```
<html>
  <head><title> string functions</title></head>
<body>
  <h1 align=center><u> String Functions</u></h1>
  <script language="JavaScript">
    var str = new String("Hello World");
    document.writeln("<br><br>Length :"  +str.length);
    document.writeln("<br><br>Upper Case:"  +str.toUpperCase());
    document.writeln("<br>Lower Case :"  +str.toLowerCase());
    document.writeln("<br><br>indexOf String :"  +str.indexOf("l"));
    document.writeln("<br><br>Substring:"  +str.substring(1,2));
    var rep=str.replace("World","JavaScript");
    document.writeln("<br>Replace String :"+rep);

  </script>
</body>
</html>
```

## MATHEMATICAL FUNCTIONS:

-  The Math object provides properties and methods for mathematical constants and functions
  -  Math is static and can be called by using Math as an object without creating it
- Syntax:

```
var pival = Math.PI;
var sinval = Math.sin(30);
```

### Mathematical Functions:

1. **abs():** Returns the absolute value of a given number  
**Ex:    var    r = Math.abs(-1);**  
         document.write(r);                    output: 1
2. **max():**    Returns the largest of zero or more number  
**Ex:    var    r = Math.max(10, 20, 30);**  
         document. write(r)                    output: 30
3. **min():** Returns the smallest of zero or more number  
**Ex:    var    r = math.min(10, 20, 30);**  
         document. write(r) ;                    output: 10
4. **pow():** Returns base to the exponent power, that is base exponent  
**Ex:    var    r = Math.pow(2,3);**  
         document. write(r) ;                    output: 8
5. **round():** Returns the value of a number rounded to the nearest integer  
**Ex:    var    r = Math.round(0.5);**                    output: 1
6. **sqrt():** returns the square root of a number  
**Ex:    var    r = Math.sqrt(16);**                    output:4

7. **ceil()**: Returns the smallest integer greater than or equal to a number

**Ex:**    **var   r = Math.ceil(45.20);**                   output: 46

8. **floor()**: Returns the largest integer less than or equal to a number

**Ex: var r= Math.floor(10.3);**                   output:10  
document. write(r);

9. **Math.PI**: Returns the PI value 3.14159

**Ex:    var   r = Math.PI;**  
document. write(r); output:3.14159

10. **sin()**: Returns the sine of a number

**Ex:    var   r = Math.sin(30);**

11. **cos()**: Returns the cosine of a number

**Ex:    var   r = Math.cos(30);**

12. **tan()**: Returns the tangent of a number

**Ex:    var   r = Math.tan(45);**

13. **asin()**: Returns the arcsine (in radians) of a number

**Ex:    var   r = Math. asin(30);**

14. **acos()**: Returns the arccosine(in radians) of a number

**Ex:    var   r= Math. acos(30);**

15. **atan()**:Returns the arctangent(in radians) of a number

**Ex:    var   r = Math. atan(45);**

16. **random()**: Returns the random number between 0 to 1.

**Ex:    var   r = Math. random();**

## OPERATORS IN JavaScript

**Ans:** Javascript language supports following operators

1. **Arithmetic Operators**
2. **Relational (comparison) operators**
3. **Logical operators**
4. **Assignment operators**
5. **Conditional operators**
6. **The + operator used on strings**

1. **Arithmetic Operator:-** These operators take numerical values as their operands and return a single value. (a, b variables holds a= 10, b=20 then)

Operator	Description	Example
+	Addition	a + b will give 30
-	Subtraction	a - b will give -10
*	Multiplication	a * b will give 200
/	Division	b/a will give 2
%	Modulation	b%a will give 0
++	Increment operator, increases one value	a++ will give 11
--	Decrement operator decrease one value	a - -will give 9

2. **Relational Operators:** These operators compare operands and returns a logical value (true/false)  
(Assume variable a=10 and b = 20 then)

Operator	Description	Example
=	Equal operator	(a==b) is not true
!=	Not equal operator	(a!=b) is true
>	Greater than	(a>b) is not true
<	Less than	(a<b) is true
>=	Greater than or equal operator	(a>=b) is not true
<=	Less than or equal operator	(a<=b) is true

3. **Logical Operators:** These operators are typically used with Boolean (logical) values returns (true/False) values  
(Assume variable a = 10, b=20 then)

Operator	Description	Example
&&	Logical AND operator (all conditions must true)	(a&&b) is true
	Logical OR operator. If any one condition is true	(a  b) is true
!	Logical NOT operator (it converts reverse results)	!(a&&b) is false

4. **Assignment Operators:** The assignment operators(=) to assign a value to a variable or constant or expression assigned for given variable

Operator	Description	Example
=	Assign operator	c=a+b
+=	Shortcut Addition assignment	c+=a is equal to (c=c+a)
-=	Shortcut subtraction assignment	c-=a is equal to (c=c-a)
*=	Shortcut multiplication assignment	C*=a is equal to (c=c*a)
/=	Shortcut division assignment	c/=a is equal to (c=c/a)

5. **Conditional Operator(?:) :** Which is used for comparing two expressions, also contains a conditional operator that assigns a value to a variable based on some condition

**Syntax:** variable = (condition)? Value1:value2;

**Ex:** result = (marks>=35)?"passed":"failed";

6. **The “+” operator used on strings:** The ‘+’ operator can also be used to add string variables or text values together

**Ex:** txt1 = ‘aditya’;

txt2 = “Degree college”;

txt3 = txt1+txt2;

## FUNCTIONS IN JavaScript

A function is a piece of code that performs a specific task. You may call a function from anywhere within a page.

Functions can be defined both in the <head> and in the <body> section of a document.

### Defining Functions:

Functions are defined using the function key word. The function name can be any combination of digits, letters and underscore but not a white space

#### Syntax:

```
function funname(parameters....)
{
    Body of function.....
}
```

**Example:-**

```
< script language = "javascript">
    fuction display()
    {
        alert("hello");
    }
</script>
```

**Parameter passing to functions:** When a function receives a value as a parameter. The parameter are taken from the function definition.

The "return" key word is used to return values from functions.

- The function can return only single value

**Example:-**

```
<script language = "javascript">
    document. write("The product of 2 no's:" + product(5,3));
    function product(a,b)
    {
        return a * b;
    }
</script>
```

**Program-1:-**

```
<html>
    <head>
        <script language = "javascript">
            document. write("The product of 2 no's:" + product(5,3));
            function product(a, b)
            {
                return a * b;
            }
        </script>
    </head>
    <body> <h1> Function Returning Demo </h1>
    </body>
</html>
```

**Program-2 :**

```
<html>
    <head><title> function demo</title>
    < script language = "javascript">
        fuction display()
        {
            alert("hello");
        }
    </script>
    </head>
    <body>
        <form name = "f1">
            <input type = "botton" value = "click me" onclick = "display()">
        </form>
    </body>
</html>
```

The above program whenever user hit the input button immediately display() function will be execute and the message box with "hello" message will be displayed on the browser window.

## Objects in JavaScript:

1. Built-in objects provide information about
  - ➔ Currently loaded web page
  - ➔ Its contents
  - ➔ Current session of Navigator
  - ➔ Methods for working with properties.
2. Built-in objects in java script are part of the Navigator object hierarchy.
3. The following are some of the built-in objects in Java script.
  - a. Navigator – The navigator object provides information about the current browser.
  - b. Window – The window object provides methods and properties to work with navigator window which includes objects for each frame.
  - c. Location – The location object provides methods and properties to work with the URL.
  - d. History – The history object provides methods and properties about the history list and previous, next visited web page information.
  - e. Document – The document object is the most frequently used java script object, which contains methods and properties to work with form elements, links and applets.

## DOCUMENT OBJECT:

The following are the some of the properties associated with document object:

1. bgColor -- used to set or get the background color
2. fgColor -- used to set or get the foreground color
3. anchor -- object reference of an anchor contained in the document
4. alinkColor -- used to set or get the value of alink attribute
5. form -- object reference of a form contained in the document
6. forms -- used to get an array of all the form objects contained in the document
7. image -- object reference of an image contained in the document
8. images -- used to get an array of all the image objects contained in the document
9. link -- object reference of a link contained in the document
10. links -- get an array of all the link object references contained in the document

The following are the some of the Methods associated with document object:

1. close( ) -- used to close a document stream opened using document.open( )
2. getElementById( ) -- used to access any element on the page using ID attribute.  
**document.getElementById("ID").value;**
3. open( ) -- used to open a document stream to display or write something.  
**document.open( )**
4. write( ) -- used to write a given string on to the document.  
**document.write("String");**
5. writeln( ) -- used to write a given string on the document and inserts a new line character at the end.  
**document.writeln("STRING");**

## WINDOW OBJECT:

Window object is the top-level object for each document, location and history object. The following are the some of the properties associated with window object.

1. closed -- used to identify whether a window is closed or not. Returns true if closed.
2. status -- used to get the current status of the browser window.
3. document -- returns the document object of the current browser page.
4. length -- returns number of frames in a window.
5. name -- used to find the name of the browser window.
6. opener -- used to identify the object from which the window was created.

The following are the methods associated with window object.

1. alert( ) -- used to pop up an alert message.  
**alert("Message");**



2. `confirm()` -- used to display a confirm dialog box.  
**`confirm("Message");`**
3. `prompt()` -- used to get user input through keyboard.  
**`prompt("Message");`**
4. `open()` -- used to open a new window using javascript.  
**`window.open(url,name);`**
5. `close()` -- used to close a window using javascript.  
**`window.close();`**

### DATE OBJECT:

Date object is used to date and time in milliseconds from 1<sup>st</sup> January 1970 UTC. The Date object is created in different ways as shown below.

1. `Date()` -- Creates an empty date object.
2. `Date(milli-seconds)` -- creates a new date object based up on the number of milliseconds since 00:00:00 hours on 1-1-1970.
3. `Date(year,month,day[,hour,minute,second])` – Create a new Date object based up on the specified values.

The following are the methods associated with Date Object:

1. `date()` -- returns the current date.
2. `getDate()` -- returns the date between the days 1 and 31.
3. `getDay()` -- returns the day of the week between 0 and 6.
4. `getMonth()` -- returns the month between 0 and 11.
5. `getFullYear()` -- returns the last two digits of the year.
6. `getFullYear()` -- returns the year as four digit number.
7. `getHours()` -- returns the current hours from 0 to 23
8. `getMinutes()` -- returns the current minutes from 0 to 59
9. `getSeconds()` -- returns the current seconds from 0 to 59
10. `getMilliseconds()` – returns the milliseconds from 0 to 999
11. `setDate()` -- specifies the day of the month from 1 to 31
12. `setMonth()` -- specifies the month from 0 to 11
13. `setFullYear()` -- specifies the four digit year in a date object
14. `setHours()` -- specifies the hours from 0 to 23 in a date object
15. `setMinutes()` -- specifies the minutes from 0 to 59 in a date object.
16. `setSeconds()` -- specifies the seconds from 0 to 59 in a date object.
17. `setMilliseconds()` – specifies the milliseconds from 0 to 999 in a date object.
18. `toString()` -- converts a date object into a string.

### HISTORY OBJECT:

1. History object contains the complete list of url/links visited during a session.
2. History object methods are used to move forward or backward.

`next` -- represents the next page URL in the history object list.  
`previous` -- represent the previous page URL in the history object list.  
`current` -- represent the current page URL in the history object.  
`length` -- returns the number of objects in the history list.

## UNIT-IV

DHTML with Javascript : Data Validation, Opening a new window, messages and confirmations, the status bar, different frames, rollover buttons, moving images.

### DHTML

- ❖ DHTML stands for Dynamic HTML.
- ❖ Using DHTML we can develop interactive web pages i.e., the content of the web page can be changed according to the user input.
- ❖ A separate HTML file will be generated for each and every user when they try to use a static web page.
- ❖ To overcome this problem DHTML came into the existence.
- ❖ DHTML included Javascript along with HTML and CSS to make the page dynamic.

### DATA VALIDATION

- ❖ Data validation is the process of ensuring that users submit only the set of required characters.
- ❖ It is not the process of ensuring that the data is accurate.
- ❖ Most of the data that the scripts get from users will be textual and not possible to verify.
- ❖ We just ensure that the user has provided formatted input or not.

Consider a login form which accepts username and password from a user. The following constraints are applied on form.

1. Username should contain only characters either upper case or lowercase.
2. Username contains only underscore ( \_ ) symbol.
3. Password must contain a uppercase character
4. Password must contain a digit
5. Password should be minimum 8 characters and maximum 14 characters.

Now the validation procedure performs the following operations.

- a. First it will validate whether the user specified values for the required fields or not.
- b. Username field contains a character or not.
- c. Username contains any other special symbol other than underscore ( \_ )
- d. Password length should be between 8 and 14 or not.
- e. Password contains a digit or not.
- f. Password contains a any uppercase character or not.

If the user doesn't follow any of the constraints then the validation procedure will be failed.

- a. Validation can be defined by many validation methods, and deploy in many different ways.
- b. Server Side Validation : It is performed by a web browser after input has been specified.
- c. Client Side validation : It is performed by a web browser before input sent to the web browser.

Example:

```
<html>
<head>
<title>Data Validation Demo</title>
<script language="javascript">
function validate()
{
```

```

var uname=document.getElementById("txt1").value;
var pwd=document.getElementById("txt2").value;
if(uname=="" || pwd=="")
{
    alert("UserName/Password should not be empty");
}
else if(pwd.length<8 || pwd.length>14)
{
    alert("Password Should be Minimum 8 and Maximum 14 characters");
}
else
{
    alert("Log In Successful");
}
}
</script>
</head>
<body>
<h1 align=center>Log In Form</h1>
<form name="loginform">
<table align=center border=0>
<tr><th>UserName</th><td><input type="text" name="uname" id="txt1"></td></tr>
<tr><th>Password</th><td><input type="password" name="pwd" id="txt2"></td></tr>
<tr><td colspan=2 align=center><input type="button" value="LogIn" onclick="validate()"></td></tr>
</table>
</body>
</html>

```

## OPENING A NEW WINDOW

1. To display data dynamically we can open a new window.
2. A new window has a set of options to define what is displayed on the window or not.
3. A new window can be opened using open( ) method of window object.
4. Open( ) method takes three parameters.
  - a. URL
  - b. Name
  - c. List of attributes.
5. URL specifies what should be displayed on the new window.
6. Name specifies the name of the window.
7. List of attributes specify various options and all these options are enclosed in a single or double quotes and all options are separated by a comma.
8. The following are the various options and all these options takes the value 0 or 1, or yes or no.
  - i. toolbar - whether to display toolbar in the new window or not.
  - ii. location - whether to display address bar in the new window or not.
  - iii. directories - whether to display buttons for favorites in the new window or not.
  - iv. status - whether to display status bar in the new window or not.
  - v. menubar - whether to display menu bar in the new window or not.
  - vi. scrollbar - whether to display scrollbars in the new window or not.
  - vii. resizable - whether to change the size of the new window or not.
  - viii. height - specifies the height of the new window.
  - ix. width - specifies the width of the new windows.

Syntax:

```
window.open("URL", name, "attribute list");
```

Example:

```
<html>
<head>
<title>Open Demo</title>
<script language="javascript">
function load(url)
{
    window.open(url,"newwin",'status=0,toolbar=0,esizable=0,width=300,height=3000');
}
</script>
<body>
<a href=" " onclick="load('aditya.jpg')">Next Page</a>
</body>
</html>
```

### MESSAGES AND CONFIRMATIONS:

1. Java script provides three built-in window types that can be used to display alert messages and user confirmations.
2. The three kinds of popup windows in javascript are
  - a. alert box
  - b. confirm box
  - c. prompt box

#### ALERT BOX:

1. alert box is used to display alert messages to the user for doing some action.
2. When a alert box is invoked, a small window is displayed along with a message and a button.
3. The message is the user specified text message and the button is OK button.

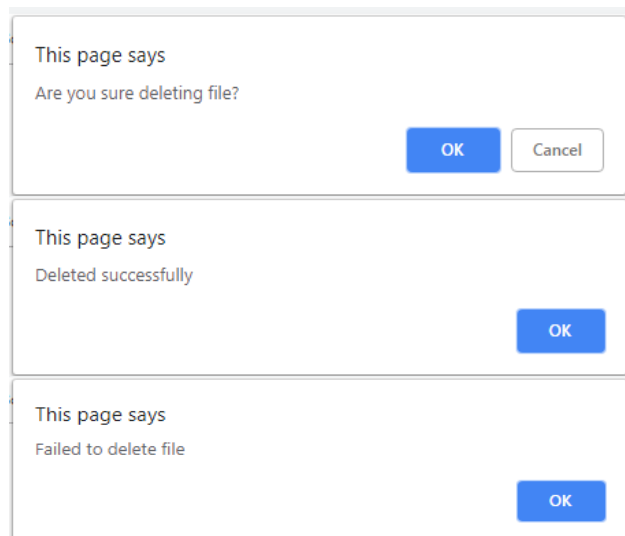
```
alert("Message");
alert("Welcome");
```

#### CONFIRM BOX:

1. Confirm box is used to take user confirmation whether to perform a task or not.
2. Confirm box display a user specified message along with two buttons i.e., OK and CANCEL.
3. Confirm box returns a boolean value i.e., true / false.
4. When user select OK button it return true and return false when CANCEL button is selected.

```
confirm("message");
```

```
<html>
<head>
<title>Messages and Confirmations</title>
</head>
<body>
<script language="javascript">
var x=confirm("Are you sure deleting
file?");
if(x)
alert("Deleted successfully");
else
alert("Failed to delete file");
</script>
</body>
</html>
```



#### PROMPT BOX:

1. Prompt box is used to accept input from the user through keyboard.
2. Prompt box contains a text field and two buttons OK and CANCEL.
3. Any type of data read using prompt box will be treated as string.

4. If user selects OK button then input specified data will be returned as a string.
5. If user selects CANCEL then null value will be returned.  
prompt("message","default value");

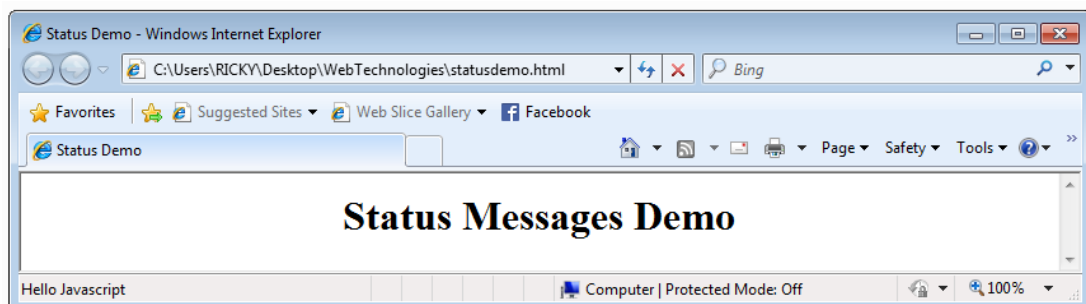
```
<html>
<head>
<title>Prompt Demo</title>
</head>
<body>
<h1 align=center>Prompt Demo</h1>
<script language="javascript">
var x=parseInt(prompt("Enter first number"));
var y=parseInt(prompt("Enter second number"));
var z=x+y;
alert("sum is :"+z);
</script>
</body>
</html>
```

The image shows three sequential browser prompts. The first prompt says 'This page says' and 'Enter first number' with a text box containing '10' and 'OK'/'Cancel' buttons. The second prompt says 'This page says' and 'Enter second number' with a text box containing '20' and 'OK'/'Cancel' buttons. The third prompt says 'This page says' and 'sum is :30' with an 'OK' button.

### THE STATUS BAR:

1. Javascript provides you the ability to modify the status bar.
2. The status property sets the text in the status bar at the bottom of the browser or return the previously set status.
3. The status bar basically displays helpful information about the operation of the browser.

```
<html>
<head>
<title>Status Demo</title>
<script language="javascript">
self.status="Hello Javascript";
</script>
</head>
<body>
<h1 align=center>Status Messages Demo</h1>
</body>
</html>
```



## DIFFERENT FRAMES:

```
<html>
<head>
<title>Writing to Different Frames</title>
<frameset rows="50,50">
<frame name="top" src="add.html">
<frame name="bottom" src="result.html">
</frameset>
</html>
```

**wdf.html**

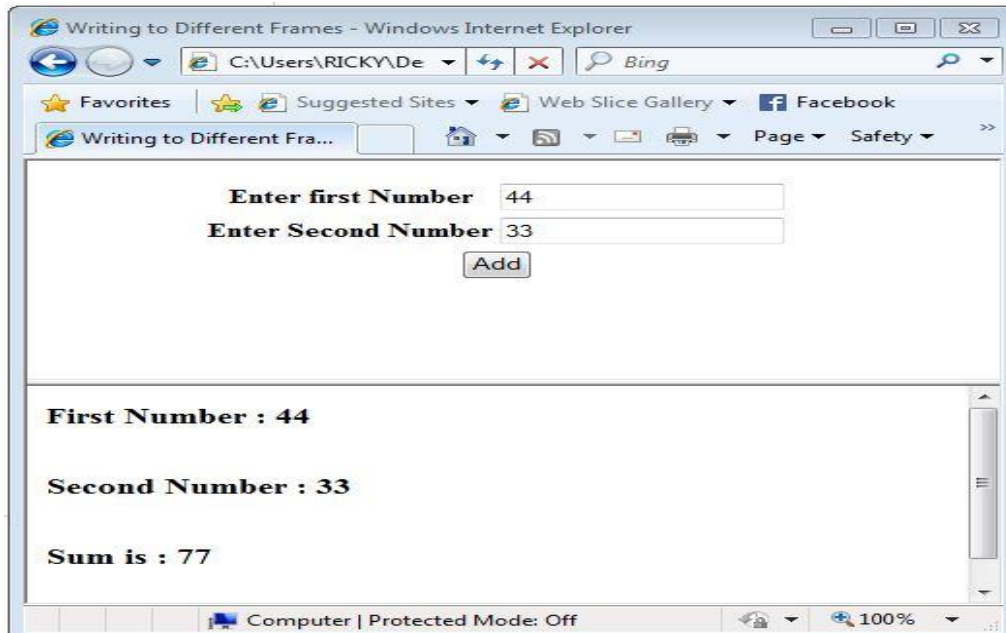
```
<html>
<head>
<title> Result Page</title>
</head>
<body>
</body>
</html>
```

**result.html**

```
<html>
<head>
<script language="javascript">
function sum()
{
    var x=parseInt(document.forms[0].n1.value);
    var y=parseInt(document.forms[0].n2.value);
    var z=x+y;
    alert("sum :"+z);
    var frm=parent.frames['bottom'].document;
    frm.open();
    frm.write("<body>");
    frm.write("<h3>First Number : "+x+"</h3><br>");
    frm.write("<h3>Second Number : "+y+"</h3><br>");
    frm.write("<h3>Sum is : "+z+"</h3><br>");
    frm.write("</body>");
    frm.close();
}
</script>
<body>
<form name="addition">
<table border=0 align=center>
<tr><th>Enter first Number</th><td><input type="text" name="n1"></td></tr>
<tr><th>Enter Second Number</th><td><input type="text" name="n2"></td></tr>
<tr><th colspan=2><input type="button" value="Add" onclick="sum()"></th></tr>
</table>
</form>
</body>
</html>
```

**add.html**

1. In the above example a web page contains two frames.
2. The upper frame contains a form which is used to accept data from user.
3. The lower frame contains a html page which display the user specified input and addition of those two values.



## ROLLOVER BUTTONS

1. Javascript is commonly used to create rollover effects, using the onmouseover event handler to trigger the rollover graphic and the onmouseout event handler to return the graphic to its original state.
2. If just the navigation button changes, it is referred as a single rollover.
3. If the button changes and another graphic anywhere on the page also changes, it is referred as a multiple rollover.
4. Rollover buttons is typically done for navigation buttons.
5. An image rollover occurs when an image is moused over and that image is replaced by another image.
6. Rollovers indicate to the user what button is being selected.
7. To create a single image rollover all that is necessary to modify the "src" property of the image.

```
<html>
  <head>
    <title>Rollover Buttons Demo</title>
  </head>
  <body>
    
  </body>
</html>
```

## MOVING IMAGES:

1. Javascript can be used to move a number of elements around the page according to a pattern determined by a logical equation or function.
2. The setTimeout( ) function is used to call a user specified function after completion of specified time.
3. setTimeout( ) function accepts two parameters i.e., function name and time duration.

Syntax:

```
setTimeout("function-name( )", "time-duration");
```

Example:

```
setTimeout("display( )", "20");
```

```
<html>
<head>
<title>Moving Images Demo</title>
<script language="javascript">
var i=1;
function starttimer()
{
    document.getElementById('myimg').style.position='relative';
    document.getElementById('myimg').style.left+=i;
    i++;
    timer=setTimeout("starttimer()",10);
}
</script>
</head>
<body onload="starttimer()">

</body>
</html>
```



## UNIT-V

1. XML is a software and hardware independent tool for storing and transporting data.
2. XML (eXtensible Markup Language) is a markup language.
3. XML is designed to store and transport data.
4. XML was released in mid 1990's.
5. XML was created to provide an easy to use and store self describing data.
6. XML is recommended by W3C and there is only one version of XML is available.
7. XML is not a replacement of HTML.
8. XML is designed to carry data, not to display data.
9. XML tags are not predefined. We can define our own tags.
10. XML is platform and language independent.

### **Features and Advantages of XML:**

- a. XML separates data from HTML.
- b. XML simplifies data transport – xml data is stored in plain text format. So data will be compatible to all kind of platforms.
- c. XML simplifies data transport – xml data can be read by different incompatible applications, so it is very easy to transport data.
- d. XML simplifies platform changes – changes in operating system, applications, browser will not make data lose because xml data is stored in simple text format.
- e. XML increases data availability – xml data can be available to all kinds of machines such as handheld computers, voice machines, news feeds etc.
- f. XML can be used to create new internet languages – new languages like XHTML, WSDL, WML SMIL etc are created with XML.

### **RULES FOR WELL FORMED XML:**

A well formed XML document can be validated against DTD. A well formed XML document is an XML document with correct syntax.

It is necessary to know about valid XML document before knowing XML validation.

The following rules must be followed to make XML as well formed document.

- a. It must begin with the XML declaration.
- b. It must have a unique root element.
- c. All start tags of XML documents must match end tags.
- d. XML tags are case sensitive.
- e. All elements must be closed.
- f. All elements must be properly nested.
- g. All attribute values must be quoted.
- h. XML entities must be used for special characters.

### **DTD – DOCUMENT TYPE DEFINITION**

1. A DTD defines the legal elements of an XML document.
2. DTD defines the document structure with a list of legal elements and attributes.
3. XML schema is a XML based alternative to DTD.
4. DTD and XML schema both are used to form a well formed XML document.
5. A DTD contains the following elements.
  - ➔ Element
  - ➔ Attributes
  - ➔ Entities

### **ELEMENTS:**

- ❖ XML element can be defined as building blocks of an XML.
- ❖ Elements can behave as containers to hold text, elements, attributes, media objects.
- ❖ Each xml document contains one or more elements, the scope of which are start and end tags.
- ❖ We can create an element using `<!ELEMENT element-name (content-type)>`
- ❖ Here element-name is any user defined xml tag name and content type specifies the type of data it can store or list of child elements.

Ex: <!ELEMENT student (roll\_number,name,marks) >  
<!ELEMENT roll\_number(#PCDATA)>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT marks(#PCDATA)>

#### ATTRIBUTES:

- ❖ An xml element can have attributes.
- ❖ With the help of attributes we can add the information about the element.
- ❖ XML attributes enhances the properties of the elements.
- ❖ XML attributes must always be quoted. We can use single or double quote.
- ❖ Metadata should be stored as attribute and data should be stored as element.  
<element attribute="value"> data </element>  
<book publisher="TMH"></book>

#### ENTITY REFERENCES:

- ✚ In XML some special characters have a specific meaning. If we change the meaning of those characters that will leads to errors.
- ✚ To avoid these errors we have to use entity references in those special character place in XML document.
- ✚ All the entity references are ends with semi colon (;).
  - &lt; → To place < symbol
  - &gt; → To place > symbol
  - &amp; → To place & symbol
  - &apos; → To place ' symbol
  - &quot; → To place " symbol
- ✚ We can create our own entities in DTD as shown below.  
<!ENTITY name definition>  
<!ENTITY footer "Thank you for visiting, please come again">

DTDs are classified into two types.

- a. Internal DTD
- b. External DTD

#### INTERNAL DTD:

In internal DTD, DTD and xml code are placed in a single file and we save that file with xml extension.

```
<!DOCTYPE student  
[<!ELEMENT student (RollNo,Name,Marks)>  
<!ELEMENT RollNo (#PCDATA)>  
<!ELEMENT Name (#PCDATA)>  
<!ELEMENT Marks (#PCDATA)>]>
```

```
<student>  
  <RollNo>10</RollNo>  
  <Name>aditya</Name>  
  <Marks>94.45</Marks>  
</student>
```

#### EXTERNAL DTD:

- a. External DTDs are used to separate the DTD from XML.
- b. The drawback of using internal DTDs is, these are not reusable and specific to a particular XML.
- c. Using External DTDs we can place DTD rules in a separate file and we link that DTD with xml document.
- d. External DTDs are classified into two types.
  1. Private DTDs – these are specific to a particular project
  2. Public DTDs – these are global DTDs and can be used by any other application.

### Private DTD:

Private DTDs are linked with the help of DOCTYPE declaration as shown below.

```
<!DOCTYPE rootElement SYSTEM "DTDFile-Location">
```

### Example:

<pre>&lt;!ELEMENT student (RollNo,Name,Marks)&gt; &lt;!ELEMENT RollNo (#PCDATA)&gt; &lt;!ELEMENT Name (#PCDATA)&gt; &lt;!ELEMENT Marks (#PCDATA)&gt;</pre>	student.dtd
<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE student SYSTEM "student.dtd"&gt; &lt;student&gt;   &lt;RollNo&gt;10&lt;/RollNo&gt;   &lt;Name&gt;aditya&lt;/Name&gt;   &lt;Marks&gt;94.45&lt;/Marks&gt; &lt;/student&gt;</pre>	Student.xml

### Public DTD:

Public DTDs are global DTDs which can be used any of the application. Public DTDs are of two categories i.e., registered DTDs and unregistered DTDs.

Public DTDs are linked using DOCTYPE declaration using PUBLIC as shown below.

```
<!DOCTYPE rootElement PUBLIC "Additional Information" "DTD File">
```

Here Additional information contains 4 fields.

1. DTD is registered or not. If it is registered then + sign will be placed, if it is not registered then '- ' sign will be placed.
2. Vendor name
3. Version of DTD
4. Language of DTD

Note: If we don't have any additional information you need place empty double quotes.

<pre>&lt;!ELEMENT student (RollNo,Name,Marks)&gt; &lt;!ELEMENT RollNo (#PCDATA)&gt; &lt;!ELEMENT Name (#PCDATA)&gt; &lt;!ELEMENT Marks (#PCDATA)&gt;</pre>	student.dtd
<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE student PUBLIC " " "student.dtd"&gt; &lt;student&gt;   &lt;RollNo&gt;10&lt;/RollNo&gt;   &lt;Name&gt;aditya&lt;/Name&gt;   &lt;Marks&gt;94.45&lt;/Marks&gt; &lt;/student&gt;</pre>	Student.xml

### Cardinality Operators:

In a XML document, to specify how many number of times an element can occur is represented using cardinality operators. XML uses three different cardinality operators.

- \* represent an element to be occur 0 to n number of times.
- + represent an element to be occur 1 to n number of times.
- ? represent an element to be occur 0 or 1 time.

Note: If no cardinality operator is specified then it should be used only once.

## DIFFERENCES BETWEEN HTML AND XML:

### HTML

1. HTML is used to display data and focuses on how data looks.
2. HTML is a markup language itself.
3. HTML is not case sensitive.
4. HTML is a presentation language.
5. HTML tags are predefined.
6. In HTML, it is not necessary to use a closing tag.
7. HTML is static because it is used to display data.
8. HTML does not preserve white spaces.

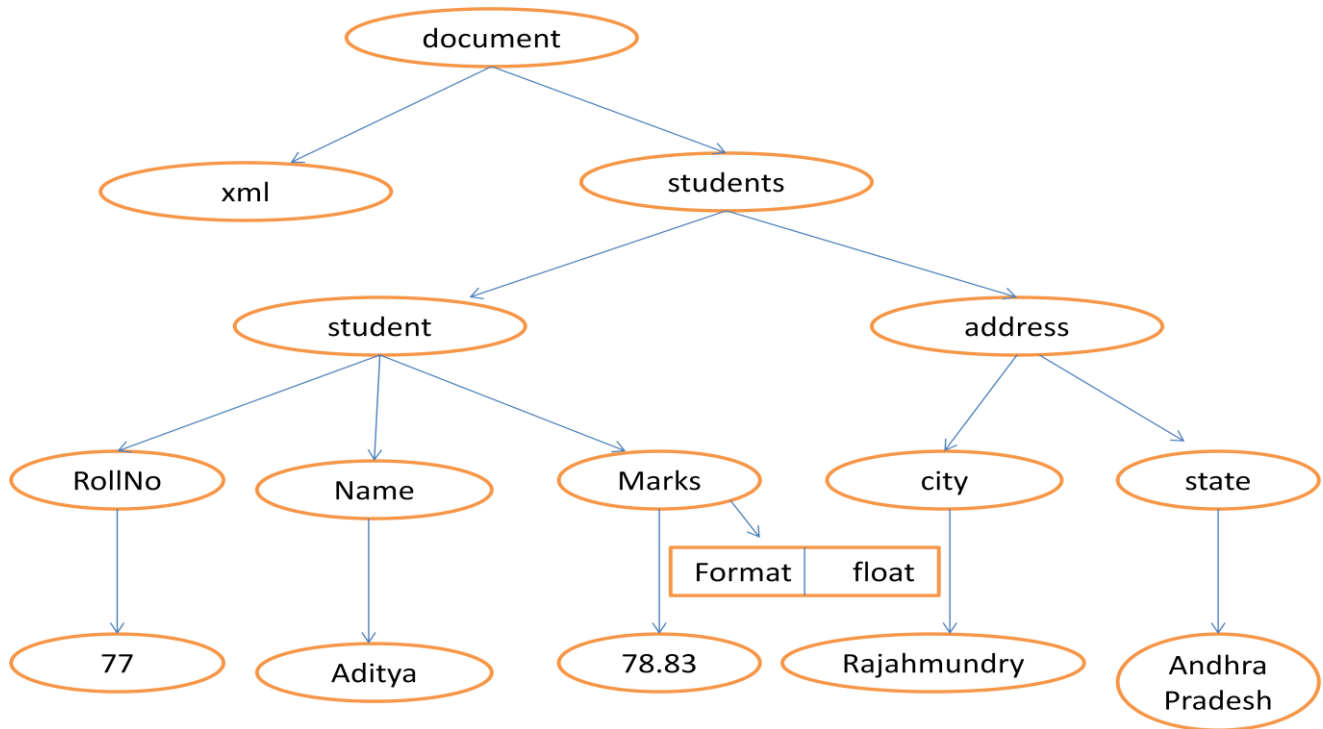
### XML

1. XML is used to transport and store data and focuses on what data is.
2. XML provides a framework to define markup languages.
3. XML is case sensitive.
4. XML is neither a presentation nor a programming language.
5. XML tags are user defined.
6. XML makes it mandatory to use a closing tag.
7. XML is dynamic because it is used to transport data.
8. XML preserves white spaces.

### XML DOM [ Document Object Model ]

- ❖ DOM stands for Document Object Model.
- ❖ DOM defines a standard way to access and manipulate documents.
- ❖ The DOM is a programming API for HTML and XML documents.
- ❖ DOM defines the logical structure of documents and the way a document is accessed and manipulated.
- ❖ The main objective of DOM is to provide a standard programming interface that can be used in a wide variety of applications.
- ❖ The DOM can be used with any programming language.
- ❖ XML DOM defines a standard way to access and manipulate XML documents.
- ❖ The XML DOM makes a tree structure view for an XML document.
- ❖ We can access all elements through the DOM tree.
- ❖ We can modify or delete their content and also create new elements.
- ❖ The elements, their content are known as nodes.

```
<?xml version="1.0">
<students>
  <student>
    <RollNo>77</RollNo>
    <Name>Aditya</Name>
    <Marks format="float">78.83</Marks>
  </student>
  <address>
    <city>Rajahmundry</city>
    <state>Andhra Pradesh</state>
  </address>
</students>
```



## WEB SERVICES:

1. Web Service is a way of communication or Technology which allows us to develop interoperable distributed applications.
2. Interoperability means platform independent and language independent.
3. Web services are common to all programming languages and technologies.
4. Web services can be published, found, and used on the web.
5. The following are the web services provided by xml and all these web services are recommended by W3C.

### ➔ WSDL:

- ➔WSDL stands for Web Services Description Language.
- ➔WSDL is an XML-Based language for describing Web Services.

### ➔ SOAP:

- ➔SOAP stands for Simple Object Access Protocol.
- ➔SOAP is an XML based protocol for accessing Web Services.

### ➔ RDF:

- ➔RDF stands for Resource Description Framework.
- ➔RDF is a framework for describing resources on the web.
- ➔RDF is written in XML.

### ➔ RSS:

- ➔RSS stands for Really Simple Syndication.
- ➔RSS allows you to syndicate your site content.
- ➔RSS defines an easy way to share and view headlines and content.
- ➔RSS files can be automatically updated.
- ➔RSS is written in XML.