HTML INTRODUCTION

HTML stands for Hypertext Markup Language used to create web pages. It is a markup language which identifies the elements of the web pages. It is to be used as the coding for the internet. It allows internet users to create Web pages that contain text, graphics, pictures, images, sound and videos and interrelated hypertext links to other Web pages. HTML contains the set of labels and tags which has predefined meanings used to render the information in a browsers (A Web browser is used to view Web pages or Web sites on the internet or intranet. Exp: IE, Mozilla, Netscape Navigator etc). Please refer WWW, Web browsers, Web Servers, URL, HTTP, TCP/IP, CGI, Client-Server Concepts, etc for the better understanding of subject.

It is combination of **Hypertext** and **markup**. *HyperText* is the method by which you move around on the web - by clicking on special text called **hyperlinks** which bring you to the next page and Markup is what **HTML tags** do to the text inside them. They mark it as a certain type of text (*italicized* text, for example) .HTML has two types of markup: **tags** and **character entities**.

In HTML a tag tells the browser what to do. When you write an HTML page, you enter tags for many reasons…to change the appearance of text, to show graphics, or to make a link to another page. TAGS are constructed with brackets between which the tag is placed. Tags are placed around segments of text, so there is usually a companion end tag which is identical to the start tag except it includes a forward slash. Here are start and end tags for a title:

<TITLE>Introduction to HTML</TITLE>

HTML also includes markup called **character entities**. These are used to include international characters as well as characters usually included in tags as markup. Entities start with an ampersand, followed by the entity name and end with a semicolon. Here is a character entity for an ampersand **(&amp;)**

- Each HTML document is contained within the **<HTML>** tag. If you leave it out, your document will probably work fine today, but someday it might not.
- Each HTML document also includes a header section indicated by the **<HEAD>** tag which contains things like title and keywords. It should always be present and at least contain the **<TITLE>** with the document title.

- Everything left will be part of the document body, enclosed by the **<BODY>** tag.

You should put a document type declaration at the top of your HTML document so that the browser will know what version of HTML you are using. If you are using HTML 3.2 it will always look like this:

```
<html>
    <head><title> royal college </title></head>
<body>
      this is my first webpage.
</body>
</html>
```

CORE HTML ELEMENTS

Let's start by taking the four main elements that form the basic structure of every document. <HTML>, <HEAD>, <TITLE> and <BODY>.

The <HTML> Element: The <HTML> element is the containing element for whole HTML document. Each HTML document should have an opening and closing tag </HTML>.

The <HEAD> Element: The <HEAD> element is just container for all the other header elements and it should be appear first after the <HTML> tag. Each <HEAD> element should contain a <TITLE> element indicating the title of the document. It also consists the others elements such as <STYLE>, <SCRIPT>, <OBJECT> <META> etc.

The <TITLE> Element: The <TITLE> element specifies the title of the document window at very top of the browser. It is important to use a title element to usually describe the content of your site. It may not contain any other elements.

The <BODY> Element: The <BODY> element appears after the <head> element and contains the part of the web pages that you actually see in the main browser which is referred as *body content.* It may contain any things such as paragraphs, headings, forms, tables, lists and images etc.

BASIC TEXT FORMATTING ELEMENTS (TAGS)

The basic text formatting elements are used to structure the web pages which are important for representation.

1) HEADING ELEMENTS
2) LINE BREAK
3) PARAGRAPH ELEMENT
4) PRE FORMATTED TEXT ELEMENT

1. **HEADING ELEMENTS :** HTML offers six levels of headings, which use the elements <h1>,<h2>,<h3>,<h4>,<h5>and <h6>.Which can be used to display the text in different sizes. I.e the element <h1>is used to display the largest and <h6> is the smallest.

   Attribute:- The ***align*** attribute indicates whether the heading appears to the left(default),right or center of the document.

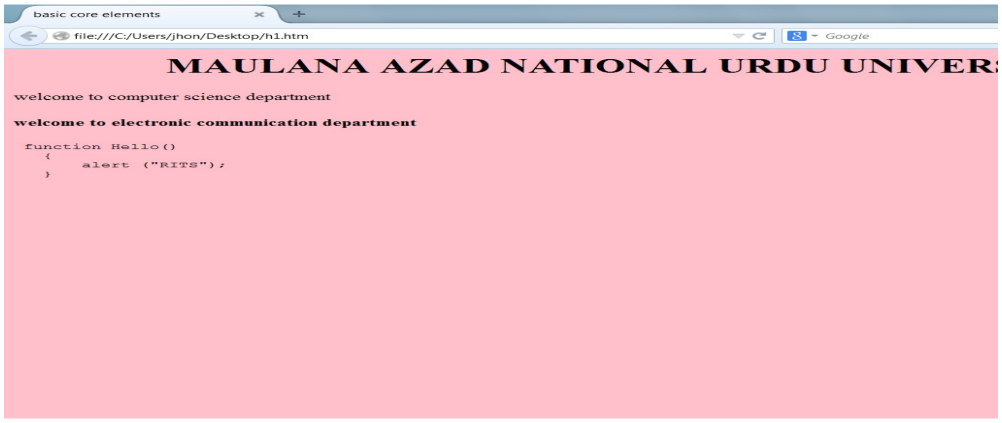
3 **PARAGRAPH:** The **<p>** element offers another way to structure your text in paragraphs. Each paragraph of text should be written in between opening<p> and closing </p> tag.

4 **PREFORMATTED TEXT:** Sometimes your text follow the exact format of how it is written in the HTML document. **<pre>** is used to preserve the formatting.

2

3 **LINE BREAKING:** Sometimes if we need to break the line of text in a document to a new line **<br>** element is used to accomplish this task.

```
<html>
<head>
<title> basic core elements </title>
</head>
<body bgcolor="pink">
<h1 align="center">MAULANA AZAD NATIONAL URDU UNIVERSITY</h1>
  <p>welcome to computer science department</p>
    <h4>
              <p>welcome to electronic communication department</p>
   </h4>
<pre>
 function Hello()
    {
       alert ("RITS");
    }
</pre>
</body>
</html>
```



MAULANA AZAD NATIONAL URDU UNIVER

welcome to computer science department

welcome to electronic communication department

```
 function Hello()
   {
       alert ("RITS");
   }
```

PRESENTATIONAL ELEMENTS

- **<b> :** Anything appear in a <b> elements is displayed in *bold*.

  ex: The following word uses an <b>bold</b> typeface.
- **<i> :** Anything appear in a <i> elements is displayed *italic*.

  ex: The following word uses a <i>italic</i> typeface.
- **<u>:** Anything appears in <u> elements is *underlined* with a simple line.

  ex: The following word would be <u>underline</u>.
- **<s> or <strike>:** The contents of an <s> or <strike> elements is displayed with a *strikethrough.*

  ex: The following word would have a <s>strike line </s>.
- **<sup>:** The contents of a <sup> elements is written in *superscript*.

  ex: Written on 31<sup>st</sup> December.
- **<sub>:** The contents of a <sub> elements is written in *subscript*.

  ex: The equation p<sub>1</sub> + p<sub>2</sub>
- **<big> :** The content of the <big> element is displayed one font size larger than the rest of the surrounding text.

  ex: The following word is displayed in <big> bigger</big> font than other.
- **<small> :** The content of the <big> element is displayed one font size smaller than the rest of the surrounding text.

  ex: The following word is displayed in <small> bigger</small> font than other.
- **<hr> :** This element created the horizontal rule across the page.
- **<em>:** The content of <em> is intended to be emphasis in your document and it is usually displayed in italicized.

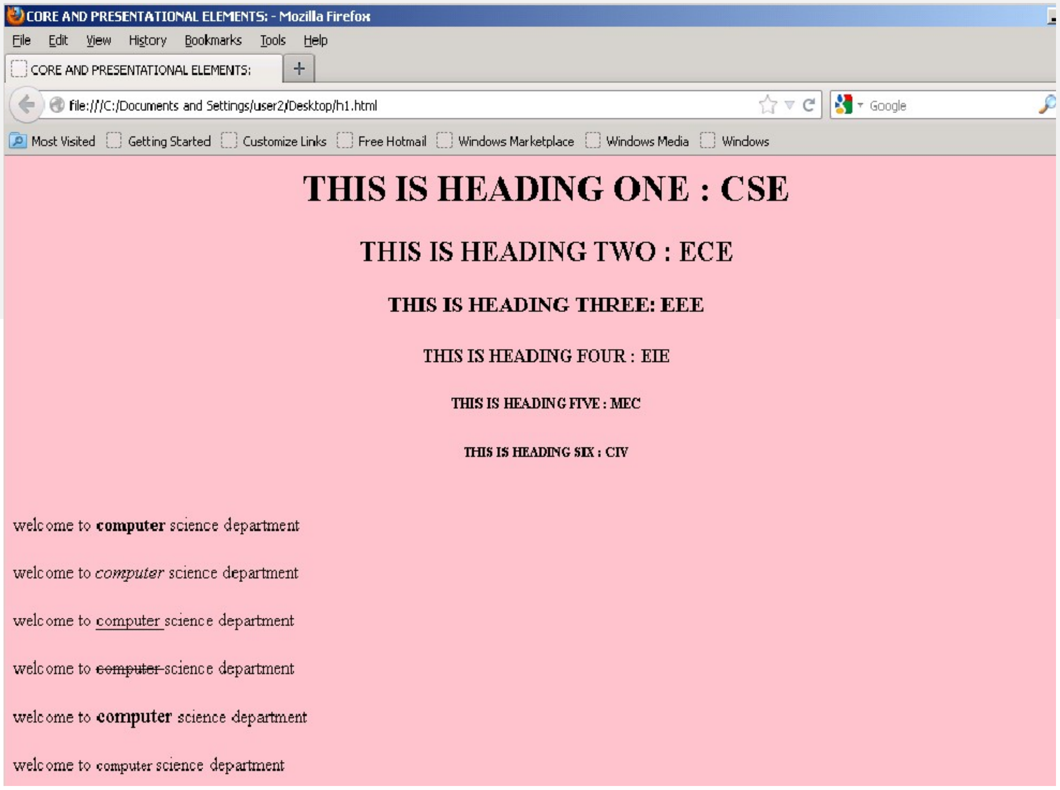
- **<strong> :** The <strong> elements is intended to show strong emphasis for its contents than <em> element. It is usually displayed in bold font.
- **<abbr> :**It is used for abbreviation of a text in a document. It has the attribute called title which has full version of acronym.

  ex: My name is <abbr title=”Mister ”>Mr.</abbr>
- **<acronym>:** This element allows you to indicate that the text between an opening <acronym> and closing </acronym> tags is an acronym. It has the attribute called title which has full version of acronym.

  ex: This chapter covers the marking the text in <acronym title=”Hyper text markup language”>HTML</acronym>

```
<html>
<head>
<title> CORE AND PRESENTATIONAL ELEMENTS: </title>
</head>
<body bgcolor="pink">
<h1 align="center">THIS IS HEADING ONE  : CSE </h1>
<h2 align="center">THIS IS HEADING TWO  : ECE </h2>
<h3 align="center">THIS IS HEADING THREE: EEE </h3>
<h4 align="center">THIS IS HEADING FOUR : EIE </h4>
<h5 align="center">THIS IS HEADING FIVE : MEC </h5>
<h6 align="center">THIS IS HEADING SIX  : CIV </h6>
<p>
<br>welcome to <b> computer </b> science department</br>
<br>welcome to <i> computer </i> science department</br>
<br>welcome to <u> computer </u> science department</br>
<br>welcome to <strike> computer </strike> science department</br>
<br>welcome to <big> computer </big> science department</br>
<br>welcome to <small> computer </small> science department</br>
<br>welcome to <strong> computer </strong> science department</br>
<br>welcome to <emp> computer </emp> science department</br>
```

```
<br><hr>welcome to  computer  science department </hr></br>
<br>welcome to <abbr title=”Mister ”>Mr.</abbr>science dept</br>
<br>Mr.Ghandhi born on 2<sup>nd</sup>october</br>
<br>x<sub>1</sub>-x<sub>2</sub> is 10.</br>s
</p>
</body>
</html>
```

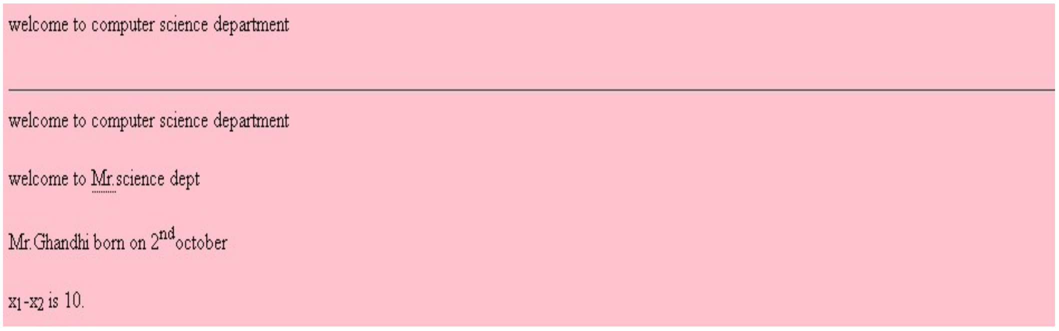


CORE AND PRESENTATIONAL ELEMENTS: - Mozilla Firefox

file:///C:/Documents and Settings/user2/Desktop/h1.html

```
THIS IS HEADING ONE : CSE
THIS IS HEADING TWO : ECE
THIS IS HEADING THREE: EEE
THIS IS HEADING FOUR : EIE
THIS IS HEADING FIVE : MEC
THIS IS HEADING SIX : CIV

welcome to computer science department
welcome to computer science department
welcome to computer science department
welcome to computer science department
welcome to computer science department
welcome to computer science department
```



```
welcome to computer science department
________________________________________
welcome to computer science department
welcome to Mr.science dept
Mr.Ghandhi born on 2nd october
x1 -x2 is 10.
```

LIST: List allows us to display information in a compact specific format, such as a list of shopping items, a list of names of employees in an organization, a list of places names etc. You can create three types of List in HTML as follows:

- Unordered lists
- Ordered lists
- Definition lists

Unordered List : Unordered lists can be created using the element <ul>(unordered list) with bullet point for each line of text.<ul> element consists the <li> elements which specify the list items for representation in bullet form and you should close the element </li> for each opening <li>.

```
<ul>
  <li>CSE</li>
  <li>ECE</li>
  <li>EEE</li>
  <li>MECH</li>
</ul>
```

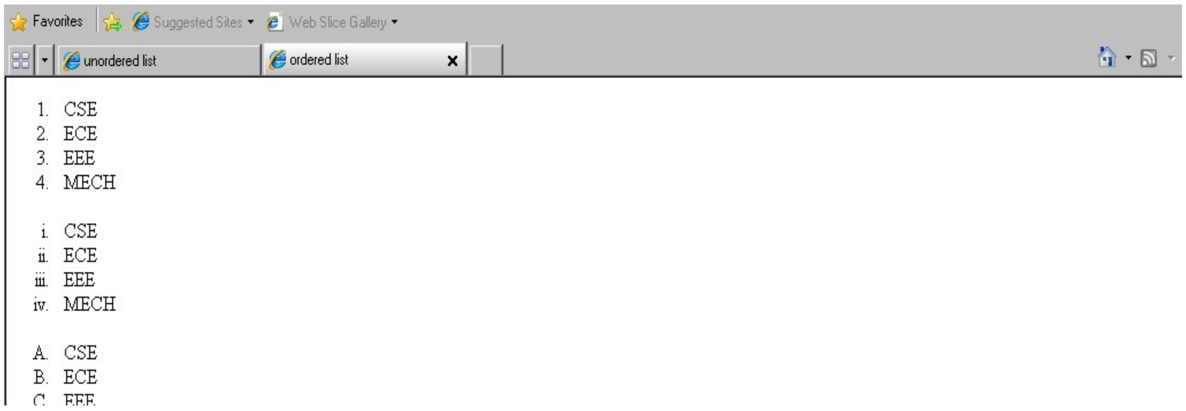
Ordered List: In an ordered list each item is started with numbers, letters or roman numerals rather than with a bullet. An ordered list is contained in <ol> element and each list item is contained between opening <li> and closing </li> tags.

***type* -Attribute:** The *type* attribute specifies the type of ordered style it represents such as numbered, letters or Roman numerals.

Value for *type* attribute	Description	Examples
1	Arabic Numerals(default)	1,2,3,4…….
A	Capitals Letters	A,B,C,D…..
A	Small Letters	A,b,c,d…..
I	Large Roman Numerals	I,II,III,IV…..

I	Small Roman Numerals	i,ii,iii,iv

```
<ol>
<li>CSE</li>
<li>ECE</li>
<li>EEE</li>
<li>MECH</li>
</ol>
<ol  type="i">
<li>CSE</li>
<li>ECE</li>
<li>EEE</li>
<li>MECH</li>
</ol>
<ol type="A">
<li>CSE</li>
<li>ECE</li>
<li>EEE</li>
<li>MECH</li>
</ol>
```



```
 1. CSE
 2. ECE
 3. EEE
 4. MECH

  i. CSE
 ii. ECE
iii. EEE
 iv. MECH

 A. CSE
 B. ECE
 C. EEE
```

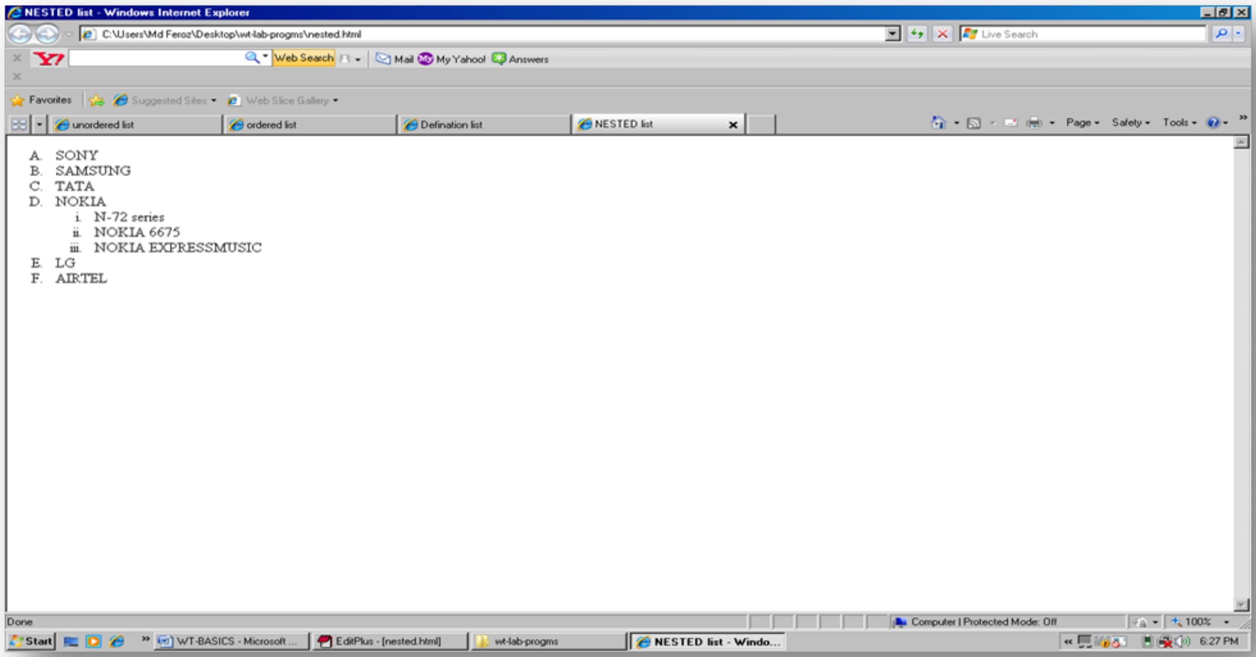
Definition List: It is a special kind of list for providing terms followed by a short text definition or description for them. It contained inside the <dl> element that contains alternating <dt> and <dd> elements. The content <dt> element is the term you will be the defining. The <dd> element contains the definition of the previous<dt> element.

```
<dl>
 <dt> Unordered List</dt>
   <dd>A list of bullet points.</dd>
 <dt>Ordered List</dt>
   <dd> A list of points such as numbers , letters, and roman steps...</dd>
```

```
</dl>
```

Nested List: A nested list is a combination of different or same list. i.e a list inside another list is called Nesting list. For example you might want a numbered list with separate point corresponding to one of the list items.

```
<ol  type="A">
<li>SONY</li>
<li>SAMSUNG</li>
<li>TATA</li>
<li>NOKIA
          <ol type="i">
          <li>N-72 series</li>
          <li>NOKIA 6675</li>
          <li>NOKIA EXPRESSMUSIC</li>
   </li>
  </ol>
  <li>LG</li>
  <li>AIRTEL</li>
</ol>
```


NESTED list - Windows Internet Explorer
C:\Users\Md Feroz\Desktop\wt-lab-progms\nested.html
unordered list
ordered list
Defination list
NESTED list
Done
Computer | Protected Mode: Off

```
A. SONY
B. SAMSUNG
C. TATA
D. NOKIA
      i.  N-72 series
      ii. NOKIA 6675
      iii. NOKIA EXPRESSMUSIC
E. LG
F. AIRTEL
```

FONT ELEMENT: The ***<font>*** tag is used to control the text which appears in a web page such as font size, style of the text and color of the text etc.

```
<FONT SIZE=”10” COLOR=”pink”   FACE=”Arial, verdana”>
```

```
            Maulana Azad national Urdu University
</FONT>
```

Grouping Elements with <div> and <span> : The <div> and <span> elements allow you to group several elements to create sections or subsection of a s web page without affecting the appearance of the page, but they are commonly used with **CSS.**

The <div> element is used to group block – level elements such as to put all of the footnotes on a page within <div> to indicate that all of the elements within that <div> element relate to the footnotes.

```
<div class=”footnotes”>
  <h2>Footnotes</ht>
  <p><b>1</b> The world wide web was invented by Tim Burners - lee</p>
  <p><b>2</b> The W3C is the best consortium for maintaining the web standards</p>
 </div>
```

The <span > element on the other hand, can be used to group inline elements only.i.e. It is used to group the sentence or paragraph.

```
<div class=”footnotes”>
  <h2>Footnotes</ht>
  <p><span class=”inventor”><b>1</b> The world wide web was invented by Tim Berners –
        lee</span></p>
  <p><b>2</b> The W3C is the best consortium for maintaining the web standards</p>
 </div>
```

HYPERLINK: The **<a>** element is an anchor element which can be used to create the link between different web documents. The<a> element consists an attribute ***href*** which specifies the **URL** of the web page to which link is specified.

A link can be created on a text that enclosed between the opening <a> tag and closing</a> tag.

```
<body>
    return to main page <a href =”index.html” >index</a>
</body>
```

href is an attribute which is used to specified the URL. The URL can also be written as http://www.ritsengg.com/index.html

IMAGE ELEMENT: Images and graphics can really bring your site to life and more interactively. We can add different formats of images such as GIFs, JPEGs and PNGs. Once you insert a right kind of images into your web pages then you can easily create them to link. Images are usually added to a site using the ***<img>*** tag. It consists the following attributes.

The ***src*** attribute indicating the source of the image which required the URL of the image to load. The URL can be an absolute or a relative. ***i.e.*** **src = "URL"**

1. The ***alt*** attribute whose value is an alternate description for the image. ***i.e.*** **alt = " RITS-LOGO "**
2. The ***align*** attribute is used to align the images within the page. It can take one of the following values left, right, middle, bottom and top. ***i.e.*** **align = "top".**
3. The ***border*** attribute specifies the width of the border around the image in pixel. ***i.e.*** **border = "3".**
4. The ***height*** and ***width*** attribute specifies the height and width of the image in pixel. ***i.e.*** **height="40" width="60".**
5. The ***hspace*** and ***vspace*** attribute can be used to control the amount of whitespace around an image. **i.e. hspace="10" and vspace ="12".**
6. The ***longdesc*** attribute is used to indicate the URL of a document containing a description for the image in more details. **i.e. longdesc = " ../rits/images-doc/profile.text"**

Example:

```
<html>
<head><title>Image Demp</title></head>
<body bgcolor = "cyan"><h4 align="center">TAJ MAHAL PHOTO </h4>
<p>
<center>
<img src="taj.JPG" widht = "227" height = "300">
</p>
</body>
</html>
```

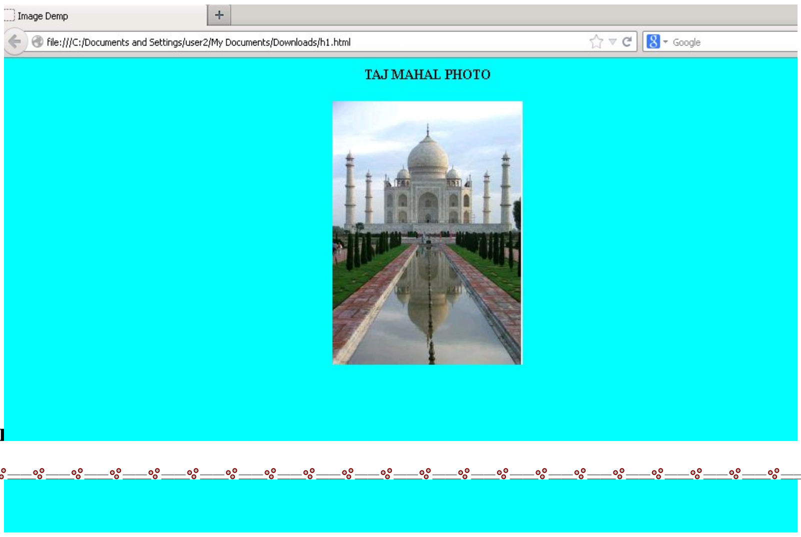


Image Demp
file:///C:/Documents and Settings/user2/My Documents/Downloads/h1.html
Google
TAJ MAHAL PHOTO

TABLE ELEMENT : In HTML tables are used to organize the data in the form of grids such as ***rows*** and ***columns***. Tables are commonly used to display of data such as time tables, financial reports and spreadsheets results. Tables can represent the data in rectangular grid. Each rectangle grid is known as ***cell.*** A *row* is made up of a set of cells on the same line from left to right, while *column* is made up of cells going from top to bottom. A table is created using the element **<table>.** A table row is created using **<tr>** element within <table> tag and the table data is created using **<dt>** tag. we can create many rows and columns as required.

```
<TABLE>
   <TR>
    <TD>ROW-1 </TD>
</TR>
    <TR>
    <TD>ROW-2 </TD>
</TR>
</TABLE>
```

BASIC ATTRIBUTES OF TABLE ELEMENT

Attribute	Purpose
Align	This attribute is used to align the table to be left(default), right and center of the page.
bgcolor	This attribute is used to set the background color of the table ,the color is specified by either constant values(PINK,GREEN,BLACK,RED…) or by six digits hex-code(#F0C9Ck).
Border	It is used to specify the border around the table and each individual cell. the default value of border is **"0".**
Bordercolor	This attribute supplies the border color for the given table.
Background	The attribute specified the background image for better visualization of the table.
Cols	It specifies the number of columns in a given table.
Width	It specifies the width of the table.
Height	It specifies the height of the table.
Cellspacing	It used to create a space between the borders of each cell. The values can be in an amount of cell or it can be a percentage.

ATTRIBUTES OF <TR> ELEMENT

Attribute	Purpose
Align	This attribute specifies the position of the content of all of the cells in the row. The values of *align* can be either *left,right,center,justify* and *char*.
bgcolor	The ***bgcolor*** attribute is used to set the background color of the table row, the color is

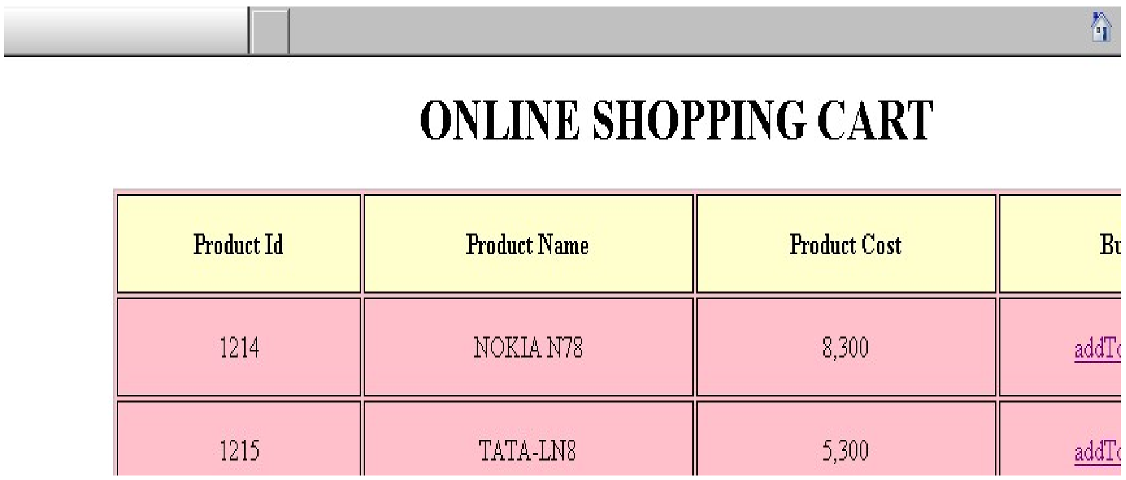
	specified by either constant values(PINK,GREEN,BLACK,RED…) or by six digits he-code(#F0C9Ck).
Valign	It specifies the vertical alignment of the content of each cell in the row. Such as top, bottom, middle and baseline.
Bordercolor	The ***bordercolor*** attribute supplies the border color for the given table row.
Char	The ***char*** attribute is used to specify that the content of each cell within the row will be aligned around the first instance of a particular character known as an ***axis character.***

ATTRIBUTES OF <TD> and <TH> ELEMENTS

Attribute	Purpose
Align	It allows to sets the horizontal alignment for the content of the cell.
bgcolor	The bgcolor attribute sets the background color for the cell.
Abbr	The ***abbr*** attribute is used to provide an abbreviated version of the cell's content.
Bordercolor	The ***bordercolor*** attribute supplies the border color for the given table row.
Nowrap	The ***nowrap*** attribute is used to stop text from wrapping onto a new line within a cell.
Rowspan	The ***rowspan*** attribute specifies the number of rows of the table a cell will span, the value of the attribute being the numbers of rows the cell stretches across.
Colspan	This attribute specifies that how many columns of the table a cell will span across.
height & width	The ***height*** and ***width*** attributes allows you to specify the height and width of a cell in pixel or as a percentage.

```
<table align="center" border="1" bgcolor="pink" width="70%" height="40%">
    <tr bgcolor = "#ffffcc" align="center" bordercolor="#000000">
        <th> product id  </th>
         <th> product name </th>
          <th> product cost </th>
         <th>buy</th>
     </tr>
    <tr bordercolor="#000000" align="center">
      <td>1214</td>
       <td>nokia n78</td>
       <td>8,300</td>
        <td><a href="buy.html">addtocart</a></td>
     </tr>
  <tr bordercolor="#000000" align="center">
      <td>1215</td>
      <td>tata-ln8</td>
     <td>5,300</td>
     <td><a href="buy.html">addtocart</a></td>
  </tr>
<tr bordercolor="#000000" align="center">
      <td>1214</td>
      <td>samsung </td>
      <td>7,350</td>
     <td><a href="buy.html">addtocart</a></td>
  </tr>
```

```
</table>
```



ONLINE SHOPPING CART

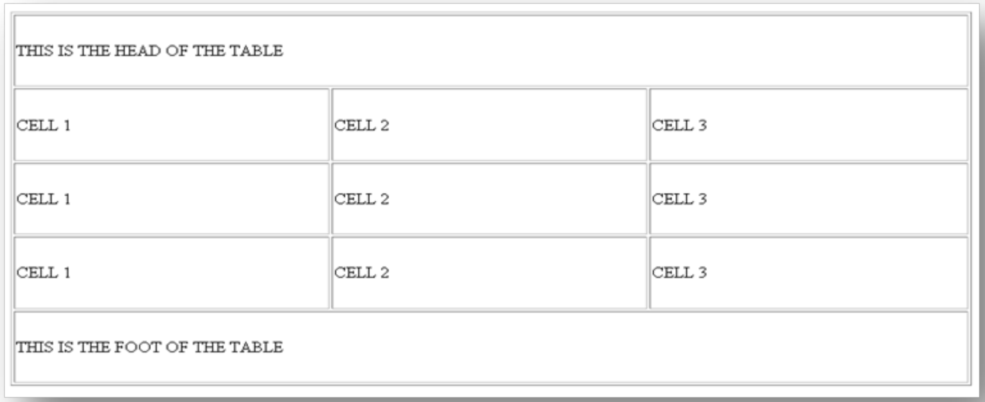
Product Id	Product Name	Product Cost	Bu
1214	NOKIA N78	8,300	addT
1215	TATA-LN8	5,300	addT

ADVANCED TABLES: The advanced tables can be divided into three portions: a ***header***, a ***body***, and a ***foot***. The head and foot are similar to the headers and footers in a word-processed document, which remain the same for every page, while the body is the main content of the table. The Three elements for separating the head, body and foot of a table are.

- **<thread>** to create a separate table header.
- **<tbody>** to indicate the main body of the table.
- **<tfoot>** to create separate table footer.

```
<table border="1" width = "70%" height="70%" align = "center">
  <thead>
    <tr>
        <td colspan="4">this is the head of the table </td>
    </tr>
  </thead>
  <tfoot>
   <tr>
      <td colspan = "4" > this is the foot of the table </td>
   </tr>
  </tfoot>
  <tbody>
   <tr>
     <td>cell 1 </td>
     <td>cell 2 </td>
    <td>cell 3 </td>
   </tr>
  </tbody>
  <tbody>
   <tr>
     <td>cell 1 </td>
```

```
      <td>cell 2 </td>
    <td>cell 3 </td>
   </tr>
  </tbody>
</table>
```



THIS IS THE HEAD OF THE TABLE		
CELL 1	CELL 2	CELL 3
CELL 1	CELL 2	CELL 3
CELL 1	CELL 2	CELL 3
THIS IS THE FOOT OF THE TABLE		

SPANNING ROWS AND COLUMNS OF A TABLE

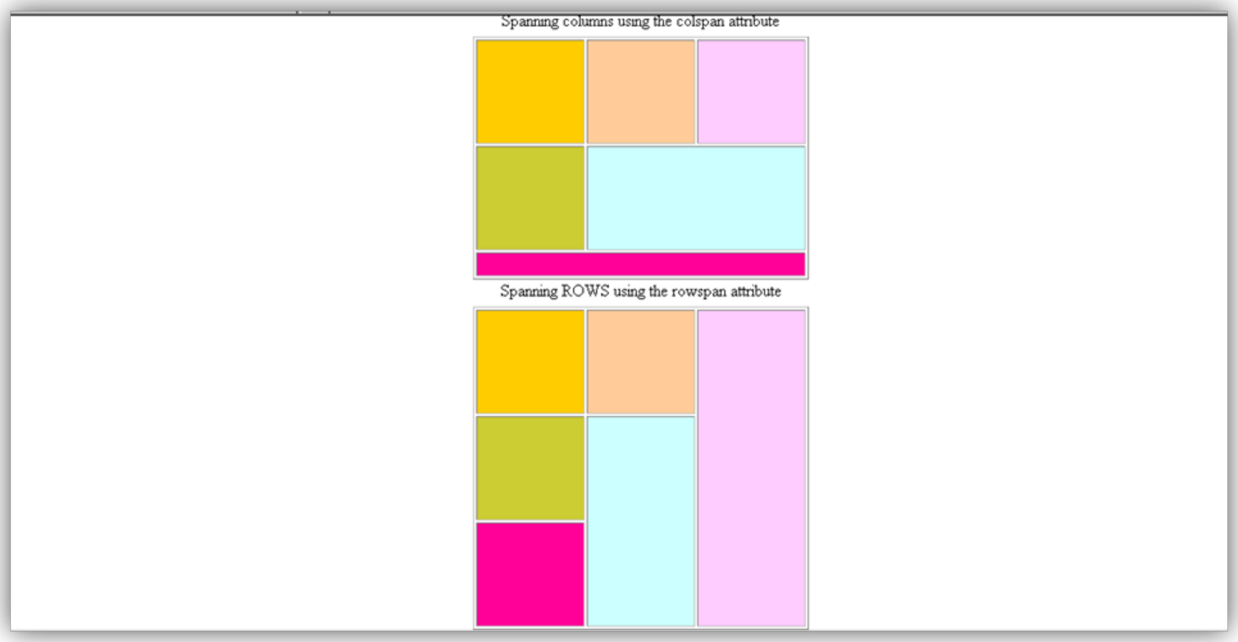
1. Spanning Columns Using the *colspan* Attribute: The colspan attribute allows a cell to stretch across more than one column, which means it can stretch across more than one rectangle horizontally in grid.

```
<table border="1"  align = "center">
  <caption>spanning columns using the colspan attribute</caption>
  <tr>
    <td bgcolor="#ffcc00" width="100" height ="100" >&nbsp;</td>
    <td bgcolor="#ffcc99" width="100" height ="100" >&nbsp;</td>
    <td bgcolor="#ffccff" width="100" height ="100" >&nbsp;</td>
</tr>
  <tr>
    <td bgcolor="#cccc33" width="100" height ="100" >&nbsp;</td>
    <td colspan="2" bgcolor="#ccffff">&nbsp;</td>
  </tr>
  <tr>
    <td colspan="3" bgcolor="#ff0099">&nbsp;</td>
  </tr>
</table>
```

2. Spanning Rows Using the *rowspan* Attribute: The rowspan attribute does the same thing as colspan attribute, but it works in the opposite direction; it allows cell to stretch vertical across cells.

```
  <table border="1” align = "center">
```

```
 <caption>Spanning ROWS using the rowspan attribute</caption>
  <tr>
  <td bgcolor="#ffcc00" width="100" height ="100" >&nbsp;</td>
  <td bgcolor="#ffcc99" width="100" height ="100" >&nbsp;</td>
  <td rowspan  = "3" bgcolor="#ffccff" width="100" height ="100" >&nbsp;</td>
</tr>
 <tr>
  <td bgcolor="#cccc33" width="100" height ="100" >&nbsp;</td>
  <td rowspan="2" bgcolor="#ccffff">&nbsp;</td>
 </tr>
 <tr>
  <td bgcolor="#ff0099" height ="100">&nbsp;</td>
  </tr>
</table>
```


Spanning columns using the colspan attribute
Spanning ROWS using the rowspan attribute

FORMS : Many websites collects information from the users who visits the site, forms are used to collect the information ,while browsing several websites demands the details to be entered into specific locations, for example text fields, checkboxes, buttons etc. A form contains fields where an end user can input information and send it to web server.A form can be used for different purposes such as registrations, order entry, subscription etc.A form can be created by using the element **<FORM>. </FORM>** which consists the basic optional attributes ***ACTION*** and ***METHOD.***

Action Attribute: This attribute indicates that what an action to be performed when a user submits the form. i.e. when a user submits the form which consisting the details such as username and password are get passed to the web server which consists the script to perform processing on the data contained in a form such as validating the user authentication.

action = " http://www.ritsengg.com/login.jsp"

Method Attribute : The data can be send to the server in two ways

- ∆ The **GET** method , which sends data as part of the URL(Default method).
- ∆ The **POST** method, which hides data in the HTTP headers.

METHOD = " POST/GET "

ID ATTRIBUTE: The **id** Attribute allows you to identify the <form> elements within a page. It is a good practice to give every <form> element an **id** attribute.

NAME ATTRIBUTE: The name attribute is the predecessor to the id attribute.

```
<form  method="post" action="..rits/login.jsp" >
  Name :       <input type ="text"  name="user"/>
  Password : <input type="password" name="pwd"/><br>
                  <input type="submit" value="submit"/>
</form>
```

FORM CONTROLS: There are different form controls that collect the data from the visitor that visit the website as follows:

Text Input controls,
Buttons,
Checkboxes and Radio buttons,
select boxes and list boxes,
File select box, and
Hidden controls.

1.Text Input controls :- There are three text input control used in form as follows.

Single-line text input controls:-It is a single line user input control such as search box or username. They are created using the element***<INPUT>.***

Attributes	Purpose
Type	Indicates that type of input you want to create. The value of this attribute could be a *text* when you want to create single line text input control.

Name	Used to give the name part of the name/value pair that is sent to the server, representing each form control and the value the user entered.
Value	It provides the initial value for the text field control that the user will see when the form loads.
Size	It allows to specify the width of the text-input control in terms of characters.
maxLength	It allows you to specify the maximum number of characters a user can enter into the text-box.

<INPUT TYPE=”TEXT”   SIZE=”10”/>

- **Password input control:-**It is just like single line input text controls but the characters that you entered cannot be seen on the screen, it tends to show the characters in asterisk or dot of each user types instead. These controls are used for entering the password on login form and which are mainly used to collect the sensitive data such as password and credit card numbers during online transactions.

<INPUT TYPE=”PASSWORD”   SIZE=”10”/>

- **Multi-line text controls :-** It is a multiple line editing text field that is used to enter the data larger than single line text. It can be used to get the Address of the visitor. It can be created using <TEXTAREA> element.

Attributes	Purpose
Name	Used to give the name part of the name/value pair that is sent to the server.
Rows	Used to specify the size of a <textarea>,it indicates the numbers of rows of text area should have corresponding to its height.
Cols	Used to specify the size of a <textarea>,it indicates the numbers of columns of text area should have corresponding to its width.

2. Buttons: Buttons are most commonly used in a form which can create the user interactive application that responds to the user when he submits/click the button. Buttons are created using the following ways.

❖ <INPUT>element with a ***type*** attribute whose value is ***submit, reset, or button.***

Attributes	Purpose
Type	Specify the type of button you want to create such as submit, reset , or button.
name	Provides the name to the button.
Value	Enables you to specify what the text on the button should read.
Size	It allows to specify the width of the button
onClick	It is used to trigger a script when the user clicks on the button.

3. Checkboxes: - checkboxes are ideal form controls that allow a user to provide a simple on or off response with one control and also select several items from the list of possible options.

A checkboxes can be created using the <input> element whose *type* attribute has a value of **checkbox**.

<INPUT TYPE =”checkbox” name=”skill” value=”HTML”/>

Attribute	Purpose
type	Indicates that you want to create a checkbox
Name	Gives the name of the control.
Value	The value that will sent to the server
checked	Indicates that the checkbox is selected.
Size	Indicates the size of the checkbox in pixel

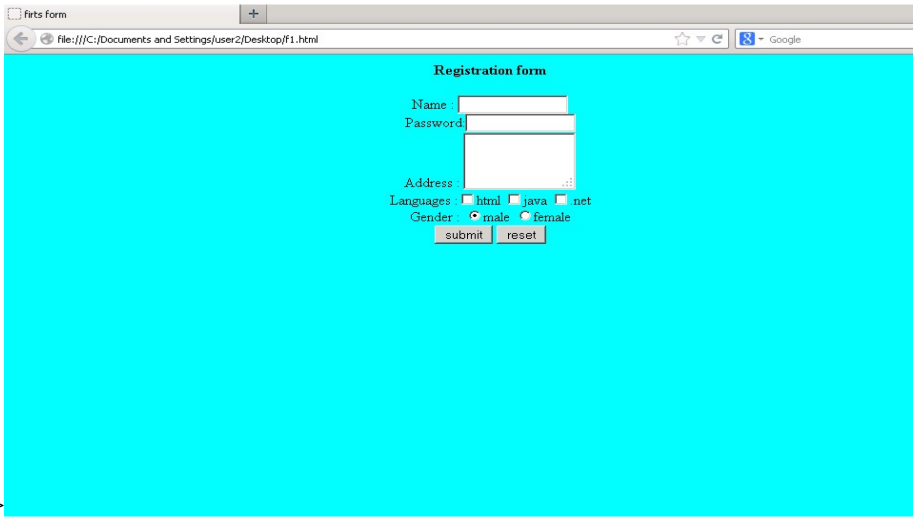
4.Radio Buttons: It is a collection of checkboxes that can share a name and only one of them can be selected. Once the radio button has been selected, the user clicks another option, the new option is selected and the old one is deselected. A radio button can be created using the <input> element whose *type* values is **radio.**

<INPUT TYPE =”radio” name=”gender” value=”male”/>

Attribute	Purpose
type	Indicates that you want to create a radio button
Name	the name of the form control.
Value	The value that will sent to the server
Checked	Indicates that the option is selected by default.
Size	Indicates the size of the radio button in pixel

```
<html>
<head><title> firts form</title></head>
<body bgcolor="cyan" ><h4 align = "center"> Registration form</h4>
<form>
<center>
  Name : <input type="text" value="" size="15"/><br>
  Password:<input  type="password" value="" size="15" maxlength="15"/></br>
  Address : <textarea cols="12" rows="3"/></textarea><br>
  Languages :<input type="checkbox" name="skill" value="html"/>html
              <input type="checkbox" name="skill" value="java"/>java
             <input type="checkbox" name="skill" value=".net"/>.net<br>
 Gender :    <input type="radio" name="gender" value="male" checked />male
            <input type="radio" name="gender" value="female"/>female<br>
            <input type="SUBMIT" value="submit">
          <input type="RESET" value="reset">
</form>
```

</body>



Registration form

Name :
Password:
Address :
Languages : ☐ html ☐ java ☐ .net
Gender : ● male ○ female
submit reset

</html>

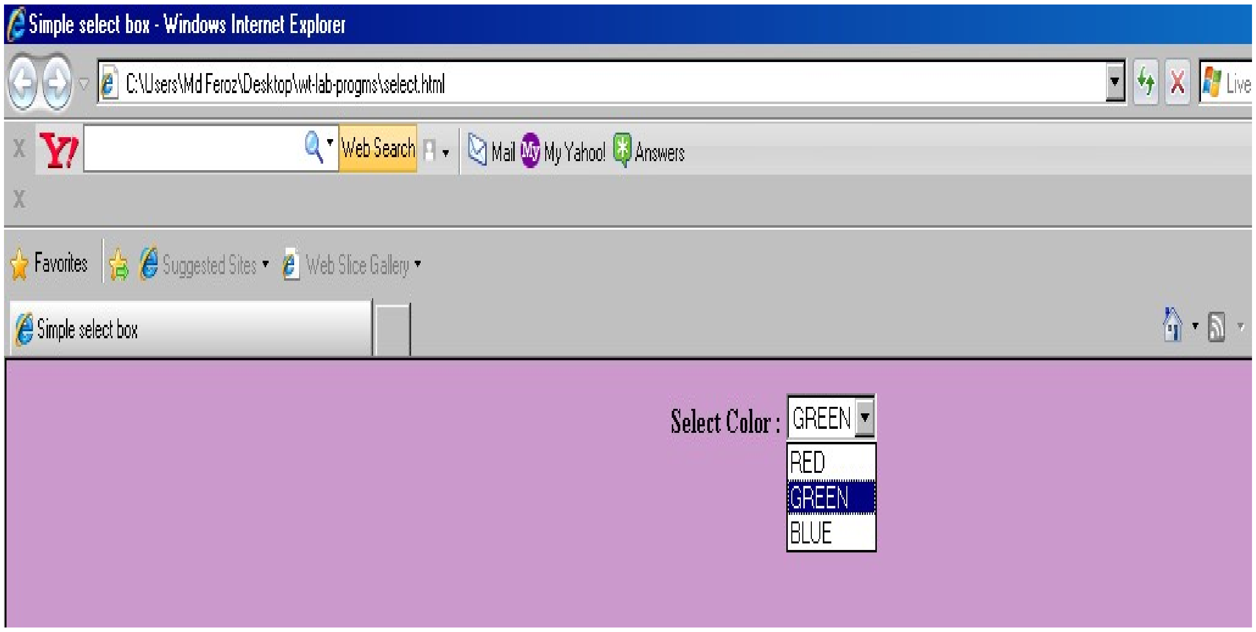
4.Select Boxes : A drop down select box allows users to select one item from a drop-down menu. It takes less space than radio button controls. A drop-down select box is created by using <SELECT> element while each individual option is contained in <OPTION> element within <SELECT> tag.

Attributes	Purpose
Name	The name of the control
Size	It is used to present the scrolling bar to the list box.
Multiple	It allows users to select multiple options from the list.

<OPTION> Attribute

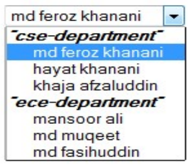
Attributes	Purpose
Value	The Value that is sent to the server
Selected	It specifies that this option is selected initially.
Label	An alternative way of labeling options.

```
<SELECT NAME=”color”>
   <OPTION VALUE=”red”>RED</OPTION>
   <OPTION VALUE=”green” SELECTED>GREEN</OPTION>
   <OPTION VALUE=”blue”>BLUE</OPTION>
</SELECT>
```



Grouping Options with the <optgroup> element : If your list box consists long list of items then you can group them together using the <optgroup> element. The <optgroup> carry a ***label*** attribute whose value is a label for that group of options.

```
 <select name=”faculty”>
  <optgroup  label=”cse-department”>
      <option  value=”feroz>md feroz khanani</option>
     <option  value=”hayat”>hayat khanani</option>
     <option  value=”afjal”>khaja afzaluddin</option>
</optgroup>
<optgroup  label=”ece-department”>
      <option  value=”mansoor ali”>mansoor ali</option>
      <option  value=”md muqeet”>md muqeet</option>
     <option  value=”md fasihuddin”>md fasihuddin </option>
</optgroup>
</select>
```



5.File Belect Box : File select Box is also known as *File upload box***,** which allows a user to upload a file to your web site from his computer. It can be created by using the <INPUT> element whose ***type*** attribute value is **file**. The ***accept*** attribute has been added to the <INPUT> element to indicate the MIME types of the files that can be selected for upload.

```
<INPUT TYPE=”file” NAME =”fileUpload”  ACCEPT=”image/*”/>
```

6. Hidden Controls : Hidden controls can use to pass information between pages without the user seeing it. i.e. It will not visible to the user while it will displayed in the browser, but it can be seen by looking at the source code of the page. Therefore ,Hidden controls are mainly used for sending the any sensitive information that you do not want the user to seethe hidden control can be created using the <INPUT> element whose ***type*** attribute value is **hidden. *name*** and ***value*** can still send to the server for a hidden form control, the hidden control must carry name and value attributes.

```
<input type=”hidden” name=”hide page sent from” value=”rits home”/>
<input type=”submit” value=”click to see”/>
```

HTML FRAMES: Frames divide a browser window into several separate pieces or panes, each pane containing a separate HTML page. It allows that you can load a reload single page without having to reload the entire contents of the browser window. A collection of frames in the browser window is known as a ***frameset.* The <frameset> Element:** The <frameset> element contains a <frame> element for each document. It should be included within <head> tags of the code but not <body>.

Attribute	Purpose
cols	It specifies how many columns are contained in the frameset and the size of each column. For example, to create three columns in browser window, the first take 20 percent, second takes 60% and third takes 30%. **COLS=”20%,60%,30%”**
rows	It is used to specify the rows in the frameset which works like ***cols*** attribute. **ROWS=”20%,60%,30%”**
border	The border attribute specifies the width of the border of each frame in pixel. **BORDER=”1”**

frameborder	It specifies whether a three-dimensional border should be displayed between frames. **FRAMEBORDER=”YES”**
framespacing	This attribute specifies the amount of space between frames in a frameset. **FRAMESPACING =”10”**

The <FRAME> Element: The <frame> element indicates what is in the <frameset> element.i.e It is used to specifies the frame in a frameset element which represents the **.html** page.

Attribute	Purpose
src	This attribute indicates the file that should be used in the frame. **SRC=”main.html”**
name	It allows you to give the name of the frame to indicate to which frame a document should be loaded. **NAME=”main”**
noresize	This attribute prevents a user from resizing the frame **NORESIZE=”NORESIZE”**
frameborder	It specifies whether a border of the frame shown or not. **FRAMEBORDER=”YES/NO”**
scrolling	This attribute allows a user to controls the scrollbars in a possible values. **SCROLLING =”YES/NO/AUTO”**

<NOFRAMES> Element: If a user browser doesn’t support frames, the contents of the <noframes> element should be displayed to the user.

```
<NOFRAMES><BODY>THIS IS VISIBLE TO THE USER IF BROWSER DOES NOT SUPPORT
FRAMES</BODY></NOFRAMES>
```

USES OF FRAMES

One of the most popular uses of frames is to place navigation bars in one frame and then load the pages with the content into a separate frame. This is particularly helpful in three situations:

1. When your navigation bar is rather larger in size, by using frames, the user does not need to reload the navigation bar each time when we views a new page.
2. When your main document is very long and navigation bar provides shortcuts to parts of the main document.
3. When you do not want to reload the whole page.

Example - index.html

```
<FRAMESET COLS=”200,*” FRAMESPACING=”10”>
   <FRAME SRC=”navigation.html” NAME=”navigation” FRAMEBORDER=”0” NORESIZE/>
  <FRAME SRC=”main.html” NAME=”main” FRAMEBORDER=”0” NORESIZE/>
```

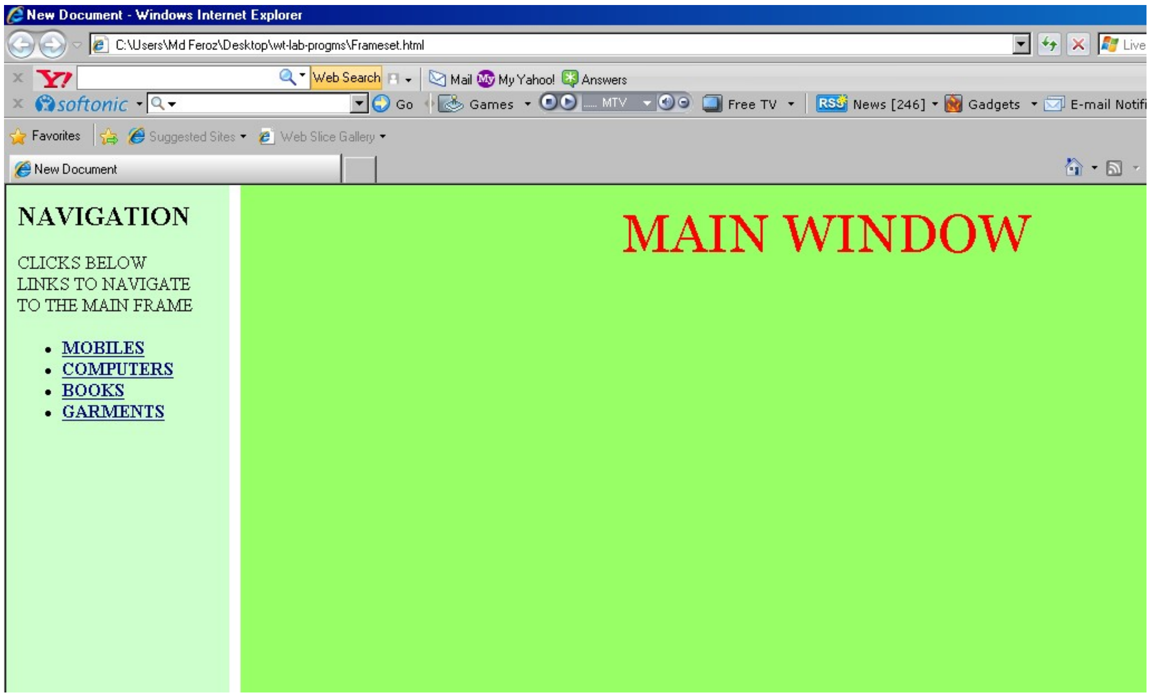
```
<FRAMESET>
```

navigation.html

```
<body bgcolor="#6666ff">
<h2>navigation</h2>clicks below links to navigate to the main frame
<ul>
<li><a href="mobile.html" target="main"><b>mobiles</a></li>
<li><a href="computers" target="main">computers</a></li>
<li><a href="books.html" target="main">books</a></li>
<li><a href="garments.html" target="main">garments</a></li>
</ul>
 </body>
```

main.html

```
<body bgcolor="#99ff66">
<font color="red" size="8" face="georgia">MAIN  WINDOW</font>
</body>
```



NAVIGATION

CLICKS BELOW LINKS TO NAVIGATE TO THE MAIN FRAME

- MOBILES
- COMPUTERS
- BOOKS
- GARMENTS

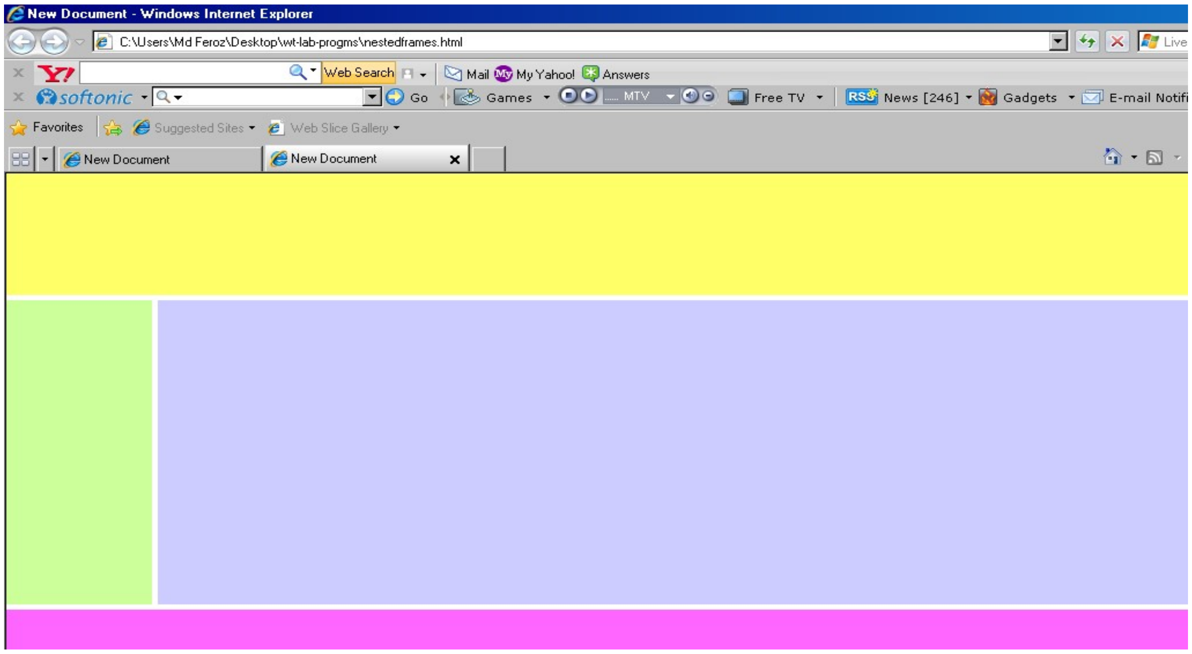
MAIN WINDOW

The **target** attribute can also takes the other attributes values as follows

Value	Purpose
_self	Loads the page into the current frame
_blank	Loads the page into new browser window.
_parent	Loads the page into the parent Window.
_top	Loads the page into the browser window, replacing any current window.

NESTED FRAMSETS: You can create a nested frameset by using <frameset> element in the place of one of the <frame> elements.

```
<frameset rows="20%,50%,*" framespacing="5">
   <frame src="top.html" name="navigation" frameborder="0" noresize/>
     <frameset cols="*,80%,*" framespacing="5">
  <frame src="left.html" name="left" frameborder="0" noresize/>
  <frame src="mid.html" name="mid" frameborder="0" noresize/>
  <frame src="right.html" name="right" frameborder="0" noresize/>
  </frameset>
  <frame src="bottom.html" name="bottom" frameborder="0" noresize/>
<frameset>
</frameset>
<frameset>
```

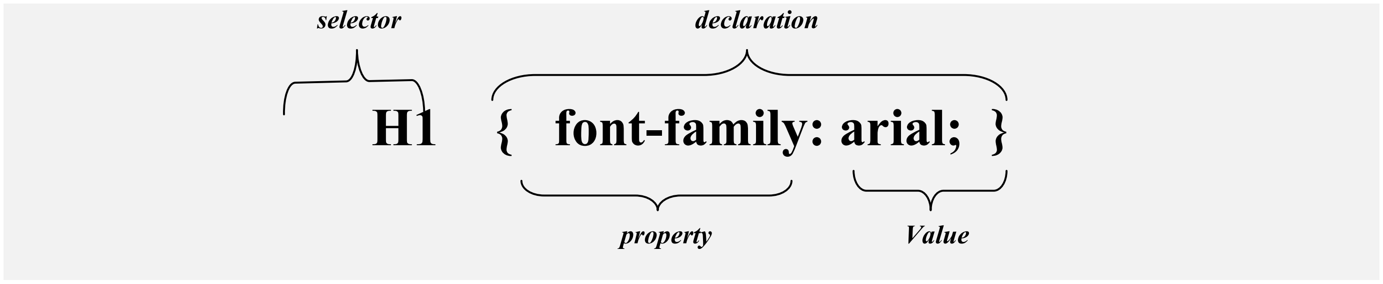


CASCADING STYLE SHEETS (CSS): Cascading style sheets (CSS) allows you to control the style of the web page. It can controls the colors, size of the fonts, the width and colors of lines and the amount of white space between items on the page. Cascading style sheets are also known for specifying the style of the page elements like spacing, margins etc., separately from the structure of the document like text, links etc., This makes pages more manageable and easy to change when required.

1. Using style sheets makes all pages of a web site have a same look and CLASS is used to apply styles.

2. Using CSS makes the programmer specify the precise font type, size, color and other properties of displayed text. CSS are reusable, creating once and reusing them reduces programming effort.
3. CSS allows to specify element background and colors.
4. CSS truly separate content and presentation along with creating own style sheets CSS is a powerful tool for applying universal formatting.
5. CSS with box model are able to control the margins, borders, padding etc.,

CSS SPECIFICATION (RULES): CSS works by allowing you to associate ***rule*** with the elements that appear in the document. These rules specify that how the content of those elements should be rendered.



The CSS rule is made up of two parts as shown above

- The ***selector*** , indicates which elements the declaration applies to.
- The ***declaration***, which sets out how the elements should be styled.

The declaration parts is further splits into two sub parts

- A ***property***, which is the property of the selected elements that you want to effect, in this case the **font-family** property.
- A ***value***, which is the specification for this property, in this case it is **arial** typeface.

Note : The rule body begins and ends with curly brace ({ and }) and values are assigned to corresponding property using a colon (:) not an equal (=) sign.

There are three types of Style sheets as follows

❖ **Inline Style Sheets**
❖ **Internal or embedded Style Sheets**
❖ **External Style Sheets.**

Inline Style Sheets: In Inline Style Sheets the configuration of CSS can be rendered by using The **STYLE** attribute within the elements.

```
<H1 STYLE=”FONT-FAMILY :  ARIAL; FONT-SIZE:12pt;COLOR : GREEN;
TEXT-ALIGN: CENTER”> THIS TEXT UNDER THE INFLUENCE OF CSS RULES SPECIFIED
</H1>
```

Internal or Embedded Style Sheets: In Internal Style Sheets, we need to specify the CSS specification in <STYLE> Element within the <HEAD> tag. The <STYLE> element must include the selectors which are declared in <BODY> region. Internal style sheet is specified in the web page itself. They collect the styling elements in one place that is applied throughout the page.

```
<head>
<style type=”text/css”>
  body {
              color : #f0f900;
              background-color: cyan;
             }
  h1   {
              font-family : verdana, arial, sans-serif;
         }
p {font-size: 20pt ;}
</style>
</head>
```

External Style Sheets: An External Style Sheets is a separate Style Sheets with **(.CSS extension)** which includes the style configuration which can be applicable to many documents at a time. If two or more documents are going to use a style sheets, you should always use an external CSS.

Advantages of external CSS over Internal or Inline CSS

- The same style sheet can be reused by all of the web pages in your site.
- The style written only once, rather than appearing on every elements in every document, the source document is smaller.i.e. Once the CSS style sheet has been downloaded with the first document that uses it, the subsequent documents will be quicker to be download, which put less strain to the server.
- You can change the appearance of several pages by altering the single style sheet rather than each individual page.
- The style sheet can act as a template to help different authors achieve the same style of a document without learning all of the individual style settings.

- ❑ Because the source document does not contain the style sheet rules, different style sheets can be attached to the same document.
- ❑ A style sheet can import styles from other style sheet which makes modular development and good reusability.

The <LINK> Element: The ***<LINK>*** element creates an link to a CSS style sheets. It describes the relationships between two document(a CSS and HTML page).

Attribute	Purpose
rel	It specifies the relationships between the two documents e.g. : **REL="STYLESHEET"**
type	It specifies the MIME type of the document **TYPE="TEXT/CSS"**
href	It specifies the URL for the document being linked **HREF="../FerozOne/MyStyle.css"**

MyStyle.css

```
body {
        color : pink;
        background-image : D:/FerozOne/Images/rits.jpg;
        }
table{
background-color:#efefef;
border-style:solid;
border-width:2px;
border-color:#999999;
}
th {
background-color:#efefef;
font-weight:bold;
padding:5px;
}
td{padding:5px}
```

Link to the web page as follows

```
<HEAD>
  <LINK TYPE="TEXT/CSS" REL="STYLESHEET" HREF="D/FerozOne/MyStyle.css" />
</HEAD>
```

STYLE *CLASS* SELECTOR : Style classes is an extension of Internal Style sheet which carries the CSS rules and allows you to match with an element carrying a ***class*** attribute.

Note: A class can be created within <HEAD> element preceded by dot or full stop(.)

THE *ID* SELECTOR: The ***id*** selector works just like a ***class*** selector but with the value of ID attributes. Rather than using a period or full stop before the value of the ***id*** attribute, you just use a hash**(#)** sign.

```
<head>
<style type="text/css">
.large   ----------------------class selector
{
   font-size : 35pt;
   font-family : georgia;
   font-style : italic;
}
.background   ------------------ class  selector
{
    background-color:#66ccff;
    text-align:center;
 }
#timetable-------------------------------///id  selector
{
  background-color:#ffccff;
  text-align:center;
  padding:4px;
  width:60%;
  border-style:solid;
 }
 td{font-size:20pt}
}
 </style>
<head>
<body class="background">
 <p class="large">royal institute of technology and science</p>
<h2> this is under id selector....</h2>
 <table id="timetable">
 <tr>
          <th>name</th>
          <th>roll number</th>
          <th>branch</th>
          <th>percentage</th>
 </tr>
 <tr>
          <td>md feroz</td>
          <td>02661a1214</td>
          <td>csit</td>
          <td>74.01%</td>
 </tr>
 <tr>
          <td>md feroz</td>
```

```
        <td>02661a1214</td>
        <td>csit</td>
        <td>74.01%</td>
 </tr>
 <tr>
        <td>md feroz</td>
        <td>02661a1214</td>
        <td>csit</td>
        <td>74.01%</td>
 </tr>
 </table>
</body>
```

CSS PROPERTIES

1. Controlling Font

Property	Value
font-family	arial,verdana,sans-serif,courier,impact,georgia…
font-size	2%,2px,x-small,medium,3pt,x-large…..
font-weight	Normal,bold,bolder,lighter,100,200..
font-style	Normal,italic,oblique
font-stretch	Wider,narrower,condensed,extra-condensed,expanded..
font-variant	Small-caps

2. Text Formatting

Property	Value
Color	red,green,#ff0000……
text-align	left,right,center,justify
vertical-align	baseline,sub,super,top,middle,bottom….
text-decoration	underline,overline,line-through,blink
text-transform	none,capitalize,uppercase,lowercase
letter-spacing	10px…….. some value
word-spacing	10px….. …some value
white-spacing	normal,pre,nowrap
text-shadow	0.3em,0.5em black

Note :
px: A *pixel* is the smallest unit of resolution on a screen.
em:An *em* unit corresponds directly to the font-size of the reference element.
ex:The *ex* should be the height of a lower case ***x***.

3. Border Property

Property	Value
border-color	red,green,#ff0000……
border-style	none,solid,dottd,dashed,double,groove,inset,outset
border-width	4px--------some value.

4. Padding Property

Property	Value
Padding	4px,5px,22px….some value of padding

5. Margin Property

Property	Value
Margin	4px or margin-left,margin-right,margin-top/bottom.

6. Link Property

Pseudo-class	Purpose
Link	styles for links in general
Visited	styles for links that have already been visited
Active	styles for links that are currently active(clicked)
Hover	styles for when someone is hovering over a link.

7. Background Property

Property	Value
background-color	red,green,#FFFCCC,#CCCCCC..
background-image	url("FerozOne/images/rits.jpeg")
background-repeat	Repeat,repeat-x, repeat-y, no-repeat
background-attachment	Fixed,scroll
background-position	Left,right,xy,x% y%,top,bottom,center.

8. List Property

Property	Value
list-style-type	none,disc,circle,square,decimal,lower-alpha,lower-roman,upper-roman,decimal-leading-zero.
list-style-position	Inside,outside.
list-style-image	url("FerozOne/images/rits.jpeg")
marker-offset	2em.

9. Table Property

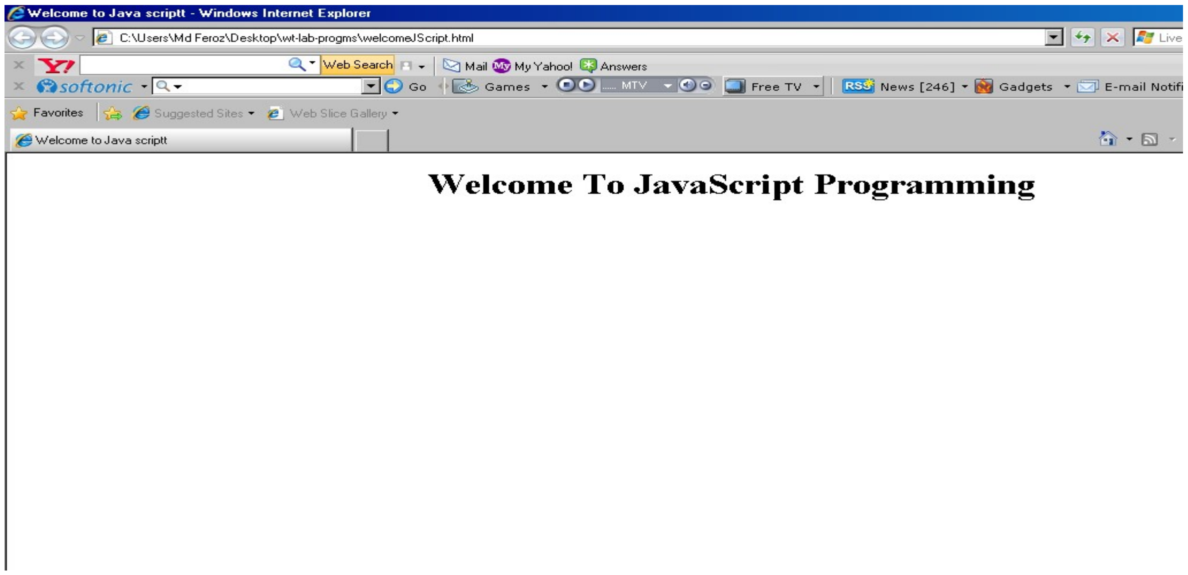
Property	Value
border-collapase	collapase, separate
border-spacing	12px,,,,
caption-side	left,right,top,bottom
empty-cell	show,hide,inherit
table-layout	fixed,auto,inherit.

INTRODUCTION TO JAVSCRIPT

JavaScript is also called as JScript. It is originally created by Netscape. Microsoft’s version of JavaScript is called as *JScript*. The Internet Explorer browser contains the “***JavaScript Interpreter***”, which processes the commands of a script written in JavaScript.

Definition: JavaScript is the most popular web scripting language in use today. It lets us to combine programs in our web pages and run these programs in a web browser. It allows us to design HTML programs that enhance the functionality and appearance of web pages.

```
<html>
  <head>
      <title>welcome to java script </title>
        <script language="JavaScript">
           document.writeln("<h1><center><b>Welcome To JavaScript Programming</h1>");
        </script>
   </head>
<body>
</body>
</html>
```



Welcome To JavaScript Programming

JavaScript Features

1. JavaScript is an object-based language that is confined to run within the web browsers only.
2. JavaScript is an interpreted language and requires no compilation steps.
3. JavaScript can directly be embedded in HTML files. The HTML files with embedded JavaScript commands interpreted by any browser that is JavaScript enabled.
4. JavaScript is a loosely typed language i.e. one data type can be automatically converted into other types without explicit conversion.
5. JavaScript is platform independent, but, browser dependent. The JavaScript applications work on any machine that has an appropriate JavaScript enabled browser installed. A JavaScript program developed on a Unix machine will work perfectly on a Windows machine.
6. JavaScript supports event-based programming.
7. Performance is good since the JavaScript programs are included in the same file as the HTML code for a Web Page, the down load time for the client is minimum.
8. JavaScript is multi functional i.e. it can be used at a client side scripting as well as server side also.

JavaScript Objects

JavaScript is an object-oriented language. Using objects makes the JavaScript programming very much easier. JavaScript comes with a number of predefined objects, such as the "documents" object which we had used in the above example. The "document" object refers to the body of the current page in the browser, giving us an easy way to access the HTML in that page. We used the "writeln" method of that object to write the text "welcome to JavaScript".

There are two important aspects of objects. They are :

1. Methods.
2. Properties.

We can use a method of an object by using a dot **(.)** followed by the method name, such as

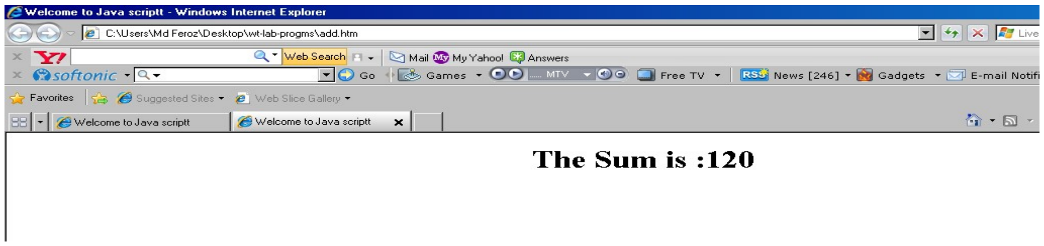
e.g.: **document.writeln ("Welcome");**

A '***property'*** holds some setting of an object. For example, the "*document.linkcolor*" property holds the color of unvisited hyperlinks in the current web page and by changing the "*document.linkcolor*" property, we can change that color.

DATA TYPE: A data type is used to defines the type of data a variable hold the value. JavaScript is a loosely typed language -- you do not have to specify the data type of a variable when you declare it, and data types are converted automatically as needed during script execution.

- **Numbers:** The numbers can be either positive or negative.
  e.g. : Any number, such as 17, 21, or 54e7
- **Booleans:** The possible Boolean values are true and false.
- **Strings:** Strings are a series of letters and numbers enclosed in quotation marks. JavaScript uses the string literally; it doesn't process it.
- **Null:** is an empty value. **Null** is not the same as 0 **--** 0 is a real, calculable number, whereas **null** is the **absence of any value**.
- **Objects :** myObj = new Object();
- **Undefined:** A value that is undefined is a value held by a variable after it has been created, but before a value has been assigned to it.

```
<script language="JavaScript">
    var first,second,n1,n2,sum;
    first=window.prompt("Enter the first number :","0");
    second=window.prompt("Enter the second number :","0");
    n1=parseInt(first);
    n2=parseInt(second);
    sum = n1+n2;
    document.writeln("<h1><center><b>The Sum is :"+sum+"</h1></center></b>");
</script>
```


Welcome to Java scriptt - Windows Internet Explorer
C:\Users\Md Feroz\Desktop\wt-lab-progms\add.htm
Welcome to Java scriptt

The Sum is :120

FUNCTIONS: In JavaScript functions are very important, which is piece of code mean to provide definite functionality to the whole program? We can write the code into a function and call that function to execute the code in it. Functions can also be called as ***modules***. The programmer can write functions to define specific tasks that may be used at many places in a ***script***. Functions are basically of two types

1. User defined or Programmer defined functions.
2. Predefined or Global or built-in functions

User defined Function: User-defined functions are those functions that are defined by the user/programmer specific to the application. Functions allow the programmer to divide the entire big program into smaller modules. There are several reasons why a big program needs to be divided into smaller modules.

1. This approach makes the program development more **manageable.**
2. **Reusability** using existing functions to create new programs.
3. It avoids repetition of the code in the program.

These functions can also be referred as "programmer-defined functions". The syntax for defining a function is:

```
function   function_name  [argument list]
  {
     code
  }
```

To return a value from the function, we use the "***return***" statement. A function is invoked (or called) by a function call. The function call specifies the function name and provides information as arguments that the called function needs to do its task.

Write a JavaScript program that change the background color dynamically (DHTML)

```
function changeColor()
  {
            var color=document.getElementById("colors").value;
            if(color=="red"){
            document.bgColor=color;}
            else if(color=="green"){
            document.bgColor=color;}
            else if(color=="blue"){
            document.bgColor=color;}
            else{
            var color=prompt("Enter Your Favourite Color :","cyan");
            document.bgColor=color;
  }
```

Predefined function: These functions are built-in function which has some predefined meanings which provide many functionalities to our program

1.isFinite(): It takes numeric value as an argument and returns ***true*** only if the given number is **finite** numeric. Otherwise ***false.***

2.isNaN() : It takes an argument and returns a ***true*** status only if the argument is not a number else ***false.***

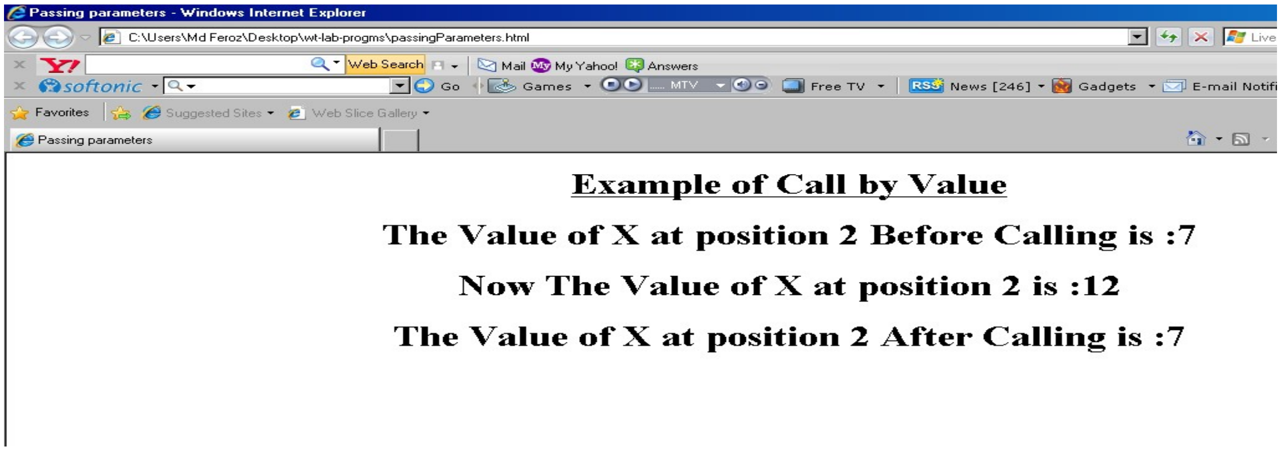
3.parseInt(): It accept a string and converts into its equivalent numeric.

******Passing an arguments to an function******

There are two ways to pass an arguments into a functions

1. **Call by Value and 2. Call by reference.**
1. **Call by value:** In call by value, a copy of a value is passed to a called function such that any modification done by the called function will not reflect to the calling function.

```
<script language="JavaScript">
document.writeln("<center><h1><u>Example of Call by Value</u>");
  function change(x)
  {
      x=12;
     document.writeln(" <h1> Now The Value of X  at position 2 is :"+x);
  }
  function show()
  {
    var x=[2,4,7,9];
    document.writeln("<h1>The Value of X at position 2  Before Calling is :"+x[2]);
    change(x[2]);
    document.writeln("<h1>The Value of X at position 2 After Calling is :"+x[2]);
  }
  show();
</script>
```


Passing parameters - Windows Internet Explorer
C:\Users\Md Feroz\Desktop\wt-lab-progms\passingParameters.html
Favorites
Passing parameters

Example of Call by Value

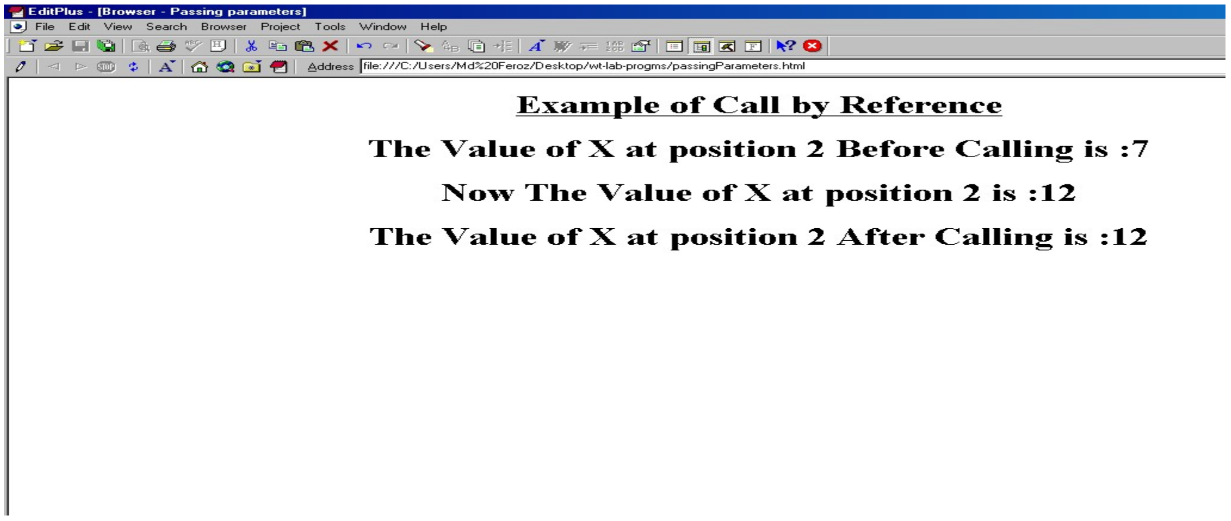
The Value of X at position 2 Before Calling is :7

Now The Value of X at position 2 is :12

The Value of X at position 2 After Calling is :7

2. **Call by Reference :** In call by reference an array is passed as an argument to a function. Therefore changes made to this array will be permanent and it will be reflected in the entire program. Passing an array as parameters is nothing but passing of address. It is also called **"pass by reference".**

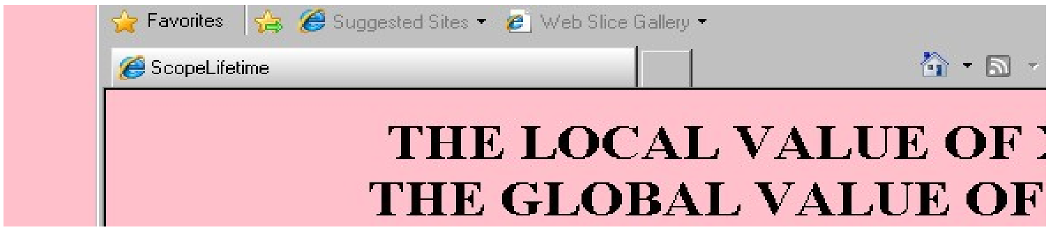
```
<script language="JavaScript">
document.writeln("<center><h1><u>Example of Call by Reference</u>");
  function change(x)
  {
     x[2]=12;
     document.writeln(" <h1> Now The Value of X  at position 2 is :"+x[2]);
  }
  function show()
  {
     var x=[2,4,7,9];
     document.writeln("<h1>The Value of X at position 2  Before Calling is :"+x[2]);
     change(x);
     document.writeln("<h1>The Value of X at position 2 After Calling is :"+x[2]);
  }
  show();
</script>
```


EditPlus - [Browser - Passing parameters]
File Edit View Search Browser Project Tools Window Help
Address file:///C:/Users/Md%20Feroz/Desktop/wt-lab-progms/passingParameters.html

```
Example of Call by Reference
The Value of X at position 2 Before Calling is :7
Now The Value of X at position 2 is :12
The Value of X at position 2 After Calling is :12
```

Scope and Lifetime of variables: An identifier's duration(lifetime) is the period during which it exists in the memory. The scope of the identifiers for a variable or function is a portion of the program in which the identifier can be referenced. The scope for an identifier are global scope and for function local scope.

```
<script>
   var x=10;
 function demo()
 {
   var x=20;
   document.writeln("THE LOCAL VALUE OF X :-" + x);
 }
   demo();
   document.writeln("<br>THE GLOBAL VALUE OF X :-" + x);
</script>
```



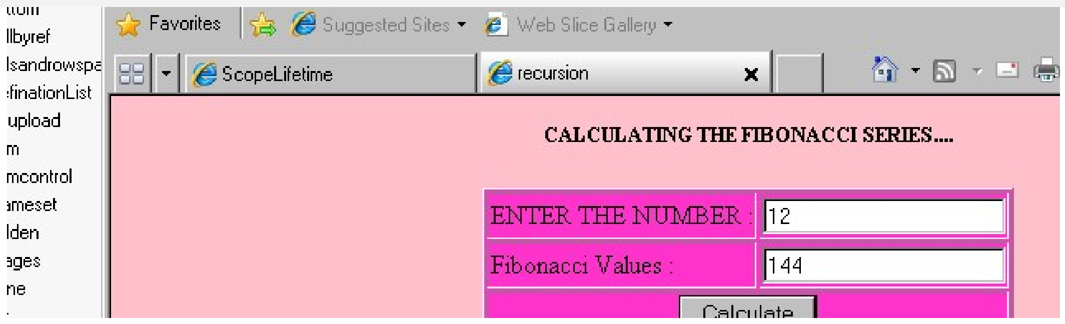
```
THE LOCAL VALUE OF
THE GLOBAL VALUE OF
```

Recursion: A function that calls itself is known as recursion. The function is called recursive function.

Write a script to calculate the Fibonacci series

```
<head>
```

```
<title> recursion </title>
<script>
  function getFibonacci()
  {
     var value= parseInt(document.myform.number.value);
     window. status = "Calculating  Fibonacci  value for "+ value;
     document.myform.result.value=fibonacci(value);
     window.status = "Calculation Done";
  }
  function fibonacci( n )
  {
     if(n==0 || n==1 )
      return n;
     else
     return fibonacci(n-1)+fibonacci(n-2);
  }
</script>
</head>
<body bgcolor="pink">
<form name="myform">
  <table border="1" bgcolor="#ff33cc" align="center">
  <caption><h5>calculating the fibonacci series....</h5></caption>
  <tr>
     <td>enter the number :</td>
     <td><input type="text" name="number"/></td>
</tr>
<tr>
     <td>fibonacci values :</td>
    <td><input type="text" name="result"/></td>
</tr>
<tr>
<td colspan="2" align="center"><input type="button" value = "calculate"
onclick="getFibonacci()"/></td>
<table>
</form>
</body>
```



CALCULATING THE FIBONACCI SERIES....

ENTER THE NUMBER :	12
Fibonacci Values :	144

ARRAYS : In JavaScript, an array is an built-in object which can be created using ***new*** keyword. An array is a collection of contiguous memory address locations which shares a common name. An array starts with **'0'** index and ends with **size-1**

var name = new Array(20);

Where name=array_name, new=keyword(an operator which allocates memory),Array=object,size=20

Write a JS to calculate the smallest and largest of given numbers.

```
<script>
var i=0,min,j,temp,max;
var num = new Array();
for(i=0;i<5;i++)
{
     var n=window.prompt("Enter the number  : "+(i+1)+"","0");
      num[i]=parseInt(n);
 }
 max=num[0];
 min=num[0];
 for(i=1;i<num.length;i++)
  {
     if(num[i]>max)
        max=num[i];
     else
    if(num[i]<min)
        min=num[i];
    }
document.writeln("<CENTER><hr><b>Entered Numbers :");
 for(i=0;i<num.length;i++)
       {
            document.write("<b> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;"+num[i]);
       }
 document.writeln("<BR><B>MAXIMUM : "+max+"<BR>MINIMUM </u>  : "+min);
 document.writeln("<hr>");
</script>
<body bgcolor="#A9D1E0">
</body>
```

ARRAY METHODS

1.push() : It is method to insert data into an array.

e.g : var student = new Array("Feroz","Fasi","Afzal");

student**.**push("Atif");

output : Feroz,Fasi,Afzal,Atif size=4.

2.pop() : This method is used to remove the elements from an array.

e.g : student.pop();

output : Feroz,Fasi,Afzal size=3. removes-Atif

3.sort() : It sorts or arrange the elements of a given array in ascending order

e.g: student.sort();

output : Afzal,Fasi,Feroz

4.reverse() : It reverse the elements of an Array.

e.g : student.reverse();

output : Afzal,Fasi,Feroz.

5.join() : It is used to join all the elements by the separator between them

e.g :student.join("\");

output : Feroz/Fasi/Afzal.

MATH OBJECT: JavaScript provides an Math object which consists several methods to perform some mathematical functions to our program.These functions can be accessed by using dot(.) operator

Methods Name	Description	Example	Output
min(n1,n2)	It displays minimum of two numbers.	Math.min(3,5)	3
max(n1,n2)	It displays maximum of two numbers.	Math.max(3,5)	5
abs(n)	It displays the absolute value of the number entered.	Math.abs(5)	5
ceil(n)	It displays the rounded value of the integer entered.	Math.ceil(5.4)	6
pow(n1,n2)	It calculate the power of n1 over the n2	Math.pow(2,3)	8
round(n)	It round the value to its nearest integer	Math.round(5.5)	6
floor(n)	It rounds to the largest integer	Math.floor(5.2)	5.0
sqrt(n)	It displays the square root of entered number	Math.sqrt(9)	3
sin(n)	It displays the trigonometric sine value of a number	Math.sin(90)	1
cos(n)	It displays the trigonometric cosine value of a number	Math.cos(0)	1
tan(n)	It displays the trigonometric tangent value of a number	Math.tan(45)	1
exp(n)	It displays the exponential(e^x) value entered	Math.exp(2)	7.38
log(n)	It displays the logarithmic ($\log_x$) value entered	Math.log(2)	0.639

STRING OBJECT: A string is a series of characters. A string may include letters, digits and special characters like +, -, *, /,&,and others. A string is an object of type '**string**'. String literals (constants) are written as a sequence of characters in double quotation marks or single quotation marks as follows.

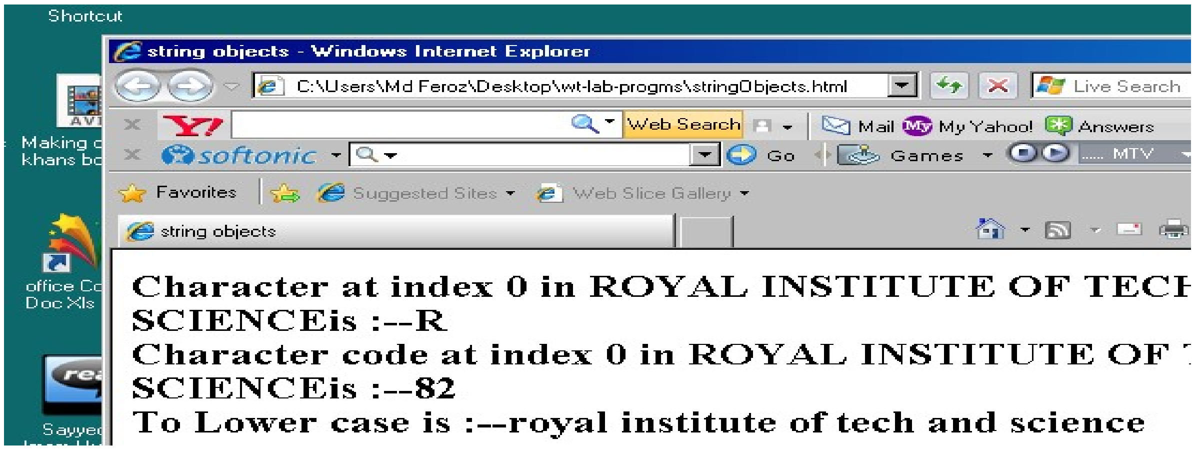
var s1 = "FEROZ"; or s2="MD"; s3="javascript"
var address = "H-No :8-77,Azampura";

Methods Name	Description	Example	Output
charat(index)	It returns the character at specified ***index*** value	s1.charAt(2)	R
concat(string)	It concat two strings.	s2.concat(s1)	md feroz
indexof (character)	It returns the ***index*** of the character specified	s1.index('R')	2

lastindex(character)	It returns the ***index*** of last occurrence of the character specified in a String.	s1.lastIndex('o')	3
slice(start,end)	It returns the start and end specified characters.	s1.slice(1,3)	ERO
split(character)	It returns the string at specified split character	s1.split('R')	FE,OZ
substr(start,length)	It returns the substring of arguments specified	s1.substring(0,3)	FERO
tolowercase()	It converts the upper case string to upper case	s1.toLowerCase()	Feroz
touppercase()	It converts the lower case string to upper case	s3.toUpperCase()	JAVASCRIPT
charCodeAt(index)	Returns the Unicode value of the character		
fromCharCode(value1,,,,)	Converts the list of Unicode values to a string		

Write a script to demonstrate string objects methods

```
<script>
  var s1 = "ROYAL INSTITUTE OF TECH AND SCIENCE";
  var  s2 = "Welcome to JavaScript";
  document.writeln("<H2>");
  document.writeln("Character at index 0 in " +s1+ "is :--" +s1.charAt(0));
  document.writeln("<br>Character  code at index 0 in " +s1+ "is :--" +s1.charCodeAt(0));
  document.writeln("<br>To Lower case is :--" +s1.toLowerCase());
  document.writeln("<br>Concat of s1 and s2 is :--" +s1.concat(s2));
  document.writeln("<br>substring of s2 :--" +s2.substring(3,7));
</script>
```


string objects - Windows Internet Explorer

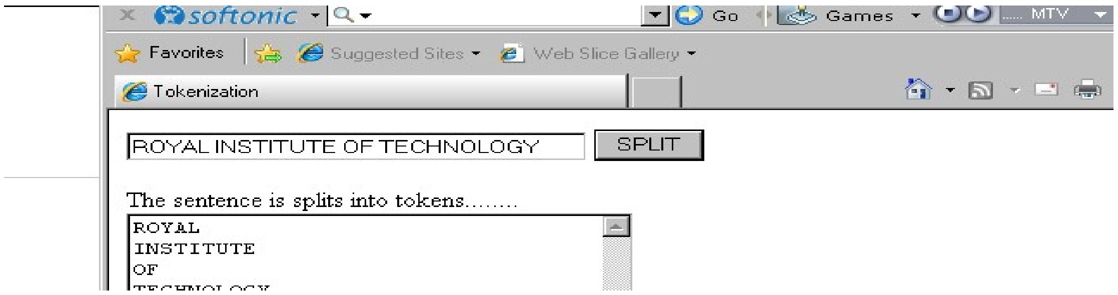
C:\Users\Md Feroz\Desktop\wt-lab-progms\stringObjects.html

```
Character at index 0 in ROYAL INSTITUTE OF TECH
SCIENCEis :--R
Character code at index 0 in ROYAL INSTITUTE OF
SCIENCEis :--82
To Lower case is :--royal institute of tech and science
```

String Tokenization: The process of breaking the string into a small unit of characters called ***token*** called Tokenization.

```
<html>
<head>
<title>Tokenization</title>
<script language="javascript">
 function tokenization()
 {
     var str = myform.inputVal.value.split(" ");
```

```
    myform.output.value=str.join("\n");
 }
</script>
</head>
<body>
  <form name="myform">
   <p>
  <input type="text" name="inputval" size="40"/>
  <input type="button" name="splitbutton" value=" split " onclick="tokenization()"/>
   </p>
<p>
the sentence is splits into tokens........
    <br>
 <textarea name="output" rows="8" cols="34"></textarea>
 </p>
 </form>
</body>
</html>
```



ROYAL INSTITUTE OF TECHNOLOGY

The sentence is splits into tokens........

```
ROYAL
INSTITUTE
OF
TECHNOLOGY
```

DATE OBJECT: In JavaScript, Date is an object which provides methods for date and time processing.It can be performed based on the local time zone or based on UTC(Universal Time) ,GMT(Green Mean Time). Date object can be created as follows:

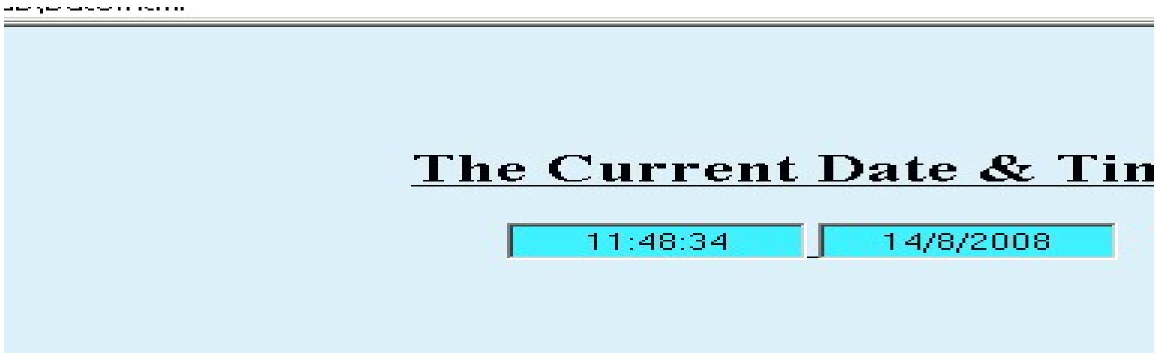
var date = new Date();

Methods Name	**Description**
getDate()	Returns a number from 1 to 31 representing the day of the month
getUTCDate()	Returns a number from 1 to 31 representing the day of the month
getDay()	Returns the number from 0(Sunday) to 6(Saturday)
getFullYear()	Returns the year as four-digit number
getHours()	Returns the number from 0 to 23 representing hour.

getMilliseconds()	Returns the number from 0 to 999 representing milliseconds.
getMonth()	Returns the number from 0(january) to 11(december) representing month.
getMinutes()	Returns the number from 0 to 59 representing minutes.
getSeconds()	Returns the number from 0 to 59 representing hour.
setDate()	Set the day of the month(1 to 31)
setFullYear()	Sets the full year in local time
setHours()	Sets the hours in local time
setMilliseconds()	Sets the number of milliseconds in local time
setMinutes()	Sets the minutes in local time
setSeconds()	Sets the seconds in local time
getTime()	Returns the number of milliseconds between jan 1,1970 and the time in the Date object.
toString()	Returns a string representation of the date and time locale to the system
toLocaleString()	Returns a string representation of the date and time locale to the system
toUTString()	Returns a string representation of the date and time. in the form of 28 July,2009 20:17:55
valueOf()	The time in numbers of milliseconds

```
<html>
<head>
<title>Date & Time </title>
<style>
#clockcss
{
background-color:#3ff0ff;
width:100px;
text-align:center;
}
</style>
<script>
function setClock()
{
        var hour,minutes,seconds;
        date=new Date();
        hour=date.getHours();
        minutes=date.getMinutes();
        seconds=date.getSeconds();
        var time=hour+":"+minutes+":"+seconds;
        var dd=date.getDate();
        var mm=date.getMonth()+1;
        var yy=date.getYear();
        var today=dd+"/"+mm+"/"+yy;
        //alert("Time is "+hour+":"+minutes+":"+seconds);
        document.clocktext.time.value = time;
        document.clocktext.date.value = today;
        setTimeout("setClock()");
}
```

```
</script>
</head>
<body onload = "setclock()" bgcolor="#dcf0f9">
<form name = "clocktext"><center><br><br><br>
<h2><u><b>The current Date & Time</h2>
<input type="text" name="time" value="" size=8 id="clockcss">
<input type="text" name="date" value="" size=8 id="clockcss">
</form>
</body>
</html>
```



The Current Date & Tin

```
11:48:34    14/8/2008
```

EXCEPTION HANDLING : Exception handling is one of the prominent feature of object-oriented programming. In JavaScript, an exception is an abnormal event which occurs during the program execution. i.e it refers to run-time errors. An exception causes the program halt and gives the wrong result.In order to handle the exception, we usually use the exception handlers i.e **try….catch**.

```
<script language="javascript">
 var a=0;
 var b=1;
 try
 {
   document.writeln(b/a)
 }
 catch(ArithmeticException)
 {
  document.writeln("Division-by-Zero leads to Infinity");
 }
</script>
```

Explanation : In the above script, the try block contains (b/a) which causes an exception which is caught by catch block.

throw keyword : The ***throw*** is a keyword which is used to throw an exception explicitly by the user specific to his application.

syntax : throw throwableInstance

```
<script language="javascript">
 var marks=prompt("Enter the marks (0 to 100)","45");
 try
 {
   if(marks<0)
   throw "value1";
   else
   if(marks>100)
   throw "value2";
 }
 catch(e)
 {
   if(e=="value1")
   alert("You Entered less than 0 marks : " + marks);
   else
   if(e=="value2")
   alert("You Entered more than 100 marks : " + marks);
   else
   alert("You Entered  marks : " + marks);
   }
</script>
```

Dynamic HTML (DHTML): Dynamic HTML (DHTML) is a set of innovative features originally introduced in Microsoft Internet Explorer 4.0. It enable us to dynamically change the rendering and content of a Web page as the user interacts with it, DHTML enables users to create visually compelling Web sites without the overhead of server-side programs or complicated sets of controls to achieve special effects. With DHTML, you can easily add effects to your pages that previously were difficult to achieve. For example

- You can hide content until a given time elapses or the user interacts with the page.
- Animate text and images in your document, independently moving each element from any starting point to any ending point, following a predetermined path or one chosen by the user.
- Embed a ticker that automatically refreshes its content with the latest news, stock quotes, or other data.
- Use a form to capture user input, and then instantly process and respond to that data.

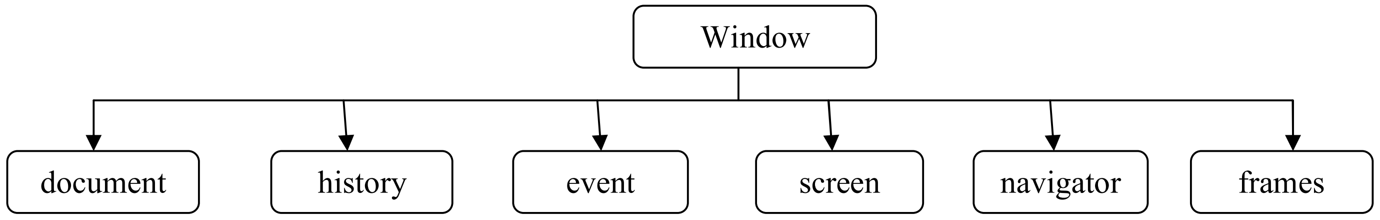
DHTML achieves these effects by modifying the in-memory representation of the current document and automatically reformatting it to show changes. It does not reload the document, or load a new document, or require a distant server to generate new content. Instead, it uses the user's computer to calculate and carry out changes. This means a user does not wait for text and data to complete time-consuming roundtrips to and from a server before seeing the results. Furthermore, DHTML does not require additional support from applications or embedded controls to make changes. Typically, DHTML documents are self-contained, using styles and a script to process user input and directly manipulate the HTML elements, attributes, styles, and text of the document.

In short, DHTML eliminates the shortcomings of static pages. You can create innovative Web sites, on the Internet or on an intranet, without having to sacrifice performance for interactivity. Not only does DHTML enhance the user's perception of your documents, it also improves server performance by reducing requests to the server.

HTML	**DHTML**
HTML is used to create static web pages	DHTML is used to create dynamic web pages
It consists of simple HTML tags	It consist of HTML+CSS+Javascripts
It does not alter the text and graphics on web pages unless web page gets changed	It does alter the text and graphics on web pages unless web page gets changed
It is very simple to create but less interactive	It is complex to create but more attractive

JAVASCRIPT OBJECT MODEL (BOM) AND COLLECTION

The central idea of object model is to create the objects of some HTML elements, present them on a webpage and retrieving the properties of the HTML elements to change the position on a web page dynamically. The object reference can be created using ***ID*** attribute and using the ***innerText*** property we can access the corresponding HTML element.



1. **document:** It corresponds to the current web page's body. Using this object, we can access to the HTML of the page itself, including all the links, images, forms and anchors in it.

   Examples:

   - document.write : Write text to the current web pages.
   - document.writeln : Write text to the current web page and adds a carriage return.
   - document.bgcolor : Background color of the current page.
   - document.fgcolor : Foreground color of the current page.
   - document.lastmodied : Date when the page last modified.
   - document.title : Title of the current page.

2. **form :** It holds information about HTML forms in the current page; forms can contain buttons, text fields and all kinds of other HTML elements.
3. **history :** Holds the record of the sites, which the web browsers has visited before reaching the current page. It gives us access to methods that help us in moving back to previous page.

   Example s:

   - history.go()

4. **location :** It holds information about the location of the current Web Page, such as its URL,the domain name, path, server port, etc.

   Examples:

   - location.hostname : Name of the Internet Service Provider (ISP) host.

5. **frame :** Refers to a frame in the browser window.
6. **navigator :** Refers to the browser itself, letting us determine what browser the user has.

   Examples:

   - navigator.appName: Name of the browser, which we can see to determine, what browser the user has.

7. **window:** Refers to the current window.

   Examples:

   - window.alert : Displays an alert dialog box.

- window. open : Opens a new browser window.
- window.prompt : Displays a prompt dialog.

EVENT MODEL : Events are mechanism by which browsers respond to user actions. Every element on a web page has certain events which can trigger invocation of event handlers. Attributes are inserted into HTML tags to define events and event handlers.

Example : Submitting an HTML form, moving mouse a browser window, clicking the mouse button will generate an event informing the browser that an action has occurred and that further relevant processing is required. The browser waits for event to occur, and it performs actions to those events. The processing that is performed in response to the occurrence of an event is known as **event handling.** The code that performs this processing is called an **event handler.**

HTML event handlers can be divided into two types

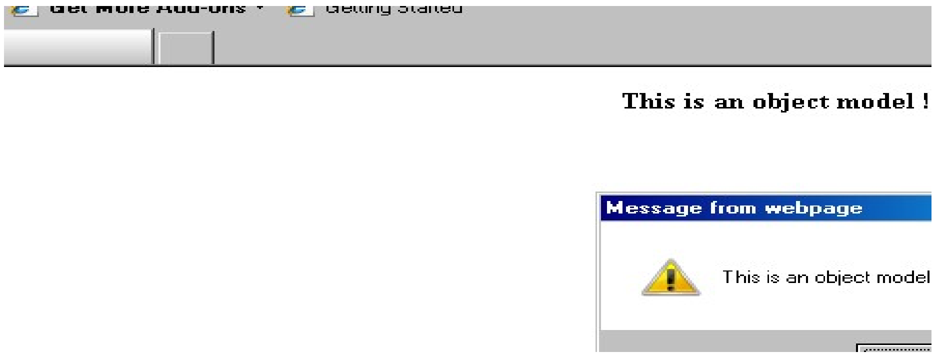
1. **Interactive :** An interactive event handler depends on the user interaction with an HTML page.
2. **Non-Interactive :** Non-Interactive event handler does not need user interaction.

Event Handlers	Description
OnClick	This event will get fired when the user clicks the mouse.
onLoad	This event is generated when the page is loaded initially. It used in body element.
onUnLoad	This event is generated when the page is closed. It used in body element.
onError	This event is generated when an error dialog is displayed by the JavaScript code.
OnMouseMove	This event gets fired repeatedly when the user moves the mouse over the web page.
onMouseOver	This event gets fired when the user moves the mouse over the page element.
onMouseOut	This event gets fired when the user moves out the mouse cursor over the element of page.
onFocus	It invokes when an element gains focus.
onBlur	It occurs when an element looses focus.
onSubmit	It gets fired/execute when the user click on the submit button in <FORM> element.
onReset	It generates when the user clicks on the Reset button, an alert dialog is generated.
onChange	It gets fired whenever the data gets changed(textfield,or textarea)
onAbort	Invokes whenever any of the processes gets aborted or halted.
onDbClick	It gets fired when the user clicked twice.
onDragDrop	It is generated when there is an drag and drop event occurred.
onKeyUp	Invokes as soon as the user releases the key
onKeyDown	Invokes as soon as the user presses the key
onResize	Invokes as soon as the user resize the browser window.
onSelect	Invokes whenever certain text of a page element is selected.

DHTML Program

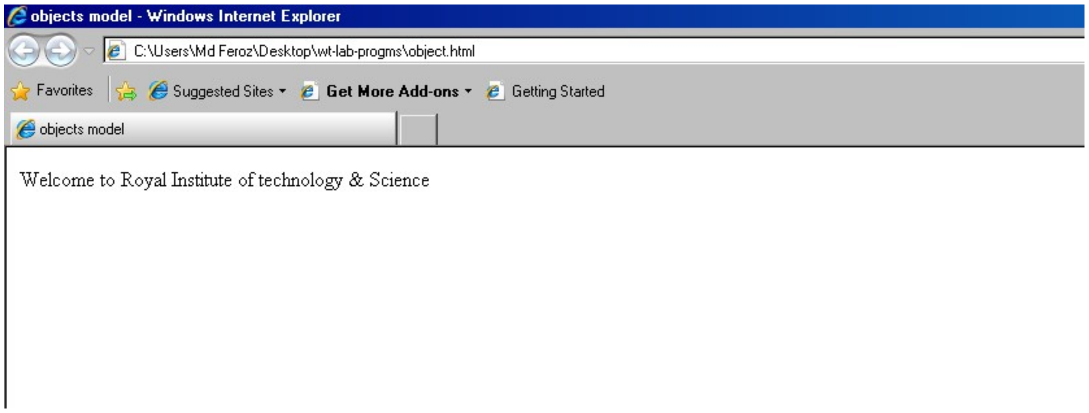
```
 <html>
<head>
```

```
<title>
<script=”javascript”>
function show()
{
   alert(pmsg.innerText);
   pmsg.innerText=” Welcome to Royal Institute of technology & Science”;
}
</script>
</head>
<body onload=” show()”>
<p id =”pmsg”> This is an object model !!!!</p>
</body>
</html>
```



```
This is an object model !
```

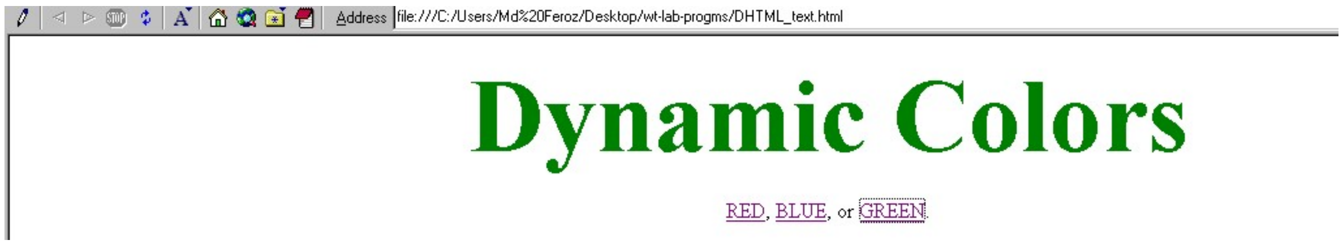
```
This is an object model
```


objects model - Windows Internet Explorer
C:\Users\Md Feroz\Desktop\wt-lab-progms\object.html
Favorites
Suggested Sites
Get More Add-ons
Getting Started
objects model

```
Welcome to Royal Institute of technology & Science
```

DHTML program

```
<html>
<head>
   <title>Dynamic Colors</title>
<script language="JavaScript">
function changeColor(c)
{
  h=document.getElementById ("head1");
  h.style.color=c;
}
</script>
</head><body><CENTER>
<h1 style="color: black; font-size:5em" id="head1">Dynamic Colors</h1>
<a href="javascript: changeColor('red');">RED</a>,
<a href="javascript:changeColor('blue');">BLUE</a>,
or <a href="javascript:changeColor('green');">GREEN</a>.
</body>
</html>
```


Address
file:///C:/Users/Md%20Feroz/Desktop/wt-lab-progms/DHTML_text.html

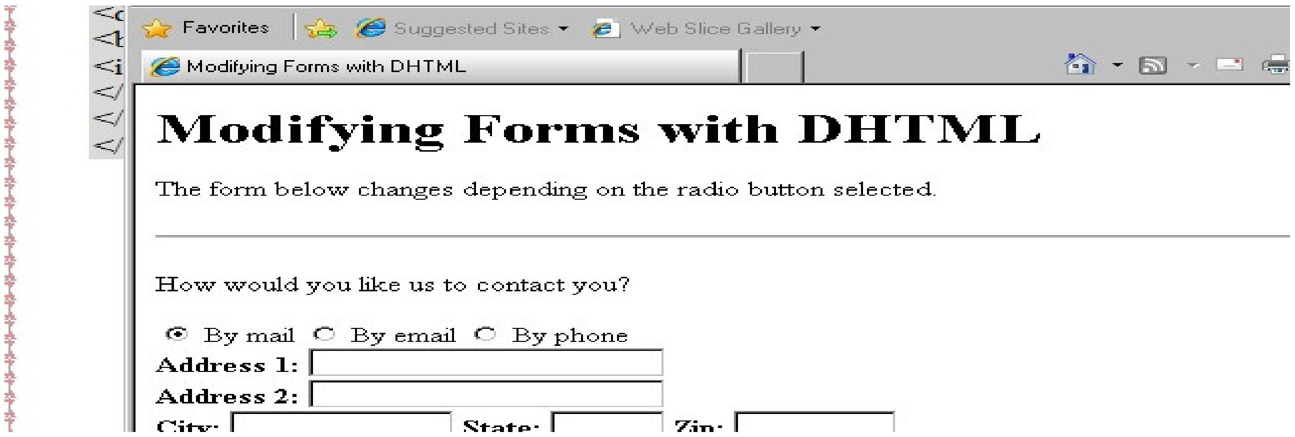
Dynamic Colors

RED, BLUE, or GREEN

DHTML Form

```
<html>
<head>
<title>Modifying Forms with DHTML</title>
<script language="javascript">
function Display(which) {
  ma=document.getElementById("mail");
  em=document.getElementById("email");
  ph=document.getElementById("phone");
  if (which=="mail") ma.style.display="block";
    else ma.style.display="none";
  if (which=="email") em.style.display="block";
    else em.style.display="none";
  if (which=="phone") ph.style.display="block";
    else ph.style.display="none";
}
</script>
</head><body>
<h1>Modifying Forms with DHTML</h1>
<p>The form below changes depending on the radio button
selected.</p>
<hr>
<form name="form1">
<p>How would you like us to contact you?</p>
<input name="type" value="mail" checked="checked" onclick="Display('mail');" type="radio">
 By mail
<input name="type" value="email" onclick="Display('email');" type="radio">
 By email
```

```
<input name="type" value="phone" onclick="Display('phone');" type="radio">
 By phone
<br>
<div id="mail" style="display: none;">
<b>Address 1:</b> <input name="address1" size="25" type="text">
<br>
<b>Address 2:</b> <input name="address2" size="25" type="text">
<br>
<b>City:</b> <input name="city" size="14" type="text">
<b>State:</b> <input name="state" size="5" type="text">
<b>Zip:</b> <input name="zip" size="9" type="text">
</div>
<div id="email" style="display: none;">
<b>Email address:</b>
<input name="email" size="25" type="text">
</div>
<div id="phone" style="display: block;">
<b>Phone:</b>
<input name="phone" size="15" type="text">
</div>
</form>
</body>
</html>
```



Modifying Forms with DHTML

The form below changes depending on the radio button selected.

How would you like us to contact you?

[✓ By mail] By email By phone

Address 1:

Address 2:

City: State: Zip:

DHTML Images

```
<html>
<head>
<script type="text/javascript">
cc=0;
function changeimage()
{
if (cc==0)
  {
```

```
  cc=1;
  document.getElementById('myimage').src="ferozOne.jpg";
  }
else
  {
  cc=0;
  document.getElementById('myimage').src="feroz.jpg";
  }
}
</script>
</head>
<body>
<img id="myimage" onclick="changeimage()" border="0" src="feroz.jpg" width="100" height="180" />
<p>Click to turn on/off the light</p>
</body>
</html>
```

DHTML Mouse Events

```
<html>
<head>
<script type=”text/javascript”>
function  one()
{
  myimg.src=”D:\ferozone\flower1.jpg”;
}
function two()
{
  myimg.src=”D:\ferozone\flower2.jpg”;
}
</script>
</head>
<body>
<center>
 <img id =”myimg” onmousedown=”one()” onmouseup=”two()”  src=”D:\ferozone\flower2.jpg”
height=”200” width=”200”/>
<p> Click this image to change it </p>
</center>
</body>
</html>
```

DHTML Styles

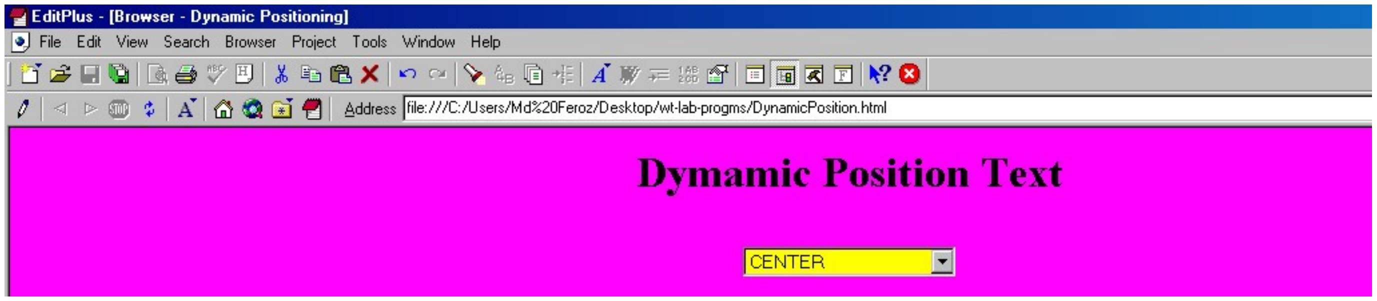
The Style of an HTML element on the web page can be changed dynamically when the user interacts through events.

```
<HTML>
<HEAD>
<TITLE>Dynamic Styles </TITLE>
<SCRIPT>
  function start()
  {
          var text = prompt("Enter your name","");
            para.innerHTML = "Welcome -<u><i> " + text;
  }
</SCRIPT>
</HEAD>
<BODY onload="start()" BGCOLOR=”cyan”>
   <CENTER><h1 id="para"></h1>
</BODY>
</HTML>
```

DHTML Positioning: HTML elements can be positioned dynamically on a web page by using script.

```
<HTML>
<HEAD>
<TITLE>Dynamic Positioning </TITLE>
<STYLE>
  .left  { text-align:left;  }
  .right { text-align:right; }
  .middle { text-align:center;  }
</STYLE>
<SCRIPT>
  function start()
  {
var position = document.getElementById("position").value;
    para.className=position;
  }
</SCRIPT>
</HEAD>
<BODY  BGCOLOR="magenta">
<h1 id="para">Dymamic Position Text</h1>
<br>
<center><select id="position" onchange="start()">
<option value="">SELET ALIGNMENT...</option>
<option value="left">LEFT</option>
<option value="right">RIGHT</option>
<option value="middle">CENTER</option>
```

```
</select>
</BODY>
</HTML>
```


EditPlus - [Browser - Dynamic Positioning]
File Edit View Search Browser Project Tools Window Help
Address file:///C:/Users/Md%20Feroz/Desktop/wt-lab-progms/DynamicPosition.html
Dymamic Position Text
CENTER

FILTERS AND TRANSITIONS

Filters give a great variety of visual effects in the web pages dynamically and transitions gives vertical and horizontal blinds effects etc. we can also convert colored images to gray in response to user actions and we can also create a shadows to a text to appears like three dimensional view. *Filters* and *Transition* are **STYLE** object properties.

Advantages:

1. To achieve special effects.
2. To be able to create animated visual transitions between web pages.
3. To be able to modify the filters dynamically using DHTML.

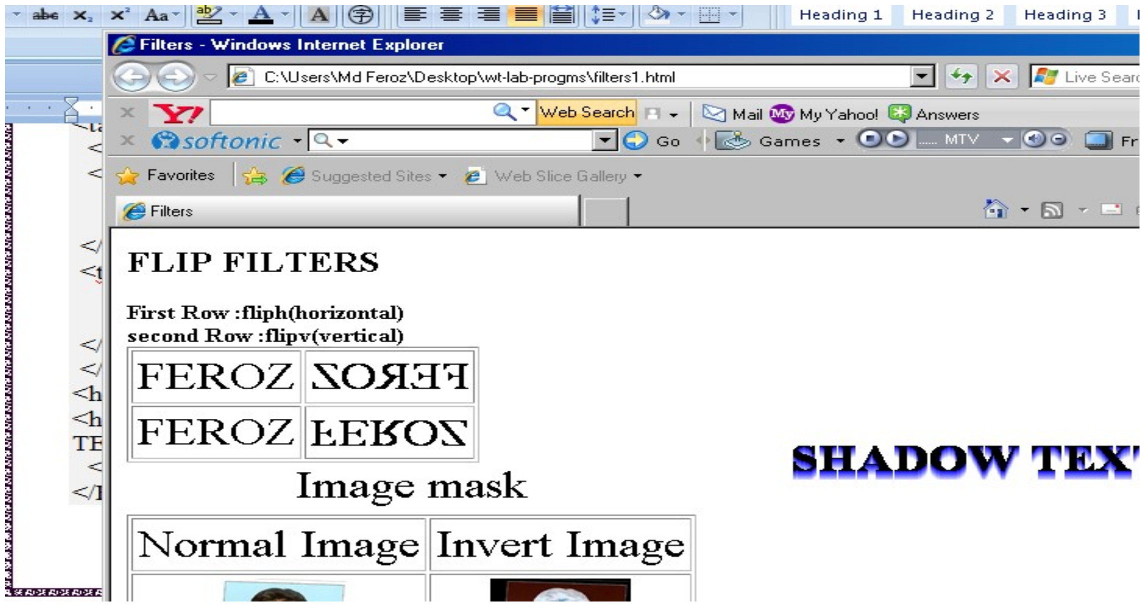
1. **Flip Filters : *filph*, *flipv*, Transparency with the *chroma* filter:-** To get the mirror effects on text or images we have to use flip filters such as filph(horizontal),flipv(vertical).Filters are applied in STYLE attribute.

```
<TD STYLE=”filter:fliph”>MD FEROZ<TD> Flip the text horizontally.
```

2. **Image mask and Image Filters** : *invert, gray and Xray* : Image mask filter is used to create to mask the image. And image filters invert, gray and Xray are used to filters the images as negative effect, gray scale image and inversion of gray scale image respectively.
3. **Adding shadow to text :** The shadow filter effects the text shadow around the text which gives the 3-dimensional appearance.

```
<HTML>
<HEAD>
```

```
<TITLE> Filters </TITLE>
<STYLE>
  table {
                      font-size:2em;
                         border-style:groove;
                         text-align:center
           }
  td{padding:0.1em;}
  </STYLE>
</HEAD>
<BODY>
 <h2>FLIP FILTERS</h2>
 <b>First Row :fliph(horizontal)</b><br>
  <b>second Row :flipv(vertical)</b><br>
  <TABLE border="1">
     <tr>
       <td>FEROZ</td>
        <td STYLE="filter:fliph">FEROZ</td>
      </tr>
      <tr>
        <td> FEROZ</td>
        <td STYLE="filter:flipv">FEROZ</td>
       </tr>
 </table>
<table border="1">
  <caption>Image mask</caption>
  <tr>
     <td>Normal Image</td>
     <td>Invert Image</td>
 </tr>
 <tr>
     <td><img  src="feroz.jpg" width="90" height="90"/></td>
     <td><img src="feroz.jpg" width="90" height="90" style="filter:invert"/></td>
 </tr>
 </table>
<h2>Adding shadow effects to the text</h2>
<h1 style=" position:absolute; top:40%; left:50%; filter:shadow(direction=180,color=blue)"> SHADOW
TEXT EXAMPLE</h1>
  </BODY>
</HTML>
```


Filters - Windows Internet Explorer
C:\Users\Md Feroz\Desktop\wt-lab-progms\filters1.html
Filters

FLIP FILTERS

First Row :fliph(horizontal)
second Row :flipv(vertical)

FEROZ	FEROZ
FEROZ	FEROZ

Image mask

Normal Image	Invert Image

SHADOW TEX

Object Models & Collections in DHTML: DHTML is the combination of HTML, CSS, a scripting language and the DOM.

Object Model : An element can be referred by using its ID attribute. Each element in the web page is an object and its attributes becomes properties.

i.e. <INPUT TYPE="text" ID="name"> here, the id=name is used to refer the text value.

Collection :- Collections are basically an array of objects on a web page. There are several collections in the object model like all,children,forms,images etc.

1. **Collection All: The** DHTML object model includes a special collection called ***'all'***, which is a collection of all HTML elements in a document in order in which appear.

   Example: HTML-HEAD-TITLE-SCRIPT-BODY-P

DHTML program

```
<html>
<head>
<title>collection </title>
<script type="text/javascript">
var wpe="";
function show()
{
for(i=0;i<document.all.length;i++)
wpe = wpe + "<br>"+document.all[i].tagName;
pmsg.innerHTML+=wpe;
}
```

```
</script>
</head>
<body onload="show()">
<p id ="pmsg"><CENTER><strong>Various tags in this web page</strong></center>
</body>
</html>
```

2. **Collection Children: -** It is similar to 'all' except for which a specific element contains that element's child element.for example, a HTML element has two child element ***head*** and ***body***.

DATA BINDING: In DHTML Data binding is a process of binding the data for the databases. i.e we can load the data from the server machine into a database files stored at client machine, later on, we can write a DHTML program from which the database files can be invoked and the data can be presented on a web page.

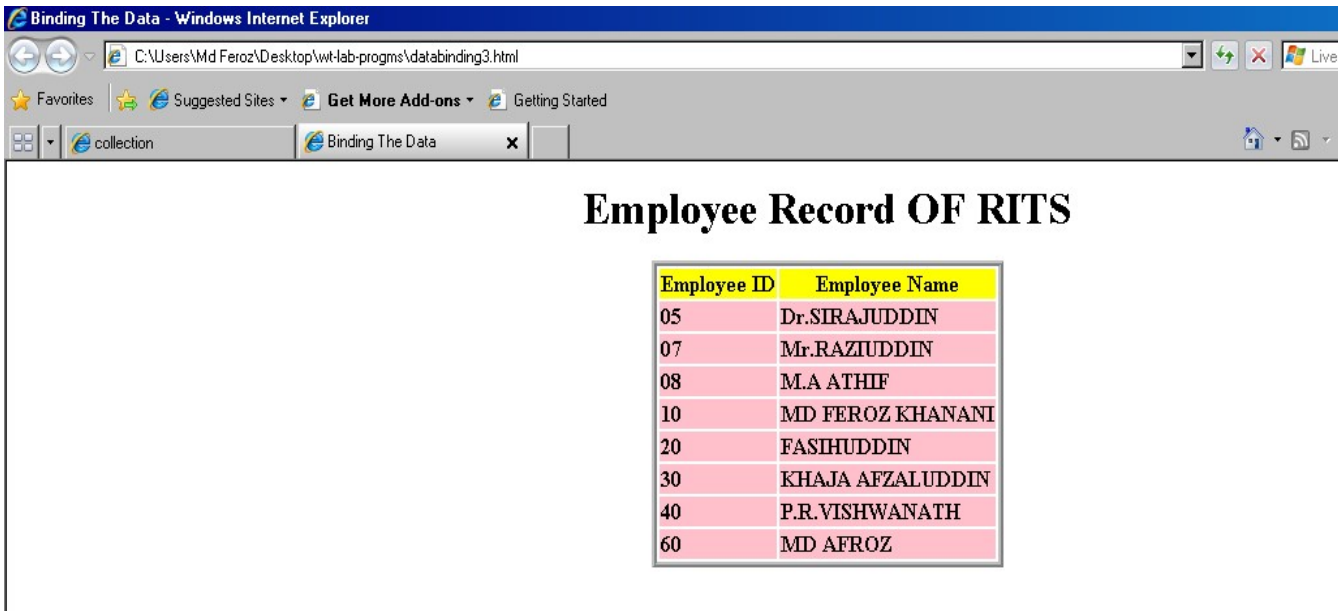
DHTML program

```
<html>
<head>
<title>Binding The Data</title>
<OBJECT ID = "EmpDB" ClassID="CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name="DataURL" value="rits.txt"/>
<param name="UseHeader" value="TRUE"/>
<param name="TextQualifier" value="#"/>
<param name="FieldDelim" value="|"/>
</OBJECT>
</head>
<body>
<center>
<h1>Employee Record OF RITS</h1>
<table datasrc="#EmpDB" style="border-style:ridge">
<thead>
<tr style="background-color:yellow">
<th>Employee ID</th>
<th>Employee Name</th>
</tr>
</thead>
<tbody>
<tr style="background-color:pink">
<td><span datafld="Emp_Id" style="font-weight:bold"></span></td>
<td><span datafld="Emp_Name" style="font-weight:bold"></span></td>
</tr>
</tbody>
```

```
</table>
</body>
</html>
```

rits.txt

```
#Emp_Id#|#Emp_Name#|#Emp_Salary#
#05#|#Dr.SIRAJUDDIN#
#07#|#Mr.RAZIUDDIN#
#08#|#M.A ATHIF#
#10#|#MD FEROZ KHANANI
#20#|#FASIHUDDIN#
#30#|#KHAJA AFZALUDDIN#
#40#|#P.R.VISHWANATH#
#60#|#MD AFROZ#
```


Binding The Data - Windows Internet Explorer
C:\Users\Md Feroz\Desktop\wt-lab-progms\databinding3.html
Live
Favorites
Suggested Sites
Get More Add-ons
Getting Started
collection
Binding The Data

Employee Record OF RITS

Employee ID	Employee Name
05	Dr.SIRAJUDDIN
07	Mr.RAZIUDDIN
08	M.A ATHIF
10	MD FEROZ KHANANI
20	FASIHUDDIN
30	KHAJA AFZALUDDIN
40	P.R.VISHWANATH
60	MD AFROZ

Explanation : The above database file(rits.txt) consists of two attributes Emp_Id and Emp_Name enclosed within special characters # and these are separated by the separator | and hence in our DHTML program we have mentioned

<param name="TextQualifier" value="#"/>

<param name="FieldDelim" value="|"/>

and in the database file(rits.txt) we have mentioned two headers such as Emp_Id and Emp_Name which can be represented in DHTML using the tag <param name="UseHeader" value="TRUE"/>

In our program we used Tabular Data Control object which is an ActiveX control which can be used to bind the database file(rits.txt) and In the body section, we used the span element in which data field is referred by **datafld="Emp_Name"** which is mentined in our database file(rits.txt).