Scripting Language

A high-level programming language that is interpreted by another program at runtime rather than compiled by the computer's processor as other programming languages (such as C and C++) are. Scripting languages, which can be embedded within HTML, commonly are used to add functionality to a Web page, such as different menu styles or graphic displays or to serve dynamic advertisements. These types of languages are client-side scripting languages, affecting the data that the end user sees in a browser window. Other scripting languages are server-side scripting languages that manipulate the data, usually in a database, on the server.

Client Side Vs Server Side Script

When you create a dynamic html page the code that makes the changes to the web page must be run in one of two places. It can be client-side code executed by the user's web browser, or it can be server-side code executed by the server. There are advantages and disadvantages to using one over the other, and a well-planned mix of the two can have a dramatic effect on your web page.

The advantages of server-side scripting are many. Because all of the server side code is executed before the HTML is sent to the browser, your code is hidden from client/ web page viewer. Server-side code is also the only choice if you want to access files and directories on the local machine. Server-side code is also browser independent. Because the HTML code returned by the server is simple HTML, you do not have to worry about the version of browser the client is using. The disadvantage to server-side scripting is that the server must use valuable resources to parse each page it sends out. This could slow down your web site.

Client-side scripting can be very useful. The major advantage is that each web browser uses its own resources to execute the code found on the web page. This eases the burden on the server. The disadvantages are that you can not prevent the user from seeing your code and that you can not use client-side code to access local files, directories, or databases.

Java Script

JavaScript is an object-oriented scripting language used to enable programmatic access to objects within both the client application and other applications. It is primarily used in the form of client-side JavaScript, implemented as an integrated component of the web browser, allowing the development of enhanced user interfaces and dynamic websites. Combined use of HTML, CSS and Javascript can be called Dynamic HTML.
It has following features:

1) It is untyped language. So a variable hold any kind of value.
2) Javascript is case sensitive language.
3) It is interpreted language.
4) Every statement must be closed by semi-colon.

It is can be used in two way:
1) Embedded in an HTML document between script tags

```
<script language="javascript">
        JavaScript statements go here
</script>
```

2) In an external file which is loaded using

```
<script language = "javascript" src="program.js" ></script>
```

Where do we put <script>?

In the head section of the HTML document

Here it is read before the HTML document in the body is parsed. Any code, except function definitions, will be executed immediately.

In the body section of the HTML document

Here it is read while the HTML document is being parsed. When the parser sees the **<script>** tag it stops parsing the document and interprets the JavaScript code.

document.write() method

By this method we can write anything on the document or say web page. For example

document.write("IIPS");

the above javascript statement will write the IIPS in bold letter on web page client area. One thing we have to remember for this method is this the place where this statement is written, when this statement is executed in the script tag open area or say outer the any ***function body*** then it will show the arguments information and other html information but if this statement is executed with in the ***function body*** then it will clear the other information already written on the page.

```
<script language="javascript">
        document.write("<b>IIPS</b>");
</script>
<body>
        Some text….
</body>
```

or

```
<script language="javascript">
        function fun()
        {
                document.write("<b>IIPS</b>");
        }
</script>
<body onload="fun()">
        Some Text.……..
</body>
```

document.getElementById() method

By this method we will receive the element in javascript code by its id value and then modify or say perform some operation over it. For example we have a tag whose id is "d1" in our HTML code, then in javascript code it could be…

```
var x  =document.getElementById("d1");   /*By this we receive d1 element in x variable.*/
x.innerHTML = "<b>Some New Information which is shown in bold letters.</b>";
x.style.backgroundColor = "#abababab";
x.className="CSSClassName";
```

Javascript Event functions

onAbort - invoked when the user aborts the loading of an image by clicking the STOP button

onClick - invoked when the user clicks the specified object

onFocus - invoked when the target object receives focus

onBlur - invoked when the target object loses focus

onMouseOver - invoked when the user passes the mouse over the target object

onMouseOut - invoked when the mouse pointer leaves the target object

onSubmit - invoked when the user clicks the Submit button in a form

onChange - invoked when the user changes the contents of a text field

onSelect - invoked when the user selects the contents of a text field

onReset - invoked when the user clicks the Reset button in a form

onLoad - invoked when the target image or document is loaded

onUnload - invoked when the target image or document is unloaded

The following table shows you what events are supported on the HTML Form Elements.

JavaScript Events and HTML Form Elements

Events / HTML Elements	Blur	Click	Change	Focus	Load	Mouseover	Select	Submit	Unload
Button		X							
Checkbox		X			X				X
Document								X	
Form						X			
Link		X							
Radio		X							
Reset		X							
Selection	X		X	X					
Submit		X							
Text	X		X	X			X		
Textarea	X		X	X			X		

get the javascript time

The Date object has been created, and now we have a variable that holds the current date! To get the information we need to print out, we have to utilize some or all of the following functions:

- **getTime()** - Number of milliseconds since 1/1/1970 @ 12:00 AM
- **getSeconds()** - Number of seconds (0-59)
- **getMinutes()** - Number of minutes (0-59)
- **getHours()** - Number of hours (0-23)
- **getDay()** - Day of the week(0-6). 0 = Sunday, ... , 6 = Saturday

- **getDate()** - Day of the month (0-31)
- **getMonth()** - Number of month (0-11)
- **getFullYear()** - The four digit year (1970-9999)

```
<script type="text/javascript">
<!--
var currentTime = new Date();
var month = currentTime.getMonth()+1;
var day= currentTime.getDate();
var year = currentTime.getFullYear();
document.write(month+"/"+day+"/"+year);
//-->
</script>
```

To open a new window, you will need to use yet another ready-made JavaScript function. Here is what it looks like:
window.open('url to open','window name','attribute1,attribute2')

This is the function that allows you to open a new browser window for the viewer to use. Note that all the names and attributes are separated with a comma rather than spaces. Here is what all the stuff inside is:

1. **'url to open'**
   This is the web address of the page you wish to appear in the new window.
2. **'window name'**
   You can name your window whatever you like, in case you need to make a reference to the window later.
3. **'attribute1,attribute2'**
   There are lot of attribute that you can set to change the behavior of the opened window.

Window Attributes

Below is a list of the attributes you can use:

1. **width=300**
   Use this to define the width of the new window.
2. **height=200**
   Use this to define the height of the new window.
3. **resizable=yes or no**
   Use this to control whether or not you want the user to be able to resize the window.
4. **scrollbars=yes or no**
   This lets you decide whether or not to have scrollbars on the window.
5. **toolbar=yes or no**
   Whether or not the new window should have the browser navigation bar at the top (The back, foward, stop buttons..etc.).
6. **location=yes or no**
   Whether or not you wish to show the location box with the current url (The place to type http://address).
7. **directories=yes or no**
   Whether or not the window should show the extra buttons. (print, Page buttons, etc...).
8. **status=yes or no**
   Whether or not to show the window status bar at the bottom of the window.

9. **menubar=yes or no**
   Whether or not to show the menus at the top of the window (File, Edit, etc...).

All right, here's an example code for opening a new window:

<body onLoad="window.open('http://www.google.com','mywindow','width=400,height=200')">
Other HTML Code
</body>

HTML Form

An HTML form provide data gathering functionality to a web page. This is vary useful if the web site is used to advertise and sell products. HTML forms provide a full range of GUI (Graphical User Interface) controls. Additionally, HTML forms can automatically submit data collected in its controls to a web server.

The data submitted can be processed at the web server by *ASP, JSP, Java Servlet* or any other server side programming language.

JavaScript allows the validation of data entered into a form at the client side. JavaScript can be used to ensure that only valid data is returned to a web server for further processing.

User input is captured in a 'Form'. HTML provide the <FORM> </FORM> tags with which an HTML form can be created to capture user input.

<form name="formName" method="GET" action="TargetWebPagePath">
form element / form body
</form>

action is the target web page address. This web page could be Any server-side script page like .ASP,.JSP etc.

method attribute can be GET / POST depending on user requirement. As per functionality both GET and POST methods were same. Difference is GET method will be showing the information to the users. But in the case of POST method information will not be shown to the user.

The data passed using the GET method would be visible to the user of the website in the browser address bar but when we pass the information using the POST method the data is not visible to the user directly.

Also in GET method characters were restricted only to 256 characters. But in the case of POST method characters were not restricted.
Get method will be visible to the user as it sends information appended to the URL, but Post will not be visible as it is sent encapsulated within the HTTP request body.

About the data type that can be send, with Get method you can only use text as it sent as a string appended with the URL, but with post is can text or binary.

About form default, GET is the default method for any form, if you need to use the post method; you have to change the value of the attribute "method" to be Post.

Text Fields

Before we teach you how to make a complete form, let's start out with the basics of forms. Input fields are going to be the meat of your form's sandwich. The <input> has a few attributes that you should be aware of.

- *type* - Determines what kind of input field it will be. Possible choices are text, submit, and password.
- *name* - Assigns a name to the given field so that you may reference it later.

- *size* - Sets the horizontal width of the field. The unit of measurement is in blank spaces.
- *maxlength* - Dictates the maximum number of characters that can be entered

<form method="post" action="targetServerSideWebPage Path">
Name: <**input type**="text" **size**="10" **maxlength**="40" **name**="name">
Password: <**input** type="password" **size**="10" **maxlength**="10" **name**="password">
</form>
For Reading the values in Javascript:
document.forms('formName').txtTextboxName.value
document.forms('formName').txtPasswordTextboxName.value

Type = "text" will create simple text box and Type = "Password" will create text box, which accept password. It is also a text box but it show the information in start / dot means it hides the information for display.

HTML Radio Buttons

Radio buttons are a popular form of interaction. You may have seen them on questionnaires, and other web sites that give the user a multiple-choice question. Below are a couple attributes you should know that relate to the radio button. In Radio button, name attribute is same for the group of radio button. Radio button values are mutually exclusive for the specific group.

- *value* - specifies what will be sent if the user chooses this radio button. Only one value will be sent for a given group of radio buttons (see *name* for more information).
- *name* - defines which set of radio buttons that it is a part of. Below we have 2 groups: shade and size.
- checked-This attribute define that the Radio Button is checked by default

HTML Code:
<form method="post">
*What kind of shirt are you wearing?
*
*Shade: <**input type**="radio" **name**="shade" value="dark">Dark*
*<**input type**="radio" **name**="shade" value="light" Checked>Light
*
*Size: <input type="radio" **name**="size" value="small">Small*
*<input type="radio" **name**="size" value="medium" Checked>Medium*
*<input type="radio" **name**="size" value="large">Large
*
</form>

The output of above HTML code will be as follows :

Radios:
What kind of shirt are you wearing?
Shade: Dark Light
Size: Small Medium Large

For Reading the values in Javascript:
document.forms('formName').radioButtonName[indexOfRadioButton].checked==true/false

HTML Check Boxes

Check boxes allow for multiple items to be selected for a certain group of choices. The check box's *name* and *value* attributes behave the same as a radio button.

HTML Code:
<form method="post" >

Select Programming Language Known.
<input type="checkbox" name="LangC" value="C"> C
<input type="checkbox" name="LangC++" value="C++"> C++
<input type="checkbox" name="LangHTML" value="HTML"> HTML
<input type="checkbox" name="LangJava" value="Java"> Java
</form>

CheckBox

For Reading the values in Javascript:

document.forms('formName').checkBoxName.checked==true/false

HTML Drop Down Lists

Drop down menus or combo boxes are created with the <select> and <option> tags. <select> is the list itself and each <option> is an available choice for the user.

HTML Code:

<form method="post" >
College Course?
*<**select** name="course">*
<option>Choose One</option>
<option>MCA</option>
<option>M.Tech.</option>
<option>MBA</option>
<option>B.Com(Hons.)</option>
</select>
</form>

For Reading the values in Javascript:

First we get the Selected Index of element, then we get the text associated with that option.

var x=document.forms('formName').selectName.selectedIndex;
var optionSelected = document.forms('formName').selectName.options[x]

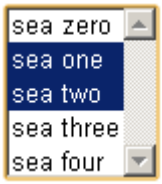
HTML Selection Forms

Yet another type of form, a highlighted selection list. This form will post what the user highlights. Basically just another type of way to get input from the user. Its is variation of Drop Down list, in which only one item /option is shown to user. But in Selection forms user can see more than one item or option at a time. It can be possible to select more than one element at a time in Selection.

The *size* attribute selects how many options will be shown at once before needing to scroll, and the *selected* option tells the browser which choice to select by default.

HTML Code:

<form method="post">
Musical Taste
*<**select multiple** name="music" size="4">*
<option value="zero" >sea zero</option>
<option value="one" selected >sea one</option>
<option value="two" selected > sea two</option>
<option value="three" > sea three</option>
<option value="four" >sea four</option>
</select>
</form>



For Reading the values in Javascript:

First we find that if the option is selected or not with loop , then we get the text associated with that option.

```
var obj = document.getElementById('selectID');
var i;
var count = 0;
for (i=0; i<obj.options.length; i++) {
   if (obj.options[i].selected) {
     alert(obj.options[i].value+ " is Selected." ;
     count++;
   }
  }
```

HTML Text Areas

Text areas serve as an input field for viewers to place their own comments onto. Forums and the like use text areas to post what you type onto their site using scripts. For this form, the text area is used as a way to write comments to somebody.

Rows and **columns** need to be specified as attributes to the <textarea> tag. Rows are roughly 12pixels high, the same as in word programs and the value of the columns reflects how many characters wide the text area will be. i.e. The example below shows a text area 5 rows tall and 20 characters wide.

HTML Code:

```
<form method="post" >
    <textarea rows="5" cols="20" name="comments">
            Enter Comments Here
    </textarea>
</form>
```

For Reading the values in Javascript:

document.forms('formName').txtTextAreaName.value

HTML Buttons

There are 3 type of HTML button used in HTML

One is Submit button: which sends the information to the Target Web page which is specified in action attribute of <form> tag.

<input type = "submit" value = "Save Information">

Second is Reset button: which resets the forms elements to default values. Say empty the text box, password field.

<input type = "reset" value = "Clear">

Third is Button : its is a general button. We have to associate the onClick event to it.

<input type = "button" value = "Display" onClick= "JavaScriptFunction">

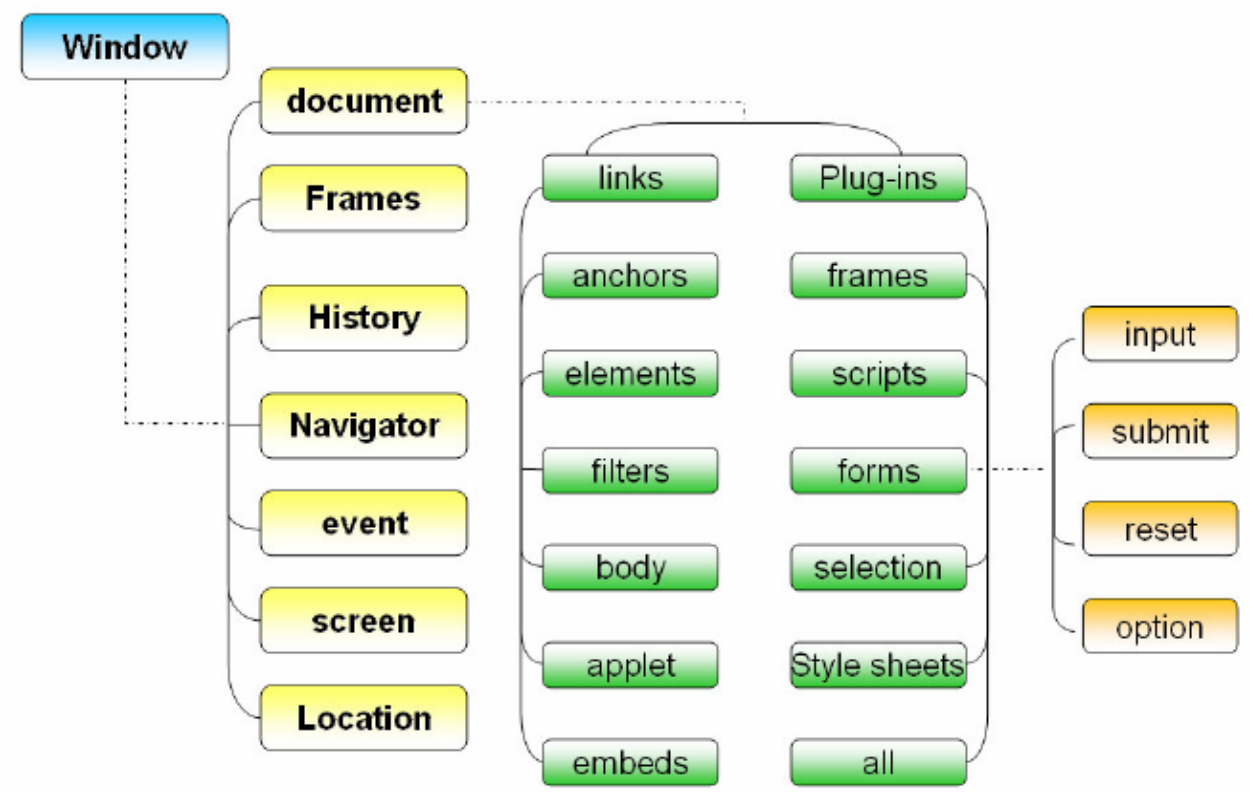
What is Document Object Model (DOM)?

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The HTML Document Object Model (HTML DOM) defines a standard way for accessing and manipulating HTML documents. The DOM presents an HTML document as a tree structure (a node tree), with elements, attributes, and text. The DOM (Document Object Model) gives generic access to most elements, their styles and attributes in a document to change anything on a page, JavaScript needs access to all elements in the HTML document. This access, along with methods and properties to add, move, change, or remove HTML elements, is given through the Document Object Model (DOM).

According to DOM

1 The entire document is a document node
2 Every HTML tag is an element node
3 The texts contained in the HTML elements are text nodes
4 Every HTML attribute is an attribute node

Complete Document Object Model diagram



Window Object

The **top level object** in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag. The window object contains everything that is accessible to programs through the object model: the element, frames, images, browser and almost everything that needs to access through the browser.

Location object-

The Location object is automatically created by the JavaScript runtime engine and **contains information about the current URL**. Location object is part of the Window object and is accessed through the **window.location** property. The location object contains all the information on the location that the window is currently displayed and all the details on that location like port, the protocol.

window.location - location is a property of window object. Location property returns current location of a web file. We can transfer location from one web page to another.

<input type = “button” value=”Go To Google” onClick = “***window.location = ‘http://www.google.com’;***”

Navigator Object

The navigator object enables to access general information about the browser program. – What the browser does and does not support. The Navigator object is automatically created by the JavaScript runtime engine and contains information about the client browser.

appName – returns name of the browser that is processing the script

appVersion – returns version number of the browser

History Object

The History object is automatically created by the JavaScript runtime engine and consists of an array of URLs. These URLs are the URLs the user has visited within a browser window. The History object is part of the Window object and is accessed through the **window.history** property. **history.length** – The length property specified how many URLs are contained in the current history object. The URLs saved are identical to those shown in browser history list

Methods of history object

history.forward() – Loads the next URL in the history list

history.back() - Loads the previous URL in the history list

Event Object

An event is a browser way of telling user that user is interacting with browser

clientX -The X position of the mouse relative to the client area of the window

clientY -the Y position of the mouse relative to the client area of the window