



**GOVERNMENT DEGREE COLLEGE**

**NARASANNAPETA-SRIKAKULAM DIST.-532421.**

Accredited by NAAC 'B' Grade  
(Affiliated to DR.B.R.Ambedkar University)



# STUDY MATERIAL

**web  
technology**



**Detartment of  
Computer Science**

## HTML:-

- HTML is a Hyper Text Markup Language.
- It is used to develop web pages on websites.
- One of the most important features is **hypertext**. Hypertext allows web pages to contain links that are used to connect and access other web pages.
- HTML consists of elements known as **TAGS**.
- It is not based on the What You See Is What You Get (WYSIWYG) concept.
- It is case-insensitive language. (i.e. lower case & uppercase letters are same)
- It is a platform independent language.
- It allows you to create both Ordered Lists and Unordered List.
- It helps in creating forms that allow a user to input data in a webpage.
- You can add graphics and pictures in web pages using HTML.
- It provides the data in tabular form.
- It provides URL's (Uniform Resource Locator) that helps retrieve web pages from a computer connected to Internet.
- Its tags are simple and require no special programming.
- It is easy to compile.
- It is less time consuming
- These websites are easy to design and have less developing costs.

## How to Design a web page

A web page is basically made up of text and HTML tags that can be interpreted by a web browser. A web page is also known as HTML page or HTML document. A webpage is marked by an opening HTML tag <HTML> and a closing HTML tag </HTML> and is divided into three sections.

1. Comment section (Optional)
2. Head section (Optional)
3. Body section

**Comment Section:-** This section contains comments about the web page. Comment tells us what is going on in the web page. A comment line begin with a <! - and ends with -->. Comments are optional and can be included anywhere in the webpage.

**Head Section:-** The head section is defined with a starting <HEAD> tag and a closing </HEAD> tag. This section contains a title for the web page. Each document has a header and a body. <HEAD> tag in an HTML document denotes the header of a document.

**Title:-** The <TITLE> tag is placed in between the opening and closing <HEAD> tags. This tag is used to assign a title for your HTML document. This tag is not compulsory in an HTML document.

**Body Section:-** After the head section comes the body section. This section contains the entire information about the web page and its behavior. It encloses the body of text, images and other objects.

**Syntax:-**

```
<HTML>
<HEAD>
    <TITLE>    HTML INFORMATION </TITLE>
</HEAD>
<BODY>
    INFORMATION ABOUT THE WEB PAGE
</BODY>
</HTML>
```

**FORMATTING TEXT USING HTML:-** Formatting usually includes indenting text in separate lines for enhancing readability, creating paragraphs, creating bold, italic, underline on text, etc.

1. **<P> :-** This tag is used to create a new paragraph. You can use the paragraph `<P>` and the corresponding closing tag `</P>` to create paragraphs. `<P>` tag has ALIGN attribute. ALIGN property has mainly 4 values. These are **right**, **left**, **center** and **justify**.
2. **<B> .....</B>:-** This tag is used for the text makes BOLD.  
Ex:- `<B> TARAKESH </B>` O/P: **TARAKESH**
3. **<I>.....</I>:-** This tag is used for the text to italics.  
Ex:- `<I> TARAKESH </I>` O/P: *TARAKESH*
4. **<U>.....</U>:-** This tag is used for the text to UNDERLINE.  
Ex:- `<U> TARAKESH </U>` O/P: TARAKESH
5. **<H<sub>n</sub>>....</H<sub>n</sub>**
  - It allows you to create 6 types of heading in the web pages.
  - Each type of heading has a different size and can be used as subheading for the preceding the one.
  - HTML provides the `<H1>`, `<H2>`, `<H3>`, `<H4>`, `<H5>` and `<H6>`.
  - `<H1>` tag denotes the highest level heading with the biggest font size and the `<H6>` tag denotes the lowest level heading with the smallest font size.EX:- `<H1> TARAKESH </H1>`
6. **<STRIKE>..</STRIKE>:-**  
It allows you to strikethrough text in our web pages. EX:-`<STRIKE> TARAKESH </STRIKE>` O/P: ~~TARAKESH~~
7. **<SUP>...</SUP> :-** This tag is denoting for super scripts.  
Ex:- `(A+B) <SUP> 2 </SUP>` O/P:  $(A+B)^2$
8. **<SUB> ...</SUB>:-** This tag is denoting for sub scripts.  
Ex:- `H <SUB> 2 </SUB> O` O/P:  $H_2O$

**9. <FONT>:-**

- HTML allows you to change the characteristics of the fonts used in web pages.
- You use the <FONT> tag in HTML to change the font properties.
- It has 3 attributes.
  - **Face** is used to change the font type.
  - **Color** is used to change the font color.
  - **Size** is used to change the font size. Size has 1 to 7 values only.
- Ex:- <FONT face="Comic Sans MS" COLOR="red" SIZE="10">

**10. <HR>:-** HTML allows you to create Horizontal Lines in your web page. Horizontal lines are required in pages to separate content into various sections. HTML allows you to change the size and width of the line using **size** and **width** attributes.

Ex:- <HR width=75% size=20 noshade>

**11. <BR>:-** It allows you to break the lines in a web page. The closing tag for this is not required.**12. <PRE>..</PRE>:-** You can use to exactly display text as typed in the HTML code. Ex:- <PRE> TARAKESH O/P: TARAKESH

JHANSI

JHANSI

SAHITHI

SAHITHI

</PRE>

**13. <CENTER>.. </CENTER>:-** You can align the text center by using this tag.**LISTS:-**

- List is a collection of items.
- HTML allows you to creating lists to add clarity to pages.
- These lists can be simple lists without numbering or order and lists with numbering or ordered lists.

HTML allows you to create three basic types of lists

- Unordered Lists
- Ordered Lists
- Definition Lists

**Unordered Lists:-**

- Unordered list is used to represent a group of items.
- It displays the items in a bulleted format.
- It does not require any predefined order.
- <UL> tag is used to create a Unordered Lists. A closing </UL> tag is required for close the list.
- <LI> tag is used to create a new item in an unordered list.
- <UL> tag has **TYPE** attribute. It has 3 values. The values are **disc**, **circle** and **square**.

Value	Description
Disc	Displays the bullets in the form of an circles
Circle	Displays the bullets in the form of an empty circles
Square	Displays the bullets in the form of an squares

**Example:-**

```
<HTML>
<HEAD>      Creating Unordered List </HEAD>
<BODY>
<UL TYPE="disc">
  <LI> TARAKESH </LI>
  <LI> JHANSI RANI </LI>
  <LI> SAHITHI </LI>
  <LI> KEERTHANA</LI>
</UL>
</BODY>
</HTML>
```

**Output:-**

- TARAKESH
- JHANSI RANI
- SAHITHI
- KEERTHANA

**Ordered Lists:-**

- It contains group of items.
- It has a specific sequence order.
- <OL> tag is used to create an Ordered Lists in HTML.
- <LI> tag is used for each item in an ordered list.
- <OL> tag has 2 attributes. These are **type** and **start**.
- **TYPE** specifies the type of numbering system to be used.
- **Start** indicates the starting number or alphabet of the ordered list.
- The **TYPE** attribute can have 5 values denoting each type of numbering system.

Value	Description
1	Displays the type 1, 2, 3, 4, etc
A	Displays the type A, B, C, D, etc
a	Displays the type a, b, c, d, etc
I	Displays the type I, II, III, IV, etc
i	Displays the type i, ii, iii, iv, etc

**Example:-**

```
<HTML>
<HEAD>      Creating Ordered Lists </HEAD>
<BODY>
<OL TYPE="I">
  <LI> TARAKESH </LI>
  <LI> JHANSI RANI </LI>
  <LI> SAHITHI </LI>
</OL>
</BODY>
</HTML>
```

**Output:-**

- I. TARAKESH
- II. JHANSI RANI
- III. SAHITHI

**Definition Lists:-**

- HTML allows you to create a list without bullets and numbers.
- HTML provides the definition list tag **<DL>** to create unmarked lists to denote definitions.
- **<DT>** tag is used to create a Definition Term.
- **<DD>** tag allows you to insert the definition of the terms.

Example:- **<DL>**

**<DT>** E-Mail **<DD>** Emails are used for sending message.

**</DL>**

Output:- **Email**

Emails are used for sending message.

**Creating Tables using HTML:-**

- Table is a collection of rows & columns.
- HTML allows you to create tables in multiple formats with a variety of displays.
- You can create simple tables with a number of rows and columns, tables with a variety of border styles.

**<TABLE>**:-It is used to create tables. The corresponding closing tag **</TABLE>** that specifies the table ends. The ending tag for a table is compulsory.

**<TR>**This tag specifies the beginning of a new row in a table. The closing tag **</TR>** is not compulsory.

**<TD>**This tag is used to insert content (column) in a table and specifies the start of the column. The use of the **<TD>** tag creates a cell that can contain content.

**1. Creating Borders:-**

- It is used to give the lines of the table borders.
- The border attribute supports values in pixels.

Example:- **<TABLE border="5">**

**2. The width attribute:-**

- It is used to change the width of the table.
- The **width** attribute can have two types of values **pixels** or **percentage**.

Example:- **<TABLE border="5" width="50%">**

**3. The height attribute:-**

- It is used to change the height of the table.
- The **height** attribute can have two types of values **pixels** or **percentage**.

Example:- **<TABLE border="5" height="50%">**

**4. The Align attribute:-**

- It is used to align the table with in the web page.
- The align attribute can have three values: left, right and center.
- By default, browsers set the table alignment to the left of the page.

Example:- `<TABLE border="5" align="center">`

5. The bgcolor attribute:-

- It sets or changes the background color of the table.

Example:- `<TABLE bgcolor="red">`

6. The bordercolor attribute:-

- It allows you to change the border color of tables.
- It can have a value of any of the 16 colors.

Example:- `<TABLE bordercolor="green">`

7. The Cellspacing Attribute:-

- This tag to increase or decrease the space between cells in a table.
- The Cellspacing attribute can have values specified in pixels.

Example:- `<TABLE Cellspacing="10">`

8. The Cellpadding Attribute:-

- This tag is used to change the space between the cell border and the content within the cell.

Example:- `<TABLE Cellpadding="2">`

HTML allows you to create two types of headings for tables.

- Headings outside a table
- Headings within a table

Creating Headings outside a table:-

HTML provides the `<CAPTION>` tag to set a caption for a table. It insert the `<CAPTION>` immediately after the starting `<TABLE>` tag and before inserting rows and columns in a table. You can change the alignment of the CAPTION using the ALIGN attribute. The values are top, bottom, left and center.

Ex:- `<CAPTION align="top"> Student Report </CAPTION>`

Creating Heading within a Table:-

HTML allows you to create headings within a table. HTML provides the `<TH>` tag to create headings within the table. The `<TH>` tag is used within the `<TR>` tag within the `<TABLE>` tag.

Ex:- `<TR>`

`<TH> SNO </TH>`

`<TH> SNAME </TH>`

`</TR>`

COLSPAN:- HTML allows you to combine the two or more columns into a single column using COLSPAN attribute in the `<TD>` tag.

Example:-

```
<html>
<head> Colspan </head>
<body>
<table border=3 bordercolor="green" width=100% >
<tr>
    <td colspan=3> Cricket Teams </td>
</tr>
<tr>
    <td> India </td>
    <td> Srilanka </td>
</tr>
<tr>
    <td> Australia </td>
    <td> Newzealand </td>
</tr>
</table>
</body>
```

Cricket Teams	
India	Srilanka
Australia	Newzealand

**ROWSPAN:-** HTML allows you to combine the two are more rows into a single row using ROWSPAN attribute in the <TD> tag.

**Example:-**

```
<table border=3 bordercolor="green" width=100% >
<tr>
    <td rowspan=3> Cricket Teams </td>
    <td> India </td>
    <td> Srilanka </td>
</tr>
<tr>
    <td> Australia </td>
    <td> Newzealand </td>
</tr>
</table>
```

Cricket Teams	India	Srilanka
	Australia	Newzealand



EXAMPLE:-

```
<HTML>
<HEAD> TABLE
<TITLE> CREATING TABLES </TITLE>
</HEAD>
<BODY>
<TABLE BORDER=5 BORDERCOLOR=BLUE ALIGN=CENTER WIDTH=75%
CELLSPACING=2 CELLPADDING=2 >
<TR>
    <TH> SNO
    <TH> SNAME
    <TH> ADDRESS
</TR>
<TR>
    <TD> 100
    <TD> TARAKESH
    <TD> TEKKALI
</TR>
<TR>
    <TD> 101
    <TD> JHANSI
    <TD> BHIMILI
</TR>

<TR>
    <TD> 102
    <TD> SAHITHI
    <TD> TEKKALI
</TR>
</TABLE>
</HTML>
```

SNO	SNAME	ADDRESS
100	TARAKESH	TEKKALI
101	JHANSI	BHIMILI
102	SAHITHI	TEKKALI

**UNIFORM RESOURCE LOCATOR (URL):-** It is used to access a website or a page of a web site. The URL is the name of the web site with some prefixes and suffixed to access the particular site. URLs are of 3 types. They are absolute, relative or fragment.

Absolute URLs provide the complete address of the location of a resource on the Internet. It is the absolute URL that you use to open the home page of a web site. For example, <http://www.google.com>

**www** refers to world wide web, **http** denotes Hypertext Transfer Protocol is used to access the resource.

Relative URLs are used to you don't require the complete address to access the resource.

Fragment URLs are used to the linked resource is present on the same page and the scope is restricted only to the same document. A fragment URL is a kind of relative URL. The fragment URL is identified by a preceding #.

**Creating Hyperlinks:-**

- Hyperlinks are required to link pages on the web.
- Hyperlink is a hypertext in a web page that provides links to other web pages.
- HTML allows you to create various types of hyperlinks.
- HTML provides the ANCHOR or <A> tag to create hyperlinks.

Syntax:- < A href="LOCATION OF THE FILE "> Email </a>

Example: <a href="c:\tara\1.html"> tara </a>

It links the file is stored in c:\tara\1.html

**CREATING FORMS:-**

- Forms allow a user to enter various types of data.
- Each field in a form is meant for a specific data type.
- Forms make it easy to collect data from the users.
- Users can enter data in a form by using form elements, such as text fields, lists, radio buttons and check boxes.

**1. The FORM tag:-**

- You can create forms in HTML using the `<FORM>` tag and the corresponding closing tag, `</FORM>`, which signifies the end of the form.
- The method is the attribute used in the FORM tag.
- The method attribute can support two values **get** and **post**.
- The **get** value sends information to a server part of the URL.
- The **post** value sends information entered in a form to the server.

POST Method	GET Method
1. Information sent along with body.	1. Information sent along with URL.
2. Data is not visible while sending	2. Data is visible
3. It provides security	3. No Security
4. Can sent any number of characters	4. Can sent limited characters
5. Special characters can also sent	5. Special characters are not possible to send.

**1. Adding a Text Box:-**

- A Text Box is a box where a user can enter text in the specified area.
- HTML allows you to create a text box using the `<INPUT>` tag.
- The `<INPUT>` tag is the most important tag used for creating forms.
- You can create a text box using the `<INPUT>` tag and specify various properties to the text box size and text.
- It has 5 Properties. These are
  - **Type** specifies the type of the form element.
  - **Name** specifies the name of the textbox.
  - **Value** specifies the initial value of the textbox.
  - **Size** specifies the total lengths of characters are displayed.
  - **Maxlength** allows the text of maximum length.

Syntax:- `<INPUT TYPE = " " NAME = " " VALUE=" " SIZE=" " MAXLEGTH=" " >`

Example:-Your Name:

```
<INPUT TYPE="text" NAME="T1" SIZE="10" VALUE="Tara" >
```

**2. Adding Password:-**

- HTML allows you to create password fields by using the `<INPUT>` tag having the type assigned as password.
- It is same as a TEXTBOX, but it displays the text is in '\*' form.
- It has 5 Properties. These are
  - **Type** specifies the type of the form element.

- **Name** specifies the name of the textbox.
- **Value** specifies the initial value of the textbox.
- **Size** specifies the total lengths of characters are displayed.
- **Maxlength** allows the text of maximum length.

Syntax:- `<INPUT type="password" name="password" size=15>`

### 3. Text Area:-

- The text box has a limitation only a small text can be created.
- TEXTAREA tag is used to allow users to enter large amounts of data.
- The text box created using the TEXTAREA tag has a scrollbar and a user can enter large amounts of data.
- It has 3 attributes. These are **name**, **rows** and **cols**.
  - **Rows** specifies the no. of rows can enter
  - **Cols** specifies the no. of characters are displayed.

Example:- Address: `<TEXTAREA name="t1" rows=6 cols=6>`

4. Adding radio Button:- Radio buttons allows a user to select and make only one choice from the list. Radio buttons created by using the `<INPUT>` tag.

Syntax:- Gender: `<INPUT type='radio' name="radio1" checked> Male`  
`<INPUT type='radio' name="radio1"> Female`

The *type attribute* specifies the radio, **name** specifies a name for the group of radio button. Each radio button in a group should have the same name of other radio buttons.

4. Adding Checkbox:- Checkboxes allow used to make choices from a list of options. You can select any or all the options in a list of checkboxes. A check box can be in two states **checked** and **unchecked**.

Syntax:- Gender: `<INPUT type='checkbox' name="ck1" checked> Male`  
`<INPUT type='checkbox' name="ch2"> Female`

### 5. Adding a Drop Down Menu:-

- HTML allows you to create drop down menu by using the `<SELECT>` tag.
- The `<SELECT>` tag has an attribute called name that assigns the name for the drop-down list.
- `<OPTION>` tag is used for each option of the list.
- HTML provides another way of presenting lists.
- The size attributes defines the size of the list and displays all the items on the screen.

Syntax:-Qualification: `<SELECT name="dropdown">`  
`<OPTION> M.Sc`  
`<OPTION> MCA`  
`<OPTION> MBA`

</SELECT>

7. Adding Buttons:- The button can also be used to create buttons that allow a user to submit the data filled in the form.

Syntax:- <Input type="button" name="button" value="v1">

The **type** of button that may be **submit** or **reset**. A **submit** button submits the data in the form to a server and the **reset** button clears the data in the form.

<INPUT type="SUBMIT" name="button" value="SUBMIT">

<INPUT type="RESET" name="button" value="RESET">

Example:-

```
<html>
<head> <center>Forms</center>
      <title> Creating Forms </title>
</head>
<body bgcolor="lightgreen" text="blue">
<form>
<font face="Bodoni MT Black">
<pre>
Employee Number :      <input type="text" name="t1" size=20>
Employee Name      :      <input type="text" name="t2" size=20>
Employee Address :      <textarea type="textarea" rows=5 cols=20>
                        </textarea>
Qualification      :      <select name="dropdown">
                        <option> S.S.C.
                        <option> Intermediate
                        <option> Degree
                        <option> Post Graduate
                        <option> Others
                        </select>
Gender              : Male <input type="radio" name="r1" checked>
                    Female<input type="radio" name="r1">
Hobbies              : <input type="checkbox" name="c1"> Playing Cricket
                    <input type="checkbox" name="c1"> Playing Chess
                    <input type="checkbox" name="c1"> Reading Books
                    <input type="checkbox" name="c1"> Watching TV
<input type="submit" name="s1" value="SUBMIT">
<input type="reset" name="r1" value="RESET">
</form>
</body>
</html>
```

**Frames**

With frames, you can display more than one HTML document in the same browser window.

Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- ☞ The web developer must keep track of more HTML documents
- ☞ It is difficult to print the entire page

**The Frameset Tag**

- ☞ The <FRAMESET> tag is to divide the windows into frames.
- ☞ It has two properties i.e. ROWS & COLUMNS
- ☞ Each frameset defines a set of rows or columns.
- ☞ The values of the rows/columns indicate the amount of screen area each row/column will occupy.

**The Frame Tag**

- ☞ This tag is used in the <FRAMESET> tag.
  - ☞ The <frame> tag defines what HTML document to put into each frame.
1. SRC:- It has a value as a path or URL of the document to be displayed in the frameset document.  
Syntax:- < FRAME SRC="FILE LOCATION OR URL">
  2. NORESIZE:- This property is used to want the frames to be fixed and not allow the user to resize the frames in the browser window.  
Syntax:- <FRAME NORESIZE>
  3. SCROLLING:- You can insert or remove the scrollbar by using the scrolling property. It has 3 values YES, NO and AUTO.
    - ☞ You can also remove the scrollbar by using SCROLLING attribute is set to NO.
    - ☞ You can also insert the scrollbar by using SCROLLING attribute is set to YES.Syntax:- <FRAME SCROLLING=YES>
  4. FRAMEBORDER:- It is used to set the display of the border in frames. You can either set the border to be displayed or removed. To remove the border between the frames you set the value of the FRAMEBORDER=0. You can set the value of the FRAMEBORDER=1, then border to be displayed.  
Syntax:- <FRAME FRAMEBORDER=1>
  5. MARGINWIDTH:- It sets the spacing of text in a frame towards the LEFT and RIGHT sides from the border. It can support values in pixels.  
Syntax:- <FRAME MARGINWIDTH=10>

6. MARGINHEIGHT:- It sets the spacing of text in a frame towards the TOP and BOTTOM sides from the border. It can support values in pixels.

Syntax:- <FRAME MARGINHEIGHT=10>

Example:-

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET COLS="50%,50%">
<FRAME SRC="1.HTML" FRAMEBORDER=1
      MARGINHEIGHT=10      MARGINWIDTH=10 >
<FRAME SRC="2.HTML"      SCROLLING=YES >
</FRAMESET>
</HTML>
```

In the example below we have a frameset with two columns. The first column is set to 50% of the width of the browser window. The second column is set to 50% of the width of the browser window. The HTML document "1.html" is put into the first column, and the HTML document "2.html" is put into the second column:

Nesting of Frameset:-

You can divide a frame into two or more parts. You would like to leave one row or columns are undivided and the other is divided. In such situations, you can simply insert another FRAMESET. This is known as nesting of the FRAMESET tag.

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET ROWS="50%,50%">
  <FRAME SRC="1.HTML" >
    <FRAMESET COLS="50%,50%">
      <FRAME SRC="2.HTML">
        <FRAME SRC="3.HTML">
      </FRAMESET>
    </FRAMESET>
  </FRAMESET>
</HTML>
```

1.HTML	
2.HTML	3.HTML



Meta Elements:-

- Search engines (Google) are used to find the web sites to search the particular information.
- Meta is used to specify the information about a document, when the search engines searches the content in each page.
- Meta elements are not visible to users.
- Meta elements are must be placed inside the head section of your html document.
- Meta element has mainly 2 attributes.
  1. name
  2. content
  - Name:- It identifies the type of the meta element.
  - Content:- It provides the information, search engines use to catalog pages.
  - Name="keywords" The content attribute of a **Meta element** provides search engines with a list of words that describe a page.
  - Name="description" The content attribute of a Meta element provides a three-to-four line description of a site, written in sentence form.

Example:-

```
<HTML>
<HEAD>
    <META name="keywords" Content="webpage, design, help, form">
    <META name="description" content="This website will help you">
</HEAD>
<BODY>
</BODY>
</HTML>
```

**CSS** stands for **C**ascading **S**tyle **S**heets.

- It was developed by W3C (World Wide Web Consortium).
- It was launched in 1996.
- Styles define **how to display** HTML elements.
- Styles are normally stored in **Style Sheets**.
- Style Sheet is a collection of style rules that applies the element.
- One Style sheet can be used by multiple HTML documents to provide the same features.
- CSS was the style sheet included with HTML to make it DHTML.

### **CSS Style Syntax**

The CSS syntax is made up of three parts:

- ♦ A selector or HTML Tag
- ♦ A property
- ♦ A value:

Syntax:-     **SELECTOR {    PROPERTY:     VALUE    }**

- ♦ The **SELECTOR** is normally the HTML element/tag.
- ♦ The **PROPERTY** is the attribute you wish to change.
- ♦ Each property can take a **VALUE**. The property and value are separated by a colon, and surrounded by curly braces:

Example:-    **H1    {       COLOR : GREEN    }**

- ♦ CSS rules can also appear in two places inside the HTML document:
- ♦ Inside a <STYLE > Tag, which sits inside the < HEAD > tag of a document?
- ♦ As a value of a STYLE attribute on any element that can carry the style attribute.

HTML allows you to use CSS in three Types. These are

1. Linked Style Sheets or External Style Sheets
2. Embedded Style Sheets or Internal Style Sheets
3. Inline Style Sheets.

### **Linked Style Sheets:-**

- ♦ It is also called as External Style Sheets.
- ♦ We can create a new text document, inserting some Style rules. These style rules are saved with the **.css** extension file.
- ♦ These **.css** file contains the opening and closing <STYLE> tag that contain properties that defines the styles.

- ♦ The main document contains <LINK> tag that has reference to the style sheet.
- ♦ The <LINK> should be placed within the <HEAD> tag of the document.

**Syntax:-** <LINK href = "file name" rel = "stylesheet" type = "text/css">

#### The rel Attribute

- ♦ The **REL** attribute specifies the relationship between the document containing the link and the document being linked to.
- ♦ The key value for working with style sheets is stylesheet.

#### The href Attribute

- ♦ The **HREF** attribute has the path of the stylesheet location (i.e filename.css file location)

#### The type Attribute

- ♦ The **TYPE** attribute specifies the type of the stylesheet language. The type is "text/css".

TARA.HTML

Example:-

##### 1. Css

```
<STYLE>
  H1
  {
      COLOR: GREEN;
  }
</STYLE>
```

```
<HTML>
  <HEAD>
      <LINK href="1.css" Rel = "stylesheet"
      Type = "text/css">
  </HEAD>
  <BODY>
      <H1> TARAKESH </H1>
  </BODY>
</HTML>
```

#### Embedded Style Sheets:-

- ☺ These are also called as Internal Style Sheets.
- ☺ These Style sheets are placed in the main document.
- ☺ No reference or links are required in this type.
- ☺ All the style sheet properties are mentioned in the main document itself.
- ☺ The style sheet syntax is used in the opening and closing <STYLE> tags, which are used in the <HEAD> opening and closing.

Example:-

```
<HTML>
  <HEAD>
      <STYLE>
          H1 { COLOR: RED; }
      </STYLE>
  </HEAD>
  <BODY>
      <H1> TARAKESH </H1>
  </BODY>
</HTML>
```

**Inline Style Sheets:-**

- ☺ These styles are specified inline in the main document by using the STYLE attribute.
- ☺ The STYLE attribute can be specified in any of the HTML tags.
- ☺ You can have multiple declarations in the STYLE attribute and a semicolon separates each declaration.

**Syntax:-**

<P STYLE : { PROPERTY : VALUE } >

**Example:-** < P Style : { color : green; } >

```
<HTML>
  <HEAD>
    INLINE STYLE SHEETS
  </HEAD>
<BODY>
  <H1 STYLE="COLOR : RED"> TARAKESH </H1>
</BODY>
</HTML>
```

**CSS TEXT PROPERTIES:-****1. Color:-**

- ☺ The color property allows you to specify or change the color of the text.

Example:- H1 { color : red }

The above rule would make the content of heading elements color RED.

**2. Text-indent:-**

- ☺ It allows you to indent the first line of text within an element.
- ☺ This property supports values as a length or percentage.

Example:- P { text-indent : 5% }

The paragraph starts with a small indentation (5% of a webpage) towards the left.

**3. Text-Align:-**

- ☺ You can align text in web pages by using this property.
- ☺ It has 4 values. These are **left**, **right**, **center** and **justify**.

Value	Purpose
left	Aligns the text with the left border of the containing element
right	Aligns the text with the right border of the containing element
center	Centers the content in the middle of the containing element
justify	It aligns the text both left & right sides of the element

Example:- P { text-align : justify }

#### **4. Text-Transform:-**

- ☺ You can Specifies that the content of the element should all be uppercase, lowercase, or capitalized.
- ☺ It has 4 values. These are

Value	Purpose
None	No change should take place.
Capitalize	The first letter of every word should be capitalized.
UPPERCASE	The entire text of the element should be uppercase.
Lowercase	The entire text of the element should be lowercase.

Example:- H1 { TEXT-TRANSFORM : UPPERCASE }  
H2 { TEXT-TRANSFORM : LOWERCASE }

#### **5. Text-decoration:-**

- ☺ You can specifies whether the text should be underlined, overlined, strikethrough, or blinking text.
- ☺ It has some values. These are

Value	Purpose
underline	Adds a line under the content.
Overline	Adds a line over the top of the content.
line - through	Adds a line through the middle of the content.
blink	Creates blinking text (its not work in IE)

Example:- P { TEXT-DECORATION : OVERLINE }  
H1 { TEXT-DECORATION : LINE-THROUGH }

#### **6. Letter-Spacing:-**

- You can increase or decrease the space between the letters using this property.

Example:- P { LETTER-SPACING : 10 PX }

#### **7. Word-Spacing:-**

- You can increase or decrease the space between the words using this property.

Example:- P { WORD-SPACING : 10 PX }

#### **8. Text-Shadow:-**

- It provides special effects with shadows around the text.
- This property does not work in Internet Explorer.

Example:- H1 { TEXT-SHADOW : BLUE 10px 10px 3px;}

#### **9. Line-height:-**

- It provides the space between the lines.

Example:- P { line-height : 150% }

It gives 1.5 line space between the lines.

Example:-

```
<HTML>
  <HEAD>
    <STYLE>
      P { COLOR           : RED;
          TEXT-ALIGN      : JUSTIFY;
          TEXT-INDENT     : 5%;
          TEXT-DECORATION : UNDERLINE;
          TEXT-TRANSFORM  : CAPITALIZE;
          LETTER-SPACING  : 20 PX;
          WORD-SPACING    : 10 PX;
          LINE-HEIGHT     : 150%;
        }
    </STYLE>
  </HEAD>
  <BODY>
    <P> CSS stands for Cascading Style Sheets. It was developed by
    W3C (World Wide Web Consortium). It was launched in 1996.
    Styles define how to display HTML elements. Styles are normally
    stored in Style Sheets. Style Sheet is a collection of style rules
    that applies the element. One Style sheet can be used by multiple
    HTML documents to provide the same features.
    </P>
  </BODY>
</HTML>
```

**FONT PROPERTIES:-****1. Font-Family:-**

- ☺ You can specify more than one font type in this property.
- ☺ The browser starts searching for the font type from the left side in the list of this property.
- ☺ It selects the first available font type is used for displaying the text.

Example:- P { font-family : "Comic Sans MS", Algerian, Arial }

The double quotes (") are used with those values that have two or more words.

**2. Font-Size:-**

- ☺ You to specify a size for the font. The size values will be given different ways.

Length (along with pixels, there are several other units of length)

Px em ex pt in cm pc mm

Example:- P { font-size : 10 Px; }

Absolute size (each of these values corresponds to a fixed size)

xx-small x-small small medium large x-large xx-large

Example:- P { font-size : large; }

Relative size (this value is relative to the surrounding text)

smaller larger

Percentage

2% 10% 25% 50% 100%

**3. Font-Style:-**

- ☺ You can give the different styles of a font.
- ☺ It has mainly 3 values. These are **normal** , **italic** , or **oblique**.

Example:- p { font-style : normal; }  
 p { font-style : italic; }  
 p { font-style : oblique; }

**4. Font-Weight:-**

- ☺ CSS allows you to alter the weight (width) of the font used in the web page.
- ☺ Font-Weight has values are **Normal** **bold** **bolder** **lighter** **100** **200** **300** **400** **500** **600** **700** **800** **900**

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

**5. Font-Variant:-**

- ☺ You can give variant styles of a font.
- ☺ font - variant property has 2 values : **normal** and **small - caps**.
- ☺ A small caps font looks like a smaller version of the uppercase letter set.

Example:- p { font-variant : small-caps }









**CSS BORDERS Property:-**

CSS allows you to add borders with a variety of colors, width and styles of your web pages. Borders can be placed over a table, over some text or over a picture.

**1. Border-Style:-**

- This property allows you to specify the line style of the border.
- It has 10 border styles by CSS. These are

Value	Description
none	No border. (Equivalent of border - width:0 ;)
solid	Border is a single solid line. 
Dotted	Border is a series of dots. 
dashed	Border is a series of short lines. 
double	Border is two solid lines; 
groove	Border looks as though it is carved into the page.
ridge	Border looks the opposite of groove.
inset	Border makes the box look like it is embedded in the page.
outset	Border makes the box look like it is coming out of the canvas.

Example:- `P { border-style: solid; }`

- You can specify different styles for all the sides of the border.
- You can change the style of the bottom, left, right and top borders.

Example:- `P { Border-left-style : solid;  
Border-right-style : double;  
Border-top-style : dashed;  
Border-bottom-style: dotted;  
}`

**2. Border-Width:-**

- It allows you to alter the width of the borders.
- The width is specified in pixels.
- You can use **thin**, **medium** and **thick** keywords to set the width of the border. Example:- `P { border-width : 10 px; }`
- You can change the different border width to the sides of the border.

Example:- `p { border-left-width : 10px;  
Border-right-width : 20 px;  
Border-top-width : 10 px;  
Border-bottom-width : 20 px;  
}`

- A border-width property can accept up to 4 values or parameters.
- Different border styles are produced based on the numbers and the values of parameters passed.

Parameter Passed	Action
One	All the four sides get the same value
Two	<b>Top &amp; Bottom</b> get the first value and <b>right &amp; left</b> get second value
Three	<b>Top</b> side gets the first value, the <b>right &amp; Left</b> sides get the <b>second</b> , and <b>bottom</b> gets the <b>third</b> value
Four	<b>Top</b> side gets the first value, the <b>right</b> side get the <b>second</b> , and <b>bottom</b> gets the <b>third</b> value and <b>left</b> gets the fourth value.

### 3. Border-color:-

- You to change the color of the border.
- Color values can be given in various formats using RGB(), color names etc.  
Example:- `P { border-color : green }`
- You can change the different colors for different sides of the border.

```
P {
    Border-left-color      :    green;
    Border-right-width    :    blue;
    Border-top-width      :    pink;
    Border-bottom-width   :    red;
}
```

### 4. Border:-

- It is used to set all the sides of the border in one style.
- We can change the border color, border style and border width.  
Syntax:- `Tag { BORDER : WIDTH STYLE COLOR }`  
Example:- `P { border : 10 px double red }`

### 5. Border-sides:-

- This property can be altered to change the border appearance.
- Each side has a fixed property such as **border-top**, **border-right**, **border-bottom** and **border-left**.
- You can use these properties to change the appearance of any side of the border.

Syntax:- `H1 { BORDER-BOTTOM: 10PX DOUBLE RED }`

**USING CSS FOR LISTS**

- CSS provides various properties to change the appearance of the list in your web pages.
- HTML provides Ordered Lists and Unordered Lists, to change the style and position of your list.

**1. LIST-STYLE-TYPE:-**

- You can use this to alter the type of marker or bullet displayed in front of list.
- The bullet type is dependant on the type of the list.
- For Ordered lists, the bullets include **decimal**, **lower-roman**, **upper-roman**, **lower-alpha** and **upper-alpha**.

Value	Meaning	Example
decimal	Number	1, 2, 3, 4, 5
lower - alpha	Lowercase alphanumeric characters	a, b, c, d, e
upper - alpha	Uppercase alphanumeric characters	A, B, C, D, E
lower - roman	Lowercase Roman numerals	i, ii, iii, iv, v
upper - roman	Uppercase Roman numerals	I, II, III, IV, V

- For Unordered Lists include **circle**, **square** and **disc** can be used as bullets.

Value	Marker
none	None
disc (default)	A filled - in circle
circle	An empty circle
square	A filled - in square

Example:-

```

OL { LIST-STYLE-TYPE : LOWER-roman }
UL { LIST-STYLE-TYPE : CIRCLE }
```

**2. LIST-STYLE-POSITION:-**

- It is used for changing the position of the list markers or bullets.
- It has 2 values. 1. Inside 2. Outside
- The **INSIDE** value places the bullets inside the list box.
- The **OUTSIDE** value places the bullets outside the list box.

Syntax:- OL { LIST-STYLE-POSITION : INSIDE }

**3. LIST-STYLE-IMAGE:-**

- You can use an image as a bullet for your list by using this property.
- The syntax is similar to the Background-Image property.
- The value starts with the letters URL and is followed by the image location.

Syntax:- UL { LIST-STYLE-IMAGE : URL ("LOCATION"); }

---

**CSS TABLE PROPERTIES:-****1. Border-Width:-**

- It allows you to alter the width of the table borders.
- The table border width is specified in pixels.

Syntax:-       TABLE       {       BORDER-WIDTH :10 px;}

**2. Border-Color:-**

- You can change the table border color by using RGB() method.

Syntax:-       TABLE       {       BORDER-COLOR : RGB(255,0,0);       }

**3. Border-style:-**

- It allows you to change the styles of the table borders.
- The width is specified in pixels.

Syntax:-       TABLE       {       BORDER-STYLE       :       DOUBLE;       }

**4. Background-Color:-**

- You can change the background color of the table.

Syntax:-       TABLE       {       BACKGROUND-COLOR : RED;       }

**5. Background-image:-**

- You can change the background image of the table.

Syntax:-       TABLE       { BACKGROUND-IMAGE : URL("1.JPG");       }

**6. Width:-**

- You can change or set the width (Horizontal space) of the table.

Syntax:       TABLE       {       WIDTH: 100%}

**7. Height:-**

- You can change or set the height (Vertical Space) of the table.

Syntax:-       TABLE       {       HEIGHT : 100% }

**8. Padding:-**

- You can to set or change the amount of space between the border of a table cell and its content.

Syntax:       TABLE       {       PADDING: 10 PX;}

**9. Border-Spacing:-**

- You can change the space between the adjacent cells.

Syntax:-       TABLE       {       Border-spacing : 10 PX ; }

**10. Border-collapse:-**

- It is used to place the border around a table.
- CSS allows you to place borders over the cells by using this property.
- It has 2 values. **Separate** and **collapse**.
- The border-collapse property is set to **separate**; the borders appear around each cell. It is set to **collapse** is used to create borders around the table.

Syntax:-       TABLE       {       border-collapse       :       separate       }

### 11. Table-layout :-

- It is used to manage the width of the table.
- It has two values; **fixed** and **auto**.
- The **table-layout** property is set to **fixed**, the border of the table is sized according to the text of the header row.
- It is set to **auto** the width of the table appears as per the setting of the browser.

Syntax:- TABLE { Table-layout: fixed ; }

### SELECTOR:-

- Selector is a style rule.
- Selectors are mainly 3 types.
  1. Universal Selector
  2. Type Selector
  3. Class Selector

#### 1. Universal Selector:-

- The universal selector is an asterisk;
- It is like a wildcard and matches all element types in the document.
- If you want a rule to apply to all elements, you can use this selector.

Syntax:- \* { Style rules }

Eg:-

```
* { color : red;
    Text-transform : uppercase;
}
```

</style>

#### 2. The Type Selector:-

- The type selector matches all of the elements specified in the comma - delimited list.
- It allows you to apply the same rules to several elements.

Eg:- H1, H2, H3 {

```
Color : red;
}
```

- H1, H2 and H3 has same rule i.e. Text color is **red**.

#### 3. The Class Selector

- The class selector allows you to match a rule with an element carrying a class attribute.
- you had a < p > element with a **class** attribute whose value was **back**.
- :

---

## CSS FOR POSITIONING:-

### 1. Position:-

- You can use this property for placing the tags in different positions.
- This property can be attached with any tag.
- The values are **static**, **absolute**, **relative**

- Javascript is a Scripting Language. (It has a Interpreter rather than the compiler)
- It is an Interpreted Language.
- It was designed to add interactivity to HTML pages.
- It was invented by **Brendan Eich** at Netscape.
- It is an Object Based Scripting Language.
- These programs are browser dependent. (Run with web browsers)
- Every one can easily use java script without purchasing a license.
- It was originally developed by **Netscape Corporation**.
- These programs do not require compilation.
- It is case-sensitive language. i.e. lower-case and upper-case letters are different.
- It is available in the web browser Netscape Navigator 2. Initially, it was called **LiveScript**.
- Microsoft version of JavaScript is called as **JScript**, to Internet Explorer
- JavaScript referred to as ECMAScript (European Computer Manufacturing Association).

**Structure of Java Script:**

- Java script is added to an HTML page using the <SCRIPT> tag.
- The <script> tag should be placed inside of the <HEAD> tag of the document.

```
<HTML>
  <HEAD>
    <script type = "text/javascript">
      Script body
    </script>
  </HEAD>
  <BODY>    Body of the HTML page  </BODY>
</HTML>
```

- You can add scripts to your page inside the <script> element. The **type** attribute on the opening <script> tag indicates scripting language will be found inside the element. Its value will be **text/JavaScript**

Eg:-

```
<HTML>
  <HEAD>
    <script type="text/javascript">
      document.writeln ("Tarakesh");
    </script>
  </HEAD>
  <BODY> </BODY>
</HTML>
```

`document.write('This is a document');`

`document` is an object, `write` is a method. It is used to print the text

Method Name	Purpose
<code>write(string)</code>	Allows you to add text or elements into a document
<code>writeln (string)</code>	The same as <code>write()</code> , but adds a new line at the end of the output

We can use '+' operator to concatenate the two strings.

Eg:- `document.write("Tarakesh" + "Polaki");`      o/p:- TarakeshPolaki

### **Java Script Tokens:-**

- Smallest individual units in a program are known as Tokens.
- A Javascript program is a collection of tokens, comments and white spaces.

### **Keyword:-**

- These are predefined words or reserved words.
- This meaning was defined by the System.
- Java script has mainly 25 keywords.
- We can not change this meaning.

### **Identifiers:-**

- These are programmed designed tokens. These are used for naming classes, methods, objects, packages, variables, etc..

Javascript Identifiers follow the following rules.

- They must not begin with a digit.
- They can have alphabets (A..Z & a..z), digits (0-9), underscore (\_) and dollar sign (\$).
- They can be any length.
- Javascript is Case sensitive. Upper case and lower case letters are different.
- We can not use a key word as an identifier name.

### **Variable:-**

- It is an identifier.
- It can store some value.
- This value can be changed during execution.
- We can use `var` keyword to declare the variables.

Syntax:- `var    variable name`

Ex:- `var    a,b;`

- We can assign a value to a variable.

Syntax:- `var    variablename = value;`

Eg:- `var    a = 10;`



**Java Script Data types:-**

- Javascript datatype can be classified into mainly four types.
  1. Boolean Data type
  2. String Data type
  3. Null Data type
  4. Number Data type
- 1. **Boolean Data Type:-**
  - This data type consists of two values. These are **True** or **False**.
  - Eg:- `var c1 = true;`
- 2. **String Data Type:-**
  - String is a collection of characters.
  - String values are declared within the double quotes.
  - Eg:- `var sname = "Tarakesh";`
- 3. **Null Data Type:-**
  - The user does not want to initialize the value of a particular variable.
  - Eg:- `var empno = null;`
- 4. **Number Data Type:-**
  - It is a collection of positive/negative integers and decimal values.
  - These values are declared without double quotes.
  - Eg:- `var i=10;     var p = 2.5;`

**Operators:-**

- An Operator is a symbol that tells the computer to perform mathematical and logical manipulation.
- Operators are used in programs to manipulate data and variables.
- Javascript supports mainly 6 types of Operators. These are
  1. Arithmetic Operators
  2. Relational Operators
  3. Logical Operators
  4. Assignment Operators
  5. Increment/Decrement Operators
  6. Conditional Operators
  7. Bitwise Operators

**1. Arithmetic Operators:-**

- Javascript provides all basic arithmetic operators + (addition), - (subtraction), \* (Multiplication), / (Division) and % (Modulus Operator).
- These operators can operate on any numeric data type.

**2. Relational Operators:-**

- Javascript supports six relational operators.
- These are used to check the relations or conditions between two operands.

- It gives the results is in TRUE or FALSE.
- These are <, >, >=, <=, == and !=.
- A simple relation or condition contains only one relational operator.

Syntax:- **arithmetic expression2 Operator arithmetic expression2**

Example:- a>b, a>c.

When arithmetic expressions are used on either side of a relational operator, the expression is evaluated first and then the results compared.

### 3. Logical Operators:-

- Like in C/C++/Java, Javascript also three logical operators.
- These are && (Logical And), || (Logical OR) and ! (Logical Not).
- The Logical operators && and || are used combining two or more relations.
- These are mainly used to check two or more conditions at a time.

A & B are two conditions.

<u>A</u>	<u>B</u>	<u>A&amp;&amp;B</u>	<u>A  B</u>
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

If all the conditions are **true** then the Logical And (&&) will be **true** otherwise **false**.

If all the conditions are **false** then the Logical Or (||) will be **false** otherwise **true**.

### 4. Assignment Operators:-

- It is used to assign the some value to a variable and an expression.
- The Assignment operator is "=". Javascript has a set of shorthand assignment operators +=, -=, \*=, /= and %=.

Syntax:- **variable operator = expression**

Example:- a +=b => a=a+b

The use of shorthand assignment operators has results in a more efficient and easier to read.

### 5. Increment/Decrement Operators:-

- These are called as Unary operators.
- Javascript has two unary operators.
- ++ Uses incremented by 1 and -- uses decremented by 1.

Syntax:- **++Variable (Pre-Increment) Variable++ (Post Increment)**

Example:- ++a => a=a+1      --a => a=a-1

### 6. Conditional Operators:-

- Javascript has two conditional Operators.
- ? and : are called as ternary operators.
- This operator is used to construct conditional expression.

Syntax:- **variable = (Condition) ? Expression1 : Expression2.**

Example:- `big = (a>b) ? a : b`

If condition is true then the Expression1 is evaluated otherwise Expression2 is evaluated.

### **CONTROL STRUCTURES:-**

- These are collection of tools and techniques to process the data efficiently.
- A Javascript program is a set of statements executed sequentially.
- Javascript supports mainly 4 control structures. These are
  1. Single Conditional Control Structure
  2. Multi Conditional Control Structure
  3. Loop Conditional Control Structure
  4. Un Conditional Control Structure

### **Single Conditional/Decision/Directional Control Structure:-**

They are **if** and **if-else**.

#### **If Statement:-**

- The **If** statement is a powerful single decision making statement.
- It checks the **TRUE** condition only.

Syntax:-     **if (Condition)**

**Statement1;**

The **if** statement checks the condition first, If the condition is **true** then the **statement1** will be executed, otherwise it skips another block.

#### **If-Else Statement:-**

- The If-Else statement is also single decision control statement.
- It is basically two-way decision statement.
- It checks the condition is either **TRUE** or **FALSE**.

Syntax:-     **if (Condition)**

**Statement1;**

**Else**

**Statement2;**

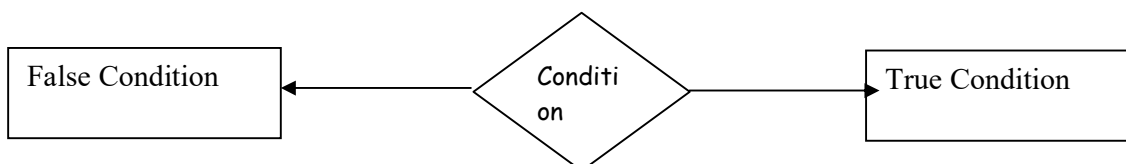
If the condition is **true** then the **statement1** will be executed, otherwise **statement2** will be executed.

Example:- `if (a>b)`

`document.write (a + "is big");`

`else`

`document.write (b + "is big");`



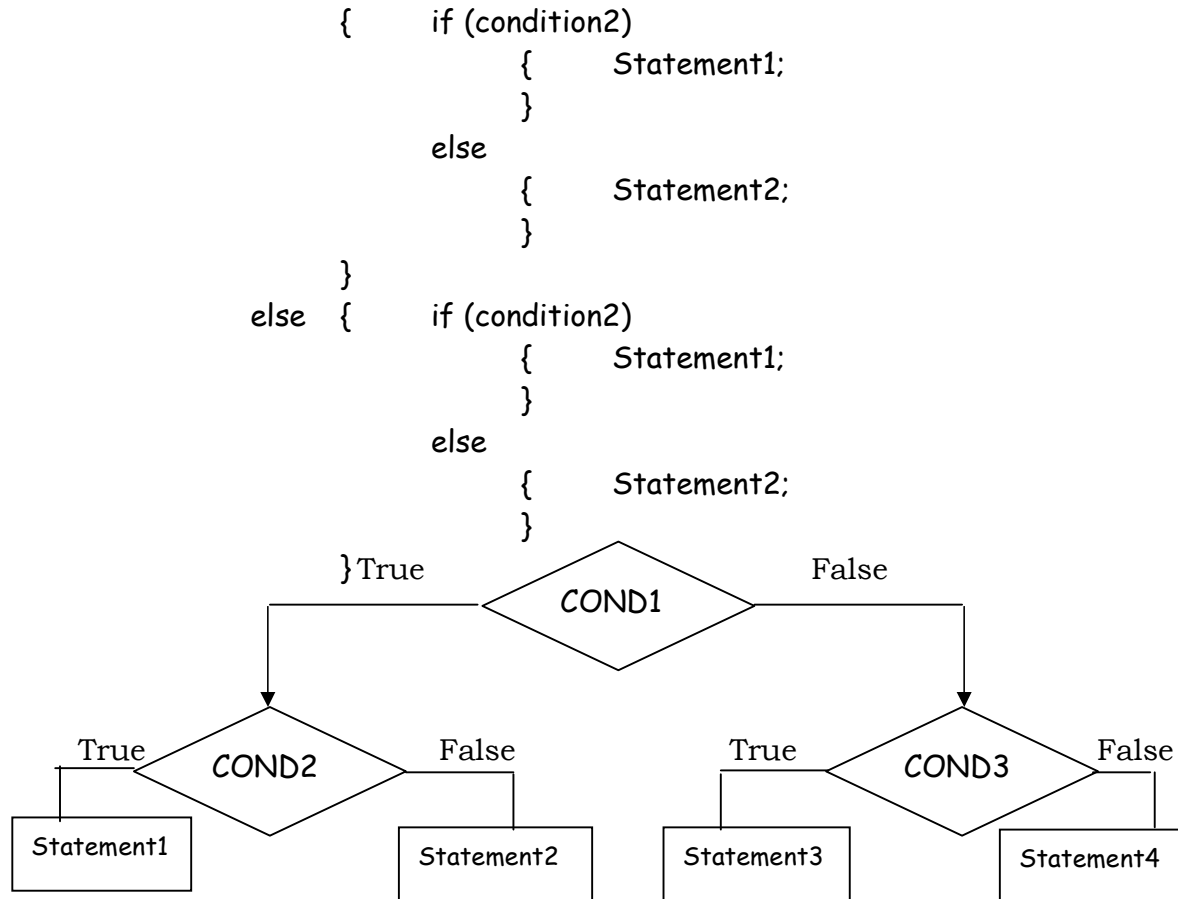
**Multi Decision Control Structure:-**

- It checks two or more conditions at a time.
- These are **Nested If** and **Switch** statements.

**Nested If:-**

- We check the two or more conditions at a time.
- **Nested If** is an if statement within another if statement.

Syntax1:- if (condition1)

**IF-Else-If Ladder:-**

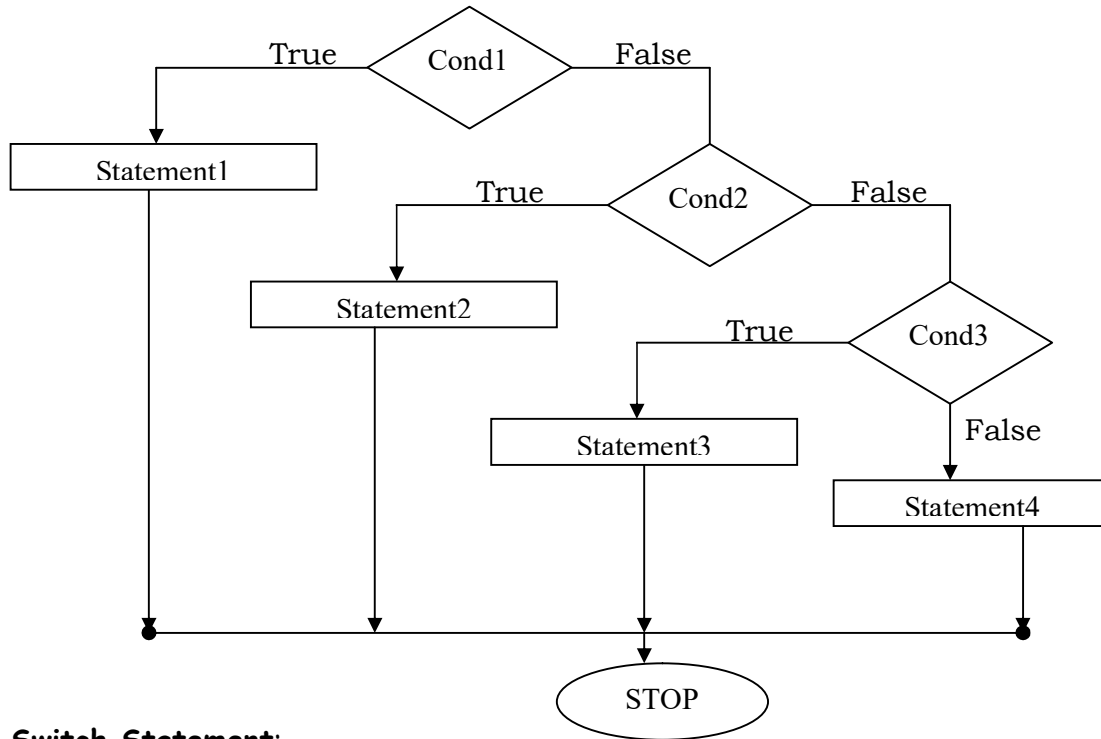
- It is also multi decision control statement.
- It checks two or more conditions at a time.
- It checks another condition in FALSE salutation.

Syntax2:-

```

    if (Condition1)
        Statement1;
    else if (Condition2)
        Statement2;
    else
        Statement3
  
```

If the **condition1** is true then **statement1** will be executed otherwise it checks the **condition2**, if the **condition2** is true then **statement2** will be executed otherwise checks the **condition3**, if the **condition3** is true then **statement3** will be executed otherwise **statement4** will be executed.



#### Switch Statement:-

- It is a multiple conditional control structure. We can check two or more conditions at a time.
- It is used instead of if-else-if statement.
- The switch statement test the value of a given variable or expression against a list of **case** values.
- When a match is found, a block of statements with that **case** is executed.
- When the match is found, the remaining conditions are not executed, then we use **break** statement.
- Break is used to terminate the loop.
- The value of a given variable is not matched, and then executes the **default** case.

The general form of the **switch** is

Syntax:-

```
switch(expression | value)
{
    case value1:    statement1; break;
    case value2:    statement2; break;
    case value3:    statement3; break;
    default:        statementn;
}
```

Example:-

```
switch(ch)
{
    case 1:    document.write("sum=" + (a+b)); break;
    case 2:    document.write ("Mul=" + a*b); break;
    default:   document.write ("Invalid Choice:");
}
```

**Loop Control Structure:-**

- The process of repeatedly executing a block of statements is known as looping.
- The statements in the block may be executed any number of times from 0 to infinite number.
- If a loop is not ended, it is called as an infinite loop.
- In looping a sequence of statements are executed until condition for the termination of the loop.
- A loop consists of two segments, one is **Body of loop** and the second one is **control statement**. Control statement tests condition and then directs the repeated execution of the statements in the body of the loop.
- A looping process contains 3 steps. These are
  1. Starting of the Loop (**Initialization**)
  2. Ending of the Loop (**Condition**)
  3. Execution body of the loop (**Body of the Loop**)

The Looping control structure may be two types.

**Entry level loop:-**

- In the Entry level loop, the conditions are tested before the start of the loop.
- **while** and **for** are entry leveled loop.

**Exit level loop:-**

- In the Exit level loop, the condition is tested at the end of the loop.
- **Do-while** loop is exit level loop.

Java language provides 3 loop control structures. They are

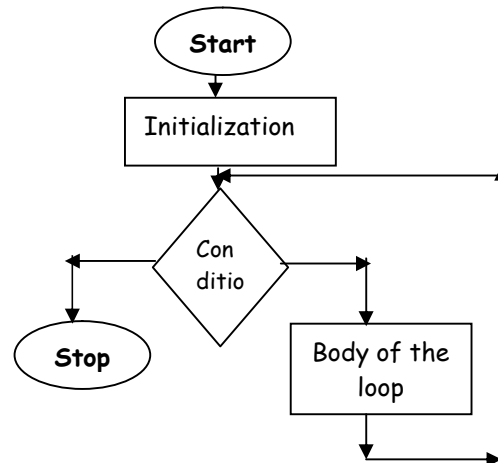
1) while loop      2) do while loop      3) for loop.

**The While Statement :-**

- The while statement is an Entry Controlled loop statement.
- It initializes the variable first, and then checks the condition.
- If the condition is true, then the body of the loop is executed one time. After execution of the body, it checks the condition again.
- This process was repeated until the condition is failed.

Syntax:-     initialization;  
               While (Condition)  
               {  
                   Body of the loop;  
               }

Example:-    i=1;  
               While(i<=10)  
               {     document.write ("i=" + i);  
                       i++;  
               }



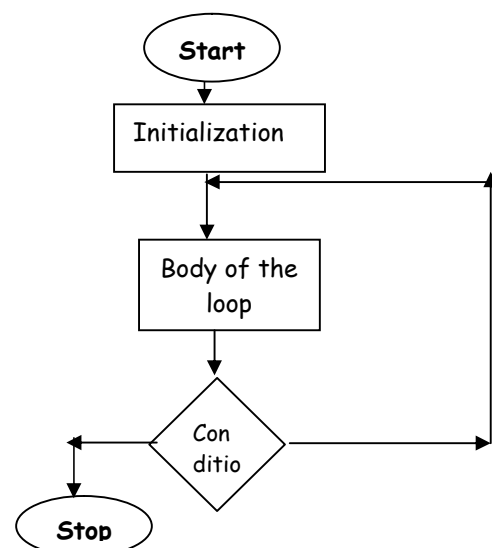
1. Start
2. Initialization of a variable
3. Check the condition. If the condition is true Go to step4; Otherwise Goto step5.
4. Execute body of the loop then go to step3.
5. stop.

#### The Do While Statement:-

- It is Exit-controlled Loop.
- In this loop was executed first and after checks the condition.
- The **Do** statement to evaluate the body of the loop first.
- At the end of the loop, checks the condition in the **while** statement.
- If the condition is TRUE then the loop was executed again once.
- This process was repeated until the condition is failed.

Syntax:-     Initialization;  
               do  
               {  
                   body of the loop;  
               } while (condition);

Example:-    i=1;  
               do  
               {  
                   document.write("i=" + i);  
                   i++;  
               } while (i<=10);



Do while loop executes the body of the loop at least one time even though the condition is failed.

**The For Statement:-**

- The **for** loop is another entry-controlled loop.
- It Initialize the variable first, then check the condition; then after execute body of the loop repeatedly until the condition is false.

The general form of for loop is

```
Syntax:-   for (initialization; test condition; increment/decrement)
            {
                body of the loop;
            }
```

The three sections enclosed within parentheses must be separated by semicolons. One of the important point is all three sections are placed in the **for** statement with in the one line.

```
Example:-   for(i =1; i<=10; i++)
            {
                document.write (" i= " + i);
            }
```

```
Example:-   for( i=1, s=0; i<=10; i++, s=s+i)
            { body of the loop
            }
```

**Nesting of for loops:-** One for statement within another for statement is called nested for.

```
Syntax:-   for(initialization1; condition1; inc/dec1)
            {
                for (initilization2; condition2; inc/dec2)
                {
                    body of the loop;
                }
            }
```





**3. pop()**:- It causes an element to be removed or popped out of an array.

Eg:-  
var a = new Array (10,20,30);  
a.pop();  
document.write ("array elements are="+ a); o/p:- 10 20 30

**4. sort()**:- It sorts or arranges the elements of a given array in ascending order of magnitude. It works effectively, if the data has strings.

Eg:-  
var a = new Array ("tara","jhansi","sahithi");  
a.sort();  
document.write ("array elements are="+ a); o/p:- jhansi,sahithi,tara

**5. reverse()**:- It prints the array elements in reversed order.

Eg:-  
var a = new Array(10,20,30,40,50);  
a.reverse();  
document.write("array elements are="+a); o/p:- 50.40,30,20,10

### **Functions in Java script:**

- It is a sub program.
- It contains some code that performs a particular task.
- Functions are basically two types. They are
  - Predefined Functions or built in functions or global functions
  - User defined functions

### **Pre-Defined Functions:-**

- These functions are defined by the JAVASCRIPT.
- They are referred as global; they can be called any part of a program.
- These are mainly Mathematical, String and Date functions.

### **a) Mathematical Functions:-**

- These mathematical functions are entities of "Math" Object.

Syntax:- **Math. method(value)**

1) min(a,b ) :- It displays minimum value of two numeric values.

Eg:- document.write(Math.min(10,4)) o/p:- 4

2) max(a,b) :- It displays the maximum value of two numeric values.

Eg:- document.write(Math.max(10,4)) o/p:- 10

3) abs(a ):- It displays the absolute value (Remove its sign).

Eg:- document.write(Math.abs(-123)) o/p:- 123

4) pow(x,y) :- It displays the  $x^y$  value.

Eg:- document.write(Math.pow(5,4)) o/p:- 625

5) sqrt( ):- It displays the square root of a given number.

Eg:- document.write(Math.sqrt(625)) o/p:- 25

6) ceil( ):- It displays the next integer value.

Eg:- document.write(Math.ceil(123.67)) o/p:-124

7) floor( ):- It returns the nearest integer less than or equal.

Eg:- document.write(Math.floor(123.67)) o/p:- 123

8) round( ):- It rounds the nearest integer value.

- Eg:- `document.write(Math.round(123.67))`      o/p:- 124
- 9) `sin( )` :- It displays the trigonometric sine value. The value is in Radians.  
We can convert radians into degree.  
Eg:- `document.write(Math.sin(22*90/7*180))`      o/p:- 1
- 10) `Cos()`:-It displays the trigonometric cosine value.
- 11) `Exp( )` :- It displays the  $e^x$  value.  
Eg:- `document.write(Math.exp(1))`      o/p:- 2.718
- 12) `Log()`:- It displays the  $\text{Log}_e$  value.  
Eg:- `document.write(Math.Log(10))`      o/p:- 2.302
- 13) `Tan()`:- It displays the Tangent value.  
Eg:- `document.write(Math.tan(45*22/7*180))`      o/p:- 1

**b) String Functions:-**

- It is collection of characters enclosed in double quotes.
  - Like Math, even strings are treated as objects.
- 1) `toLowerCase()`:- It converts all uppercase letters into lowercase letters.  
Eg:- `var s1 = "TARAKESH"`  
`document.write(s1.toLowerCase());`      o/p:- tarakesh
- 2) `toUpperCase()`:- It converts all lowercase letters into uppercase letters.  
Eg:- `var s1 = "tarakesh"`  
`document.write(s1.toUpperCase());`      o/p:- TARAKESH
- 3) `concat(string)`:- It combines two strings.  
Eg:- `var s1 = "Jhansi"`  
`document.write(s1.concat(" Rani"));`      o/p:- Jhansi Rani
- 4) `charAt(pos)`:- It displays the character depending on position.  
Eg:- `var s1 = "TARAKESH"`  
`document.write(s1.charAt(3));`      o/p:- A
- 5) `indexOf(char)`:- It displays the position of a first occurrence of a character in a string.  
Eg:- `var s1 = "TARAKESH"`  
`document.write(s1.indexOf('A'));`      o/p:- 1
- 6) `substr(start,length)`:- It displays some portion of a string. It extracts the characters from start position to number of characters.  
Eg:- `var s1 = "TARAKESH"`  
`document.write(s1.substr(0,4));`      o/p:- TARA
- 7) `substring(index,end)`:- It displays some portion of a string. **Index** argument specifies the initiating character and the **end** specifies the last character to be displayed.  
Eg:- `var s1 = "TARAKESH"`  
`document.write(s1.substring(2,6));`      o/p:- RAKE

8) split(char):- It splits the string into substrings with respect to the character.

Eg:- var s1 = "TARAKESH"  
document.write(s1.split('A')); o/p:- T,R,KESH

9) charCodeAt(pos):- It displays the Unicode equivalent of the character.

Eg:- var s1 = "TARAKESH"  
document.write(s1.charCodeAt(3)); o/p:- 65

### **Date Methods:-**

- The date object helps you work with dates and times.
- For example date is 22-11-2011

Syntax:- Date d1 = new Date()

1) getDate():- It returns the date of a Date object (from 1 to 31)

Eg:- Date d1 = new Date();  
document.write(d1.getDate()); o/p:- 22

2) getDay():- It returns the day of a Date object (from 0 to 6; 0=Sunday, 1=Monday, and so on)

Eg:- Date d1 = new Date();  
document.write(d1.getDay()); o/p:- 2 (Tuesday)

3) getMonth():- It returns the month of a Date object (from 0 to 11; 0=January, 1=February, and so on)

Eg:- Date d1 = new Date();  
document.write(d1.getMonth()); o/p:- 1

4) getFullYear():- It returns the year of a Date object (four digits)

Eg:- Date d1 = new Date();  
document.write(d1.getFullYear()); o/p:- 0

5) getHours():- It returns the hours of a Date object (from 0 to 23)

Eg:- Date d1 = new Date();  
document.write(d1.getHours()); o/p:- 15

6) getMinutes():- It returns the minutes of a Date object (from 0 to 59)

Eg:- Date d1 = new Date();  
document.write(d1.getMinutes()); o/p:- 20

7) getSeconds():- It returns the seconds of a Date object (from 0 to 59)

Eg:- document.write(d1.getSeconds()); o/p:- 50

8) toLocaleString():- It converts the Date object to a string, set to the current time zone of the user.

Eg:- document.write(d1.toLocaleString());  
o/p:- Monday, January 03, 2011 3:21:14 PM

9) toString():- It converts the Date object to a string.

10) setDate():-It sets the date of the month in the Date object (from 1-31).

11) setFullYear():-It sets the year in the Date object (four digits)

12) setHours():-It sets the hours in the Date object (from 0 to 23)

13)setMinutes():- It sets the minutes in the Date object (from 0 to 59)

14)setMonth():- It sets the months in the Date object (from 0 to 11;

User-Defined Functions:-

- These functions are names and defined by the user.
- They are initially declared and coded, later they are called.
- Every user defined function has two steps. These are
  1. Declaration of the function
  2. Definition of the function

Declaration of the function:-

- In this we can create a new function.

Syntax:-      **functionname** (parameters)

Definition of the function:-

- In this we can give description of the function.
- We can use **function** keyword to define the function.

Syntax:-      **function**      **functionname** (parameters)  
                         {  
                              Body of the function  
                         }

Eg:-    `function sum(a,b) { return(a+b); }`

Recursion:- A function can call itself is called as recursion.

```
<html>
<head>
<script type="text/javascript">
var f,n;
n=parseInt(window.prompt("enter n value"));
f=fact(n);
document.write("factorial="+f);
    function fact(n)
    {
        var n,f;
        if(n==1) return(1);
        else
            f=n*fact(n-1);
        return(f);
    }
</script>
</head>
<body> </body>
</html>
```

**Exception Handling:-**

An error is wrong information.

Errors may broadly be classified into two categories

- compile time errors
- runtime errors

Compile time errors:- These errors was occurred at the compile-time.

- Missing semicolons
- Missing brackets in classes and methods
- Misspelling of identifiers
- Missing double quotes in strings
- Use of undeclared variables

Run-time errors:- These errors was occurred at the run time.

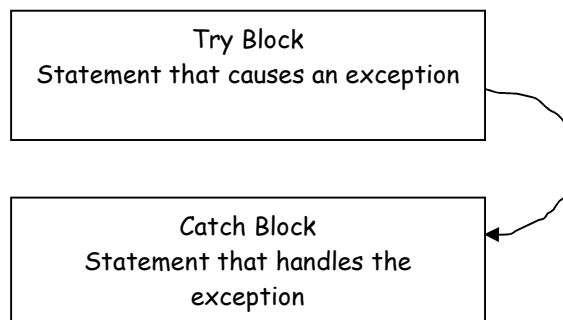
Most common run time errors are

- Dividing an integer by zero
- Accessing an element that is out of the bounds of an array
- Trying to store a value into an array of an class type

**Exception: -**

- An exception is a condition that is caused by a run-time error in the program.
- Exception handling mechanism is to provide to detect and report runtime errors, so that appropriate action can be taken.
- This mechanism suggests error-handling code. That performs the some task.
  1. Find the problem
  2. Inform that an error has occurred
  3. Receive the error information
  4. Catch &Take corrective actions.

The error handling code basically consists of two segments, one to detect errors and to throw exceptions and the other to catch exception and to take corrective actions.



It has mainly 3 keywords like **try**, **throw** and **catch**. It uses a keyword **try** to a block of code that is cause an error condition and **throws** an exception. A catch block defined by the keyword **catch** "catches" the exception "thrown by the try block and handles it.

Eg:-           try  
                {  
                    Block of statements;  
                }  
                catch(exception)  
                {  
  
                }  
            }

#### User Defined Exception

```
<html>
<head>
<script type="text/javascript">
var a;
try
{
a=parseInt(window.prompt("enter a value"));
if(a>5)
    throw "error1";
else if(a<0)
    throw "error2"
else
    document.write("a="+a);
}
catch(e)
{
if(e=="error1")
    alert("Number is greater than 5");

if(e=="error2")
    alert("Number is less than 0");
}
</script>
</head>
<body>

</body>
</html>
```

**Event Handling:-**

- Event is a user action within the browser.
- Responses made by the browser on account of user's interactions are often referred to as events.
- Once the event is generated, requirement of code to process these events. Such code is known as event handler.

Eg:- click, doubleclick, mousemove etc..

- onChange:- It invokes whenever data is changed in Textbox/TextArea box.
- onClick:- It invokes when the item was single clicked.
- onDbClick:- It invokes when the item was double clicked.
- onLoad:- It invokes when the file was opened or loaded.
- onUnload:- It invokes when the file was closed.
- onFocus:- It invokes whenever any of the page elements get focus.
- onKeyUp:- It invokes whenever user releases the key.
- onKeyDown:- It invokes whenever user presses the key.
- onMouseMove:- It invokes the user moves the mouse pointer on the document.
- onSubmit:- It invokes when the user presses the SUBMIT button.
- onMouseDown:- It invokes when the user presses the mouse key.
- onReset:- It invokes when the user presses the RESET button.
- onMouseOver:- It invokes when the mouse is over the object or image.
- onMouseOut:- It invokes when the mouse is out of the object or image.

Eg:-

```
<html>
<head>
</head>
<body onLoad='alert("Welcome to GDC, Tekkali")'
      onUnload='alert("Good Bye")'>
</body>
</html>
```



**Call-By-Value:-**

- In this, the **values** are sending from function-declaration (calling function) to function- definition (called function).
- The values are changed in the called function (definition);
- The changes are done in the called function will not affecting to **calling function** (declaration).
- Called function and calling function values are stored in different places.

Eg:-

```
<html>
<head>
<script type="text/javascript">
var a,b;
function swap(a,b)
{
var t;
t=a;
a=b;
b=t;
document.write("<br>Swapping the values");
document.write("<br>a="+a);
document.write("<br>b="+b);
}
a=10;
b=20;
document.write("<h1><u> CALL BY VALUE</u></h1>");
document.write("<br><h2>Before swapping the values");
document.write("<br>a="+a);
document.write("<br>b="+b);
swap(a,b);
document.write("<br>After swapping the values");
document.write("<br>a="+a);
document.write("<br>b="+b);
</script>
</head>
<body>
</body>
</html>
```

**CALL BY VALUE****Before swapping the values****a=10****b=20****Swapping the values****a=20****b=10****After swapping the values****a=10****b=20**

**Call-By-Reference: -**

- In this, the **objects** are sending from function-declaration (calling function) to function- definition (called function).
- The values are changed in the called function (definition);
- The changes are done in the called function will affecting to **calling function** (declaration).
- Called function and calling function values are stored in same place.

```
<html>
<head>
<script type="text/javascript">
var a = new Array();
function swap(a)
{
var t;
t=a[0];
a[0]=a[1];
a[1]=t;
}
var a=new Array (10,20);
document.write ("<h1> <u>CALL BY REFERENCE</u></h1>");
document.write ("<br><h2>Before swapping the values");
document.write ("<br>a="+a[0]);
document.write ("<br>b="+a[1]);
swap (a);
document.write ("<br>After swapping the values");
document.write ("<br>a="+a[0]);
document.write ("<br>b="+a[1]);
</script>
</head>
<body>
</body>
</html>
```

<p style="text-align: center;"><b><u>CALL BY REFERENCE</u></b></p> <p><b><u>Before swapping the values</u></b></p> <p><b><u>a=10</u></b></p> <p><b><u>b=20</u></b></p> <p><b><u>After swapping the values</u></b></p> <p><b><u>a=20</u></b></p> <p><b><u>b=10</u></b></p>
---

```
<!--BUBBLE SORT -- >
```

```
<html>
<head>
<script>
var i,n,t;
a = new Array();
n=parseInt(window.prompt("enter n value"));

for(i=0;i<n;i++)
{
a[i]=parseInt(window.prompt("enter n value"));
}

document.write("<br>"+"Array Elements Are: "+ a);

for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if (a[i]>a[j])
{
t=a[i];
a[i]=a[j];
a[j]=t;
}
}
}

document.write("<br>"+" Sorted List is: "+a);
</script>
<body onLoad='alert("Bubble Sort")' onUnLoad='alert("Thank U")'>
</body>
</html>
```

```
// Linear Search
<html>
<head>
<script type="text/javascript">
var i,n,search,t=0;
var a = new Array();
n=parseInt(window.prompt("enter n value"));
for(i=0;i<n;i++)
{
a[i]=parseInt(window.prompt("enter array element:"));
}
search=parseInt(window.prompt("enter search value:"));

for(i=0;i<n;i++)
{
if (search==a[i])
{
window.alert("element is found at "+ i + " Position");
t=1;
break;
}
}
if (t==0)
window.alert("element is not found ");
</script>
</head>
<body> </body>
</html>
```

**// Binary Search**

```
<html>
<head>
<script type="text/javascript">
var i,n,l,u,mid,search,t=0;
var a = new Array();
n=parseInt(window.prompt("enter n value"));
for(i=1;i<=n;i++)
    {
        a[i]=parseInt(window.prompt("enter array element:"));
    }
search=parseInt(window.prompt("enter search value:"));
document.write("<br>Printing an array");
for(i=1;i<=n;i++)
    document.write(a[i]+ " ");
l=1;
u=n;
while ((l<=u) && (t==0))
{
mid=Math.floor((l+u)/2);
    if (search==a[mid])
    {
        window.alert("element is found at  "+ mid + " Position");
        t=1;
    }
    else if (search > a[mid] )
        l=mid+1;
    else
        u=mid-1;
}
if (t==0)
    window.alert("element is not found ");
</script>
</head>
<body> </body>
</html>
```

**Pre-defined Objects in Javascript:-**

**Window:** - It provides methods for manipulating browser windows.

- a) **open():**- It is used to open a new window. It has some arguments.
- **url:-** It contains the URL address, i.e. web site address of the page is displayed in our web page.
  - **Name:-** Name of the window
  - **Height:-** Height of the window
  - **Width:-** Width of the window.

Syntax:- `window.open(www.google.com,"p1","height=100","width=100")`

- b) **close():**- It is used to closes the window.

Syntax:- `window.close()`

- c) **alert():**- It displays the alert box. A dialog box used when a programmer wants to make the information is passed to the user. The alert box pops up with an OK button.

Example:- `window.alert("Tarakesh");`

- d) **prompt():**- It prompts the dialog box, asking the user's input.

Syntax:- `window.prompt("Message","defaultvalue")`

Example:- `window.prompt("Enter ur name")`

- e) **confirm():**-

- A confirm box is often used to user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:- `confirm("Message")`

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
    document.write("You pressed OK!");
else
    document.write("You pressed Cancel!");
}
</script>
</head> <body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

**DHTML:-**

- It stands for Dynamic Hyper Text Markup Language.
- It is the combination of HTML, CSS, Javascript and DOM(Document Object Model)
- It creates interactive and animated web sites.
- Netscape and Internet Explorer browsers are supported.
- It was developed by W3C (World Web Consortium)
- It is used to make rollover buttons or drop down menus on a web page.

**Advantages:-**

- DHTML files are small compared to flash files.
- All browsers are supports the DHTML files.
- It does not require Java Virtual Machine.

**Rollover Images / Buttons:-**

- Javascript code does not directly manipulate the image. DHTML is manipulates the image.
- When the mouse cursor is over on the image or button was changed. When the mouse cursor is out of the image or button the old image or button was appeared. It is called as Rollover Image/ Button.
- One image is created for the inactive state when the mouse is not over it. A second image is created for the active state when the mouse cursor is placed over it.
- We can use **onMouseOver** and **onMouseOut** events in DHTML.
- **onMouseOver** calls a Javascript function when the cursor passes over the image.
- **onMouseOut** calls a function when the cursor moves away from the image.

Example:-

```
<html>
<head>
<script type="text/javascript">
if (document.images)
{
b1=new Image();
b2=new Image();
b1.src="1.jpg";
b2.src="2.jpg";
}
</script>
</head>
<body>


</body>
</html>
```



**Data Validation:-**

- Validation is simply the process of guarantee that some data might be correct data for a particular application.
- If a program accepts data from a remote data logger and that input is always going to be in a particular range, then the program knows that data outside the range is invalid and should not be accepted.
- Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed submit button.
- If the data entered by a client was incorrect or missing fields, the server would have to send all the data form be resubmitted with correct information.
- Javascript provides a way to validate form's data on the client's computer before sending it to the webserver.
- Form validation performs two functions.
  - i. Basic validation:- The form must be checked to make sure all the mandatory fields are filled in. Each field in the form and check for data.
  - ii. Data format validation:- The data is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

```
<html>
<head>
<script type="text/javascript">
function validate(form1)
{
var rv=true;
var t2=form1.t1.value;
var p3=form1.p1.value;
var p4=form1.p2.value;

if (t2.length<6)
{
rv=false;
alert("UserName Length less than 6 characters");
}

if(p3!=p4)
{
rv=false;
alert("Password was not match");
form1.p1.value="";
}
```

```
form1.p2.value="";  
}  
return rv;  
}  
</script>
```

```
<body>
<form name="form1" onSubmit="validate(this)">
<pre>
User Name:      <input type="text" name="t1" value="" size="20">
Password:       <input type="password" name="p1" value="" size="20">
Confirm Password: <input type="password" name="p2" value="" size="20">
<input type="submit" name="s1" value="Submit"> <input type="button"
name="r1" value="Close" onClick='window.close()>
</form>
</body>
</html>
```

**Floating Logos:-**

- It is the most important concept in DHTML.
- Use some script to place your image or logo on your website in such a way that it floating on the page as the user scrolls up and down. T
- he display location can be set within the script.
- Easy to use, and very effective.

**Moving Images:-**

- It is the important concept in DHTML.
- It is used to move the text/image from one place to another by using user's actions.

```
<html>
<head>
<script type="text/javascript">
var i=1;
function starttimer()
{
document.getElementById('i1').style.position="relative";
document.getElementById('i1').style.left+=i;
i=i+1;
timer=setTimeout("starttimer()",10);
}

function stoptimer()
{
clearTimeout(timer);
}

</script>
</head>
```

```
<body onKeyPress='starttimer()' onDbClick='stoptimer()' >  
  
</body>  
</html>
```

## XML

- XML stands for **Extensible Markup Language**.
- XML is a markup language much like HTML.
- XML was designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML extension name is **.xml**.
- It was developed by W3C (World Wide Web Consortium).
- XML is a case-sensitive language.
- XML language has no predefined tags. The tags used in HTML are predefined like (<P>,<B>,<I>,<IMG> etc...)
- XML allows the author to define his own tags and his own document structure.
- XML is not a replacement for HTML. In most web applications, XML is used to transport data, while HTML is used to format and display the data.
- XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.
- With XML, data can easily be exchanged between the systems.
- XML applications are platform Independent.
- XML is Used to Create New Internet Languages like CML (Chemical Markup Language), VoiceML (for sounds), MathML (for Maths) etc.

### The Difference between XML and HTML

XML is not a replacement for HTML.

XML and HTML were designed with different goals:

- XML was designed to transport and store data
- HTML was designed to display data, with focus on how data looks.
- HTML is about displaying information, while XML is about carrying information.

Syntax:-     <?xml version="1.0"?>  
              <ROOT>  
                  <CHILD>  
                      <SUBCHILD> ..... </SUBCHILD>  
                  </CHILD>  
              </ROOT>

```
<? xml version="1.0"?>
<STUDENT>
  <NAME>
    <FNAME> TARAKESH </FNAME>
    <LNAME> POLAKI </LNAME>
  </NAME>
  <ADDRESS>
    <STREET> AYYAPPA NAGAR </STREET>
    <CITY> TEKKALI </CITY>
    <DIST> SRIKAKULAM </DIST>
  </ADDRESS>
</STUDENT>
```

- XML documents must contain a **root element**. This element is "the parent" of all other elements.
- The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.
- All elements can have sub elements (child elements):
- **All XML Elements Must Have a Closing Tag.**
- XML tags are case sensitive. With XML, the tag <Letter> is different from the tag <letter>.
- Opening and closing tags must be written with the same case.
- In XML the attribute value must always be quoted.

Eg:- <note date="12/11/2007">  
          <to> Tara </to>  
          <from> Jhansi </from>  
          </note>

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters.
- First Character must be an alphabet.
- Names cannot start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces.
- Any name can be used, no words are reserved.

#### Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD):

XML DOCUMENT TYPE DEFINITION:-

- XML document is referenced as a Document Type Definition.
- In XML, we can define our own markup language.
- A DTD is a set of rules that allows us to specify our own set of elements, attributes and entities.
- DTD is basically a grammar of XML that indicates, what tags are allowed, what order they can appear and how they can nested.
- The purpose of a DTD is to define the structure of an XML document.
- It defines the structure with a list of legal elements:

```
<!DOCTYPE STUDENTLIST
[
<!ELEMENT STUDENTLIST (STUDENT)*>
<!ELEMENT STUDENT (NAME, ADDRESS)>
<!ELEMENT NAME (FIRST, LAST)>
<!ELEMENT FIRST (#PCDATA)>
<!ELEMENT LAST (#PCDATA)>
<!ELEMENT ADDRESS (STREET,CITY)>
<!ELEMENT STREET (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
]>
```

<! DOCTYPE name [DTD declaration]>

- The XML header is the Document Type Declaration, commonly referred to as the **DOCTYPE**.
- The Document Type Declaration must appear at the start of the document (preceded only by the XML header).
- It is not permitted anywhere else within the document.
- The DOCTYPE declaration has an exclamation mark (!) at the start of the element name.
- The XML Recommendation indicates that declaration elements must begin with an exclamation mark.
- Declaration elements may appear only as part of the DTD.
- They may not appear within the main XML content.

<!ELEMENT STUDENTLIST (STUDENT)\*>

- This rule tells us that the element STUDENTLIST consists of zero or more STUDENT elements.
- The \* after STUDENT indicates how many students can appear inside the STUDENTLIST element.
  - \* denotes zero or more occurrences
  - ? denotes zero or one occurrence
  - + denotes one or more occurrences

<! ELEMENT STUDENT (NAME, ADDRESS)>

- This rule states that STUDENT element contains a NAME element and an ADDRESS element.

<!ELEMENT FIRST (#PCDATA)>

- This rule states that a FIRST is an element that does not contain other elements, but contains actual text.
- An element type declaration has <!ELEMENT (content type)>
- The special symbol **#PCDATA**, which indicates (parsed) character data.
- The special symbol **EMPTY**, which indicates that the element has no content.
- The special symbol **ANY**, which indicates that any content is permitted.

```
<?xml version="1.0" ?>
```

```
<! DOCTYPE STUDENTLIST [
```

```
    <!ELEMENT STUDENTLIST (STUDENT)>
```

```
    <!ELEMENT STUDENT (NAME, ADDRESS)>
```

```
    <!ELEMENT NAME (FNAME,LNAME)>
```

```
    <!ELEMENT FNAME (#PCDATA)>
```

```
    <!ELEMENT LNAME (#PCDATA)>
```

```
    <!ELEMENT ADDRESS (STREET, CITY,DIST)>
```

```
    <!ELEMENT STREET (#PCDATA)>
```

```
    <!ELEMENT CITY (#PCDATA)>
```

```
    <!ELEMENT DIST (#PCDATA)>
```

```
];
```

```
<STUDENT>
```

```
    <NAME>
```

```
        <FNAME> TARAKESH </FNAME>
```

```
        <LNAME> POLAKI </LNAME>
```

```
    </NAME>
```

```
    <ADDRESS>
```

```
        <STREET> AYYAPPA NAGAR </STREET>
```

```
        <CITY> TEKKALI </CITY>
```

```
        <DIST> SRIKAKULAM </DIST>
```

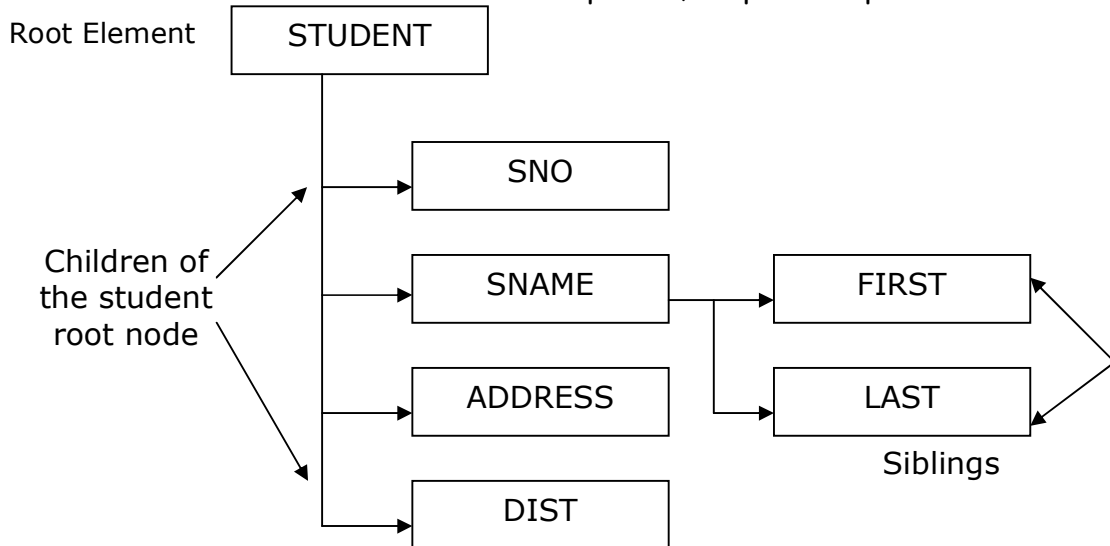
```
    </ADDRESS>
```

```
</STUDENT>
```



**Document Object Model (DOM):-**

- An XML document is a text file, retrieving data from the document using sequential file access technique.
- It creates a tree structure in memory that contains the document's data.
- The hierarchical tree structure is a Document Object Model (DOM) tree.
- Each tag name represents a node.
- A node that contains other nodes is called a **parent node**.
- A parent node can have many children, but child node can have only one parent node.
- Nodes that are peers (same level) are called as **sibling nodes**.
- A node's **descendant nodes** include its children, its children's children and so on.
- A node's **ancestor nodes** include its parent, its parent's parent and so on.



- Each node is an object that has properties, methods and events.
- **Properties** associated with a node include names, values and child nodes.
- **Methods** enable programs to create, delete and append nodes and load XML documents.

**DOM Methods:-**

1. `getParentNode ()`:- It returns the parent node.
2. `getFirstChild ()`:- It returns the FIRST child in the NodeList.
3. `getLastChild ()`:- It returns the LAST child in the NodeList.
4. `createElement ()`:- It creates a new element node.
5. `createTextNode ()`:- It creates a new text node.
6. `appendChild ()`:- It adds a child node to a node(after the last child).
7. `removeChild()`:- It removes a specified node.
8. `getLength()`:- It returns the total number of nodes in the list.
9. `getDocumentElement()`:- It returns the root node of the document.
10. `load()`:- It loads the xml file into memory.

**ASP :-**

- It stands for Active Server Pages.
- It is used to server side technologies.
- It was developed by Microsoft.
- Its extension name is .asp.
- It runs inside IIS (Internet Information Services).
- An ASP file contains text, HTML, XML and scripts.
- An ASP file contains server scripts surrounded by the delimiters <% and %>.
- **Response. Write** is used to print the output.
- Scripts in an ASP files are executed on the server.

**Uses:-**

- It dynamically edits, change or add any content of a web page.
- It access data or databases and return the results to a browser.
- It provides security, ASP code can not be viewed from the browser.
- It minimizes the network traffic.
- Its code is small and simple than CGI and Perl.

**ASP Methods:-**

- ☺ An Object is a collection of functionality and information.
- ☺ Objects are things which have defined boundaries.
- ☺ Objects provide services of other parts of an application through their methods.
- ☺ A method is a function. It has some properties.
- ☺ ASP has mainly 5 objects. These are
  1. Request
  2. Response
  3. Session
  4. Application
  5. Server

**1. Request:-**

- ☺ It is used to getting information from the user.
- ☺ This object is mainly used to retrieve the information from the FORM in a HTML page.
- ☺ It has some methods. These are **Form**, **QueryString** and **Cookies**.

**Form:-** It is used to access value from a FORM element.

**Eg:-** <INPUT TYPE="TEXTBOX" NAME="T1" VALUE=" ">

**Request.Form("t1");**

**QueryString:-** It is used to access a variables value.

**Syntax:-Request.QueryString()**

**Cookie:-** To access the value of a cookie. **Syntax:- Request.Cookie();**

**2. Response:-**

- ☺ This object is used to send information to the user(browser).
- ☺ It controls the transmission of data from the server to the web browser.
- ☺ Any type of data can be sent back, normally by sending HTML.
  1. **Write:-** It is used to send information to be displayed on the browser. **Ex:-**     Response.write("Welcome");
  2. **Redirect:-** It is used to send the user to a new HTML page.  
**Ex:-** Response.ReDirect("1.html")

**3. Session:-**

- ☺ This object is used to store information about a user's session.
- ☺ This uses cookies but they are set by the server, not by the programmer and are valid only for a limited time.
- ☺ It has 2 properties. These are **SessionID** and **TimeOut**.
  1. **SessionId:-** It created by the web application and send as a cookie to client.
  2. **TimeOut:-** To set session time out period.

It has only one method.

1. **Abandon():-** It is used to destroy the session's object.

**4. Application:-**

- ☺ This object is used to share information among the users of Web application.
- ☺ It has 2 methods. These are **Lock** and **UnLock**.  
**Lock:-** It is used to lock the variable.  
**UnLock:-** It is used to unlock the variable.

**5. Server:-**

- ☺ This object gives access to server components, its methods and properties.
- ☺ It has some methods. These are **CreateObject**, **HTMLEncode** and **URLEncode**.
  1. **CreateObject:-** It is used to create instance of server component.  
**Syntax:-** Server.CreateObject("ADODB.Connection")
  2. **HTMLEncode:-** To HTML encode a string passed.
  3. **URLEncode:-** To URL encode the string.

Accessing Database from a JSP:-

Declaring variables:-

Java is a strongly typed language which mean, that variable must be explicitly declared before use and must be declared with the correct data types. In the above example code we declare some variables for making connection.

These variables are

```
Connection con=null;
```

```
ResultSet rst=null;
```

```
Statement stmt=null;
```

The objects of type Connection, ResultSet and Statement are associated with the java.sql.

"con" is a Connection type object variable that will hold Connection type object.

"rst" is a ResultSet type object variable that will hold a result set returned by a database query.

"stmt" is a object variable of Statement. Statement Class methods allow to execute any query.

Connection to database:-

The first task of this programmer is to load database driver. This is achieved using the single line of code.

```
String driver = "org.gjt.mm.mysql.Driver"
```

```
Class.forName(driver).newInstance();
```

The next task is to make a connection. This is done using the single line of code.

```
String url="jdbc:mysql://localhost/books?user=<username> &
```

```
password=<password>"
```

```
Con=DriverManager.getConnection(url);
```

When url is passed into getConnection() method of DrivesManager class it returns connection object.

Executing Query or Accessing data from database:

This is done using following code:

```
Stmt=con.createStatement()
```

```
Rst=stmt.executeQuery("select * from books");
```

Stmt is the Statement type variable name and rst is the RecordSet type variable. A Query is always executed on a Statement object. A Statement

object is created by calling `createStatement()` method on connection object `con`.

The two most important methods of this Statement interface are `executeQuery()` and `executeUpdate()`.

The `executeQuery()` method executes an SQL statement that returns a single `ResultSet` object.

The `executeUpdate()` method executes an insert, update and delete SQL statement. The method returns the number of records affected by the SQL statement execution.

After creating a Statement a method `executeQuery()` or `executeUpdate()` is called on Statement object `stmt` and a SQL query string is passed in method `executeQuery()` or `executeUpdate`.

This will return a `ResultSet` `rst` related to the query string.

Reading values from a `ResultSet`:-

```
While(rst.next())
{
<%
<tr>
<td> <%=no%> </td>
<td> <%=rst.getString("bookname")%> </td>
<td> <%=rst.getString("author")%></td>
%>
}
```

The `ResultSet` represents a table like database result set. A `ResultSet` object maintains a cursor pointing to its current row of data. Initially, the cursor is positioned before the first row. Therefore to access the first row in the `ResultSet`, you use the `next()` method. This method moves the cursor to the next record and returns true if the next row is valid, and false if there are no more records in the `ResultSet` object.

Other important methods are `getXXX()` methods, where XXX is the data type returned by the method at the specified index, including `String`, `long` and `int`. The indexing used is 1-based. For example, to obtain the second column of type `String`, you use the following code:

```
ResultSet.getString(2)
```

You can also use the `getXXX()` methods that accept column name instead of a column index. For instance, the following code retrieved the value of the column `LastName` of the `String`.

```
ResultSet.getString("bookname");
```

The above example shows how you can use the next method as well as the `getString()` method. Here you retrieve the bookname and author columns from a table called book-details. You then iterate through the returned `ResultSet` and print all the book names and author name in the format "bookname|author" to the web page.

**Web Browsers:-**

- A Browser is a software program.
- It accesses the Internet and is able to retrieve and send information over the internet.

The most popular web browsers are

- Mozilla Firefox
- Internet Explorer:-
  - It was developed by Microsoft Corporation.
  - It is the Default web browser of Microsoft.
- Netscape Navigator
  - It is the Default web browser of LINUX.
  - It was developed by James Anderson.
- Opera

**Functions of Web browsers:-**

1. It is the user interface.
2. It sends requests and receives the information to the clients.
3. The primary purpose of a web browser is to bring information resources to the user.

**Features of Web browsers:-**

- **Back** and **Forward** buttons to go the back to the previous web page and forward web page again.
- A **Refresh** or **reload** button to reload the current web page.
- A **Home** button to return the home page.
- A **Stop** button to cancel loading the resource.
- A **History** list showing all web pages previously visited in a list.
- An **Address Bar** to input the Uniform Resource Locator (URL) of the desired web pages and display it.
- A **Search** bar to input terms into a search engine.
- A **Status Bar** to display progress in loading web pages.

**Web Servers:-**

- It is software program.
- It is responsible for sending requested information to the client.
- When the user enters a URL address into a web browser, they are requesting a specific document from a web server.
- The primary function of a web server is to deliver web pages to clients.
- The first web server is **CERNhttpd**.



The famous web servers are

- Internet Information Service
- Apache Tomcat Server
- Lighttpd
- Jigsaw
- Java Web server

**IIS:-**

- It stands for Internet Information Server.
- It was developed by the Microsoft.
- It is a high performance webserver.
- It runs on Windows NT and next windows operating systems.
- ASP debuggers allow us to find script errors.

**Apache:-**

- This is the most popular webserver.
- It was developed by the Apache Software Foundation.
- It is open source software and can be installed on almost all OS.
- Almost 60% of the web server machines run the Apache Web Server.

**Java Web Server:-**

- This webserver from Sun Micro Systems is suited for medium web sites.
- It is a free webserver but it is not open source.
- It runs on Windows, Linux and Unix.
- It supports JSP, Java Servlets, PHP, Perl, ASP etc.

**Lighttpd:-**

- It is pronounced lightly.
- It is also free web server that is distributed with the FreeBSD OS
- It is fast, secure and consumes much less CPU power.
- It runs on Windows, Linux and Sun Solaris.

**Jigsaw:-**

- Its comes from W3C (World Wide Web Consortium)
- It is open source and free and can run on various platforms.
- It has been written in java and can run CGI scripts and PHP programs.



**Features of web browsers:-**

- Virtual hosting:- It serves many websites using one IP address
- Large File Support:- It supports up to 2 GB capacity files.
- Server side Scripting:- It generate dynamic web pages.

**Protocols:-**

- These are set of rules and regulations, to pass the data between browser and server.
- It may include signaling, authentication, error detection and error correction capabilities.
- It describes the syntax, semantics and synchronization of communication and may be implemented in hardware or software.
- WWW has some protocols.
  - TCP/IP (Transmission Control Protocol / Internet Protocol)
  - HTTP (Hyper Text Transport Protocol)
  - SMTP (Simple Mail Transport Protocol)
  - POP3 (Post Office Protocol Version 3)
  - IMAP (Internet Message Access Protocol)

**TCP:-**

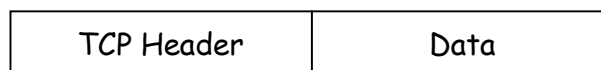
- It stands for Transmission Control Protocol.
- It was developed by a Department of Defense (DOD) research project.
- These protocols provide to transmit and receive data across complex WANs.
- It is a Transport Layer Protocol.
- It is a Connection Oriented Protocol.
- It is responsible for the data delivery of packets.

**Functions of TCP:-**

- Error Recovery
- Basic Data Transfer
- Precedence and Security
- Error Recovery
- Flexibility
- Load distribution
- Multiplexing.

**Structure of TCP:-**

- TCP consists of a header followed by a data field.
- The header length can vary between 20 and 60 Bytes, and the total size of the packet can be up to 65535 bytes.



- The Header must have 5 words of defined contents, and may have up to 10 more words of optional information.

Source Port			Destination Port		
Sequence Number					
Acknowledgement Number					
Data Offset	Reserved	Flags	Window		
Checksum			Urgent Pointer		
Options				Padding	

**Source Port:-** The port number of the source host.

**Destination Port:-** The port number of the destination host.

**Sequence Number:-** The sequence number of the first data octet in this segment.

**Acknowledgement Number:-** The sequence number of the next data octet to be sent by the source. This field is used when the ACK flag is set.

**Data Offset:-** The number of the 32 bit words in the TCP header.

**Reserved:-** These bits should be set to 0.

**Flags:-**

0	1	2	3	4	5
URG	ACK	EOL	RST	SYN	FIN

**URG** : The Urgent Pointer Field is significant

**ACK** : The Acknowledgement field is significant

**EOL** : End of Letter

**RST** : Reset the Connection

**SYN** : Synchronize sequence numbers

**FIN** : No more data from the sender

**Window:-** The maximum number of octets which the sender will accept.

**Check sum:-** It covers both the header and data portion of the TCP packet.

**Urgent Pointer:-** It points to the first urgent data byte in the packet.

**Options:-** It is used to specify various TCP options.

**Padding:-** It required to make the header length a multiple of 32 bits.

**IP:-**

- It stands for Internet Protocol.
- It was developed by Department of Defense (DOD).
- It is a network layer protocol.
- It is a connection-less protocol.
- It is responsible for the logical addressing.
- It takes communication with other computers.
- It is responsible for the sending and receiving data packets over the Internet.

**HTTP Protocol**

- It stands for Hyper Text Transport Protocol.
- It is the communication between web browsers and web servers.
- It is the text oriented protocol.
- It specifies how a client and server establish a connection, how the information is required by the browser from the server, how the connection is being closed.
- It is an Application Layer Protocol.
- It uses the TCP/IP protocol for data transfer.
- It defines 4 steps.
  - Making the connection
  - Making a request
  - Response
  - Closing the Connection

**1) Making the connection:-**

- The client establishes a TCP connection to the server.
- The connection is made on port 80.

**2) Making a request:-**

- The client sends a message to the server request.
- There are 5 types of requests.

Operations	Description
GET	Get the document identified in URL
POST	Gives the information to the server
PUT	Stores the information in specified URL
DELETE	Deletes the information
HEAD	Get the Meta data (Head data) in a URL

**3) Response:-**

- The server sends a response to the client; the response begins with a response code, followed by MIME header information.

Code	Type	Example Reasons
1xx	Information	Request Received, continuing process
2xx	Success	Action successfully received, understood and accepted
3xx	Redirection	Further action must be taken to complete the request
4xx	Client error	Request contained syntax error
5xx	Server error	Server failed to fulfill the request

**4) Closing the Connection:-**

- The connection can be closed either by the client or by the server.

**SMTP:-**

- It stands for Simple Mail Transport Protocol.
- It is the simple ASCII protocol.
- It establishes the TCP connection to port 25.
- After establishing the TCP connection, the sending machine acts as a client, waits for the receiving machine acts as the Server.
- It is a TCP/IP for sending the e-mails.
- It is an Application Layer Protocol.

**Advantages:-**

- It is well defined.
- It is simple to use.

**Disadvantage:-**

- Message should not exceed 64 KB length.

**POP3:-**

- It stands for Post Office Protocol Version 3.
- It is also ASCII protocol.
- It is also used in sending messages.
- It is an Application Layers protocol.

**IMAP:-**

- It stands for Internet Message Access Protocol.
- It is a client/server protocol running over TCP.

**MIME:-**

- It stands for Multipurpose Internet Mail Extensions used in RFC 2045.
- It is an open standard for sending multipart, multimedia through internet email.
- It can send either binary, multiple ASCII or non-ASCII character sets.

**PERL:-**

- It stands for Practical Extraction Report Language.
- It is one of the mostly widely used languages for web programming.
- It was developed by **Larry Wall** in 1987.
- Its extension name is **.pl**.
- It is a case sensitive language.
- These programs can be executed by running the Perl interpreter from the command prompt.
- It is an interpreter language.
- It is a general purpose UNIX scripting language.
- It runs on all platforms and is more portable than other environments.
- Perl has 3 built-in data types.
  1. Scalar
  2. Array
  3. Hash
- **Scalar** data type contains integers & float values. **\$** Character specifies that the variable contains a **scalar** value. Syntax: - **\$** variable name
- **Array** data type contains an ordered list of scalar values. **@** Character specifies that the variable contains an array.  
Syntax: - **@** variable name
- **Hash** data type contains strings. Syntax:- **%** variable name.
- Function **print** is used to output text.

**Features:-**

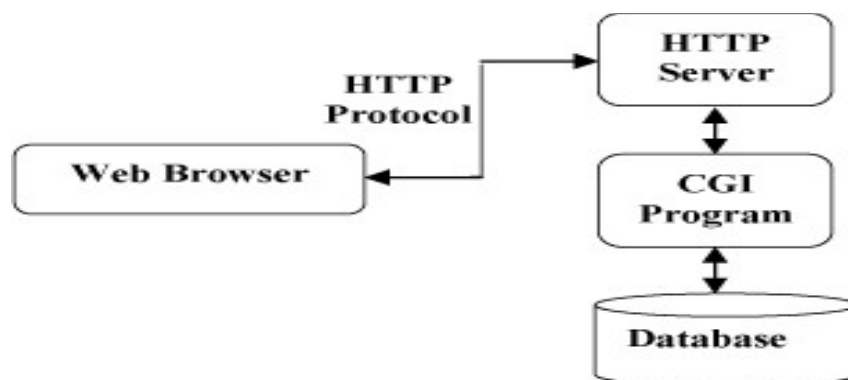
1. Open source:- It is Open source software, licensed under its Artistic License.
2. Embeddable:- The Perl Interpreter can be inserted into other systems such as web servers and database servers.
3. C/C++ Library Interface:- It interface with external C/C++ libraries.
4. Database Integration:- It supports third party databases including Oracle, Sybase, MySQL.
5. Unicode Support:- It supports Unicode version 5.
6. Text Manipulation:- It includes powerful tools for processing text, working with HTML, XML and all other mark-up languages.
7. Object Oriented:- It supports object oriented, procedural and functional programming.
8. High Quality Code:- It gives more security to code.
9. Dynamic Memory Allocation:- It gives memory allocation dynamically easily. We can increase or decrease the size of the array.

**Disadvantages:-**

1. You cannot easily create a binary image (exe) from a Perl file.
2. It is an interpreter language; it is slower to other compiling language like C.

**Common Gateway Interface (CGI):-**

- The Common Gateway Interface, or CGI, is a set of standards that define how information is exchanged between the web server and a custom script.
- The CGI specifications are currently maintained by the NCSA and NCSA defines CGI is as follows:  
The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers.
- The current version is CGI/1.1 and CGI/1.2 is under progress.
- Before you proceed with CGI Programming, make sure that your Web Server supports CGI and it is configured to handle CGI Programs.
- All the CGI Programs be executed by the HTTP server are kept in a pre-configured directory.
- This directory is called CGI Directory and by convention it is named as /cgi-bin. By convention PERL.
- CGI files will have extension as .cgi.
- Your browser contacts the HTTP web server and demand for the URL ie. filename.
- Web Server will parse the URL and will look for the filename in if it finds that file then sends back to the browser otherwise sends an error message indicating that you have requested a wrong file.
- Web browser takes response from web server and displays either the received file or error message.

**CGI Environment Variables**

The entire CGI program will have access to the following environment variables. These variables play an important role while writing any CGI program.

Variable Name	Description
---------------	-------------

<b>CONTENT_TYPE</b>	The data type of the content. Used when the client is sending attached content to the server. For example file upload etc.
<b>CONTENT_LENGTH</b>	The length of the query information. It's available only for POST requests
<b>HTTP_COOKIE</b>	Return the set cookies in the form of key & value pair.
<b>HTTP_USER_AGENT</b>	The User-Agent request-header field contains information about the user agent originating the request. Its name of the web browser.
<b>PATH_INFO</b>	The path for the CGI script.
<b>QUERY_STRING</b>	The URL-encoded information that is sent with GET method request.
<b>REMOTE_ADDR</b>	The IP address of the remote host making the request. This can be useful for logging or for authentication purpose.
<b>REMOTE_HOST</b>	The fully qualified name of the host making the request. If this information is not available then REMOTE_ADDR can be used to get IR address.
<b>REQUEST_METHOD</b>	The method used to make the request. The most common methods are GET and POST.
<b>SCRIPT_FILENAME</b>	The full path to the CGI script.
<b>SCRIPT_NAME</b>	The name of the CGI script.
<b>SERVER_NAME</b>	The server's hostname or IP Address
<b>SERVER_SOFTWARE</b>	The name and version of the software the server is running.

**ASP GET & POST Methods.****GET Method**

- The *GET* method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location box.
- Never use the *GET* method if you have password or other sensitive information to pass to the server.
- The *GET* method has size limitation: only 1024 characters can be in a request string.
- You can pass information by simply concatenating key and value pairs along with any URL or you can use HTML `<FORM>` tags to pass information using *GET* method.

Example:-

```
<FORM action="/cgi-bin/hello_get.cgi" method="GET">  
    First Name: <input type="text" name="first_name"> <br>  
    Last Name: <input type="text" name="last_name">  
    <input type="submit" value="Submit">
```

```
</FORM>
```

**POST method:-**

- A generally more reliable method of passing information to a CGI program is the *POST* method.
- This packages the information in exactly the same way as *GET* methods, but instead of sending it as a text string after a `?` in the URL it sends it as a separate message.
- This message comes into the CGI script in the form of the standard input.

Example:-

```
<form action="/cgi-bin/checkbox.cgi" method="POST" target="_blank">  
    <input type="checkbox" name="maths" value="on"> Maths  
    <input type="checkbox" name="physics" value="on"> Physics  
    <input type="submit" value="Select Subject">  
</form>
```