

8

Scripting

8.1 INTRODUCTION

In the early age of World Wide Web, HTML (Hypertext Markup Language) was the only language that could be used to create web pages. HTML a merely markup language and is limited to what it can do. HTML creates static web pages that have fixed contents (text, audio, and graphics). It does not allow user to interact with the web page. The increase in expectations, requirements has resulted in continual improvement of HTML and advent of powerful set of language called 'Scripting Language'.

Scripting language allows user to **create an active and dynamic web page** that can **display dynamic information** such as display an appropriate greeting – "Good Morning" or "Good Evening" depending on the time, **making web page interactive** for instance: Lets web page to ask questions and respond to how the user answers them, **validate data**: take input from the user and check the data to make sure it is valid or meets certain criteria and much more.

8.2 OBJECTIVES

After going through this lesson, you would be able to :

- write a JavaScript
- work with loop and conditional statements
- write functions and use of predefined methods
- handle JavaScript events
- differentiate between JavaScript and VBScript
- use VBScript in HTML document
- define basics of ASP

8.3 DEFINING SCRIPT

In computer programming, 'Script' is a program or sequence of instruction that is **interpreted**. This means that a program built with a scripting language must be run in the environment that contains the Script Language Interpreter (Interpreter: software program that helps to interpret the script line by line for the computer to understand and get the result).

Broadly scripts are categorized into two types:

- **Client-side scripting**
- **Server-side scripting**

Client-side scripting

Client-side scripting generally refers to the class of computer programs that are executed at client-side, by the user's web browser. JavaScript and VBScript are two widely used client-side scripts.

Server-side scripting

Server-side scripting is a program that executes at server end. It is usually used to provide interactive web sites that interface to

databases or other data stores. Some popular server-side scripting language are: ASP (Active Server Pages), JSP (Java server Pages), PHP, Python etc.

8.4 JAVASCRIPT

JavaScript developed by Netscape Communication Corporation is the most popular platform independent (run in windows, Linux, Macintosh...etc) Scripting Language. All most all browsers (Internet Explorer, Netscape Navigator) support JavaScript i.e. JavaScript Language Interpreter becomes part of browser.

8.4.1 Characteristics of Java Script

Scripts are embedded in a web page (HTML document) in specifically demarcated section. When browser encounters a script while opening a web page it calls a scripting interpreter, which parses and executes the scripting code.

JavaScript has the following characteristics:

- JavaScript is a **scripting language**.

A scripting language is a simple programming language having some limited features of full-blown programming language, easy to learn and easy to understand.

- JavaScript is **platform independent**.

JavaScript that embedded inside HTML document are not tied to any specific hardware platform or operating system and it is interpreted by all most all browsers.

- JavaScript is **object-based**.

JavaScript, depends on a collection of built-in objects (entities) for functionality

- JavaScript is **event-driven**.

It responds to user actions by the use of input devices for example: respond to the action of mouse: on mouse over, on mouse out, on mouse click.

8.4.2 Embedding JavaScript in HTML

There are two ways to use of JavaScript in HTML. One way, you can store your script in a separate simple text file and include it in your HTML document, other way, you can directly write your script inside HTML document.

The basic structure of an HTML file with JavaScript is as follows (JavaScript is indicated in bold):

```
<HTML>
<HEAD>
  <TITLE>Page title goes here</TITLE>
  <SCRIPT Language="JavaScript">
    Your JavaScript code goes here
  </SCRIPT>
</HEAD>
<BODY>
  Document Text goes here
  <SCRIPT Language="JavaScript">
    Your JavaScript code goes here too
  </SCRIPT>
  Document text here too
</BODY>
</HTML>
```

A sample HTML document embedding a JavaScript shown in Example 8.1 below:

Example 8.1

```
<HTML>
<HEAD>
  <TITLE>JavaScript Example 8.1</TITLE>
</HEAD>
<BODY>
  This Document shows the use of JavaScript in HTML.
  <BR>
  <SCRIPT language="JavaScript">
    document.write("Welcome to the world o f
JavaScript!")
  </SCRIPT>
</BODY>
</HTML>
```

‘Document’ – Predefined Java Script Object refer to the HTML document(web page).

‘write’ – It is a predefined method of document object that is used to write text to the HTML document.

The output of the Example 8.1 shown in Figure 8.1 below:

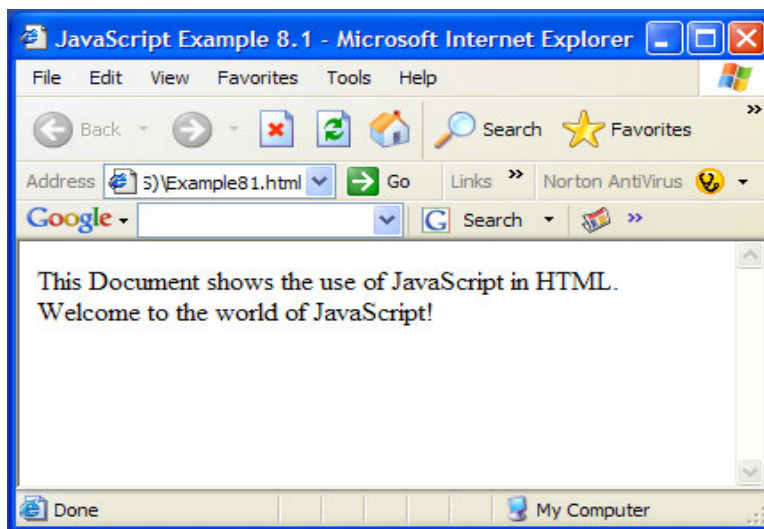


Fig. 8.1

8.4.3 Variables in JavaScript

Variables, data types, expressions and operators form the core of any programming language and JavaScript is no exception to this.

➤ **Define Variable**

Variables are storage containers to store data of various types like “computer”, 65000, and 2765.50. Like any other programming language, JavaScript allows user to declare variables and use the values stored in them.

Variable names in JavaScript can begin with an uppercase letter (A–Z), lowercase (a–z), or underscore (_) character. The rest of characters may be letters, the underscore character, or digits (0–9). JavaScript is case sensitive, the variable result not same as ‘RESULT’.

➤ **Variable Data Types**

In JavaScript, variables can store following data types:

Data types	Examples
Number	Store numeric value(for example: 3, 3.5)
String	Store alphanumeric characters(for example: "computer", "555-123")
Boolean	Store true or false values only

➤ **Declaring Variables**

In JavaScript, you can declare a variable using the *var* keyword, as shown in below:

```
var itemName;
```

```
var itemPrice;
```

You can also assign a value to the variable at the time of declaration. For example: `var itemName="computer", var itemPrice="35000"`

➤ **Arrays**

Arrays are used to store a sequence of values of same data type. These values are stored in the indexed location within the array.

For Example: If you want to store three items of the same type, you can create an array with the name of the items and store their values in the same.

The following statement declares an array of length 3:

```
itemNamees=new Array(3)
```

Three item Names can be stored by using the following statements:

```
itemNamees[0]="Computer"
```

```
itemNamees[1]="TV"
```

```
itemNamees[2]="Radio"
```

The same can also be declared as follows:

```
itemNamees= new Array("Computer","TV","Radio")
```

To access the names stored in the array, you can use the following statements:

```
document.write(itemNames[0])
```

```
document.write(itemNames[1])
```

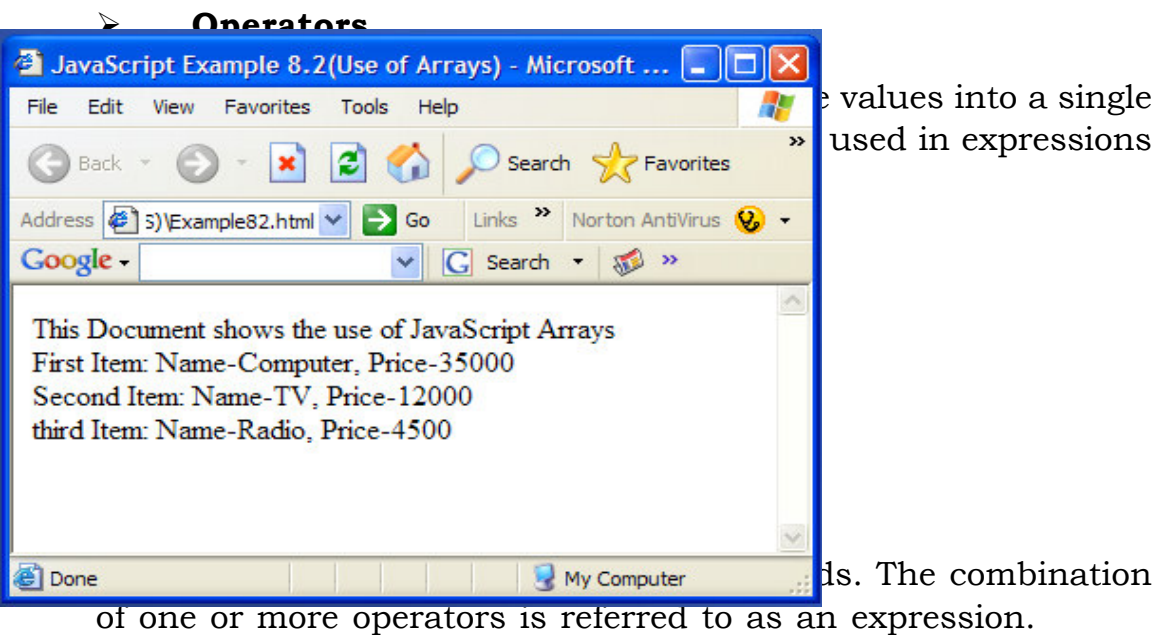
```
document.write(itemNames[2])
```

An example 8.2 shows use of arrays.

```
<HTML>
<HEAD>
  <TITLE>JavaScript Example 8.2(Use of Arrays) </TITLE>
</HEAD>
<BODY>
  This Document shows the use of JavaScript Arrays
  <BR>
  <SCRIPT language="JavaScript">
    itemNamees=new Array("Computer","TV","Radio")
    itemPrices=new Array(35000,12000,4500)
    document.write("First Item: Name-" + itemNames[0]+", Price-"
+"itemPrices[0])
    document.write("<BR>")
    document.write("Second Item: Name-" + itemNames[1]+",
Price-" +itemPrices[1])
    document.write("<BR>")
    document.write("third Item: Name-" + itemNames[2]+", Price-"
+"itemPrices[2])
  </SCRIPT></BODY></HTML>
```

The output of the Example 8.2 shown in Figure 8.2 below:

Fig. 8.2



values into a single
used in expressions

ds. The combination
of one or more operators is referred to as an expression.

For example: `netPayment=netPrice+salesTax`

In this example "+" operator operates on operand `netPrice` and `salesTax` and its result assigned to variable `netPayment`.

The following table shows the list of operators used in JavaScript:

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	—	Decrement and then return value	X=3; —X	2
		Return value and then decrement	X=3; X—	3
	&&	Logical "and" evaluates to true when both operands are true	3>2 && 5>3	False
Logical		Logical "or" evaluates to true when either operand is true	3>1 2>5	True
	!	Logical "not" evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
	+	Concatenation(join two strings together)	"A"+"BC"	ABC

8.4.4 Expressions

An expression is a part of statement that is evaluated as value. An expression can be any combination of variables, operators and other expressions. In JavaScript there are four types of expressions:

- Assignment expression
- Arithmetic expression
- String expression
- Logical expression

Assignment expression

This expression is used to assign a value to a variable.

Example: avgMarks=55, the value 55 has been assigned to the variable avgMarks using assignment operator (=)

Arithmetic Expressions

Expressions in which arithmetic operations are performed are called arithmetic expression.

Example: netPrice=1200+12

String Expression

Expressions in which operations are performed on string are called string expression.

Example: msg= "Hello," + "Mr. Rakesh", Here "+" string operator concatenate two strings "Hello, " and "Mr. Rakesh" results "Hello, Mr. Rakesh" that is assigned to variable msg.

Logical Expression

These expressions evaluate to a Boolean (true or false) value.

Example: 10<19, since 10<19, this expression evaluates to the Boolean value of true.

INTEXT QUESTION

1. Fill in the blanks :
 - (a) An active and dynamic web page can be created by using _____.
 - (b) An _____ is used to transform one or more values into a single resultant value.
 - (c) The _____ tag is used to embed JavaScript in HTML.
 - (d) _____ is an increment operator.
 - (e) The script should be written inside _____ tag.
-

8.4.5 Looping Constructs (Controlling Programming Flow)

The statements inside any program executes sequentially, line by line. This flow of execution is called as sequential execution. This flow of execution can be controlled (for example: Execute a portion of a program only once, based on a condition. Execute a portion of the program repetitively, based on condition.) by using programming constructs. This programming constructs typically are of two types:

- **Conditional**

These constructs provide the facility to execute a portion of a program only once, based on a condition. Examples of conditional constructs are: if, if...else and switch...case.

- **Iterative**

These constructs provide the facility to execute a portion of the program repetitively, based on a condition. Examples of iterative constructs are: while, do...while and for loops.

- **Conditional Constructs (if, if...else & switch...case)**

1. *The if and if...else statement*

If and if..else are the primary decision making constructs of any programming language.

Syntax:

<pre>if(condition) { Statement1 Statement2 _____ _____ }</pre> <p>In this case, if a certain condition evaluates to true, than only the statement(s) written inside the block gets execute otherwise skips the block.</p>	<pre>if(condition) { Statement(s) } else { Statement(s) }</pre> <p>Here, if a certain condition evaluates to true, the statement(s) written inside the block gets execute otherwise, the statement(s) written inside else block gets execute.</p>
---	---

N.B: more than one condition can also be used in if and if..else statement with the help of && (logical and) and || (logical or).

<p>Example 8.3: A sample program to find out the average marks secured by the student in 3 subjects, display the result PASS, only if average marks > 40.</p>	<p>Example 8.4: A sample program to find out the average marks secured by the student in 3 subjects, display the result PASS if average marks > 40 else FAIL.</p>
<pre><HTML> <HEAD> <TITLE>Example 8.3</TITLE> </HEAD> <BODY> <SCRIPT language="JavaScript"> subject1=45 subject2=70</pre>	<pre><HTML> <HEAD> <TITLE>Example 8.4</TITLE> </HEAD> <BODY> <SCRIPT language="JavaScript"> subject1=45 subject2=35</pre>

<pre>subject3=80 name="Rakesh Kumar" document.write("Name:"+name+"") document.write("
") avgMarks=(subject1+subject2+subject3)/ 3 if(avgMarks>40) { document.write("Result: PASS") } </SCRIPT> </BODY> </HTML></pre>	<pre>subject3=10 name="Manoj Kumar" document.write("Name:"+name+" ") document.write("
") avgMarks=(subject1+subject2+subject3)/ 3 if(avgMarks>40) { document.write("Result: PASS") } else { document.write ("Result: FAIL") } </SCRIPT> </BODY> </HTML></pre>
--	---

The output of the Example 8.3 and Example 8.4 are shown in Figure 8.3 and Figure 8.4 below:

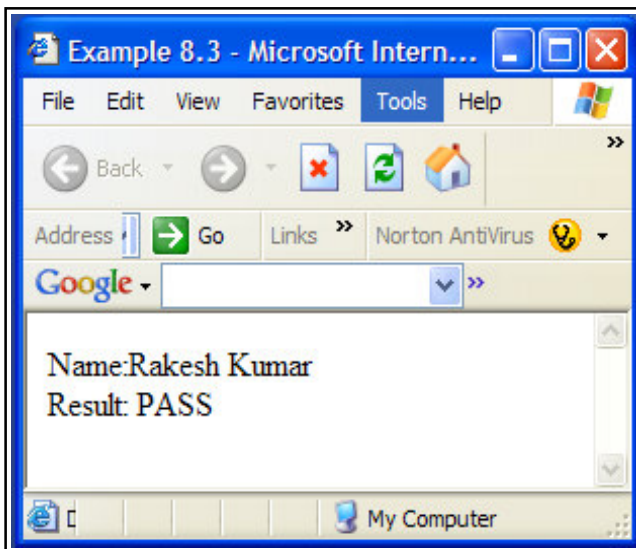


Fig. 8.3



Fig. 8.4

2. The switch Statement

The switch statement compares a value of the variable against other values. Multiple if statements can be replaced with switch.

```
switch(variablename)
{
case value1:
    statement(s)
break
case value2:
    statement(s)
    break
default:
    statement(s)
}
```

value1, value2 may be a number or string, if string it should be written inside double quote. The switch statement on execution checks if the value of variable matches with any case values i.e value1, value2 it executes the statement(s) under it. In case, it can not find a matching case statement, it executes the default statement(s)

An example 8.5 showing use of switch..case statement.

```
<HTML>
  <HEAD>
    <TITLE>Example 8.5, use of switch..case statement</TITLE>
  </HEAD>
  <BODY>
```

This example shows the use of switch..case statement.

```
<table bordercolor=olive border="1">
  <tr><b><td>Sl. No</td><td>Item Name</td><td>Price</td></b></tr>
```

```
<tr><td>1</td><td>Computer</td><td>35000</td></tr>
<tr><td>2</td><td>TV</td><td>15000</td></tr>
<tr><td>3</td><td>Radio</td><td>1500</td></tr>
</table>
<br>

<SCRIPT language="JavaScript">
  choice=1
  switch(choice)
  {
    case 1:
      document.write("You have choose the option 1")
      document.write("<BR>")
      document.write("Item Name: Computer, Price:35000")
      break
    case 2:
      document.write("You have choose the option 2")
      document.write("<BR>")
      document.write("Item Name: TV, Price:15000")
      break
    case 3:
      document.write("You have choose the option 3")
      document.write("<BR>")
      document.write("Item Name: Radio, Price:1500")
      break
    default:
      document.write("You have not choose any option")
  }
</SCRIPT>
</BODY>
</HTML>
```

The output of the Example 8.5 shown in Figure 8.5 below:

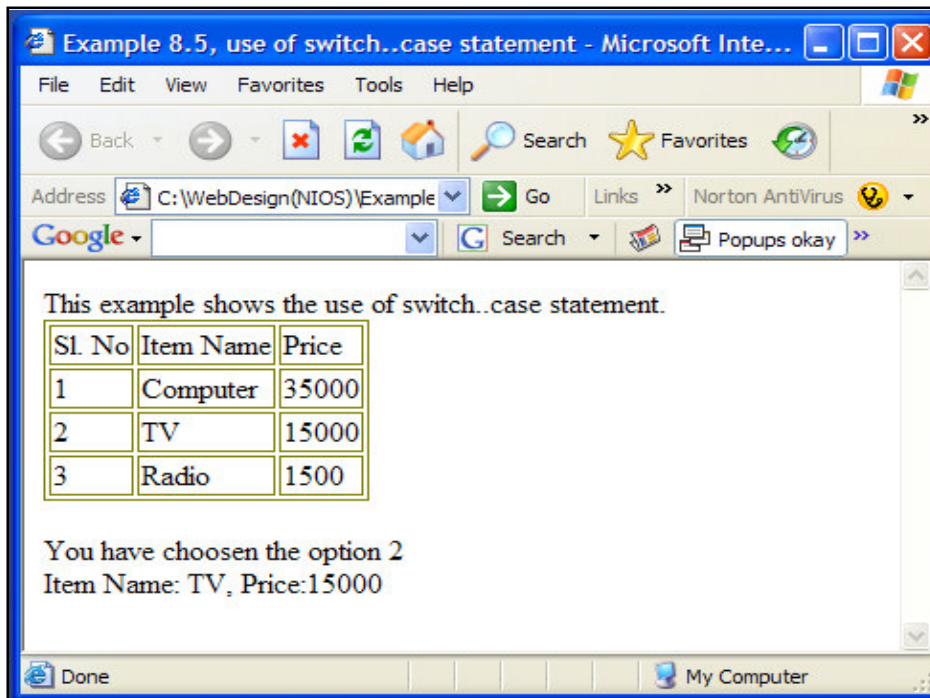


Fig. 8.5

➤ **Iterative Constructs (while, do...while and for loops)**

1. *The while statement*

The while construct is used to execute statement(s) as long as certain condition is true.

Syntax:

```
while(condition(s))
{
    statement(s)
}
```

2. The do...while statement

This statement is very similar to the while statement, except that it first execute the statement then checking the condition whereas in while first condition checking then statement execution. So in case of do... while the statement(s) execute at least once even if the condition is false.

Syntax:

```
do
{
Statement(s)
}while(condition(s))
```

An example 8.6 showing use of while and do..while.

```
<HTML>
  <HEAD>
    <TITLE>Example 8.6, use of while and do..while statement</TITLE>
  </HEAD>
  <BODY>
    This example shows the use of while and do..while statement.
    <SCRIPT language="JavaScript">
      count=1
      while(count<=3)
      {
        document.write("<BR>"+ "You are inside while loop"+"<BR>")
        document.write("The value of count is:" + count)
        count++
      }
    </SCRIPT>
  </BODY>
</HTML>
```

```
    }
    document.write("<BR>")
    document.write("You are now outside while loop")
    num=1
    document.write("<BR>")
    do
    {
        document.write("You are inside do...while! The statements inside
do..while execute at least once even if the condition is false")
        document.write("<BR>")
        document.write("The value of num is:"+num)
        num++
        }while(num>3)
        document.write("<BR>")
        document.write("You are now outside do...while loop")
    </SCRIPT>
</BODY>
</HTML>
```

The output of the Example 8.6 shown in Figure 8.6 below:

Fig. 8.6

3. *The for statement*

The for statement is also an iterative construct, used to execute statement(s) based on some condition.

Syntax:

```
for(variable initialization;checking condition;change value of vriable)
{
    Statement(s)
}
```

The for statement first initialize the variable with some value then, checks the condition. If the condition becomes true, it changes the value of the variable and the statement(s) within the braces are executed. One thing always remembers while changing the value of variable, the changes to the variable in such a way that at one point of time on checking the condition it should get false otherwise it would enter into infinite loop and the program will hang.

An example 8.7 showing use of for loop – display 1 to 5 numbers in different line.

```
<HTML>
  <HEAD>
    <TITLE>Example 8.7, use of for statement</TITLE>
  </HEAD>
  <BODY>
    This example shows the use of for statement.
    <br>
    <SCRIPT language="JavaScript">
      {
        for(i=1;i<=5;i++)
        {
          document.write("<BR>"+"The Value of i is:"+i)
          document.write("<BR>")
        }
      }
    </SCRIPT>
  </BODY>
</HTML>
```

The output of the Example 8.7 shown in Figure 8.7 below:

Fig. 8.7

8.5 FUNCTIONS

A function is a block of code that has a name. Whenever the name is used the function is called which means that the code within the function gets executed. If a same set of statements to execute number of times in different parts of the program it is better to create a function, write the statements inside it and call the function by name anywhere in program. By this way you can make your program efficient and simple.

The second purpose of JavaScript functions is to link actions on a web page such as mouse click, button presses, text selection, and other user actions can call JavaScript functions by including suitable tags inside HTML document.

Function blocks begin with the keyword function, followed by the function name and parentheses (). The statements inside function block are written inside braces {}. The function may take some values, known as parameters that are specified inside

parentheses () separated by comma one another and may also return some value.

Syntax:

```
function functionName(p1,p2...pn)
{
    block of code goes here
}
```

The keyword **function**, should be in lower case. functionName represents name of the function, p1,p2....pn are the parameters (optional), block of code goes inside braces { }. Usually functions are written in the Head part of the HTML page, so that they will be loaded first, when the page is loaded into browser.

A function can be called inside the program, called by another function, or it can be called by a user action called event such as mouse click, button presses etc.

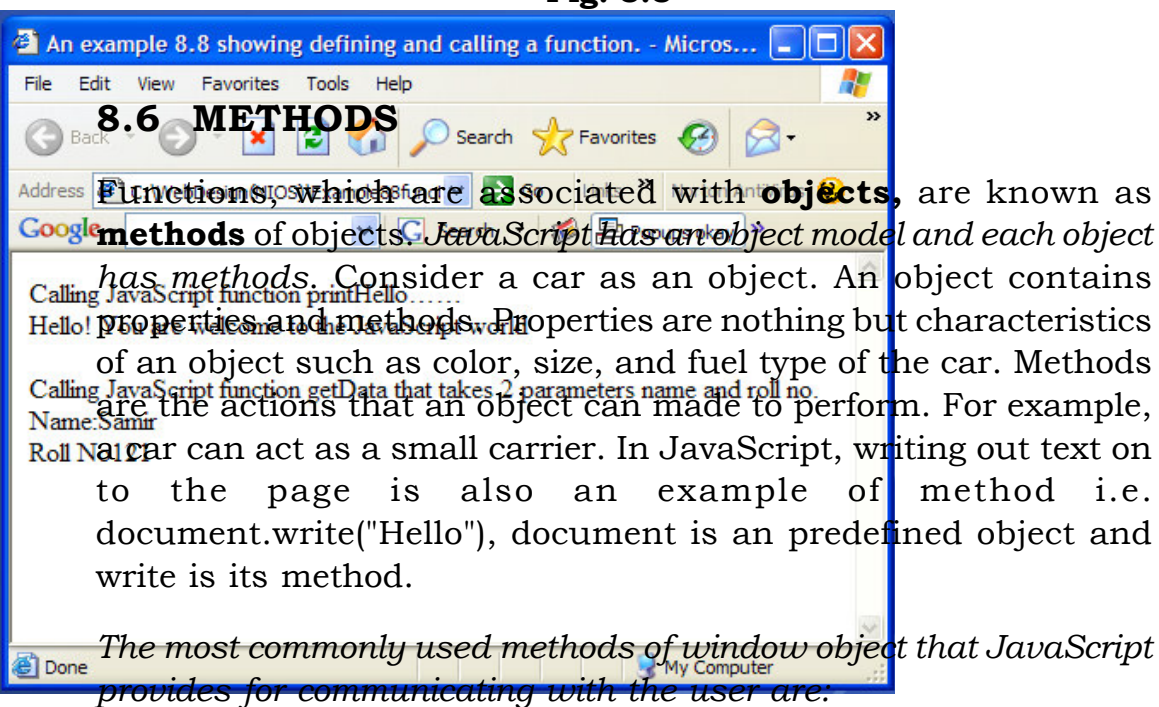
An example 8.8 showing defining and calling a function

```
<HTML>
  <HEAD>
    <TITLE> An example 8.8 showing defining and calling a function. </TITLE>
    <SCRIPT language="JavaScript">
      function printHello()
      {
        document.write("Hello! You are welcome to the JavaScript world")
      }
      function getData(name,rollNo)
      {
```

```
        document.write("Name:"+name)
        document.write("<BR>")
        document.write("Roll No"+rollNo)
    }
</SCRIPT>
</HEAD>
<BODY>
    Calling JavaScript function printHello.....
    <BR>
    <SCRIPT language="JavaScript">
        printHello()
    </SCRIPT>
    <BR>
    <BR>
    Calling JavaScript function getData that takes 2 parameters name and
roll no.
    <BR>
    <SCRIPT language="JavaScript">
        getData("Samir",121)
    </SCRIPT>
</BODY>
</HTML>
```

The output of the Example 8.8 shown in Figure 8.8 below:

Fig. 8.8



- The **alert ()** method

Used to create a dialog box with a message and OK button. Usage:
alert("Hello World")

➤ The **confirm ()** method

To get the confirmation from the user, confirm() is used, it is similar to the alert method, and in *addition to an OK button it also provides a Cancel button*. The method returns true if the user clicks OK and false if the user clicks Cancel.

```
val=confirm("Message to be displayed")
```

Usage: var ans=confirm("Continue ?")

➤ The **prompt ()** method

The prompt () method create a dialog box with a message and requests user input through a text field. Whatever user types into the text box is returned as the result of the prompt ().

Syntax:

```
Prompt("Message to the user","Default value on to the text field")
```

Usage: var name=prompt("Please enter your name","Type your name")

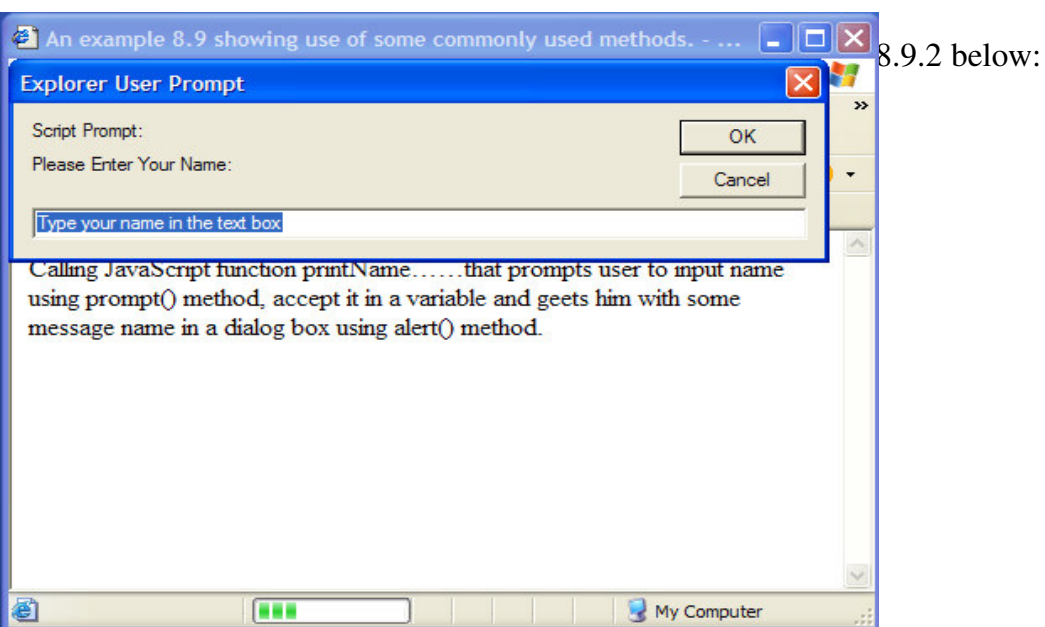
An example 8.9 showing use of some commonly used methods

```
<HTML>
  <HEAD>
    <TITLE>
      An example 8.9 showing use of some commonly used methods.
    </TITLE>
  <SCRIPT language="JavaScript">
    function printName()
    {
      var name=prompt("Please Enter Your Name:","Type your name in
the text box")
      document.write("Hi!" + name)
```

```
    alert("Hello! " + name + ", You are welcome to the JavaScript world.")
  }
</SCRIPT>
</HEAD>
<BODY>
```

Calling JavaScript function printName.....that prompts user to input name using prompt () method, accept it in a variable and gets him with some message name in a dialog box using alert () method.

```
<BR>
<SCRIPT language="JavaScript">
    printName()
</SCRIPT>
</BODY>
</HTML>
```



8.9.2 below:

Fig. 8.9

The Figure 8.9.2 shows output after input Suresh in the text box

Fig. 8.10

8.7 EVENT HANDLING

➤ Events and Event Handlers

An event is **an action** that occurs when a user interacts with a web page or other browser related activities. For Example: *when a user clicks a mouse button or enters data in a form, an event is generated. The browser waits for the event to occur, and takes an appropriate action according to the type of event. The action taken by the browser to an event is known as **event handling**. The function that is executed in response to the event is called an **event handler**.*

The following figure shows an event and the process of event handling:

Fig. 8.11 : Events and Event Handlers

Event Handling in JavaScript takes place in two steps:

1. Defining events such that they are handled by the script.
2. Connecting these events to JavaScript code.

JavaScript has predefined events for links, images, form elements, windows (a window refers to the body of a document containing HTML content) and event handlers.

The following table structure shows few events, description and the event handlers associated with HTML elements:

HTML Element	HTML Tags	Event	Description	Event Handler
document	<BODY	load	Generated when page is loaded into the browser.	onLoad
body	</BODY>	unLoad	Generated when document is closed	onUnLoad
Link	<A> - 	click	Mouse click on link	onClick
		mouseover	Generated when the mouse moved over a	onMouseOver

			link	
		mouseout	Generated when mouse is moved away from link	onMouseOut
form	<form> </form>	submit	Generated when user submit the form	onSubmit
Text field	<INPUT TYPE = "text">	blur	Text field loses focus	onBlur
		Focus	Text field receives current input focus	onFocus
		Change	Generated when user changes the value	onChange
		select	Text is selected within the text field	onSelect
Text area	<TEXTAREA> </TEXTAREA>	blur	A text area loses current input focus	onBlur
		focus	Test area gets the current input focus	onFocus
		change	Text area value changes	onChange
button	<INPUT TYPE = "button">	click	The button is clicked	onClick
submit	<INPUT TYPE = "submit">	click	The submit button is clicked	onClick
Reset	<INPUT TYPE = "reset">	click	The reset button is clicked	onClick
radio button	<INPUT TYPE = "radio">	click	The radio button is clicked	onClick

check box	<INPUT TYPE = "checkbox">	click	The check box button is clicked	onClick
Dropdown selection	<SELECT> </SELECT>	focus	A selection element receives input focus	onFocus
		change	Selection element value changed	onChange

An example 8.10 showing the use of Events and Event Handlers associated with HTML elements:

```
<html>
<head>
<title>Event Handling in
JavaScript</title>
<script language="JavaScript">
function load()
{
alert("onLoad event was just
handled")
}
function displayAlert()
{
alert("You are about to visit NIOS
site")
}
</script>
</head>
<body onLoad="load()">
<CENTER>
```

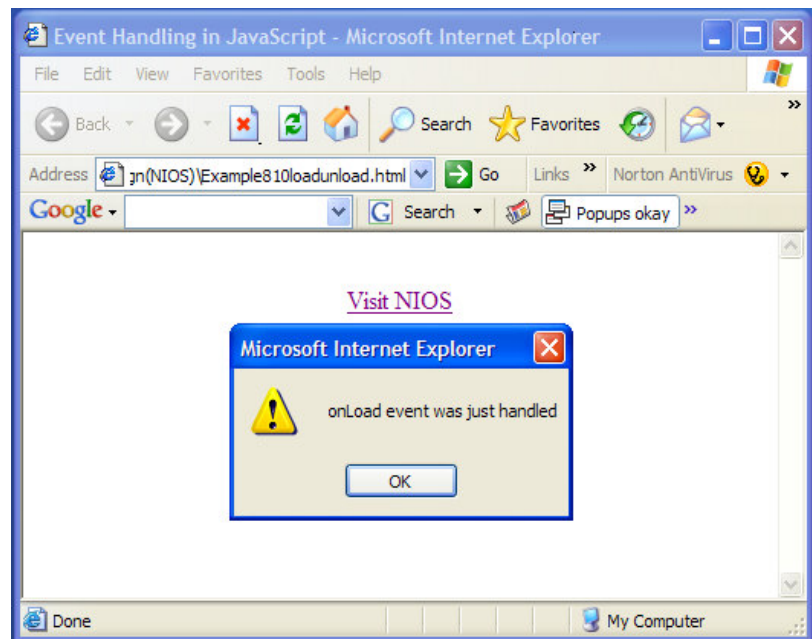


Fig. 8.12 (onLoad event handling)

```

<BR>
<A HREF="http://www.nios.ac.in"
onClick = "displayAlert()"> Visit
NIOS </A>
</CENTER>
</body>
</html>
    
```

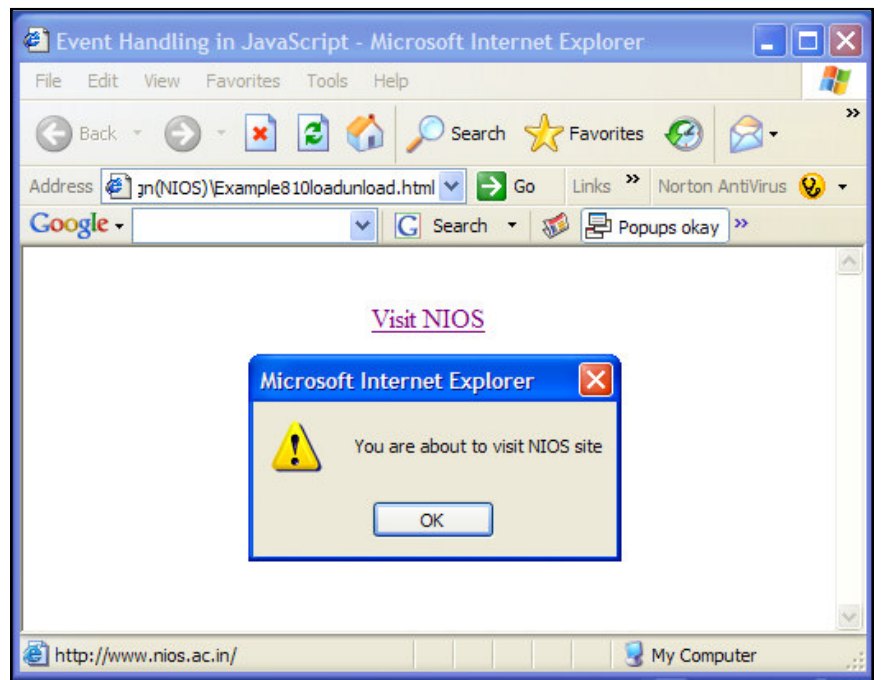


Fig. 8.13 (onClick event handling)

8.8 WINDOW OBJECT

The window object is a top level object in the JavaScript object model. It is also default object i.e. the statement `document.write("Hello! Welcome to JavaScript World")` is actually expanded as `window.document.write("Hello! Welcome to JavaScript World")` by the JavaScript interpreter.

Properties associated with the window object:

Property	Description
document	Refers to a document being displayed in a window
location	Refers to a URL of a window
status	Refers to message appearing on the status bar
name	Refers to the name of window

Methods of the window object:

Method	Description
alert(text)	Pop up a window with text as message.
confirm(text)	Pop up a message box with text displayed, along with buttons for OK and CANCEL.
prompt(text,defaultInput)	Display a dialog box with a text displayed along with text box to input user data and default value in text box.
open(URL,name,featureList)	Opens a new window, populated by URL, with the target name(optional) of the window, and whichever features (optional) are identified in the feature list.
close()	Closes the current window

Event and Event Handlers of the window object

Event	Event Handlers	Description
load	onLoad	Processes when the window finishes loading.
unload	onUnLoad	Processes when the window finishes unloading.

8.8.1 Form Validation with JavaScript

Form validation is the process of checking that a form has been filled in correctly before it is processed. This section covers how to create a JavaScript-enabled form that checks whether a user has filled in the form correctly before it is sent to the server.

First of all we should know why form validation is a useful thing, and then build up a simple example form. For example, if your form has a box for the user to type their email address, you might want your form handler to check that they've filled in their address before you deal with the rest of the form.

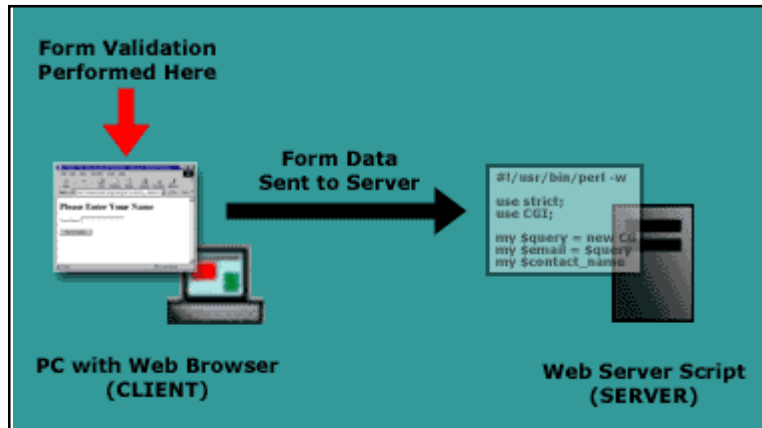


Fig. 8. 14 : Client-side form validation

Let us build a simple form with a validation script. The form will include one text field called **Your Name**, and a **submit** button. Our validation script will ensure that the user enters their name before the form is sent to the server.

An example 8.11 showing form validation with JavaScript

```
<HTML>
<HEAD>
<TITLE>JavaScript Form Validation</TITLE>
<SCRIPT language="JavaScript">
function validate_form ( )
{
    valid = true;
    if ( document.contact_form.contact_name.value == "" )
    {
        alert ( "Please fill in the 'Your Name' box." );
        valid = false;
    }
    return valid;
}
</SCRIPT>
</HEAD>
<BODY>
<form name="contact_form" method="post"
action="http://www.nios.ac.in/contact_simple.asp" onSubmit="return
validate_form ( );">
<h1>Please Enter Your Name</h1>
<p>Your Name: <input type="text" name="contact_name"></p>
<p><input type="submit" name="send" value="Send Details"></p>
</form>
</BODY>
</HTML>
```

You can see that the page consists of a JavaScript function called **validate_form ()** that performs the form validation, followed by the form itself. Let us look at the form first.

First part of the form is the form tag:

```
<form name="contact_form" method="post" action="http://  
www.nios.ac.in/contact.asp" onSubmit="return validate_form (  
);">
```

The form is given a name of "contact_form". This is so that we can refer the form by name from our JavaScript validation function.

The form uses the post method to send the data to a sever side script on www.nios.ac.in server(assumption). In reality, you would of course send the data to your own server side script JSP, ASP page, etc.

Finally, the form tag includes an onSubmit attribute to call our JavaScript validation function, validate_form (), when the Send Details button is pressed. The return allows us to return the value true or false from our function to the browser, where true means "carry on and send the form to the server", and false means "don't send the form". This means that we can prevent the form from being sent if the user hasn't filled it in properly.

The rest of the form prompts the user to enter their name into a form field called contact name, and adds a Send Details submit button:

```
<h1>Please Enter Your Name</h1>  
<p>Your Name: <input type="text" name="contact_name"></p>  
<p><input type="submit" name="send" value="Send  
Details"></p>  
</form>
```

Now let us take a look at the JavaScript form validation function that does the actual work of checking our form.

➤ The **validate_form ()** function

The form validation function, **validate_form ()**, is embedded in the **head** tag near the top of the page:

```
<SCRIPT language="JavaScript">
```

```
function validate_form ( )
```

```
{  
    valid = true;
```

We use this **valid** variable to keep track of whether our form has been filled out correctly. If one of our checks fails, we'll set **valid** to **false** so that the form won't be sent.

The next 5 lines check the value of our **contact_name** field to make sure it has been filled in:

```
    if ( document.contact_form.contact_name.value == "" )  
    {  
        alert ( "Please fill in the 'Your Name' box." );  
        valid = false;  
    }
```

If the field is empty, the user is warned with an **alert** box, and the variable **valid** is set to **false**.

Next, we return the value of our **valid** variable to the **onSubmit** attribute (described above). If the value is **true** then the form will be sent to the server; if it's **false** then the form will not be sent:

```
return valid;
```

Finally, we finish our **validate_form ()** function with a closing brace, and end our HTML comment and **script** tag:

```
}  
  
</SCRIPT>
```

8.9 VBSCRIPT

VBScript, a subset of Visual Basic Language is a scripting language. It is much easier to learn than programming language C, C++, Java, and other scripting language as JavaScript.

An important aspect of this programming model is that one can use Microsoft's ActiveX controls (graphs, charts, timer, banner etc) and intrinsic HTML controls that give web pages an attractive look and feel.

VBScript can play an important role in many ways: validating data, providing interactive response, and initiating data storage.

VBScript embedded inside HTML document might not run properly on the various platforms and operating systems, it is supported by Internet Explore only.

8.9.1 Comparison with JavaScript

JavaScript and VBScript have many similarities as well as several differences that are described below:

➤ **Embedded HTML Languages**

Like JavaScript, VBScript is a scripting language embedded in HTML document. VBScript also uses the **<SCRIPT> tag** in the same as JavaScript, using VBScript as the LANGUAGE parameter.

A Sample code th

at shows use of SCRIPT tag in VBScript.

```
<HTML>

<HEAD>
<TITLE>VBScript sample Example</TITLE>
<SCRIPT language="VBScript">
  Sub displayalert()
    MsgBox("Hello! Testing VBScript")
  End Sub
</SCRIPT>
</TITLE>
</HEAD>

<BODY onLoad="VBScript:call displayalert">
VBScript Tested
</BODY >
</HTML>
```

The above calls VBScript function displayalert on page load and displays message using Msg Box procedure.

Here, in above example it is clear that the declaration of function(JavaScript) called as procedure in VBScript. In VBScript, there are two types of procedure

- Sub Procedure
- Function Procedure

A **sub procedure** is a series of statements, enclosed by Sub and End Sub statements, which perform actions but does not return value. A sub procedure can take parameters or arguments. The sub procedure called by the statement: call subprocedurename. VBScript also have some built in object such as: MsgBox to

display messages and InputBox to accept some information.

A **function procedure** is a series of statements enclosed by Function and End Function statements. A function procedure is similar to a sub procedure, but has an added functionality for returning a value. A function procedure can also take parameters. A function returns a value by assigning a value to its name.

➤ **Identical Object Model**

Both JavaScript and VBScript use identical object models. Both JavaScript and VBScript interact with HTML in an identical manner. Like JavaScript responds to events triggered by objects, VBScript also responds similarly to events

➤ **Industry Support**

JavaScript has wider acceptance and industrial support than VBScript, due to its platform independent and run on any Operating environment nature. JavaScript is supported by all most all browsers. VBScript runs only under Internet Explorer.

INTEXT QUESTION

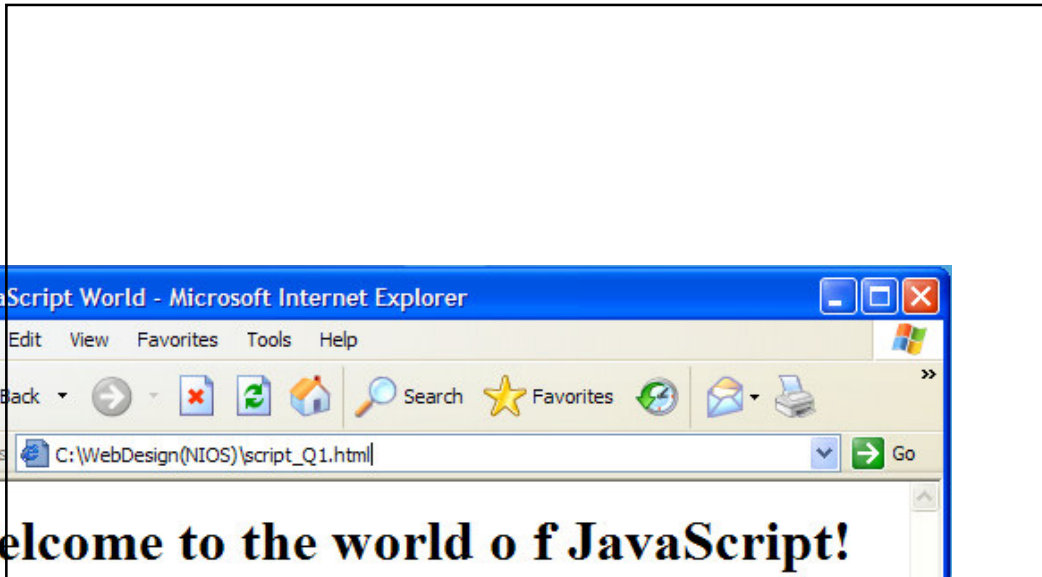
2. Write True or False for the following
- (a) The window object is a top level object in the JavaScript object model.
 - (b) A sub procedure is a series of statement.
 - (c) The function that is executed in response to the event is called an event responder.
 - (d) Load event occurs when web page load into browser.
 - (e) Open method of window object opens a new window.
 - (f) VBScript runs only under Internet Explorer.
-

8.10 WHAT YOU HAVE LEARNT

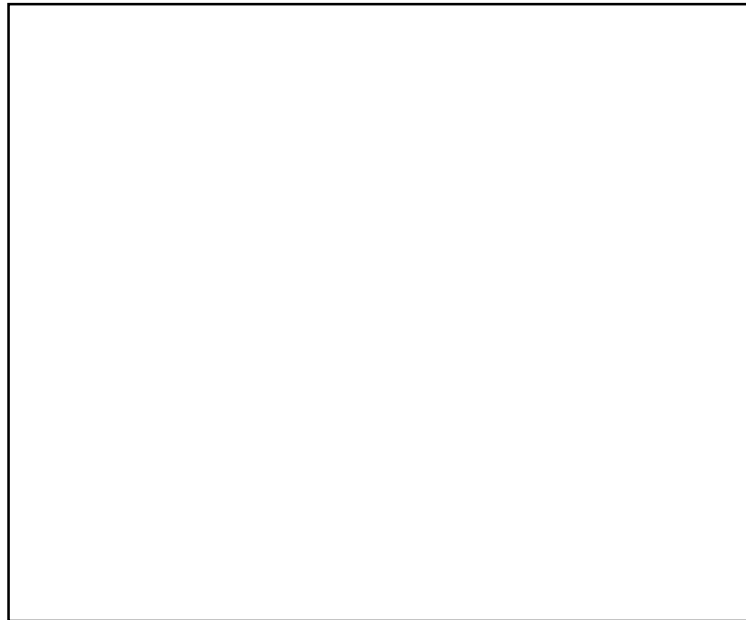
In this lesson you learnt about characteristics of Javascript, variables in Java script, and use of operators. We discussed looping concepts and Functions as well as how to handle event along with window object. The comparison between VBscript and Java script is also discussed.

8.11 TERMINAL QUESTIONS

1. What is client side and server side script?
2. Write a script that creates a document as shown in figure:



to 10



4. Create a document which on loading, automatically displays an alert dialog box informing the user that “the user is about to enter the site”.
5. Create a document and add a link to it. When the user moves the mouse over the link it should load the linked document on its own. (User is not required to click the link)
6. Create a document with a form having at least two text fields such that if the user leaves these fields blank they should be prompted to enter data in these fields when they click on the submit button.
7. Compare JavaScript with VBScript

8.12 FEEDBACK TO INTEXT QUESTIONS

1. (a) scripting language (b) Operator
(c) <SCRIPT> (d) ++ (e) <HEAD>
 2. (a) True (b) True (c) False
(d) True (e) True (f) True
-