UOK
DEPARTMENT OF COMPUTER SCIENCE

Web Application Programming
Instructor: Dr Makupi

Course objectives

- Explain introduction to internet and ways to access information form world wide web
- Create dynamic web pages by hand-coding HTML
- Find Creating simple web based applications using HTML
- Validating a web page using java script

Course Outline.

WEEK 1-2	Introduction Internet
WEEK3-4	Introduction to Html, working with the Html tags
WEEK 5	creating of tables and forms using HTML
WEEK6 -7	Introduction and working with Style Sheets and CSS
WEEK8	Working with frames
WEEK9	Introduction to java script and writing java script programs
WEEK 10-11	Data types and functions in java script , Event handling and form validation using java script
WEEK 12	introduction to php and working with forms and php

Learning and Teaching Methodology

Lectures, Presentations by members of the class, Case discussions, Tutorials, Assignments, Continuous assessment tests, manual/notes, Practicals

Instructional Materials:

Text books, handouts, chalk/white board, computers, projectors, software

Assessment Strategy:

Continuous assessment Tests		30%
• Class Tests	15%	
• Assignments	15%	
End of semester examination		70%
Total		**100%**

References:

1. **HTML for the World Wide Web - 6th Edition** by Elizabeth Castro PeachPit Press, ISBN 0321430840, 2006
2. *Web Technologies: A Computer Science Perspective*, Jeffrey C. Jackson, 2007, Pearson Prentice Hall
3. **Eric Meyer On CSS** and **More Eric Meyer On CSS** by Eric A. Meyer New Riders, ISBN 0-7357-1425-8, 2004
4. **Transcending CSS, The fine art of web design** by Andy Clarke New Riders, ISBN: 0-321-41097-1, 2007
5. **Essential ActionScript 3.0** by Colin Moock O'Reilly, ISBN 978-0-596-52694-8
6. **PPK on JavaScript** By Peter-Paul Koch New Riders, ISBN: 978-0-321-42330-6, 2007

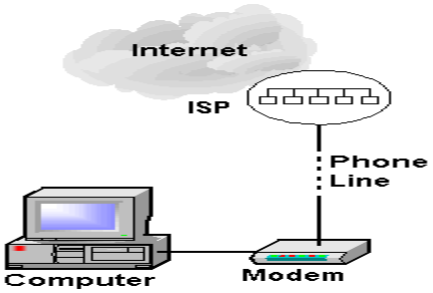
Further examples refer to
http://www.maketemplate.com/
http://www.dynamicdrive.com/

INTERNET

The **Internet** is a collection of information stored in computers that are networked together internationally. It is literally a network of networks. Physically, the Internet uses a subset of all of the resources of the currently existing public telecommunication networks.

Technically, what distinguishes the Internet as a cooperative public network is its use of a set of protocols called TCP/IP (Transmission Control Protocol/Internet Protocol).

How Do I Connect to the Internet?



- Computer
- Connection - Phone Line, Cable, DSL, Wireless, ...
- Modem
- Network Software - TCP/IP
- Application Software - Web Browser, Email, ...
- Internet Service Provider (ISP)

WAYS TO CONNECT TO INTERNET

- **Dial-up access** uses a modem and a phone call placed over the public switched telephone network (PSTN) to connect to a pool of modems operated by an ISP.

The modem converts a computer's digital signal into an analog signal that travels over a phone line's local loop until it reaches a telephone company's switching facilities or central office (CO) where it is switched to another phone line that connects to another modem at the remote end of the connection

- **Cable Internet** or cable modem access provides Internet access via Hybrid Fiber Coaxial wiring originally developed to carry television signals. Either fiber-optic or coaxial copper cable may connect a node to a customer's location at a connection known as a cable drop.

- **Digital Subscriber Line (DSL)** service provides a connection to the Internet through the telephone network. Unlike dial-up, DSL can operate using a single phone line without preventing normal use of the telephone line for voice phone calls. DSL uses the high frequencies, while the low (audible) frequencies of the line are left free for regular telephone communication
- **Wireless Fidelity (WIFI)** Individual homes and businesses often use Wi-Fi to connect laptops and smart phones to the Internet. Wi-Fi Hotspots may be found in coffee shops and various other public establishments. Wi-Fi is used to create campus-wide and city-wide wireless networks Wi-Fi networks are built using one or more wireless routers called Access Points.
- **WiMAX (Worldwide Interoperability for Microwave Access)** is a set of interoperable implementations of the IEEE 802.16 family of wireless-network standards certified by the WiMAX Forum. WiMAX enables "the delivery of last mile wireless broadband access as an alternative to cable and DSL"
- **Satellites** can provide fixed, portable, and mobile Internet access. It is among the most expensive forms of broadband Internet access, but may be the only choice available in remote areas

WORLD WIDE WEB

The World Wide Web (or WWW) is a subset of the Internet. Technically it is all the resources and users on the Internet that are using the Hypertext Transport Protocol (HTTP).

How the Web Works

Hypertext Transfer Protocol (HTTP) is a fast and efficient communication protocol that controls many different operations that take place between the Web browser client and the server.

HTTP uses the **Transmission Control Protocol (TCP)** to transport all of its control and data messages from one computer to another.

Web pages are typically grouped at a Web site, where the main page is referred to as the home page. The user navigates by mouse clicking on hyperlinks displayed as text, buttons, or images. These hyperlinks reference other information. When you click a

hyperlink, you jump to another part of the same page, a new page at the same Web site, or to another Website.

You might also execute a program, display a picture, or download a file. All of this hyperlinking is done with Hyper Text Markup Language, which works in concert with HTTP.

To connect with a Web site, you type **the Uniform Resource Locator (URL**) for the site into the Address field of a Web browser. Here is an example of the URL that retrieves the Microsoft Home Page.

When you type this request, the Web browser first gets the IP address of *www.microsoft.com* from a **Domain Name System (DNS)** server, and then connects with the target server. The server responds to the client and transfers this HTMLcoded document to your Web browser. Your Web browser then translates and displays the HTML information.

BROWSER

A **browser** program enables your computer to extract information from remote computers working on any platform through the WWW. The program on your computer is called WWW *client*. The WWW program [usually] on another computer that the browser "talks" to is called the *server*

Technically, a Web browser is a client program that uses the Hypertext Transport Control Protocol (HTTP) to make requests of Web servers throughout the Internet on behalf of the browser user

Web browsers communicate with Web servers primarily using **HTTP (Hypertext Transfer** Protocol) to fetch Web pages. HTTP allows Web browsers to submit information to Web servers as well as fetch Web pages from them.

Internet Explorer

Windows Internet Explorer (formerly **Microsoft Internet Explorer** abbreviated as **MSIE**), commonly abbreviated as **IE,** is a series of graphical web browsers developed by Microsoft and included as part of the Microsoft Windows line of operating systems starting in 1995. It has been the most widely used web browser since 1999.

3.9.1.1 Features of Internet Explorer

Internet Explorer has been designed to view the broadest range of web pages and to provide certain features within the operating system including Microsoft Update. Some of its features are as follows:

1. **Standard support**
   Internet Explorer almost fully support HTML, CSS XML implementation techniques.

2. **Usability and Accessibility**
   Internet Explorer makes use of the accessibility framework provided in Windows. Internet Explorer is also a user interface for FTP. Recent versions feature **pop – up blocking** and **tabbed browsing.**
3. **Cache**
   Internet Explorer caches visited content in the **Temporary Internet Files** folder to allow quicker access to previously visited pages.

4. **Security**
   Internet Explorer uses a zone – based security framework that groups sites based on certain conditions, including whether it is an intranet or internet – based site. Security restrictions are applied on a per – zone basis, all the sites in a zone are subject to the restrictions.
5. **Group Policy**
   Internet Explorer is fully configurable using **Group Policy (**feature that provides centralized management and configuration of computers and remote users**).** Administrators of **Windows Server Domains (**a logical group of computers running versions of the Microsoft Windows operating system that share a central directory database**)** can apply and

enforce a variety of settings that affect the user interface (such as disabling menu items and individual configuration options), as well as underlying security features such as downloading of files, zone configuration, per – site settings, etc.

Netscape Navigator

Netscape Navigator, also known as **Netscape,** is a proprietary (proprietary software is a term for computer software with restrictions on use or private modification, or with restrictions judged to be excessive on copying or publishing of modified or unmodified versions) web browser that was popular during the 1990s. It is a **"closed – source", "non – free"** web browser.

advantages to using a browser:

1) Browsers are typically free or very inexpensive. Windows 95/98 and Windows NT include a browser.
2) Browsers provide an almost universal interface for accessing and displaying information. Everyone uses the same or a similar interface, even people outside your organization and in different countries.
3) Browsers can connect with any Web server, no matter what operating system or platform.
4) Browsers require few system resources and little if any maintenance and system configuration.
5) Browsers can easily be updated by downloading the latest version from the Web page of the browser's developer or vendor.

URL

A URL (Uniform Resource Locator) (pronounced "you-are-EL" or, in some quarters, "earl") is the address of a file or other resource accessible on the Internet

- URL structure
  protocol://host.domain[:port]/path/file.extension [?optional stuff]

http://www.hellohelpme.com/index.html

http:	protocol
//www.	host

hellohelpme.com domain
/index.html the individual document

the extension html shows that it was coded with html (web documents frequently have the extension html or htm)

- Protocol might be - http, file, gopher, wais, news, telnet, https, ...
- The CASE of URLs may be important! Use only lower case in URLs that you build!
- "optional stuff" - after a ? - is usually search parameters or values for variables. Often used to pass parameters to search engines, databases, or CGI scripts.

Domains commonly end in:

- .com - commercial entity e.g., www.godiva.com - Godiva Chocolates
- .gov - government e.g., www.loc.gov - the library of congress
- .net - a network e.g., www.psi.net PSInet
- .org - a non-profit organization e.g., www.kennedy-center.org
- .mil - the military e.g., www.army.mil
- Foreign domains end in .ca (Canada), .uk (United kingdom), etc. For example This New Zealand University at www.canterbury.ac.nz - Note the ".nz" for New Zealand

- Some proposed new domains
  - .arts - Entertainment, music, culture
  - .firm - Businesses
  - .info - Information services and providers
  - .nom - Personal home pages
  - .rec - Recreation
  - .store - Online sales
  - .web - Web activities

HTTP

The Hypertext Transfer Protocol (HTTP) is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. Relative to the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), HTTP is an application protocol.

HTTP include the idea that files can contain references to other files whose selection will elicit additional transfer requests. Any Web server contains, in addition to the Web files it can serve,

Your Web browser is an HTTP client, sending requests to servers. When the browser user enters file requests by either "opening" a Web file (typing in a Uniform Resource Locator or URL) or clicking on a hypertext link, the browser builds an HTTP request and sends it to the Internet Protocol address indicated by the URL

HYPERTEXT

In a hypertext document, certain words within the text are marked as *links* to other areas of the current document or to other documents

The user moves to a related area by moving his or her mouse pointer to the link and clicking once with the mouse button

Links are used to point to another part of the same document, in which case clicking the link will cause the browser to move to a new part of the currently displayed document

HYPERMEDIA

hypermedia links uses the images tom point to HTML documents. For instance, a hypermedia link might point to an audio file, a QuickTime movie file, or a graphic file such as a GIF- or JPEG-format graphic

HYPERTEXT MARKUP LANGUAGE

HTML documents are plain-text (also known as ASCII) files that can be created using any text editor

HTML

This element tells your browser that the file contains HTML-coded information. The file extension .html also indicates this is a HTML document and must be used

The HTML tag identifies a document as an HTML document. All HTML documents should start with the <HTML> tag and end with the </HTML> tag.

Syntax

<HTML>...</HTML>

HEAD

The HEAD tag defines an HTML document header. The header contains information about the document rather than information to be displayed in the document. The web browser displays none of the information in the header, except for text contained by the TITLE tag.

Syntax

<HEAD>...</HEAD>

Example

```
<HEAD>
<TITLE> Mozilla speaks out</TITLE>
<BASE HREF="http://www.mozilla.com">
</HEAD>
```

TITLE

The title element contains your document title and identifies its content in a global context. The title is typically displayed in the title bar at the top of the browser window, but not inside the window itself.

The title is also what is displayed on someone's hot list or bookmark list, so choose something descriptive, unique, and relatively short. A title is also used to identify your page for search engines

BODY

The second--and largest--part of your HTML document is the body, which contains the content of your document (displayed within the text area of your browser window). The tags explained below are used within the body of your HTML document.

For Example:--

```
<html>
<head>                    // basic document structure
<title>
MY WEB
</title>
</head>
<body>
HTML Tutor Lesson 1
</body>
</html>
```

TEXT FORMATTING TAGS

Tags	*Meaning*
<B> SAMPLE </B>	Bold text
<I>SAMPLE </I>	Italic text
<U> SAMPLE </U>	Underlined text

```
<H1><I>Welcome Home!</I></H1>
<B><I>This is bold and italic</I></B>
```

Some Basic Logical HTML Tags

Tags	*Meaning*	*Generally Rendered as…*
<EM>, </EM>	Emphasis	Italic text
<STRONG>, </STRONG>	Strong emphasis	Bold text
<TT>, </TT>	Teletype	Monospaced text

ORDERED LIST:

```
<OL>
<LI> Item number one.
<LI> Item number two.
<LI> Item number three.
</OL>
```

```
<UL>
<LI> First item.
<LI> Second item.
<LI> Third Item.
</UL>
```

UNORDERED LIST:

IMAGE TAG

To add graphics, you use an empty tag called the <IMG> (image) tag, which you insert into the body section of your HTML document as follows:

<IMG SRC="*image URL*">

<IMG>	image	**<IMG** SRC="Earth.gif" WIDTH="41" HEIGHT="41" BORDER="0" ALT="a sentence about your site">

```
<HR>
<P>This is a test of the Image tag. Here is the image I want to
display:</P>
<IMG SRC="image1.gif">
<HR>
```

The ALT Attribute

The ALT attribute for the <IMG> tag is designed to accept text that describes the graphic, in case a particular browser can't display the graphic

using the <A> Tag

The basic link for creating hypertext and hypermedia links is the <A>, or anchor, tag. This tag is a container, which requires an </A> to signal the end of the text, images, and HTML tags that are to be considered to be part of the hypertext link. Here's the basic format for a text link:

<A HREF="*URL*">*Text describing link*</A>
Example
```
<A HREF="products.html">Our Product Information</A>
```

<FONT>	font	**<FONT** FACE="Times New Roman" SIZE="4">Example**</FONT>**
<FONT>	font	**<FONT** FACE="Times New Roman" SIZE="+3" COLOR="#FF0000">Example**</FONT>**
<H1>	heading 1	**<H1>Heading 1 Example</H1>**
<H2>	heading 2	**<H2>Heading 2 Example</H2>**
<H3>	heading 3	**<H3>Heading 3 Example</H3>**
<H4>	heading 4	**<H4>Heading 4 Example</H4>**
<H5>	heading 5	**<H5>Heading 5 Example</H5>**
<H6>	heading 6	**<H6>Heading 6 Example</H6>**
<HEAD>	heading of document	**<HEAD>Contains elements describing the document</HEAD>**
<HR>	horizontal rule	**<HR>**
<HR>	horizontal rule	**<HR WIDTH="50%" SIZE="3">**
<HR>	horizontal rule	**<HR WIDTH="50%" SIZE="3" NOSHADE>**
<HR> (Internet Explorer)	horizontal rule	**<HR WIDTH="75%" COLOR="#FF0000" SIZE="4">**
<HR> (Internet Explorer)	horizontal rule	**<HR WIDTH="25%" COLOR="#6699FF" SIZE="6">**

HTML FORMS

The **form** element has two major attributes.

The **method** attribute indicates the HTTP method to be used in sending the information and the **action** attribute gives

details of where it will be sent. In the example, the HTTP POST method is specified and the input is to be sent to the mail address specified. The possible methods are:

GET

This is the default action. If a URL is specified as the value of the **action**, a **?** followed by the input is sent to the server or whatever processing agent is specified.

POST

In this case the input is embedded in the form and sent to the processing agent

<form method= "get" action = "second.html">

This method is not save as it displays all the values on the address bar where by the passwords can be easily be seen

FORM ELEMENTS

Text box : Defines a one-line input field that a user can enter text into
Radio button: let a user select ONLY ONE of a limited number of choices
Checkboxes: let a user select ZERO or MORE options of a limited number of choices.
Select option: the user selects on one option at a time
Text area : accepts rows and columns and allows the user to scroll

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute

Example

<Html><head> <title> my form</title></head>
<body>

<form>

1.   First name: <input type="text" name="firstname" value="" size="10" maxlength="30" **/>
**

First Name:

2.Last name: <input type="text" name="lastname" value="" size="10" maxlength="10" **/>**

Last Name:

3.Password: <input type="password" name="pwd" />

Password :

4.Gender <input type="radio" name=" Gender " value="male" CHECKED **/> Male
**

<input type="radio" name=" Gender " value="female" /> Female

Gender : male female

**vehicle :<input type="checkbox" name="vehicle" value="Bike" /> I have a bike
**

<input type="checkbox" name="vehicle" value="Car" CHECKED **/> I have a car
**

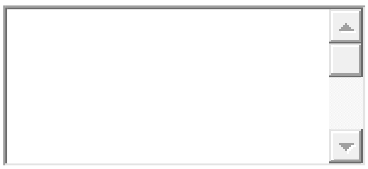
Vehicle :

bike

car

**6. text area : <textarea rows="10" cols="30"> The cat was playing in the garden. </textarea>
**

*Enter Your Comments:
 <TEXTAREA wrap="virtual" name="Comments" rows=3 cols=20 MAXLENGTH=100></TEXTAREA>
*



7. select option

```
<select name="course">
<option value="MSC ">MSC </option>
<option value="MBA">MBA</option>
<option value="BA">BA</option>
<option value="ENG">ENG</option>
</select><br />
```

Select an option: option 1

```
<input type="submit" value="Submit" />
<INPUT type="Reset" VALUE="Clear">
```

Submit Clear

```
</form>
</body>
</html>
```

The input Element

<form method= action= onsubmit= >

<input name= type= value= size= maxlength= checked= src=

The overall form of the **input** element depends on the three attributes **name**, **type** and **value**. The other attributes are either required for a subset of the devices or the meaning varies dependent on the device:

NAME

The **name** attribute gives the name to be added to the input to indicate where it has come from. In most cases, the input device has an initial value so that if you submit the input without doing anything,this value will be sent to the destination. The input device has the ability to change the initial value and this is usually what will be sent.

TYPE

The **type** attribute defines the type of input. The possible input types are: **text**, **submit**, **radio**,

checkbox, **button**, **reset**, **password**, **file**, **hidden** and **image**. We shall look at these in detail in this section. If no **type** value is given, the input type is assumed to be **text**. In the example in the previous section, the **type** attribute could have been omitted.

VALUE

The **value** attribute gives the initial value for the input from a device in most cases. The only device that must have an initial value is the **radio** device.

SIZE

This either gives the width of the input device in pixels or the number of characters allowed.

MAXLENGTH

For **text** and **password** devices, this defines the maximum size that the input can take in characters.

CHECKED

For **radio** and **checkbox** devices, this says whether the button is set initially.

SRC

For **image** devices, this points to the image.

IMAGE TAG

Image tag

<IMG SRC="Earth.gif" WIDTH="41" HEIGHT="41" BORDER="0" ALT="a sentence about your site">

Dynamic text area

Text area

TEXTAREA wrap="virtual" name="Comments" rows=3 cols=20
MAXLENGTH=100></TEXTAREA>

DEFAULT SELECTED ELEMENTS

Selected option

Select an option:
<SELECT>
<OPTION >option 1
<OPTION SELECTED>option 2
<OPTION>option 3
<OPTION>option 4
<OPTION>option 5
<OPTION>option 6
</SELECT>

Radio button

<INPUT type="radio" name="option"> Option 1
<INPUT type="radio" name="option" CHECKED> Option 2
<INPUT type="radio" name="option"> Option 3

Check box

Select an option:
<INPUT type="checkbox" name="selection"> Selection 1
<INPUT type="checkbox" name="selection" CHECKED> Selection 2
<INPUT type="checkbox" name="selection">

TABLES

The table is divided into three main parts, a header, a body and a footer. There is also an optional caption that can be added to the table.

Table is divided into rows each having a number of columns. As with most word processing systems it is possible to give groups of columns special features and to merge two or more columns together or two or more rows together.

Element Description:--

<TABLE> ... </TABLE>

defines a table in HTML. If the BORDER attribute is present, your browser displays the table with a border.
<CAPTION> ... </CAPTION>

defines the caption for the title of the table. The default position of the title is centered at the top of the table. The attribute ALIGN=BOTTOM can be used to position the caption below the table.
<TR> ... </TR> specifies a table row within a table. You may define default attributes for the entire row: ALIGN (LEFT, CENTER, RIGHT) and/or VALIGN (TOP, MIDDLE, BOTTOM). See Table Attributes at the end of this table for more information.

<TH> ... </TH> defines a table header cell. By default the text in this cell is bold and centered. Table header cells may contain other attributes to determine the characteristics of the cell and/or its contents. See Table Attributes at the end of this table for more information.

<TD> ... </TD> defines a table data cell. By default the text in this cell is aligned left and centered vertically. Table data cells may contain other attributes to determine the characteristics of the cell and/or its contents

Borders These are the lines that surround the table and each cell.

CELLSPACING. The CELLSPACING attribute tells the browser how much space to include between the walls of the table and between individual cells. (Value is a number in pixels.)

CELLPADDING. The CELLPADDING attribute tells the browser how much space to give data elements away from the walls of the cell. (Value is a number in pixels

Attribute Description :--

ALIGN (LEFT, CENTER, RIGHT) Horizontal alignment of a cell.
VALIGN (TOP, MIDDLE, BOTTOM) Vertical alignment of a cell.
COLSPAN=n The number (n) of rowa cell spans.
ROWSPAN=n The number (n) of rcolumn a cell spans

The general format of a table looks like this:--

<TABLE>
<!-- start of table definition -->

<CAPTION> caption contents </CAPTION>

```
<!-- caption definition -->

<TR>
<!-- start of header row definition -->
<TH> first header cell contents </TH>
<TH> last header cell contents </TH>
</TR>
<!-- end of header row definition -->

<TR>
<!-- start of first row definition -->
<TD> first row, first cell contents </TD>
<TD> first row, last cell contents </TD>
</TR>
<!-- end of first row definition -->

<TR>
<!-- start of last row definition -->
<TD> last row, first cell contents </TD>
<TD> last row, last cell contents </TD>
</TR>
<!-- end of last row definition -->

</TABLE>
<!-- end of table definition -->
```

EXAMPLE OF TABLE

```
<html><head><title> my food</title></head><body>

<TABLE>

<CAPTION>My favorite foods</CAPTION>

<table width="40%" height="30%" cell padding="5" border="0" align="center">

<TR>

<TH>Soup</TH>

<TD>Chicken Noodle</TD>

 </tr>

<TR>

<TH>Salad</TH>
```

```
<TD>Tossed Green</TD>

 </tr>

</TABLE></body></html>
```

Other attributes

- **ALIGN**. The ALIGN attribute is used to determine where the chart will appear relative to the browser window. Valid values are ALIGN=LEFT and ALIGN=RIGHT. As an added bonus, text will wrap around the table (if it's narrow enough) when the ALIGN=LEFT or ALIGN=RIGHT attributes are used.
- **WIDTH**. The WIDTH attribute sets the relative or absolute width of your table in the browser window. Values can be either percentages, as in WIDTH="50%", or absolute values. With absolute values, you must also include a suffix that defines the units used, as in px for pixels or in for inches (e.g., WIDTH="3.5in"). Absolute values for table widths are discouraged, though.
- **COLS**. The COLS attribute specifies the number of columns in your table, allowing the browser to draw the table as it downloads.
- **BORDER**. The BORDER attribute defines the width of the border surrounding the table. Default value is 1 (pixel).
- **CELLSPACING**. The CELLSPACING attribute tells the browser how much space to include between the walls of the table and between individual cells. (Value is a number in pixels.)
- **CELLPADDING**. The CELLPADDING attribute tells the browser how much space to give data elements away from the walls of the cell. (Value is a number in pixels.)

Adding audio to html page

```
<html><head> </head><body>

<audio controls>

  <source src="Rihanna.ogg" type="audio/ogg">

  <source src="Rihanna.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

</body>

</html>
```

Ogg is an open and standardized bitstream container format designed for streaming and manipulation.

mpeg

an international standard for encoding and compressing video images. "**Moving Picture Experts Group**" or "Audio/visual file format"

FRAMES

HTML frames allow authors to present documents in multiple views, which may be independent windows or subwindows. Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced.

For example, within the same window, **one frame might display a static banner**, a second a **navigation menu**, and a third **the main document that can be scrolled though or replaced by navigating in the second frame.**

Layout of frames

An HTML document that describes frame layout (called a *frameset document*) has a different makeup than an HTML document without frames. A standard document has one HEAD section and one BODY.

The Frameset Tag

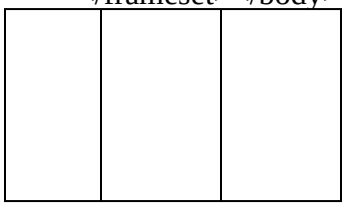
The <frameset> tag defines how to divide the window into frames,Each frameset defines a set of rows or columns

The values of the rows/columns indicate the amount of screen area each row/column will occupy

TYPES OF FRAMES

Vertical frameset

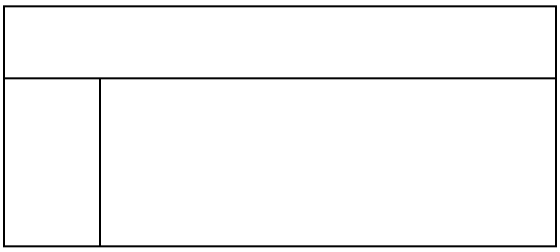
```
<html><body>

<frameset cols="25%,50%,25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
</frameset></body></html>
```



```
<html><body>

<frameset rows="25%,50%,25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">

</frameset><body>
</html>
```

Combining rows and columns

```
<html>
<frameset rows="25%,75%">

<frame src="frame_a.htm">

<frameset cols="25%,75%">
<frame src="frame_b.htm">
<frame src="frame_c.htm">
</frameset>
</html>
```



Example 2

```
<html>
<head>
<title> Second example of frames </title>
<frameset rows="20%, 80%">
<frame src="page1.html" name="top" marginheight="1" scrolling="auto">
<frameset cols="25%, 75%">
<frame src="page2.html" name="left" marginheight="1" scrolling="no">
<frame src="page3.html" name="right" marginwidth="3" marginheight="1"
scrolling="auto">
</frameset>
</frameset>
```

```
<html>

<frameset rows="25%75%">

<frame noresize="noresize" src="frame_a.htm">

<frameset cols="25%,75%">
<frame noresize="noresize" src="frame_b.htm">
<frame noresize="noresize" src="frame_c.htm">
</frameset>
</frameset>

</html>
```

EXAMPLE TOP FRAME

```
<html>
<head>
<titile></title>
</head>
<body>
<body bgcolor="wheat">
<table width="500" height="50" border="0" cellspacing="0" cellpadding="0">
<tr>
<td><img src="d:\web design\FLOWER103.JPG"width="150" height="80"></td>
<td><img src="d:\web design\ban.JPG" width="500" height="70"></td>
<td><img src="d:\web design\white_W.gif"></td>
</tr>
</table>
</BODY>
<HTML>
```

SIDE FRAME

```
<HTML>
<HEAD>
<TITLE>MAIN PAGE</TITLE>
</HEAD>
<body bgcolor="wheat"><table>
<TR><TD><a href ="about.html"target="right frame">ABOUT...</a></td></tr>
<tr><td><BR><a href ="intro.html"target="right
frame">HOME</a></td></tr><BR><BR>
<tr><td><BR><a href ="san.html"target="right frame">IN TECH</a></td></tr><BR>
```

```
<tr><td><BR><a href ="online.html"target="right frame">OnlineExam</a></td></tr><BR>
<tr><td><BR><a href ="calculate.html"target="new">jokes</a></td></tr><BR>
</table>
</body>
</html>
```

Main frame

```
<html>
<head>
<title> frameset</title>
</head>`<body>
<frameset rows="65,*">
<frame src="top.htm" name="topframe" scrolling="no">
<frameset cols="120,*">
<frame src="main.htm" name="left frame" scrolling="no">
<frame src="intro.htm" name="right frame" scrolling="yes">
</frameset>
</frameset></body></html>
```

STYLE SHEET

CSS stands for Cascading Style Sheets. It is a way to divide the content from the layout on web pages.

The CSS syntax is made up of three parts: a selector, a property and a value:

```
selector {property: value}
```

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon and surrounded by curly braces:

```
body {color: black}

p{          text-align: center;
                    color: black;
                    font-family: arial
                }
```

All the headers will be green color

```
h1,h2,h3,h4,h5,h6
                    {
                    color: green
                  font-family: arial
                    }
```

Class selector

you can define different styles for the same type of HTML element. Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles

```
p.right {text-align: right}
                    p.center {text-align: center}
```

example

```
<h1 class="center">
                    This heading will be center-aligned
                    </h1>
<p class="center">
                    This paragraph will also be center-aligned.
                    </p>
```

Id selector

With the id selector you can define the same style for different HTML elements.

The style rule below will match any element that has an id attribute with a value of "green":

h1 id="green">Some text</h1>

Some text

Example 1

```
<STYLE TYPE-"type/css">
<!--
BODY {background: wheat}
A:link {color: blue}
A:visited {color: red}
H1 {font-size: 24pt; font-family: arial; color :green}
H2 {font-size: 18pt; font-family: braggadocio; color :orange}
H3 {font size:14pt; font-family: Desdemona;b color:  back}
-->
</STYLE>
```

Example 2

```
<head>
<style>
body {
   background-color: linen;
}

h1 {
   color: maroon;
   margin-left: 40px;
}
</style>
</head>
```

Types of style sheets

(a) **inline style sheet** – is mainly embedded in the html document after the head after closing the title tag and before closing the head tag

```
<html> <head><title>MY CSS PAGE</title>
<style type="text/css">
.headlines, .sublines, .infotext {font-face:arial; color:black;
background:yellow; font-weight:bold;}
.headlines {font-size:14pt;}
.sublines {font-size:12pt;}
.infotext {font-size: 10pt;}
</style>  </head><body>

< span class="headlines">Welcome</span><br>
<div class="sublines">
This is an example page using CSS.<br>
</div>
<div class="infotext">
Example from Academic Articles.
</div> </body></html>
```

Example 2

<head><title>….</title>

```
body {
background-color: orange;
font-family: Arial, Verdana, sans-serif;
font-size: 18;
color: red;
}

a { font-family: Arial, Verdana, sans-serif; font-size: 18px; color: blue; text-decoration:
underline}

h1 { font-family: Arial, Verdana, sans-serif; font-size: 32px; color: yellow }
h2 { font-family: Arial, Verdana, sans-serif; font-size: 24px; color: brown }

hr{ color:brown; background-color:tan; width:90%; height:2px; }

table {
font-family: Arial, Verdana, sans-serif;
font-size: 18px;
color: orange;
}
```

```
<p> {
font-family: Arial, Verdana, sans-serif;
font-size: 14px;
color: purple;
font-weight: bold;
}
</head>
<body>
```

(b) **Entire Sites / External Style Sheet**

CSS can be defined for entire sites by simply writing the CSS definitions in a plain text file that is referred to from each of the pages in the site

```
<html>
<head>
<title>MY CSS PAGE</title>
<link rel=stylesheet href="whatever.css" type="text/css">
</head>
<body>

<span class="headlines">Welcome</span><br>
<div class="sublines">
This is an example of a page using CSS.<br>
The example is really simple,<br>
and doesn't even look good,<br>
but it shows the technique.
</div>
<table border="2">

<tr>

<td class="sublines">
As you can see:<br>
The styles even work on tables.
</td>

</tr>

</table>
<hr>
<div class="infotext">Example from Academic Articles.</div>
<hr>
</body>
</html>
```

This means that the browser will look for a file called whatever.css and insert it at the place where the reference was found in the html document.

whatever.css

```
headlines, .sublines, infotext {font-face:arial; color:black; background:yellow; font-weight:bold;}
.headlines {font-size:14pt;}
```

```
.sublines {font-size:12pt;}
.infotext {font-size: 10pt;}
```

Now if you just add the line **<link rel=stylesheet href="whatever.css" type="text/css">** to the **<head>** of all your pages, then the one style definition will be in effect for your entire site.

The benefits of CSS

Separation of content and presentation

CSS rules can be provided in a file that is separate to the (content) HTML. If all pages link to this centralised CSS file, then the look of a website can more easily be updated.

Smaller webpage file sizes

As the code required to style content can be removed from individual webpages, the size of each webpage file is reduced. Depending on the benchmarks, file sizes may be reduced by up to 60%.

Improved webpage download speed

Once a stylesheet has been downloaded, it is typically stored on the user's computer .

Improved rendering speed

Once a webpage has been downloaded, a browser processes the underlying code to determine how content should be displayed. This process is referred to as 'rendering'. The time a webpage takes to render is affected by the complexity of the code the browser receives.

Streamlined maintenance

As less code is required for each webpage, both the likelihood of coding errors and time required to add content to a website are reduced.

Changing presentation for different devices

The CSS specification enables different rules to be used depending on the device used to access the web. For example, a different set of rules can be used to reformat a webpage for printing or viewing on a mobile phone.

JAVA SCRIPT BASICS

Java script is used to develop dynamic web pages interactive web pages, when one uses HTML the pages are said to be static, java script is embedded into a html document using the SCRIPT tag where by it allows one to allow to add real programming to web pages by validation of the data which is submitted to the form.

Script tag

```
<script language = “java script”>
<!...
document. write(“welcome to java script programming”);
//…>
</script>
```

Where to place it

The browser will use the <script> type="text/javascript"> and </script> to tell where javascript starts and ends.

Example 1

```
<html>
<head><title>My Javascript Page</title>
</head><body>
<script type="text/javascript">
alert("Welcome to JAVA PROGRAMMING!!!");
</script>
</body>
</html>
```

Example 2

```
<html>
<head>
<title>My Javascript Page</title>
<script>
document.write("Welcome to my world!!!<br>");
```

```
</script>
Enjoy your stay...<br>
</head>
<body>
Hello!!!<br>
</body>
</html>
```

The document.write is a javascript command telling the browser that what follows within the parentheses is to be written into the document

- A prompt box is used when you want some input from the user.

USES JAVASCRIPT:

- Browser Detection
  Detecting the browser used by a visitor at your page. Depending on the browser, another page specifically designed for that browser can then be loaded.

- Cookies
  Storing information on the visitor's computer, then retrieving this information automatically next time the user visits your page. This technique is called "cookies".

- Control Browsers
  Opening pages in customized windows, where you specify if the browser's buttons, menu line, status line or whatever should be present.

- Validate Forms
  Validating inputs to fields before submitting a form.
  An example would be validating the entered email address to see if it has an @ in it, since if not, it's not a valid address.

Java script identifiers

Identifier- is a variable used to store data and symbolically use the data, each variable will allow the data to be referred easily

GUIDELINES IN CREATING THE IDENTIFIERS

- ✓ The first character of an identifier must be a letter, an underscore(_) or dollar ($)sign may be used
- ✓ An identifier must not begin with a number.
- ✓ An identifier must not be the reserved java script words(as they are used directly by the language)
- ✓ Java script is case sensitive language .eg casetest is not same as CaseTest

Declaring and Initializing the identifiers

```
var num   // declare variable, has null value( var used to represent string information)
num = 20   // assign a value, null value is replaced
//OR
var num=20   // declare AND assign value (initialize)
var a,b,c,d //multiple variables may be declared simultaneously
var a = 5, b = 6, c = 7,d=8 //multiple variables may be initialized simultaneously
```

Variable Scope

There are two types of Variable scope:-

1. Globle scope Variable,

2 . Local scope Variable.

1.Globle scope Variable:- A variable declared or initialized outside a function body has a global scope, making it accessible to all other statements within the same document.

2. Local scope Variable:- A variable declared or initialized within a function body has a local scope, making it accessible only to statements within the same function body

DATA TYPES

In java script data can be represented into the following major types are Number, String, Boolean

a) PRIMITIVE DATA types are the simplest building blocks of a program. There are following types :-

1 Number

Used to perform the arithmetic operations such as addition, subtraction, multiplication and division, any whole number or floating –point literal that does not appear between quotation marks(" ") is considered as number

Example: 67.,29,40

2.String

String represents any sequence of zero or more characters that are to be strictly text that os no mathematical operations can be performed on them.

Examples: "hallo","200","kabarak"

3.Boolean

Is used with the logical operations and can have one of two values true or false.

4. Null:-The null value represents no value that means strings are empty .

5. **Undefined:-** A variables to be declared but given no any initial value then it runtime error display

ABSTRACT DATA TYPES

1.OBJECTS: Represents a collection of data that work together to perform a related task they mainly include strings, array

2.ARRAY: They are used to hold large amounts of data of the same type logically in the same memory so that each element can be accessed directly from the array name and index

3.FUCTIONS : is a collection of one or more script statements that can be made to run at any point during the life of a program, they allow one to write very compact, easy to understand and reusable code.

OPERATORS

Java script operators are symbols and key words you use to assign values to variables or perform operations with those values ,using the operations you can combine the variables and literals values into expression that perform calculation and produce results

Arithmetic operators

They are used to perform the arithmetic operators on numbers they are divided to unary to binary

Unary operators change the value of a single value or expression while Binary operators change the value of two values or expressions ,

Assignment operator is used to store the results of the arithmetic to a variable

Unary operators

++ increment example 5++

-- decrement example 6---

Binary operators

+ addition	* multiplication	% modulus
- subtraction	/ division	

RELATIONAL OPERATORS

They are primary used with selection statement because they produce a Boolean result (either true or false)

= = equal to

!= not equal to

> greater than

>= greater than or equal to

< less than

<= less than or equal to

Example

Var var1 = 99

Var var2 = -12

If (var1 = = var2)

{

// false

}

else if (var1 > var2)

{

```
// true
}
else if (var1 < var2)
{
//false
}
```

MISCELLANEOUS OPERATORS

delete	used to delete an object and free up the memory
new	used to create an instance of a user defined object
this	used to refer to the current object
typeof	return the type of an unevaluated operand
void	evaluates without returning an object
+	string concatenation operator

SELECTION AND REPETITION STATEMENTS

The *if* Statement

The `if` statement is used to conditionally execute a single block of code

`if` statement consists of a conditional expression, known as the `if` test, and a block of code which is executed if that expression evaluates to a Boolean `true`

SYNTAX

if (<condition >)
{
Statements
}

Example
if (browser=="MSIE") {alert("You are using MSIE")}

```
<HTML><HEAD></HEAD><SCRIPT language="JavaScript">
<script language="javascript">
document.write ("checking if 5 is less than 6.....");
if (5 < 6) {
document.write ("<br>Condition is true : 5 is less than 6");
}
document.write ("<br>Done checking");
</script>
</BODY>
</HTML>
```

```
<html>
  <body>

    <script type="text/javascript">
      <!--
         var age = 20;

         if( age > 18 ){
            document.write("<b>Qualifies for driving</b>");
         }
      //-->
    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

IF ELSE

In this form, one block of code is executed if the if test passes, and a second block is executed if it fails. The format of this type of if statement is as follows:

```
if (expression){
   Statement(s) to be executed if expression is true
}

else{
   Statement(s) to be executed if expression is false
}
```

Example

```
<html>
  <body>

    <script type="text/javascript">
      <!--
        var age = 15;

        if( age > 18 ){
          document.write("<b>Qualifies for driving</b>");
        }

        else{
          document.write("<b>Does not qualify for driving</b>");
        }
      //-->
    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Example

```
<HTML>
<HEAD></head>
```

```
<script type="text/javascript">
var a = new Date()
var time = a.getHours()
if (time < 11)
{
document.write("Good morning!");
}
else{
document.write("Good day!");}
</script></BODY></HTML>
```

Nested if else statement

```
if (expression 1){
  Statement(s) to be executed if expression 1 is true
}

else if (expression 2){
  Statement(s) to be executed if expression 2 is true
}

else if (expression 3){
  Statement(s) to be executed if expression 3 is true
}

else{
  Statement(s) to be executed if no expression is true
}
```

Check Odd/Even Numbers JavaScript Program (for beginners)

This is a simple JavaScript program to check odd or even numbers with example.

```
<html>
<head>
<script type="text/javascript">
var n = prompt("Enter a number to find odd or even", "Type your number here");
n = parseInt(n);
if (isNaN(n))
{
alert("Please Enter a Number");
}
else if (n == 0)
{
alert("The number is zero");
}
else if (n%2)
{
alert("The number is odd");
}
else
{
alert("The number is even");
}
</script>
</head>
<body>
</body>
</html>
```

SWITCH STATEMENT

It is commonly know as case statement, switch matches an expression with a specified case and executes the statements defined for that case.

Syntax

```
Switch (expression)
{
case value 1:
        Statement;
        Break;
Case value 2:
        Statement;
        Break;
Case value 3:
        Statement;
        Break
…….
……


case value n:
        Statement;
        Break;
default:
        statement;
        break;

}
```

```
<script type="text/javascript">
var n=0;
n=prompt("Enter a number between 1 to 12:")
switch(n)
{
case(n="1"):
```

```
document.write("January");
break;
case(n="2"):
document.write("Febuary");
break;
case(n="3"):
document.write("March");
break;
case(n="4"):
document.write("April");
break;
case(n="5"):
document.write("May");
break;
case(n="6"):
document.write("June");
break;
case(n="7"):
document.write("July");
break;
case(n="8"):
document.write("August");
break;
case(n="9"):
document.write("September");

case(n="10"):
document.write("October");

case(n="11"):
```

```
document.write("November");


case(n="12"):
document.write("December");
break;
default:
document.write("enter  value btn 1-12");
break;
}
</script> >/head>  <body>
</body>
```

Simple Switch Case JavaScript Program (for beginners)

This is a simple switch case JavaScript example program for beginners..

```
<html>
<head>
<script type="text/javascript">
var n=prompt("Enter a number between 1 and 7");
switch (n)
{
case (n="1"):
document.write("Sunday");
break;
case (n="2"):
document.write("Monday");
break;
case (n="3"):
document.write("Tuesday");
break;
case (n="4"):
document.write("Wednesday");
break;
case (n="5"):
```

```
document.write("Thursday");
break;
case (n="6"):
document.write("Friday");
break;
case (n="7"):
document.write("Saturday");
break;
default:
document.write("Invalid Weekday");
break
}
</script>
</head>
</html>
```

LOOPING

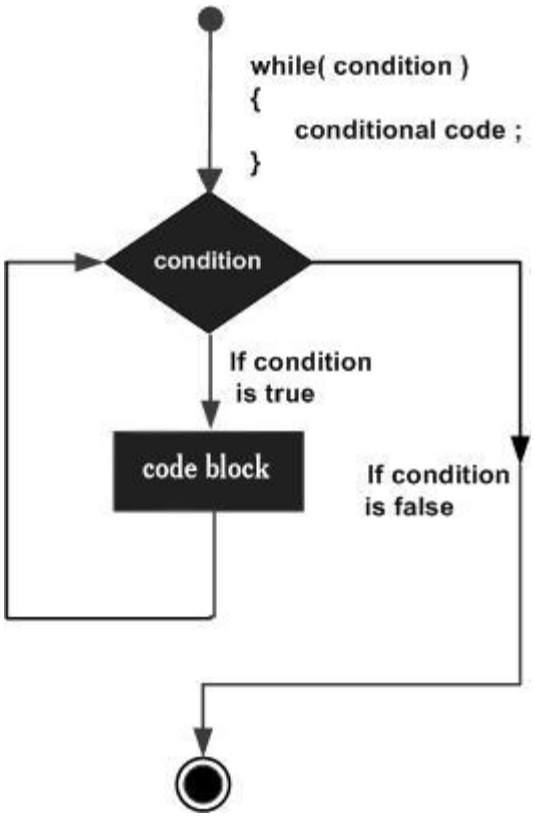
Used to repeat the given case for specific number of times (when the condition is true)

The *while* Statement

The `while` statement is used to execute a block of code while a certain condition is `true`. The format of the `while` statement is as follows:

```
while (variable<=endvalue)
{
// Here goes the script lines you want to loop.
}
```

The flow chart of **while loop** looks as follows −



Syntax

The syntax of **while loop** in JavaScript is as follows −

```
while (expression){
  Statement(s) to be executed if expression is true
}
```

Example

Try the following example to implement while loop.

```
<html>
  <body>

    <script type="text/javascript">
      <!--
        var count = 0;
        document.write("Starting Loop ");
              while (count < 10){
          document.write("Current Count : " + count + "<br />");
```

```
            count++;
          }

          document.write("Loop stopped!");
        //-->
      </script>

      <p>Set the variable to different value and then try...</p>
    </body>
</html>
```

```
<html>
<body>
<script language="Javascript">
var i=0
while (i<=10)
{
document.write("serial number" + i)
document.write("<br />")
i=i+1
}
</script>
<body>
</html>
```

The do while loop

This loop will be always executed once, even if the condition is false, because the loop code are executed before the condition is tested.

```
<html>
<body>
<script language="Javascript">
var i=0
do
{
```

```
alert("number" + i)
i=i+1
}
while (i<=3)
</script>
</body>
</html>
```

THE FOR LOOP

The for loop repeatedly through the block of statements until the test condition is false, the numbers of times it repeats depends on a counter

for (initial-statement; test; increment/decrement**)**

{

statements .

}

The initial statement is executed first; commonly this statement is used to initialize a counter variable

Test is applied to the counter variable and then the loop starts again

example

```
<html>
  <body>

    <script type="text/javascript">
      <!--
         var count;
         document.write("Starting Loop" + "<br />");

         for(count = 0; count < 10; count++){
            document.write("Current Count : " + count );
            document.write("<br />");
         }
```

```
        document.write("Loop stopped!");
      //-->
    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

FUNCTIONS

Is a collection of one or more script that can be made to run at any point during the life of a program.

The functions allow one to write very compact, easy to understand and reusable code by calling the functions whenever the action is required to perform a certain task

Defining the function

Function<identifier>(<parameter list>) // Identifier is the name of the function

{

<statements>

}

Example

Function getcolor() // function Name

{

Var color;

If(rowNumber>2)

rowNumber =1;

else if(rowNumber = = 1)

color = ”red”;

else

color = “white”;

rowNumber++;

return(color); // to pass the final to the function

```
}
```

example

```
<html>
<head>
<script language="javascript">
function showAlert()
{
var a;
var b;
a=prompt(“ enter  value of A”);
b=prompt(“ enter  value of B”);
var sum=0;
sum=a+b;
document.write(" sum= "+sum);
document.write("\t\tHello\nworld!\n");
document.write('\nWelcome to JavaScript');
}
</script>
</head>
<body>

showAlert();
</body>
</html>
```

Calling a function

A function waits in the wings until it is called unto the stage, you call a function by specifying its name followed by a parenthetical list of arguments if any

syntax

```
clearPage();

…….

,…….// arguments
```

```
<html>  <head>   <script language="javascript">

   function showmessage(){

  alert("How are you");    }

   </script>  </head>   <body>   <form>

  <input type="button" value="Click Here!"

  onclick="showmessage()" >

  </form>  </body></html>
```

JavaScript Timing Events

With JavaScript, it is possible to execute some code at specified time-intervals. This is called timing events.

It's very easy to time events in JavaScript. The two key methods that are used are:

- setInterval() - executes a function, over and over again, at specified time intervals
- setTimeout() - executes a function, once, after waiting a specified number of milliseconds

```
<html> <body>

<p>Click the button to display the current time.</p>
```

```
<button onclick="myFunction()">Try it</button>

<script>

function myFunction()

{

setInterval(function(){myTimer()},1000);

}

function myTimer()

{

var d=new Date();

var t=d.toLocaleTimeString();

document.getElementById("demo").innerHTML=t;

}

</script>

<p id="demo"></p>

</body>

</html>
```

EVENT HANDLING

Events are actions that occur on the web page usually as a result of something the user does, for example a button click is an event as is giving focus to a form element.

Java script events are divided into two groups:

The events that occurs ***automatically*** with out the input from the user such on error, onunload

User driven they occur with the input from the user such as onlick, onkeydown

Event handler specifies which java script code to execute,often event handlers are placed within the html tag which creates the object on which the event acts

Example

<a href =”www.gmail.com ” on mouseover=”popupFun(); >

// the event handler will be triggered when the mouse passes on the link therefore you place the event handler for a hyperlink ‘s mouse over inside the A tag

Common events

Event	**occurs when**	**event Handler**
Click	user clicks on the form or link	onClick
Change	user changes value of text, textarea,select element	onChange
Focus	user gives form element input focus	onFocus
Blur	user removes input focus from form element	onBlur
Mouseover	user moves mouse pointer over a link or anchor	onMouseOver
Mouseout	user moves mouse pointer off of link or anchor	onmouseout
Select	user selects form element’s input field	onSelect
Submit	user submits a form	onSubmit
Resize	user resize the browser window	onResize

Load	user loads the page in the navigator	onLoad
Unload	user exits the page	onUnload

FORM VALIDATION

JavaScript can be used to validate data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty
- has the user entered a valid e-mail address
- has the user entered a valid date
- has the user entered text in a numeric field

Required Fields

The function below checks if a required field has been left empty. If the required field is blank, an alert box alerts a message and the function returns false. If a value is entered, the function returns true (means that data is OK):

Example

```
<Script Type='Text/Javascript'>
Function Formvalidator(){
        // Make Quick References To Our Fields
        Var Firstname = Document.Getelementbyid('Firstname');
// Check Each Input In The Order That It Appears In The Form!
        If(Isalphabet(Firstname, "Please Enter Only Letters For Your Name"))
{
Function Notempty(Elem, Helpermsg){
        If(Elem.Value.Length == 0){
                Alert(“enter  some value ”);
                Elem.Focus(); // Set The Focus To This Input
                Return False;
        }
        Return True;
}
```

```
Function Isalphabet(Elem, Helpermsg){
        Var Alphaexp = /[A-Za-Z] /;
        If(Elem.Value.Match(Alphaexp)){
                Return True;
        }Else{
                Alert("enter btn a-z"rmsg);
                Elem.Focus();
                Return False;
        }
}
<HTML><BODY>
<Form Onsubmit='Return Formvalidator()' >
First Name: <Input Type='Text' Id='Firstname' /><Br />
<Input Type='Submit' Value='Check Form' />

</Form></HTML>
```

Example

```
<Script Type='Text/Javascript'>

Function Formvalidator(){
        // Make Quick References To Our Fields
        Var Firstname = Document.Getelementbyid('Firstname');
        Var Addr = Document.Getelementbyid('Addr');
        Var Zip = Document.Getelementbyid('Zip');
        Var State = Document.Getelementbyid('State');
        Var Username = Document.Getelementbyid('Username');


        // Check Each Input In The Order That It Appears In The Form!
        If(Isalphabet(Firstname, "Please Enter Only Letters For Your Name")){
                If(Isalphanumeric(Addr, "Numbers And Letters Only For Address")){
                        If(Isnumeric(Zip, "Please Enter A Valid Zip Code")){
```

```
                                        If(Madeselection(State, "Please Choose A State")){
                                                If(Lengthrestriction(Username, 6, 8)){

        Return True;
                                                        }
                                                }
                                        }
                                }
                        }
        }


        Return False;

}

Function Notempty(Elem, Helpermsg){
        If(Elem.Value.Length == 0){
                Alert(Helpermsg);
                Elem.Focus(); // Set The Focus To This Input
                Return False;
        }
        Return True;
}

Function Isnumeric(Elem, Helpermsg){
        Var Numericexpression = /^[0-9]+$/;
        If(Elem.Value.Match(Numericexpression)){
                Return True;
        }Else{
                Alert(" insert  numbers btn  0-9");
                Elem.Focus();
                Return False;
        }
}

Function Isalphabet(Elem, Helpermsg){
        Var Alphaexp = /^[A-Za-Z]+$/;
        If(Elem.Value.Match(Alphaexp)){
                Return True;
        }Else{
                Alert(Helpermsg);
                Elem.Focus();
                Return False;
        }
}
```

```
Function Isalphanumeric(Elem, Helpermsg){
        Var Alphaexp = /^[0-9A-Za-Z]+$/;
        If(Elem.Value.Match(Alphaexp)){
                Return True;
        }Else{
                Alert(Helpermsg);
                Elem.Focus();
                Return False;
        }
}

Function Lengthrestriction(Elem, Min, Max){
        Var Uinput = Elem.Value;
        If(Uinput.Length >= Min && Uinput.Length <= Max){
                Return True;
        }Else{
                Alert("Please Enter Between " +Min+ " And " +Max+ " Characters");
                Elem.Focus();
                Return False;
        }
}

Function Madeselection(Elem, Helpermsg){
        If(Elem.Value == "Please Choose"){
                Alert(Helpermsg);
                Elem.Focus();
                Return False;
        }Else{
                Return True;
        }
}

}
</Script>

<Form Onsubmit='Return Formvalidator()' >
First Name: <Input Type='Text' Id='Firstname' /><Br />
Address: <Input Type='Text' Id='Addr' /><Br />
Zip Code: <Input Type='Text' Id='Zip' /><Br />
State: <Select Id='State'>
        <Option>Please Choose</Option>
        <Option>Al</Option>
        <Option>Ca</Option>
        <Option>Tx</Option>
        <Option>Wi</Option>
```

```
</Select><Br />
Username(6-8 Characters): <Input Type='Text' Id='Username' /><Br />
<Input Type='Submit' Value='Check Form' />
</Form>
```

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server

What is a PHP File?

- PHP files can contain text, HTML, JavaScript code, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have a default file extension of ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, and close files on the server
- PHP can collect form data(using GET/ POST)
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with **<?php** and ends with **?>**:

```
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
Print “hi”;
?>

</body>
</html>
```

PHP Variables

PHP variables can be used to hold values (x=5) or expressions (z=x+y).

Variable can have short names (like x and y) or more descriptive names (age, carname, totalvolume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive ($y and $Y are two different variables)

Variables are "containers" for storing information

```
<?php
$x=5;
$y=6;
```

```
$z=$x+$y;
echo $z;
?>
```

Creating (Declaring) PHP Variables

PHP has no command for declaring a variable.

A variable is created the moment you first assign a value to it:

```
$txt="Hello world!";
$x=5;
```

Example

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

welcome.php

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br>
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

PHP echo and print Statements

There are some differences between echo and print:

- echo - can output one or more strings
- print - can only output one string, and returns always 1

DATE function

```
<?php
  $date_array = getdate();

  foreach ( $date_array as $key => $val )
  {
    print "$key = $val<br />";
  }

  $formated_date  = "Today's date: ";
  $formated_date .= $date_array['mday'] . "/";
  $formated_date .= $date_array['mon'] . "/";
  $formated_date .= $date_array['year'];

  print $formated_date;
?>
```

The HTML ACTION attribute and PHP

Create the following page, and call it basicForm2.php. This is your HTML. Notice the ACTION attribue.

```
<html>
<head>
<title>A BASIC HTML FORM</title>
</head>
<body>

<Form name ="form1" Method ="POST" Action ="submitForm.php">

    <INPUT TYPE = "TEXT" VALUE ="username" Name
    ="username">

    <INPUT TYPE = "TEXT" VALUE ="password" Name
    ="password">
```

```
            <INPUT TYPE = "Submit" Name = "Submit1" VALUE =
            "Login">

        </FORM>

        </body>
        </html>
```

Now create the following page, and call it submitForm.php:

```
        <?PHP

            $username = $_POST['username'];

            if ($username == "mercy") && ($password ==" john" {

                print ("Welcome back, friend!");

            }
            else {

                print ("You're not a member of this site");

            }

        ?>
```

LOGIN

```
<body>


    <h2>Enter Username and Password</h2>
     <?php
        $msg = '';


        if (isset($_POST['login']) && !empty($_POST['username']) &&
!empty($_POST['password'])) {


          if ($_POST['username'] == 'inte221' && $_POST['password'] == 'web') {
            $_SESSION['valid'] = true;
```

```
            $_SESSION['timeout'] = time();
            $_SESSION['username'] = 'inte221';

            echo 'You have entered valid use name and password';
          }
          else
          {
            $msg = 'Wrong username or password';
          }
        }
      ?>
    </div> <!-- /container -->

    <div class="container">

      <form class="form-signin" role="form" action="<?php echo
htmlspecialchars($_SERVER['PHP_SELF']); ?>" method="post">
        <h4 class="form-signin-heading"><?php echo $msg; ?></h4>
        <input type="text" class="form-control" name="username" placeholder="username =
tutorialspoint" required autofocus></br>

        <input type="password" class="form-control" name="password" placeholder="password
= web" required>
        <button class="btn btn-lg btn-primary btn-block" type="submit"
name="login">Login</button>
      </form>

      Click here to clean <a href="logout.php" tite="Logout">Session.

    </div>

  </body>
</html>
```

Logout.php

It will erase the session data.

```
<?php
  session_start();
  unset($_SESSION["username"]);
  unset($_SESSION["password"]);

  echo 'You have cleaned session';
  header('Refresh: 2; URL=login.php');
?>
```

2) WORKING WITH A RADIO BUTTON

The PHP code:

```
<?PHP

    $male_status = 'unchecked';
    $female_status = 'unchecked';

    if (isset($_POST['Submit1'])) {

        $selected_radio = $_POST['gender'];

        if ($selected_radio = = 'male') {

            $male_status = 'checked';

        }
        else if ($selected_radio = = 'female') {

            $female_status = 'checked';

        }

        Else echo"  No selected option"
```

```
        }

    ?>
```

PHP FORM VALIDATION

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST")
{
if (empty($_POST["name"]))
{$nameErr = "Name is required";}
else
{
$name = test_input($_POST["name"]);
// check if name only contains letters and whitespace
if (!preg_match("/^[a-zA-Z ]*$/",$name))
{
$nameErr = "Only letters and white space allowed";
}
}

if (empty($_POST["email"]))
{$emailErr = "Email is required";}
else
{
$email = test_input($_POST["email"]);
// check if e-mail address syntax is valid
if (!preg_match("/([\w\-]+\@[\w\-]+\.[\w\-]+)/",$email))
{
$emailErr = "Invalid email format";
}
}
```

```
if (empty($_POST[“website”]))
{$website = “”;}
else
{
$website = test_input($_POST[“website”]);
// check if URL address syntax is valid (this regular expression also allows dashes in the URL)
if (!preg_match(“/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i”,$website))
{
$websiteErr = “Invalid URL”;
}
}

if (empty($_POST[“comment”]))
{$comment = “”;}
else
{$comment = test_input($_POST[“comment”]);}

if (empty($_POST[“gender”]))
{$genderErr = “Gender is required”;}
else
{$gender = test_input($_POST[“gender”]);}
}

function test_input($data)
{
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span>* required field.</span></p>
<form method=”post” action=”<?php echo htmlspecialchars($_SERVER[“PHP_SELF”]);?>“>
Name: <input type=”text” name=”name” value=”<?php echo $name;?>“>
<span>* <?php echo $nameErr;?></span>
<br><br>
E-mail: <input type=”text” name=”email” value=”<?php echo $email;?>“>
<span>* <?php echo $emailErr;?></span>
<br><br>
Website: <input type=”text” name=”website” value=”<?php echo $website;?>“>
<span><?php echo $websiteErr;?></span>
<br><br>
Comment: <textarea name=”comment” rows=”5″ cols=”40″><?php echo
$comment;?></textarea>
```

```
<br><br>
Gender:
<input type=”radio” name=”gender” <?php if (isset($gender) && $gender==”female”) echo
“checked”;?>  value=”female”>Female
<input type=”radio” name=”gender” <?php if (isset($gender) && $gender==”male”) echo
“checked”;?>  value=”male”>Male
<span>* <?php echo $genderErr;?></span>
<br><br>
<input type=”submit” name=”submit” value=”Submit”>
</form>
<?php
echo “<h2>Your Input:</h2>”;
echo $name;
echo “<br>”;
echo $email;
echo “<br>”;
echo $website;
echo “<br>”;
echo $comment;
echo “<br>”;
echo $gender;
?>

</body>
</html>
```