Java Script

Introduction:

- JavaScript is a compact, object-based scripting language for developing client Internet applications.
- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language - a scripting language is a lightweight programming language.
- A JavaScript is lines of executable computer code.
- A JavaScript is usually embedded directly in HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- Everyone can use JavaScript without purchasing a license.
- All major browsers, like Netscape and Internet Explorer, support JavaScript.
- Javascript was developed by Netscape as *Live Script* - changed to *JavaScript* when endorsed by Sun 1993, version 1.0 released with Netscape 2.0.
- JavaScript is a powerful scripting language that is also capable of performing extensive form data collecting and form processing functions.

Like many other programming languages java script is programming language, which mainly used with web pages. This language is very useful when providing interactivity to a web page. A java script is a program which is included on an HTML page. In the html page, it is enclosed within <script> and ends it up with </script>. When used with a browser, it knows that this is a program written under java script and gives the result instead of giving the text of the script. The<script> tag is written with the <head> section of the html page. It can also be written within the<body> section.

Java & JavaScript:

Comparing JavaScript with Java

While comparing JavaScript with java first lets see the basic differences between them

JavaScript	Java
1. JavaScript is a small, lightweight programming language.	1. Java is a full-blown powerful, sophisticated programming language.
2. Developed by Netscape communications.	2. Developed by Sun Microsystems.
3. Scripting language interpreted at runtime.	3. True programming compiled to byte-code.
4. Object-based, has limited number of built-in objects.	4. Object-oriented, can create their own classes.
5. Not fully extensible.	5. Fully extensible.
6. Code integrated with, and embedded in HTML.	6. Applies distinct from HTML (accessed from HTML pages).
7. Variables type must not be declared.	7. Variables type must be declared.
8. JavaScript source code can be viewed by everybody using “view source command”.	8. Your Java source is hidden because it's only the compiled byte-code, which the browser uses, but this is not a *guarantee* of security.

Java & JavaScript have only name in common and nothing else. They are poles apart. Java is basically a programming language which is enhancement of C++ having a distinctive feature of being able to run on various machines and operating systems like Windows NT, Mac Os, UNIX.

Java script is an object-oriented language that allows creation of interactive web pages. Java script allows user entries, which are loaded into an HTML form to be processed as required. Java script offers several advantages to a web developer. The strengths of java script can be easily and quickly used to extend the functionality of HTML pages.

Netscape & Java Script: Java script is scripting language created by Netscape hence JavaScript works best with the Netscape suite of client and server products. The Netscape client 'browser' product is called Netscape communicator. The default scripting language that Netscape communicator understands is java Script.

Netscape server product is called Netscape commerce server. The default scripting language that Netscape commerce server understands is JavaScript.

Netscape has a product called 'Live wire', Which permits server side , JavaScript code, to connect to Relational Database management systems (RDBMS) like oracle, My SQL server, My SQL etc 'live wire' database drivers also support a no of non-relational database.

Client Side JavaScript:

Client side java Script is embedded into a standard HTML program. JavaScript is embedded between the **<script>…….</script>** html tags. These tags are embedded within the ***<head>……</head>*** *or* ***<body>…..</body>*** tags of the html program.

Java script is embedded into an html program because JavaScript uses the filename.html and http protocol to transport itself from the web serve to the client's browser where the JavaScript executes and processes client information. Only a browser that is JavaScript enabled will be able to interpret JavaScript code. Netscape communicator does this best as java script is the natural language of Netscape communicator. Microsoft is Internet explorer also interprets JavaScript.

Capturing user Input:

Web site interactivity starts from being able to capture user input. The **<form> ……</form>** html tags can be used to create a user request-form. Between these html tags, the html **<input> </input>** tags can be used in the html form to facilitate the captures of user data.

The HTML objects used in html form creation are '**Text' 'Text area' , 'Radio Buttons', 'push Button', 'check Boxes'** and so on. These will be passes as '**values**' to the types attributes of the <input> </input> tags. Once the html form has been coded in fielname.html JavaScript can be embedded in the same html file to make it interactive and facilitate client side data validation and processing.

The HTML FORM objects are used to capture user input while client side, JavaScript is due do validate and process such user INPUT. Once user INPUT posses the validation tests applied and has been processed by client side, JavaScript, the form data captured will have to be passed to the web server where from where the HTML file originated .

At the web server an appropriate program (e.g. JavaScript , VB script, Perl etc). Will accept the user Input. Based on user Input info will be assembled on the web server, converted to an HTML file and returned to the user.

JavaScript Uses

The most frequent uses of JavaScript are:

- For displaying the clock
- Used for creating Drop-down menus.
- Showing Alert messages.
- Displaying Popup windows.
- Validating HTML Form Data.

Advantages of JavaScript

- Cross Browser supporting**:**
  - This means that the javascript codes are supported by various browsers like Internet Explorer, Netscape Navigator, Mozilla etc.
- Platform Independent:
  - Javascript codes can be executed in any platform like Linux, Microsoft Windows and many other operating systems.
- Lightweight for fast downloading:
  - The javascript codes runs directly in the client machine. The code should be downloaded all the way from server to the client and this time duration is very minimum and the executing the codes of javascript is also very fast.
- Validating Data:
  - The data can be validated in the two different way:
    - Validating in server side or in server machine.
    - Validating in client side or in client machine.
  - In this two different types of validation of data the second one is much more faster, and this is done through javascript.
- Sophisticated User Interfaces:
  - By using javascript you can create a user interactive interfaces that can communicate with the user and the Browser.
- In-Built software:
  - To you don't need any extra tools to write JavaScript, any plain text or HTML editor will do, so there's no expensive development software to buy.
- Easy to Learn:
  - The javascript programmer should know the minimal syntax of javascript since it supports many syntax and data types as C and C++.
  - It's also an easy language to learn, and there's a thriving and supportive online community of JavaScript developers and information resources.
- Designed for programming User-Events:
  - Javascript supports Object/Event based programming. So the code written in javascript can easily be break down into sub-modules.

Disadvantages of Javascript:

- Launching an application on the client computer
  - Javascript is not used to create stand-alone application; it is only used to add some functionality in any web page.
- Reading or writing files:
  - Javascript cannot read and write files into the client machines. It can only be used as a utility language to develop any web site.
- Retrieving the text contents of HTML pages or files from the server.
- Reading and Writing files to the server:
  - Javascript can read and write to any file in the server as well.
- Sending secret e-mails from Web site visitors to you:
  - Javascript cannot be used to send email to the visitors or user of the web site. This can be done only with the server side scripting.
- Cannot create Database Application:
  - By using javascript you cannot connect the web site to the database. For this you need to use server-side scripting.
- Browser Compatibility Issues:
  - Not all browsers support the javascript codes. The browser may support javascript as a whole, but may not support the codes or lines of codes written in javascript and may interpret differently.
- Javascript does not implement multiprocessing or multithreading.
- Use printers or other devices on the user's system or the client-side LAN.
- Javascript has limitations of writing in a client machine. It can only write the cookie in client machine that is also of a certain size i.e. 4K.

Java Scripts Syntax:

```
<script language = "JavaScript">
        // JavaScript code snippet written here
</script>
```

- Netscape Communicator- default JavaScript
- Internet Explorer - default VB Script.

Programming Technique:

Variables: It is used to store values that can be used in other parts of a program. It can be

upper case letter (A to Z),

lowercase letter (a to z).

underscore character. (_). Or

dollor sign character ($).

*(The dollar sign (**$**) character is reserved for machine generated code and should not be used in script.)*

Variable name are case sensitive.

Adding Javascript codes

We have to focus on three major steps while adding javascript codes in HTML document.

1. Use a **<script>** tag to tell the browser that you are using javascript.

Example:
<script language = "Javascript"**>**

2. Write the javascript codes.

Example:
<script language = "Javascript"**>**
// javascript codes HERE
</script>

3. Test the scripts.

- While writing the javascript codes there can be many types of errors like:
  - Human Errors.
  - Browser compatibility issues.
  - Incorrect behavior based on the different operating systems.
- So when using javascript, be sure that you test your scripts out on a wide variety of systems and setups.
- You can add javascript codes between the **<head>** tag as well as in the **<body>** tag. And you can have multiple **<script>** tag in a single web page, but you must have the closing **</script>** tag of each opening tag.

First Example in Javascript

```
<Script language = "javascript">
        <!- -
                document.write ("Hello World");
        // - - >
</script>
```

- The first line <script language = "javascript"> tells the browser that the code written between the ***<script> and </script>*** tags are the javascript codes.
- Some browsers don't support scripts, and may display the scripts as the HTML content in the webpage.
- To prevent this you must write the javascript codes in between ***<! - -*** and ***// - ->***.
- The two forward slashes at the end of the line are the javascript comments.
- You cannot put this slashes to the starting of the comment line like (**// <!- -**), because the older browser will display it.
- Now the above code will display **Hello World** in the browser.

Using External Javascript Files

- While using external javascript files in a web-page, the javascript files must have the three main features:
  - First, the file that you are importing must be a valid javascript file.
  - Second, the file must have the extension "**.js**".
  - Lastly, you must know the location of the file.
- Here is the example of using external javascript files in a web-page.

```
<HTML>
        <HEAD>
                <Script src = "myExample.js"></script>
                <TILTE></TITLE>
        </HEAD>
<BODY>
</BODY>
<HTML>
```

- Here myExample.js contains the javascript codes and whatever is written in the file will be displayed in the browser.
- In this condition the myExample.js file and the above HTML file must be in the same location.
- For implementing the external javascript file you must have at least two files: one for HTML codes and other for javascript file.

Data Types

Number: Consists of integer and floating point numbers.
Ex: 33, 12.10,-31.24, 2E4
Boolean: Consists of logical value *true* and *false.* True to 1 and False to 0.
String: Consists of string value that are enclosed in single quotes or double quotes.
Ex:, "Anil Yadav", '222, Biratnagar, Nepal'

CREATING VARIABLE NAME:

Syntax:
var <variable name> = value;

```
Eg. var first_name;
var last_name= "Yadav"
var  phone = 1234567;
```

```
<html>
<head>
<script language = "JavaScript">
<!--
        var name=prompt ("Enter your name", "Name");
//-->
</script>
</head>
<body>
<script language = "JavaScript">
        document.write ('<h2>Hello ' + name + '</h2>');
</script>
</body>
</html>
```

Here ***prompt ( )*** method picks up a string from the user which is then assigned to the variable name. document.write embedded in the **<body> </body>** tags write the content of variable name to the client browser.

JavaScript Array:

Arrays are JavaScript objects that are ***capable of storing a sequence of values***. These values are stored in ***indexed locations*** within the array. An array must be *declared before* it is used. An array can be declared using any one of the following techniques.

```
arrayName = new Array(array length)
arrayName= new Array( )
```

Eg.

```
cust_orders=new Array ( )
cust_orders[50]= “lion pencils”
cust_order[100]= “Natraj eraser”
```

Dense Arrays: A dense array is an array that has been ***created with each of its elements being assigned a specified value***. They are declared and initialized at the same time.
e.g

```
array name = new array (value 0, value1,……., value n )
```

Arrays have several methods associated in manipulation of array with its element content.

Join ( ): returns all elements of the array joined together as a single string.
Reverse ( ): reverse the order of the element in the array.

Join ( ) Method – Reverse () Program

```
<html>
<head><title>JavaScript Lab </title></head>
<body>
<script language ="javascript">
<!--
friends=new Array(5);
friends[0] = "Ramesh Shrestha";
friends[1] = "Maninsha Sah";
friends[2] = "Bibek Baral";
friends[3] = "Sanjay Yadav";
friends[4] = "Anil Yadav";

document.write(friends[0] + "<BR>");
document.write(friends[1] + "<BR>");
document.write(friends[2] + "<BR>");
document.write(friends[3] + "<BR>");
document.write(friends[4] + "<BR>");
document.write ("<BR>");
join_crit= friends.join();
document.write(join_crit);
```

```
document.write ("<BR>");
rev=friends.reverse();
document.write(rev);
//-->
</script>
</body>
</html>
```

Operators & Expressions in Java Script:

Arithmetic operation:

+	addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
- -	Return the value then decrement
++	Return the value then increment.

Logical operation:

&&	logical AND
¦¦	Logical OR
!	Logical NOT

Comparison operation: Compare two values.

= =	equal
= = =	strictly equal
!=	not equal
! = =	strictly not equal.
<	less then
>	greater then
<=	less than or equal to
>=	greater than or equal to

Assignment operator:

a = b	It set a value to the value of b.
a+=b	It is same as a = a+b
a*=b	It is same as a = a*b
a-=b	It is same as a = a-b
a/=b	It is same as a = a/b
a%=b	It is same as a = a%b

Ternary operator:

Javascript support the conditional expression operator. They are **?** and **:**

Syntax:

condition ? value1 : value2

The condition expressed must return a true or false. If the condition is true, value1 is the result of the expression; otherwise value2 is the result of the expression.

Special Operator

The delete operator: The operator is used to delete a property of an object or an element at an array index.

E.g., delete myArray[5] -> delete the sixth element of the array.

The new operator: The new operator is used to create an instance of an object type.

E.g., myArray=new Array()

JavaScript Programming Construct

*/*Program to greets a user with a message when a user leaves this page, display the good-bye alert dialog box */*

```
<html>
<head><title>iBirat Technologies</title></head>
<script language = "JavaScript">
var name = "";
function hello()
        {
        name=prompt ("Enter your name", name);
        alert('Greeting ' + name + ' Welcome to iBirat Technologies!');
        }
function goodbye()
        {
        alert ('goodbye ' + name + ' sorry to see you go!');
        }
</script>
</head>
<body onLoad ="hello();" onUnload= "goodbye()">
<img src = "pu.gif">
<script language= "Javascript">
        document.write("Welcome to Java Script")
</script>
</body>
</html>
```

Comments: For putting comments in the program we use **/*…….*/**. Anything written between these symbols are not executed.

Using Quotes:

Character	Meaning
\'	Single quote.
\"	Double quote
****	back slash
\t	horizontal tab
\b	back space.

e.g.

```
<html>
<head><title>…….</title></head>
<body>
<script language = "JavaScript">
document.write('I Said, \" That \'s is mine! \ <br>')
document.write('He said, "NO it\'s not . <br>')
document.write('I hope that \'s clear now')
</script>
</body>
</html>
```

Output:

I said, “That\’s mine!”

He said, "No it's not.
I hope that's clear now.

Using Alert:

You can alter the user about anything by using alert symbol.

e.g. alert ("Welcome to iBirat Technologies")

Functions: These can be called as many times as needed during the running of the script. E.g . Let us say that you have same info. That a user typed into a form and you saved it using java script. If you need to use that information again and again, you could repeat the same code over and over in your script.

Eg.

```
Function Purbanchal (message)
{
        alert ("write the message")
}
```

```
<html>
<head>
<title>Java Script Alert</title>
</head>
<body>
<script language= "javascript">
function purbanchal(message)
{
alert(message)
}
</script>
<h3>faculty run by PU</h3>
<form>
<input type = "button" value= "computer" onclick= "purbanchal('Welcome to B.E Computer')">
<input type = "button" value = "E&C" onclick ="purbanchal ('Welcome to B.E electronic & comm.')">
<input type = "button" value= "civil" onclick ="purbanchal('Welcome to B.E civil')">
</form>
</body>
</html>
```

Function in Java Script:

Built in Functions:

1. **Eval function:** The eval( ) function can be used to convert a string expression to a numeric value.
E.g

```
Var grand_total = eval("10*10+5");
```

This will result the value 105 to variable grand_total.

2. **ParseInt ( ):** The parseInt ( ) function is used to ***convert a string value to integer***. ParseInt ( ) returns the *first integer contained in a string* or *0 if the string does not begin with an integer*.
Eg.

```
var string2Num = parseInt ("123xyz");
```

Results 123 assigned to variable string2Num

```
var string2Num = parseInt ("xyz");
```

Results a "**NaN**" (***not a number***) assigned to variable string2Num.

```
var string2Num =pasetFloat("1.2xyz");
```

Result 1.2 being assigned to variable string2Num.

Use as eval( ):

```
<html>
<head> <title>…….</title></head>
<body>
<script language = "JavaScript">
document.write (eval("5*6"))
document.write ('<br>')
document.write (eval("5/6"))
document.write ('<br>')
var grand_total = eval("10*10+5");
document.write (grand_total);
</script>
</body>
</html>
```

Output

30
0.8333333333333334
105

Dialog boxes:

Java Script provides the ability to pickup user input or display small amount of text to the user by using dialog boxes. These dialog boxes appears as separate windows and their contents depends on the information provided by the user.

There are 3 types of dialog boxes provided by Java Script.

1. The alert dialog box: The simplest way to direct small amount of textual output to a browser's window is to use alert dialog box. The alert dialog box displays the string passed to the alert ( ) method, as well as an **OK** button.

The Java Script and the HTML program, in which this code snippet is set, will not continue processing until **OK** button is clicked.

Syntax:

```
alert("message");
Eg.alert("click ok to continue");
```

Eg.

```
<html><head>
<title>……..</title></head>
<body>
<script language = "Javascript">
alert ("welcome to purbanchal university");
document.write("<img src = 'ibirat.gif'>");
</script>
</body>
</html>
```

Output:

When **OK** is clicked, ibirat.gif image is opened.

2. The confirmation dialog box: As the name suggest, this dialog box serves as technique for confirming user action. The confirm dialog box displays the following information.

- A pre-defined message.
- **OK** and **CANCEL** button.

The confirm dialog box, causes the program execution to halt until user action takes place. User action can be either the "**OK**" button begins clicked or the "**CANCEL**" button begins clicked which causes the following action to take place.

- Clicking on **OK** button cause TRUE to be passed to the program which is called the confirm dialog box.
- Clicking on **CANCEL** button causes FALSE to be passed to the program.

Syntax: conform ("message")

e.g confirm ("Are you sure to exit out of the system?");

```
<html>
<head> <title>Confirm</title></head>
<script language="javascript">
var question = "What is 10+10?";
var answer = 20;
var correct = '<IMG src = "image1.gif">';
var incorrect = '<IMG src = "image2.gif">';
var Response=prompt(question,"0");
for (count = 1;count < 3; count++){
                    if(Response != answer){
                                if(confirm("Wrong, press ok for another chance")){
                                            Response = prompt(question,"0");
```

```
                        }
                        else{
                        alert("Better luck next time");
                        count = 3;
                        }
                }
                else{
                alert ("Great!! You are right");
                count =3;
                }
        }
var output = ((response == answer)? correct: incorrect)
document.write("<br>");
document.write(output);
</script>
</head>
<body>
...
</body>
</html>
```

Output:

This program asks a question and accepts an answer. The user is given 3 chances. The 2nd and 3rd chance to provide an answer can be accepted, or rejected, if accepted the program prompts for an answer again.

(1)

(2)

(2 Or 3)

3. Prompt dialog box: JavaScript provides a prompt dialog box for user interaction. The **prompt ( )** method instantiates the prompt dialog box which displays a specified message. In addition, the prompt dialog box also provides data entry field, which accepts user input thus a prompt dialog box:

- Display "predefined message".
- Displays a text box and accepts user input.
- Displays on **OK** and **CANCEL** button.

The prompt dialog box causes program execution to halt until user action takes place.

- Clicking on the "**OK**" button causes text typed inside the textbox to be passed to the program.
- Clicking on the "**CANCEL**" button causes a NULL value is passed to the environment.

Syntax:

```
Prompt ("message", "default value")
```

Eg.

```
Prompt ("Enter your favorite college:", "EASCOLL");
```

The value entered into the textbox on the prompt dialog box is accepted and can be stored in a variable.

```
<html>
<head><title>…….</title></head>
<body>
<script language = "javascript">
document.write('<img src = "purbanchal.gif">');
document.write('<h1>Greeting.</h1>');
var Name=prompt ("Enter your Name Of College", "Name");
//document.write(prompt("Enter your Name of college","Name"));
document.write ('Welcome to iBirat Solution! </h1>'+ Name);

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

Output: Fig above of Prompt Dialog

The Java Script Document Object Model:

What is Document Object Model?

JavaScript enabled browsers are capable of recognizing individual elements (objects) in an HTML page, browser assembles all the elements(objects) contained, after the page has been rendered in the browser.

JavaScript enabled browser recognizes and uses the **Document Object Model** (i.e DOM). Using the **DOM** JavaScript enabled browsers identify the collection of web page objects that have to be dealt with while rendering an HTML based, web page in the browser window.

The HTML objects, which belong to the DOM, have a descending relationship with each other. The topmost object in the DOM is the '***Navigator***' (Browser itself), the next level in the DOM the browser's **window**. The next level is the '**Document'** displayed in the browsers window. If the document displayed in the browser's window have an HTML 'form' coded it , then 'form' is the next level in the DOM.

The DOM hierarchy consists downwards that consists elements on a form such as text boxes, labels, radio buttons, check boxes etc. When a web page is rendered in a JavaScript enabled browser window, JavaScript is capable of uniquely identifying each element in the web page, because elements of a web page are bound to the DOM. The DOM that JavaScript recognizes is described below:

According to DOM

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

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.

It contains various method, properties and Events

Method: item, navigator, blur, focus, alert, confirm, setTimeout, clear Timeout

Properties: document, location, history, navigator, event, length, name

Events: onfocus, onload, onunload, onblur, onhelp, onerror

Window.open() - Opens a new browser window

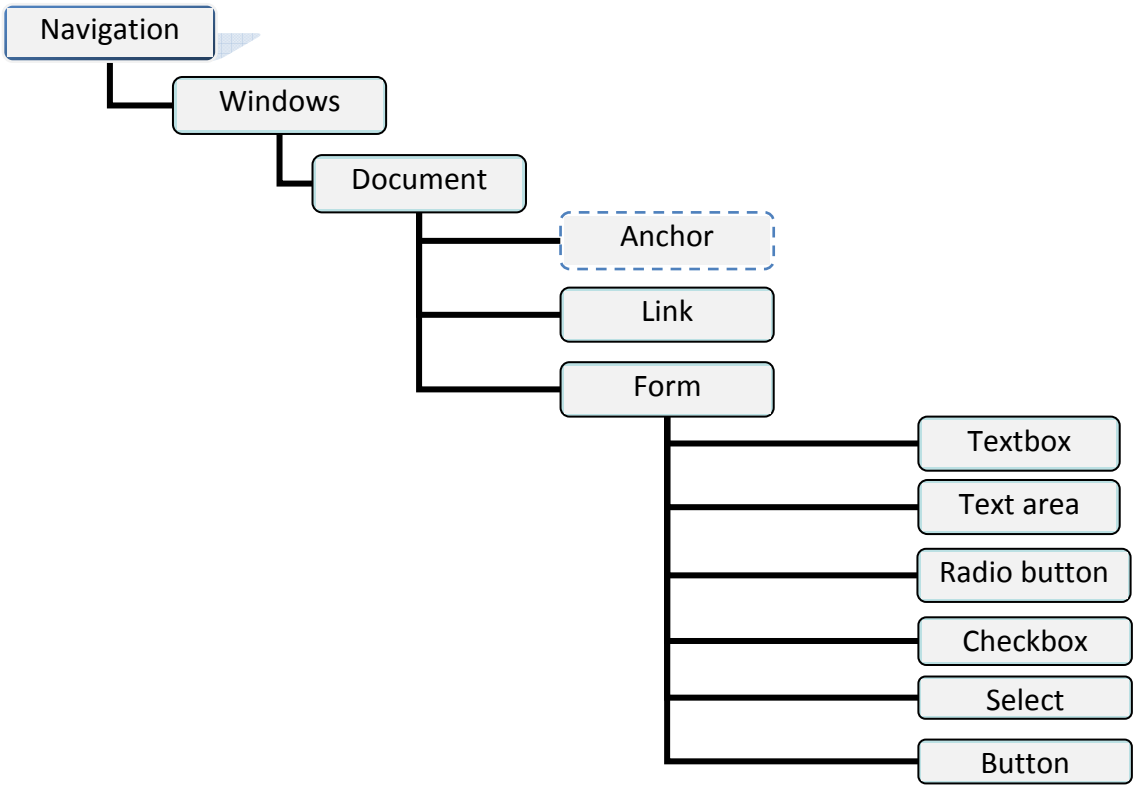


Fig: JavaScript DOM.

Java Script Assisted Style Sheet Dom [JSSS DOM].

JSSS use Java Script syntax to control a document's presentation style. When a JSSS is embedded in an HTML page within the **<head> </head>** tags, then the Java Script DOM picks up a whole new set of objects, when add to the standard DOM objects recognized by Java Script. Objects added to the DOM by use of JSSS are:

|→ Document
|→ Tags
| |→P
| |→DIV
| |→SPAN
| |→H1 through H6
|→ Classes
| |→ tag names
|→ IDS

Since JavaScript understands the DOM and can extend the DOM with the use of JSSS in a web page.

JavaScript understands '**objects**'. All objects have:

- **Properties** that determine the functionality of the object.
- **Methods** that allows access to these properties.
- **Events** that allow JavaScript code to be connected to the object by being mapped to appropriate JavaScript event handlers.

Forms

Creating a form: It is divided into 3 parts:

- Forms tag includes the URL of the CGI script that will process the form.
- Form elements, like fields and menus.
- Submit button, which sends the data to the CGI script on the server.

Steps for creating the form:

- Type **<form>** to initiate the form.
- Type **METHOD** = method, where method is either **GET** or **POST**.
- Type **ACTION** = **"Script.url"**> where *script.url* is the location on the server of the CGI Script that will run when the form is submitted.
- Create the form's contents.
- Type **</form>** to complete the form.

Working with Select & Options:

- *Type* <select
- *Type* NAME= "name", where name will identify the data collected from the menu when it is sent to the server.
- *Type* size = "n" , where n represents the number of options that should be initially visible in the menu.
- Type> to close the option.
- Type< option to start the next option.

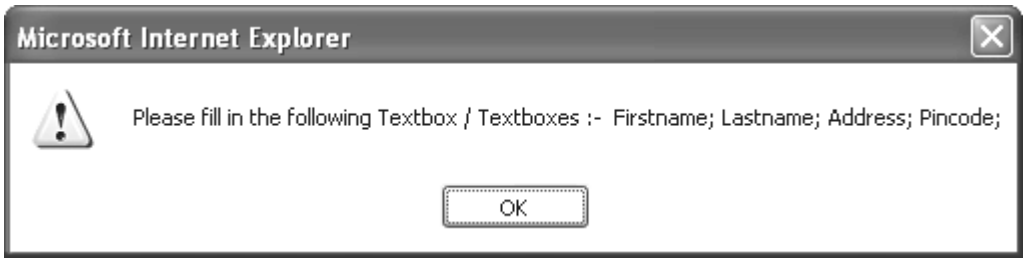
- *Type* SELECTED if you want the option to be selected by default.
- *Type* value = "value", where value specifies the data that will be sent to the server when the option is selected.
- *Type* > to close this option.
- *Type* the option name as you wish it to appear in the menu.
- Repeat step 6 – 10 for each option.
- *Type* </select> to close the select option.

Ex:

```
<select size="1" name="name">
        <option selected value="Choice1">Choice1</option>
        <option value="Choice2">Choice2</option>
        <option value="Choice3">Choice3</option>
</select>
```

```
<HTML>
    <HEAD>
 <SCRIPT LANGUAGE="JAVASCRIPT">
            function verifyData ( )
           {
                    a=0;     r="";
    for (i=0; i<=4; i++)
             {
                if (document.forms[0].elements[i].value == "")
                 {  a=1;
                            r = r +" "+ document.forms[0].elements[i].name + ";" ;
          }
                     else if ((i > 3)&&(a==0))
          {  alert("All Textboxes are filled in - Thank You !");
           }
            }
           for (i=0; i<=4; i++)
             {       if (document.forms[0].elements[i].value == "")
                 {
                            alert("Please fill in the following Textbox / Textboxes :- " + r);
                document.forms[0].elements[i].focus( );
                            break;
              }
                   }
           }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM>
        First Name : <INPUT TYPE="text" NAME="Firstname" SIZE=20>
        Last Name : <INPUT TYPE="text" NAME="Lastname" SIZE=20><P>
        Address : <INPUT TYPE="text" NAME="Address" SIZE=60><P>
        Pincode : <INPUT TYPE="text" NAME="Pincode" SIZE=6><P>
        <INPUT TYPE="button" NAME="act" VALUE="Verify"  onClick="verifyData( )">
     </FORM>
</BODY>
```

```
<SCRIPT Language="JavaScript">
document.forms[0].Firstname.focus( );
</SCRIPT>

</HTML>
```



Radio Buttons:

Instead of giving the options of selection of the options to the user, we can provide buttons to click one of them as a feedback. These buttons are also known as radio buttons.

```
<HTML>
 <HEAD>
        <TITLE>FORMS1</TITLE>

        <!-- Using Reset Button //-->

<SCRIPT>
        function func(f1)
{
        alert("The Form Elements have been cleared");
}
</SCRIPT>

 </HEAD>

<BODY>
<FORM onReset="func(this.form) ">
        Client Name : <Input Type=Text Name="Text" Value=""><BR><BR>
        Client Address: <Input Type=Text Name="Text1" Value=""> <BR><BR>
        <Input Type="radio" Name="Radio" Value="Male"> Male
        <Input Type="radio" Name="Radio" Value=" Female "> Female<BR><BR>
        <Input Type="CheckBox" Name="Check" Value="Employed"> Employed <BR><BR>
        <Input Type="Reset" Name="Rst" Value="Reset" >
</FORM>
</BODY>
</HTML>
```

Check boxes:
Check boxes are similar to radio buttons. Instead of rounded hollow buttons, they are square boxes. The only thing to change in the text is that replace radio with checkbox.

Text Area:
It is the palace where the user can write comments. The parameters of text area contains the no of rows and columns.
Eg.

```
<p><textarea name= “answer” rows= “4” cols = “60”> your comments……..</text area>
```

Password Boxes:
This provides space for creating a password.

```
<p> password: <input type = “password” name= “password” size = “10”>
```

Submit buttons:
This is used to submit the form. Along with submit buttons, we can place image too which the user will click to respond.
Eg.

```
<button type = “Submit” Name = “Submit” value = “Submit”>
<img src = “crane.jpg” width = 10 height= 30> press </button>
```

Resetting the form:
While filling up the form, it is very likely that the visitor may make a mistake which can be spelling or second thought. You can give visitors a reset button so that they can start over with a fresh form.

Steps:

- Type <input type = “reset”
- If desired type value = “reset message” where reset message is the text that appears in the button. The default reset message is reset.
- Type > to close the tag.

Eg.

```
<input type = “reset” value = “reset message”>
```

Understand Objects in HTML:

Properties of HTML Objects:

Just like real world objects HTML objects have a no of properties that determine the behavior of the object.

Eg. Textbox have properties like name, size and so on.

Methods of HTML objects: Methods of an object are used to set or get a value of an object's property. It can be references as: Object Name. Method Name

Browser Objects: when any JavaScript enabled browser leads a web page, the browser automatically creates a no of JavaScript objects that map to the DOM. It is the DOM, which provides JavaScript access to the HTML objects that are contained in the web page.

The JavaScript objects created by Netscape communicator are:

Object Name | its use:

Navigator: To access information about the browser that is executing the current script.
Window: To access a browser window or a frame within the window.
Document: To access the document currently loaded into a window. Tags used for document in HTML are head and body.
Location: To represent a URL
History: To maintain a history of the URL's accessed within a window.
Event: To access information about the occurrence of an event.
Event: The Event object provides constants that are used to identify events.
Screen: To access information about the size and color depth of a client computer's screen in which the current browser is running.

How a java script enabled browser handles the Document object:

Any document can contain various HTML objects such as:

- images
- image map
- Hyperlinks.
- Frames
- Anchors
- Forms with various form elements.

The browser creates one array in memory per HTML object in the document. If these HTML objects are actually contained in the HTML page then these array will hold indexed elements, which will point to the context area in memory where the HTML object are otherwise the array will exist, but will be empty. The first image in the document will have the array index as [0], the next image in the document will have the array index of [1] and so on.

Java script provides access to the arrays and their elements. The java script arrays created by Netscape communicator are

- Image/images array.
- Link/links array.
- Area
- Frame/frames array.
- Anchor/anchors array.
- Form/forms array.

Access to element of a web page:

Once a web page is rendered (pointed) in a browser window, it is completely static. For any program code to be able to interact with the web page, each element of the web page should be in memory, with unique name. The unique name translates to a content area in memory where the web page element resides. When a web page is assembled in memory prior being rendered in the browser's window, a Java Script enabled browser creates several arrays. These arrays hold references to individual objects.

When java script is processed, the web page can be rendered in the browser. When re-rendered, the browser changes made to the web page elements will be visible.

Handling events using JavaScript:

Java Script has several event handlers that ate mapped to on HTML object's events. Java Script, event handlers can be divide into two types.

- Interactive and
- Non-interactive.

An **interactive** even handlers depend on user interaction with an HTML page. Eg – '**OnMouseOver'** event handlers is an interactive event handler. This requires the user to move the mouse cursor over a web page.

Non- interactive event handlers does not need user interaction to be invoked. Eg. '**OnLoad'** event handler automatically executes whenever a form is loaded into a web page.

Name of java Script event handlers:

OnAbort	the loading of an image is aborted as a result of user action.
OnBlur	a document, window, frame set or form loses the current input focus.
Onchange	A text field, text area, file upload field or selection is modified and loses the current input focus.
OnClick	A link, client-side image map area or form element is clicked.
OnDbl Click	A link, client-side image map area or document is double clicked.
On DragDrop	A dragged object is dropped in a window or frame.
OnError	An error occurs during loading of an image, window or frame.
OnFocus	A document, window, frame set or form element receives the current input focus.
OnKeyDown	The user presses a key.
OnKey Press	The user presses and releases a key.
OnKeyUP	The user releases a key.
OnLoad	An image, document or frame set is loaded.
OnMouseDown	The user presses a mouse button.
OnMouseMove	The user moves the mouse.
OnMouseOut	The mouse is moved out of a link or an area of a client side image map.
OnMouseOver	The mouse is moved over a link or an area of a client side image map.
OnMouseUP	The user releases mouse button.
OnReset	The user resets a form by clicking the form's reset button.
OnResize	The user resizes a window or frame.
OnSelect	Text is selected in a text field or a text area.
OnSubmit	The user presses a form's submit button.
OnUnload	The user exits a document or frame set.

Program using Event handler:

```
<html>
<head><title>Event Handler</title></head>
<body>
<h1>Welcome to PU</h1>
<p> <a href = "computer.htm"  onmouseover='alert("Jump to B.E computer")'>
Move your mouse over this link & a popwindow is displayed
</A> </p>
</body>
</html>
```

OnLoad and OnUnLoad Event:

```
<html>
<head><title>Event Handler</title><head>
</head>
<body onLoad = "alert('Hello Students')" onUnLoad = "alert('Good bye')">
<h1> Handling Load events in a content document </h1>
</body>
</html>
```

```
<html>
<head><title>Event Handler</title></head>
<script>
        count=0;
</script>
</head>
<body>
<h1>Event handler with count </h1>
<p><a href="computer.htm" onmouseover='++count; alert("you have moved your mouse " + count +
" times")'>Move your mouse over this link and a popwindow will tell howmany times you moved the
mouse</a></p>
</body>
</html>
```

Form Object:

When creating an interactive web site for the internet it is necessary to capture user input and process this input. HTML provides the <Form> </Form> tags with which an HTML form can be created to capture user input. As soon as the <form> </form> tags are encountered in an HTML program by a JavaScript enabled browser, the browser creates a **'forms array'** in memory. This array tracks the no of form objects described in the HTML program. The Java Script **'forms array'** holds inform about each object used within the <form></form> tags.

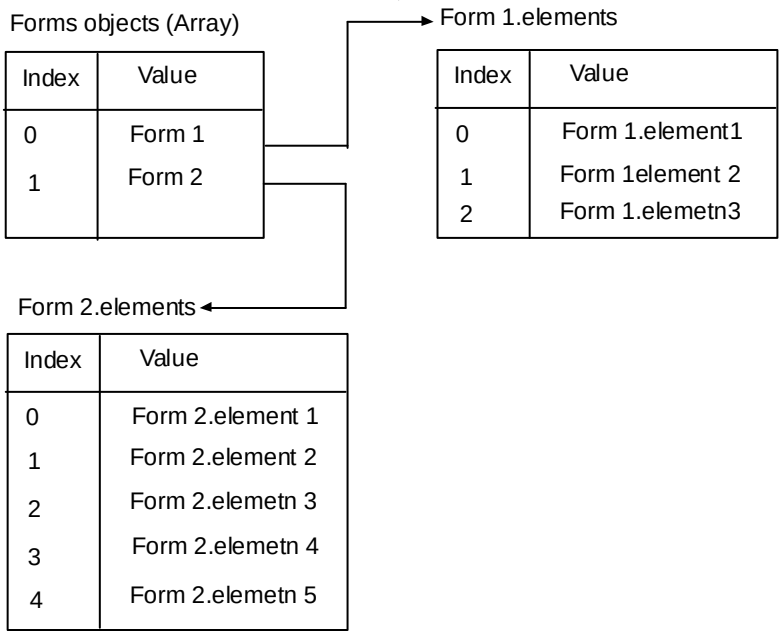
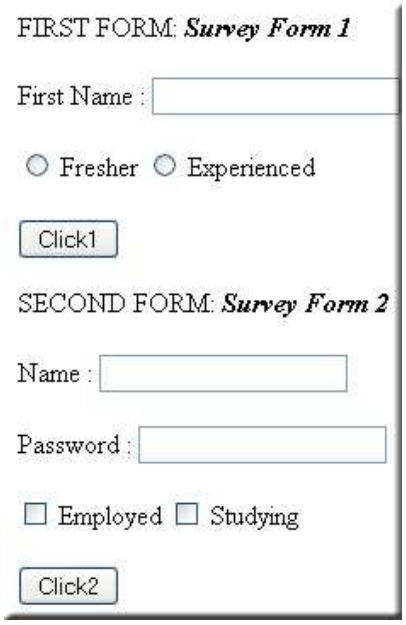


Fig:- Form element's Array.

Program:

1 . Count the number of element in **form's** element array. Check the number returned against the no. of form elements describe between the <FORM> </FORM> tags in the HTML page that is running in the browser. Recognize that the number of elements in the element array match the number of elements describe between the <FORM> </FORM> in the HTML page exactly.



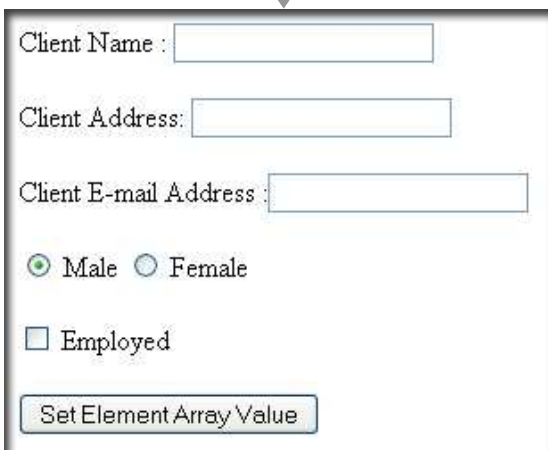

```
<HMTL>
 <HEAD>
     <TITLE>FORMS</TITLE>
          <!-- This code allows to access the Form objects Elements Array //-->
   <SCRIPT Language="JavaScript">
```

```
function Ver(form1)
{
      v=form1.elements.length;
if (form1.elements[3].name=="Button1")
{
            alert('First form name : '+document.forms[0].name);
            alert('No. of Form elements of  ' +document.forms[0].name + ' = '+v);
}
      else if (form1.elements[4].name=="Button2")
      {
      alert('Second form name : '+document.forms[1].name);
      alert('No. of Form elements of  ' +document.forms[1].name + ' = '+v);
      }
      for(i=0;i<v;i++)
      alert(form1.elements[i].name+ ' is at position '+i);
}
  </SCRIPT>
</HEAD>

<BODY>
<FORM Name="Survey Form 1">
      FIRST FORM: <i><b>Survey Form 1 </b></i><BR><BR>
      First Name : <Input Type=Text Name="Text1" Value=""><BR><BR>
            <Input Type="radio" Name="Radio1" Value=""> Fresher
            <Input Type="radio" Name="Radio1" Value=""> Experienced<BR><BR>
            <Input Type="Button" Name="Button1" Value="Click1" onClick="Ver(form)">
</FORM>

<FORM Name="Survey Form 2">
            SECOND FORM: <i><b> Survey Form 2 </b></i><BR><BR>
      Name : <Input Type="Text" Name="Text2" Value=""> <BR><BR>
            Password : <Input Type="Password" Name="Pass2" Value=""> <BR><BR>
            <Input Type="CheckBox" Name="Check1" Value="" > Employed
            <Input Type="CheckBox" Name="Check2" Value="" > Studying <BR><BR>
            <Input Type="Button" Name="Button2" Value="Click2" onClick="Ver(form)">
</FORM>
</BODY>
</HTML>
```

2. Illustrate the use of a form object's element array. The state of a radio button and a checkbox on the HTML form can be programmatically changed using event based Java Script.

(i) When the Java Script program runs an Alert draws attention to the checkbox, 'Checked property has been set to true via JavaScript code.

(ii) On clicking on the first Alert's ok button, another Alert pops up that displays a message indicating that radio button is checked when the same button was clicked.

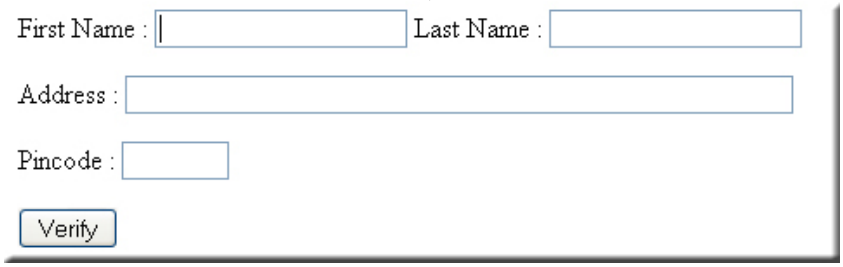



```
<HMTL>
 <HEAD>
      <TITLE>FORMS</TITLE>
      <!-- This code checks the Checkbox when the button is clicked //-->
   <SCRIPT Language='JavaScript'>
  function Chk(f1)
{
 f1.Check.checked=true;
 alert(" The Checkbox just got checked ");
 f1.Check.checked=false;
 f1.Radio[0].checked=true;
 f1.Radio[1].checked=false;
 alert(" The Radio button just got checked ");
}
  </SCRIPT>
 </HEAD>
<BODY>
<FORM>
      Client Name : <Input Type=Text Name="Text" Value=""><BR><BR>
      Client Address: <Input Type=Text Name="Text1" Value=""> <BR><BR>
      Client E-mail Address :<Input Type=Text Name="Text2" Value=""><BR><BR>
      <Input Type="radio" Name="Radio" Value=""> Male
      <Input Type="radio" Name="Radio" Value=""> Female<BR><BR>
      <Input Type="CheckBox" Name="Check" Value=""> Employed <BR><BR>
<Input Type="Button" Name="Bt" Value="Set Element Array Value" onClick="Chk(this.form)">
</FORM>
</BODY>
</HTML>
```

(3) Create a HTML form that has a no of text boxes. When the form runs, browser fills the textboxes with data. Write JavaScript code that verifies that all textboxes have been filled.

If a textbox has been left empty, popup an Alert indicating which textboxes has been left empty.

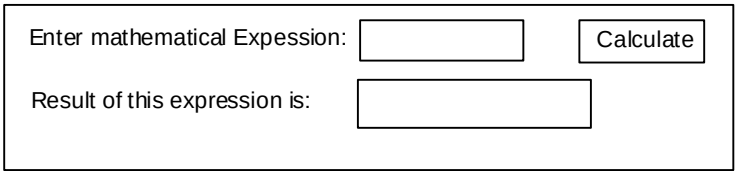
When the Alert's Ok button is clicked on, set focus to that specific textbox. If all the textboxes are filled, display a thank you alert.




```
<HTML>
    <HEAD>
 <SCRIPT LANGUAGE="JAVASCRIPT">
        function verifyData ( )
       {
            a=0; r="";
     for (i=0; i<=4; i++)
        {
           if (document.forms[0].elements[i].value == "")
           {   a=1;
               r = r +" "+ document.forms[0].elements[i].name + ";" ;
         }
           else if ((i > 3)&&(a==0))
         { alert("All Textboxes are filled in - Thank You !");
          }
       }
      for (i=0; i<=4; i++)
        {  if (document.forms[0].elements[i].value == "")
           {
               alert("Please fill in the following Textbox / Textboxes :- " + r);
           document.forms[0].elements[i].focus( );
               break;
         }
          }
       }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM>
        First Name : <INPUT TYPE="text" NAME="Firstname" SIZE=20>
        Last Name : <INPUT TYPE="text" NAME="Lastname" SIZE=20><P>
        Address : <INPUT TYPE="text" NAME="Address" SIZE=60><P>
        Pincode : <INPUT TYPE="text" NAME="Pincode" SIZE=6><P>
        <INPUT TYPE="button" NAME="act" VALUE="Verify"
onClick="verifyData( )">
      </FORM>
</BODY>
<SCRIPT Language="JavaScript">
document.forms[0].Firstname.focus( );
</SCRIPT>
</HTML>
```

(4) Develop a HTML page, which accepts

- any mathematical expression
- Evaluate the expression.
- Displays the results of the evaluation.




```
<HTML>
<HEAD>
    <TITLE>Using Text and Button objects in an HTML Form </TITLE>
   <SCRIPT LANGUAGE="JavaScript">
     function calculate(form)
{
                    form.results.value = eval(form.entry.value);
}
   </SCRIPT>
</HEAD>
<BODY>
<FORM >
Enter a Javascript Mathematical Expression:
<INPUT  TYPE = "text"   NAME = "entry"   VALUE = "">
<INPUT  TYPE="button"  VALUE = "Calculate"  onClick = "calculate(this.form);" >
<BR>
The result of this expression is:
<INPUT  TYPE = "text"   NAME = "results"   onFocus = "this.blur();" >
<BR>
</FORM>
</BODY>
</HTML>
```

(5) Create a form having textboxes radio buttons, checkbox and a reset button as shown in fig (i) on clicking the reset button, the entire form is reset.


```
<HMTL>
 <HEAD>
          <TITLE>FORMS1</TITLE>
          <!-- Using Reset Button //-->
<SCRIPT>
          function func(f1)
```

```
{
        alert("The Form Elements have been cleared");
}
</SCRIPT>
 </HEAD>
<BODY>
<FORM onReset="func(this.form) ">
        Client Name : <Input Type=Text Name="Text" Value=""><BR><BR>
        Client Address: <Input Type=Text Name="Text1" Value=""> <BR><BR>
        <Input Type="radio" Name="Radio" Value=""> Male
        <Input Type="radio" Name="Radio" Value=""> Female<BR><BR>
        <Input Type="CheckBox" Name="Check" Value=""> Employed <BR><BR>
        <Input Type="Reset" Name="Rst" Value="Reset" >
</FORM>
</BODY>
</HTML>
```

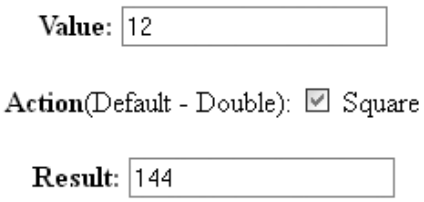
(6) Develop a HTML page, which uses **two text** fields and a checkbox. The first text object accepts a numerical value. Depending on the checked or unchecked state of the checkbox, the second text object displays.

Checkbox is not checked: Double the numeric value entered Checkbox is checked: The square of the numeric value entered.

If the second text object is loaded with a numeric value depending on the checked or unchecked state of the checkbox, the first text object displays.

Checkbox is not checked: Half the numeric value entered.

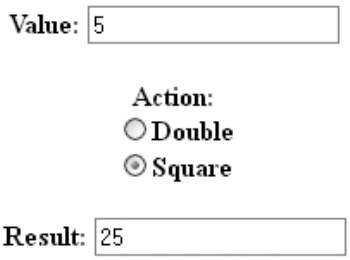
Checkbox is checked: The square root of numeric value entered.




```
<HTML>
<HEAD>
<TITLE>Working with Check Boxes</TITLE>
<SCRIPT>
function calculate(form, callingField)
          {
        if (callingField == "result")
        {
                if(form.square.checked)
                 {
                   form.entry.value = Math.sqrt(form.result.value);
                 }
        else
                 {
                   form.entry.value = form.result.value/2;
                 }
              }
```

```
else
  {
                        if(form.square.checked)
{
                              form.result.value = form.entry.value * form.entry.value;
                        }
      else
{
                              form.result.value = form.entry.value * 2;
                        }
                  }
            }

</SCRIPT>
</HEAD>
 <BODY>
<FORM>
<CENTER><BR>
<B> Value: </B>
<INPUT  TYPE = "text"  NAME = "entry"  VALUE = 0 onChange = "calculate(this.form, this.name);">
<BR><BR>

<B>Action</B>(Default - Double):
<INPUT  TYPE = checkbox  NAME = square onClick = "calculate(this.form, this.name);"> Square
<BR><BR>

<B>Result:</B>
<INPUT   TYPE = "text"  NAME = "result"  VALUE = 0 onChange = "calculate(this.form, this.name);">
</CENTER>

</FORM>
</BODY>
</HTML>
```

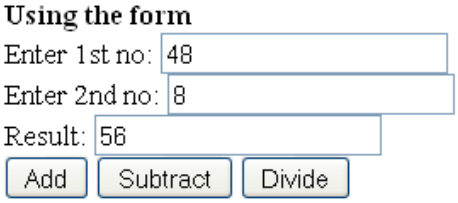
(7) An HTML form displays tow text boxes and two radio buttons. The first text box accepts a numeric value. If the first radio button if active, double the no entered in the first text field will be displayed in the second text field. If the second radio button is active the square of the no entered in the first field will be displayed in the second text field.




```
<HTML>
<HEAD>
<TITLE> Working with Radio Buttons </TITLE>
```

```
<SCRIPT LANGUAGE="JavaScript">
function calculate(form)
            {
      if(form.elements[2].checked)
            {
                        form.result.value = form.entry.value * form.entry.value;

           }
            else
                        {
                        form.result.value = form.entry.value * 2;
            }
      }

</SCRIPT>
</HEAD>
<BODY>
<FORM >
<CENTER><BR>
<B>Value:</B>
<INPUT  TYPE="text"  NAME="entry"  VALUE=0>
<BR><BR>
<SPACER Size= 190>

<B>Action:<B><BR>
<SPACER Size = 225>

<INPUT  TYPE="radio"  NAME="action1"  VALUE="twice" onClick="calculate(this.form);">Double
<BR>
<SPACER Size = 225>

<INPUT  TYPE="radio"  NAME="action1"  VALUE="square" onClick="calculate(this.form);">Square
<BR><BR>
<B>Result:</B>

<INPUT  TYPE=text  NAME="result" onFocus = "this.blur();">
</CENTER>
</FORM>
</BODY>
</HTML>
```

Arithmetic Operations with HTML DOM object.



```
<html>
<head> <title> Arithmetic operations </title>
<script Language = "Javascript">
function add(x,y)
{
var g = x+y;
window.document.forms[0].elements[2].value = g;
}
function subtract (x,y)
{
var g = x-y;
window.document.forms[0].elements[2].value =g;
}
function divide(x,y)
{
var g = x/y;
window.document.forms[0].elements[2].value =g;
}
</script>
</head>
<body>
<b> Using the form </b>
<br> <form>
Enter 1st no: <input type = "text"><br>
Enter 2nd no: <input type = "text"> <br>
Result: <input type= "text">
<br>
<input type = "button" value = "Add" OnClick =
"add(parseInt(window.document.forms[0].elements[0].value),parseInt(window.document.forms[0].
elements[1].value))">
<input type = "button" value = "Subtract" OnClick =
"subtract(window.document.forms[0].elements[0].value,
window.document.forms[0].elements[1].value)">
<input type = "button" value = "Divide" OnClick =
"divide(window.document.forms[0].elements[0].value,
window.document.forms[0].elements[1].value)">
</form>
</body>
</html>
```

Example of for loop

```
<html>
<head> <title>……</title></head>
<body>
<script>
var i;
for (i=1;i<=5;i++)
{
document.write("<br>"+i)
}
</script>
```

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

Example of while loop

```
<html>
<head> <title>……</title></head>
<body>
<script>
var num = 1
while (num<=10)
{
document.write(num);
document.write("<br>");
num++
}
</script>
</body>
</html>
```

Program to prompt dialog two box for value and print result.

```
<html>
<head> <title>……</title></head>
<body>
<script>
var x,y,sum;
x= prompt("enter any no");
y= prompt("enter any no");
sum= parseInt(x)+parseInt(y);
document.write("Total sum =" +sum);
</script>
</body>
</html>
```

USING <DIV> …..</DIV> TAG.

A web page can be divided into segments or division called DIVS. Each segments starts with <DIV> and ends with </DIV>. These segments can be positioned anywhere on the page. The <DIV> tag has a **'position'** attribute that can be one of the two values, ***Absolute*** or ***Relative***.

Absolute - positions the segment with respect to the top/left edge of the browser window.
Relative positions the segment in relation to other elements on the page.

Eg.

```
<HTML>
   <HEAD>
     <TITLE> Working with DIVs </TITLE>
   </HEAD>
   <BODY >
          <DIV ID = box1 style = "background-color:red; position:absolute; left:300;top:150;  width:50">
               <IMG SRC ="images/image1.jpg">
          </DIV>

          <DIV ID = box2 style = "background-color:red; position:absolute; left:340; top:180;  width:50">
```

```
            <IMG SRC ="images/image2.jpg">
        </DIV>

        <DIV ID = box3 style = "background-color:red; position:absolute; left:380; top:210;  width:50">
            <IMG SRC ="images/image3.jpg">
        </DIV>

        <DIV ID = box4 style = "background-color:red; position:absolute; left:420; top:240;  width:50">
            <IMG SRC ="images/image4.jpg">
        </DIV>
    </BODY>
</HTML>
```

LAYERS:

To segment a web page, there is another pair of DHTML tags, <Layers> </Layers>. These tags are designed to behave like a piece of transparent paper laid on top of a web page. Between <layer> </layer>, HTML elements ate inserted and the user is given precise control over the placement of these object. Each layers is provided with an ID and with attributes and values specify its appearance and position.

Layer Attributes:

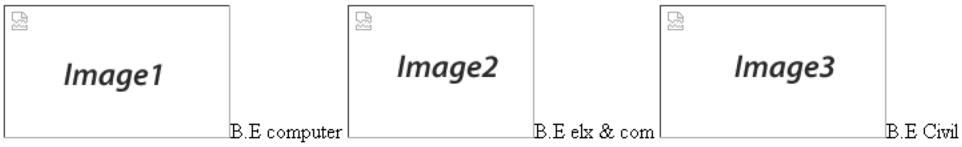
Attributes	**Values**
ID/Name	*- The name of the layers.*
Left and Top	*- horizontal and vertical positions of the layers in pixels.*
Pagex and pagey	*- horizontal and vertical positions of the layers relative to the document's window.*
Src	*- pathname of a file.*
Z-index, above, below	*- The stacking order of a layer. Only one valid at any given time.*
Width	*- Widht of layer's dispay.*
Height	*- height of layer's display.*
Clip	*- Viewable area of a layer.*
Visibility	*- Whether a layer is visible or not.*
Bgcolor	*- background color to be used by the layer.*
Background	*- image to be used as the background for the layer.*

Layers Method:

Method	***Description***
CaptureEvents(eventType)	*- Allows the layer to capture all events of the specified type.*
HandleEvents(event)	*- Invokes the event handler for the specified event.*
MoreAbove(layer)	*- Moves the layer above the identified layer.*
MoveBy(x,y)	*- Moves the layer by the specified x and y pixel increments.*
MoveTo(x,y)	*- Moves the layer to the specified position within the container.*
ReleaseEvents(eventType)	*- Ends the capturing of the specified event type.*

Layers Event handlers:

Event handlers	***Description***
onMouseOver, onMouseOut	*Event handler to use when the mouse enters or leaves the layers.*
onfocus, onBlur	*Event handler to use when the layer receives or loses keyboard focus.*
onLoad	*OnLoad:- Event handler to use when the layer is first loaded.*



Welcome to purbachal Univeristy.

The programs offered by this University are shown below:

```
<html>
<head><title>……</title></head>
<body>
<Layer id = box1 left=150 Top=150>
<img src=img1.jpg height =100 width=170>B.E computer</layer>
<Layer id = box2 Left =50 Top=200>
<img src = image2.jpg height=100 width = 140>B.E elx & com </layer>
<layer id = box3 left =250 Top=50>
<img src=img3.jpg height=100 width= 170>B.E Civil </layer>
<h1> Welcome to purbachal Univeristy.</h1>
<hr>
The programs offered by this University are shown below:
</body>
</html>
```

Factorial Program

```
<HMTL>
 <HEAD>
            <TITLE>FORMS1</TITLE>
<SCRIPT>
function factorial(n)
{   if (n == 0)
      return 1;
   else
      return n * factorial(n-1);
}
var i;
document.clear();
for (i = 0; i <= 16; i++)
   document.write(i + "! = " + factorial(i) + "<br />");
</SCRIPT>
 </HEAD>
<BODY>
</BODY>
</HTML>
```