Unit-IV

Chapter 7

Working with Java Script

1. Introduction to JavaScript

JavaScript was introduced in 1995 as a way to add programs to web pages in the Netscape Navigator browser. The language has since been adopted by all other major web browsers. It is also used in traditional websites to provide various forms of interactivity and dynamicity.

It is important to note that JavaScript has no relation with language named Java. The similar name was assigned due to inspired by marketing considerations. When JavaScript was being introduced, the Java language was marketed gaining in popularity. Someone thought it a good idea to try to ride along on this success.

JavaScript is most commonly used as a client side scripting language. This means that JavaScript code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it.

JavaScript is known as scripting language, because it can embedded in HTML page and interpreted by browser. It is client side language as it is executed on client machine (browser).

JavaScript is not a programming language in strict sense. Instead of it is a scripting language because it uses the browser to do the dirty work. If you command an image to be replaced by another one, JavaScript tells the browser to do it. Because the browser actually does the work, you only need to pull some strings by writing some relatively easy lines of code.

JavaScript is a full-fledged programming language. It is possible to write quite complex programs in JavaScript. As we know, there are the browser differences. Though modern web browsers all support JavaScript, there is no sacred law that says they should support exactly the same JavaScript. A large part of this site is devoted to exploring and explaining these browser differences and finding ways to cope with them.

2. Features and use of JavaScript

- JavaScript programs are run by an interpreter built into the user's web browser (not on the server).
- JavaScript is client side language.
- JavaScript is less programming language, but it is a script language.
- It is light weighted, object-based programming.
- It is widely used and supported.
- It is accessible to the beginner.
- JavaScript is not complied language, it is interpreted by web browser (All browsers are interpreter of HTML)

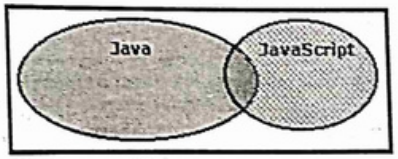
- JavaScript can dynamically modify an HTML page.
- JavaScript can react to user input and can validate user input.
- JavaScript can be used to create cookies.
- JavaScript is a full-featured programming language.
- JavaScript user interaction does not require any communication with the server.
- JavaScript code is case sensitive.
- JavaScript is platform independent.
- White space between words and tabs are ignored.
- Line breaks are ignored except within a statement.
- JavaScript statements end with a semi- colon ;
- It Requires a JavaScript-enabled browser.
- JavaScript is loosely based on Java and it is built into all the major modern browsers.
- JavaScript can be inserted into HTML documents by using the SCRIPT tag.

Uses of JavaScript

- Use it to add multimedia elements With JavaScript you can show, hide, change, resize images, and create image rollovers. You can create scrolling text across the status bar.
- Create pages dynamically Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.
- Interact with the user. It can do some processing of forms and can validate user input when the user submits the form.

3. Java and JavaScript

Although the names are similar, Java and JavaScript are different languages. Java is a full-fledged object-oriented programming language. Developed by Sun Microsystems (Now it handle by oracle). Java can be used to create stand-alone applications and a special type of mini-application called a Java applet.



Applets are written in Java, compiled, and then referenced via the <APPLET> tag.. JavaScript is an object-based scripting language. Although it uses some of Java's expression syntax and basic program flow controls, JavaScript stands alone and does not require Java. In fact, any similarities between the two are due to their use of objects. JavaScript was not developed by the same company, nor was it developed with Java in mind. Below table illustrates the similarities :

JavaScript	Java
Interpreted (not compiled) by client.	Compiled on server before execution on client.
Object-based: code uses built-in, extensible objects, but no classes or inheritance.	**Object-oriented:** applications and applets consist of object classes with inheritance.
Variable data types not declared (loose typing).	Variable types must be declared (strong typing).
Dynamic binding: object references checked at run time.	**Static binding:** object references must exist at compile time.
Secure: Not secure as whole code is view by user.	**Secure:** Secure is come to user.
Code integrated with and embedded in HTML.	Code is not integrated with HTML.

4. JAVASCRIPT SYNTAX:

A JavaScript consists of JavaScript statements that are placed within the
<script>... </script> HTML tags in a web page.
<script> tag can place anywhere within you web page but it is preferred way to keep it within the <head> tags.
The <script> tag alert the browser program to begin interpreting all the text between these tags as a script. So simple syntax of your JavaScript will be as follows

```
<script>
JavaScript Code
</script>
```

The script tag have two important attributes:

- **Language:** This attribute specifies what scripting language you are using. Typically, its value is JavaScript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript segment will look like:

```
<script language="Javascript" type="Javascript/text">
JavaScript Code
</script>
<html>
<head>
<title>Hello JavaScript</title>
</head>
<body>
<script type="text/javascript">
document.write("Hello World!!!");
</script>
</body>
</html>
```

Denoting JavaScript versions

As noted earlier, several different versions of JavaScript exist. If you need to ensure that browsers know which version of JavaScript you are using, this information can be placed in the <SCRIPT> tag using the LANGUAGE attribute. The following example demonstrates this syntax:

```
<HTML>
<HEAD>
<TITLE>Page Title</TITLE>
<SCRIPT LANGUAGE="JavaScript1.5">
//JavaScript 1.5 code goes here
</SCRIPT>
</HEAD>
<BODY>
HTML page text
</BODY>
</HTML>
```

In this example, the developer forces the user agent to use the 1.5 version of JavaScript. If the user agent cannot execute JavaScript 1.5 code, the entire script block is ignored.

4.1 POSITION TO PUT YOUR SCRIPTS

- You can have any number of scripts.
- Scripts can be placed in the HEAD or in the BODY.
- In the HEAD, scripts are run before the page is displayed.
- In the BODY, scripts are run as the page is displayed.
- HEAD is the right place to define functions and variables that are used by scripts within the BODY.

There is a flexibility given to include JavaScript code anywhere in an HTML document. But there are following most preferred ways to include JavaScript in your HTML file.

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in and external file and then include in <head>...</head> section.

```
<html>
<head>
<title>Hello World in JavaScript</title>
<script type="text/javascript">
function helloWorld()
{
document.write("Hello World!");
}
```

```
</script>
< head>
<body>
<script type="text/javascript">
helloWorld();
</script>
</body>
</html>
```

4.2 Whitespace and Line Breaks:

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. Because you can use spaces, tabs, and newlines freely in your program so you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

4.3 Semicolons and Case Sensitivity:

Statements in JavaScript are generally followed by a semicolon character (;), just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if statements are each placed on a separate line. For example, the following code could be written without semicolons

```
<script language="javascript" type="text/javascript">
var1 = 10
var2 = 20
</script>
```

But when formatted in a single line as follows, the semicolons are required:

```
<script language="javascript" type="text/javascript">
var1 = 10; var2 = 20;
</script>
```

Note: It is good programing practice to use semicolons.

Case Sensitivity: JavaScript is a case-sensitive language. This means that language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So identifiers temp, Temp and TEMP will have different meanings in JavaScript.

So care should be taken while writing your variable and function names in JavaScript.

4.4 Comments in JavaScript:

Comments is part of program which is read by compiler or interpreter but not complied or interpreted by compiler or interpreter respectively. Comments helps us to write extra information in code. JavaScript supports both C-style and C++-style comments.

Single-line comment indicator

Single line comment is use to add comments to a whole or partial line.

```
<SCRIPT LANGUAGE="JavaScript">
// Variables are defined here
var firstNum = 20
var secondNum = 0 // this value will change
```



```
</SCRIPT>
```

In this example, the portion of the coding from the // to the end of the line will be ignored by the JavaScript interpreter.

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.

Multiple-line comment indicator

If there is requirement to use a comment that extends beyond a single line. To perform this task, enclose the area that we does not want to execute with the /* and */ indicators. Note that the syntax for this comment is exact.

Note : Nesting of comments is not allowed, and it give error.

The following is an example of a multi-line comment:

```
<SCRIPT LANGUAGE="JavaScript">
() is used to calculate the two numbers that are supplied to the function by the
user.
*/
function addNumbers() {
//code for function
}
</SCRIPT>
```

- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

5. Working with Variables and Data Types:

Data Type: JavaScript allows you to work with three primitive data types:

- Numbers eg. 123, 120.50 etc.
- Strings of text e.g. "This text string" etc.
- Boolean e.g. true or false.

JavaScript also defines two trivial data types, null and undefined, each of which defines only a single value.

5.1 JavaScript Variables:

Like other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows:

```
<script type="text/javascript">
var money; // Variable declaration
```

var name;
</script>

There are rules and conventions in naming variables in any programming language. It is good practice to use descriptive names for variables. The following are the JavaScript rules:

- The variable name must start with a letter or an underscore:
  firstName or _myName
- Digits are allow in a variable name:
  name01 or tuition$
- Variable names are not start with digits:
  3_varname is wrong pattern
- You can't use space to separate characters :
  username (right) not user Name(wrong)
- Capitalize the first letter of every word except the first :
  salesTax or userFirstName
- Except underscore (_) and dollar ($). no other sign are allowed.
- To declare variables. use the keyword var and the variable name:
  var username;
- To assign values to variables. add an equal sign and the value:
  var userName = "Smith"
  var price = 100
- You should not use any of the JavaScript reserved keyword as variable name. These keywords are mentioned in the next section. For example, break or boolean variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or the underscore character. For example, 123test is an invalid variable name but _123test is a valid one.
- JavaScript variable names are case sensitive. For example, Name and name are two different variables.

5.2 JavaScript Variable Scope:

The scope of a variable is the region of program in which it is defined. JavaScript variable will have only two scopes.

- **Global Variables:** A global variable has global scope which means it is defined everywhere in your JavaScript code.
- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

5.3 JavaScript Keywords:

Keywords are some reserve words which are defined by any language. The following are reserved words in JavaScript. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	instanceof
boolean	int
break	interface
byte	long
case	native
char	new
class	null
const	package
continue	private
debugger	protected
default	public
delete	return
do	short
double	static
	super
else	switch
enum	synchronized
export	this
exrtends	throw
false	transient
final	true
finally	try
float	typeof
for	var
function	void
goto	volatile
if	while
implements	with
imports	
in	

6. Operators in JavaScript

6.1 The Arithmetic Operators:

There are following arithmetic operators supported by JavaScript language:
Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Add two operands	A + B will give 30
–	subtracts second oporand from the first	A – B will give –10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denumerator	B / A will give 2
%	Modulus operator and remainder of after an integer division	B % A will give 0
++	Icrement operator, increase integer value by one	A ++ will give 11
--	Decrement operator, decrease integer value by one	A -- will give 9

6.2 The Comparison Operators:

There are following comparison operators supported by JavaScript language
Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A ==B) is not true.
! =	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A ! = B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition become true	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A > = B) is not true.
<=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A < = B) is true.

6.3 The Logical Operators:

There are following logical operators supported by JavaScript language

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
&&	Called logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
||	Called logical OR operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT operator. Ue to reverses the Logical state of it operand. If a condition i true then logical NOT operator will make false.	! (A && B) is false.

6.4 The Bitwise Operators:

There are following bitwise operators supported by JavaScript language. Assume variable A holds 2 and variable B holds 3 then:

Operator	Description	Example
&	Called Bitwise AND operator. It performs a Boolean AND operation on each bit of its integer arguments.	(A & B) is 2.
|	Called Bitwise OR operator. It performs a Boolean OR operation on each bit of its integer arguments.	(A | B) is 3.
^	Called Bitwise XOR Operator. It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.	(A ^ B) is 1.
~	Called Bitwise NOT Operator. It is a is a unary operator and operates by reversing all bits in the operand.	(~B) is –4.
<<	Called Bitwise Shift Left Operator. It moves all bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying by 2, shifting two positions is equivalent to multiplying by 4, etc.	(A << 1) is 4.
>>	Called Bitwise Shift Right with Sign Operator. It moves all bits in its first operand to the right by the number of places specified in the second operand. The bits filled in on the left depend on the sign bit of the operand. The bits filled in on the left depend on the sign bit of the original operand, in order to preserve the sign of the result. If the first operand is positive, the result a zeros placed in the right bits; if the first operand is negative, the result has ones placed in the right bits. Shifting a value right one place is equivalent to dividing by 2 (discarding the remainder), shifting right two places is equivalent to integer division by 4, and so on.	(A >> 1) is 1.
>>>	Called Bitwise Shift Right with Zero operator. This operator is just like the >> operator, except that the bits shifted in on the left are always zero,	(A >>> 1) is 1.

6.5 The Assignment Operators:

There are following assignment operators supported by JavaScript language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND asignment operator, It add right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Ssubtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C - = A is equivalent to C = C – A
* =	Multiply and assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C* A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C/ A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A i equivalent to C = C % A

6.7 Miscellaneous Operator

The Conditional Operator (? :)

There is an operator called conditional operator. It is ternary operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X: Otherwise value Y

The type of Operator

The typeof is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The type of operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

7. External scripts

We can also include script as an external file. This is helpful in following cases:

- If your code is more complex.
- If you plan on revising the code often.
- If you plan on using the same code in multiple pages.

To create an external JavaScript file, use the .js file name extension. The .js file consists of native JavaScript code without the HTML <SCRIPT> tag. You can then include code similar to the following within the <HEAD> or <BODY> tags (note the use of the SRC attribute):

```
<SCRIPT LANGUAGE="JavaScript" SRC="JavaScriptCode.js">
  /*
Avoid adding embedded JavaScript code. It will not be executed if the .js file is unavailable.
  */
  </SCRIPT>
```

The browser will automatically read the code written in the .js file as if it were placed between the <SCRIPT> tags. As noted in the code, any additional embedded JavaScript code will not be executed if the .js file is not available in Netscape Navigator 4.x and higher and in Microsoft Internet Explorer 4.x and higher. If any additional code needs to be executed in the file, that code should reside in a different <SCRIPT> block. The JavaScript comment tags used in the previous examples will be discussed in detail in the next section.

- Scripts can also be loaded from an external file
- This is useful if you have a complicated script or set of subroutines that are used in several different documents.

```
<script src="myscript.js"></script>
```

8. Controlling Programming flow

8.1 Conditions:

If statement:

The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax:

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

if...else statement:

The if...else statement is the next form of control statement that allows JavaScript to execute statements in more controlled way.

Syntax:

```
if (expression)
{
Statement or statements to be executed if expression is true
else
{
Statement or statements to be executed if expression is false
}
```

if...else if... statement:

The **if**...else if... statement is the one level advance form of control statement that allows JavaScript to make correct decision out of several conditions.

Syntax:

```
if (expression 1)
{
Statement(s) to be executed if expression 1 is true
}
else if (expression 2)
{
Statement(s) to be executed if expression 2 is true
}
else if (expression 3)
{
Statement(s) to be executed if expression 3 is true
}
else
{
Statement(s) to be executed if no expression is true
}
```

switch statement:

The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, then control goes to **default statement.**

Usually default is last statement, so break is optional after default statement. Default can be use in between then we have use break.

Conclusion is that break is optional after last statement.

```
switch (expression)
{
case condition 1: statement(s)
break;
case condition 2: statement(s)
break;
```



```
case condition n: statement(s)
break;
default: statement(s)
}
```

8.2 Looping

JavaScript performs several types of repetitive operations, called "looping". Loops are set of instructions used to repeat the same block of code till a specified condition returns false or true depending on requirement. To control the loops use counter variable that increments or decrements with each repetition of the loop.

JavaScript supports three loop statements: for, do-while and while. The For statements are best used when you want to perform a loop a specific number of times. The while statements are best used to perform a loop an undetermined number of times. In addition, you can use the break and continue statements within loop statements.

If we want to execute a statement or group of statements are execute multiple times till any condition is reached, then we use loop.

8.2.1 While-Loop

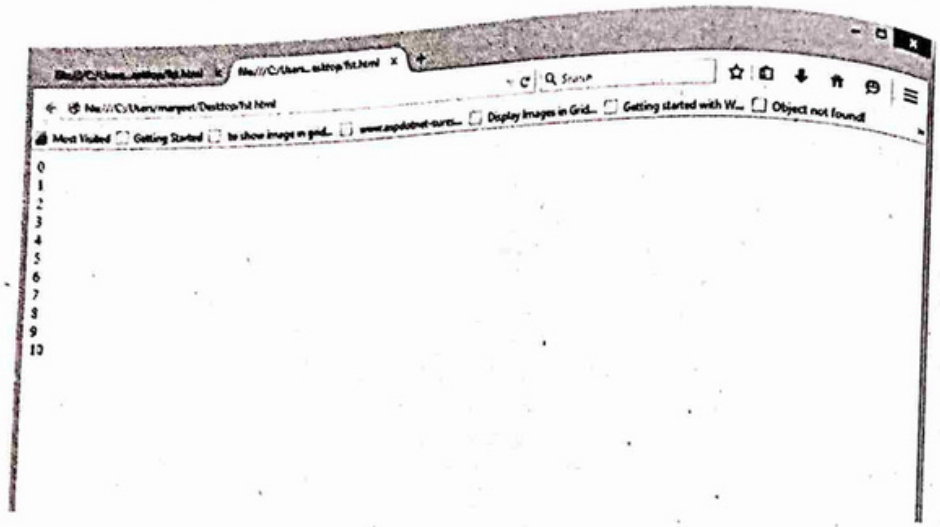
The While loop is another commonly used loop after the For loop. The while statement repeats a loop as long as a specified condition evaluates to true. If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. It is entry control loop, means first check condition then enter the loop. The while statement looks as follows:

Syntax:-

```
while(condtion)
{
Statements.
}
```

Example:-

```
<html><head><script type="text/javascript">
var i=0;
while (i<=10) //Output the values from 0 to 10
{
 document.write(i + "<br>")
 i++;
}
</script></head>
</body></html>
```



8.2.2. The do...while Loop:

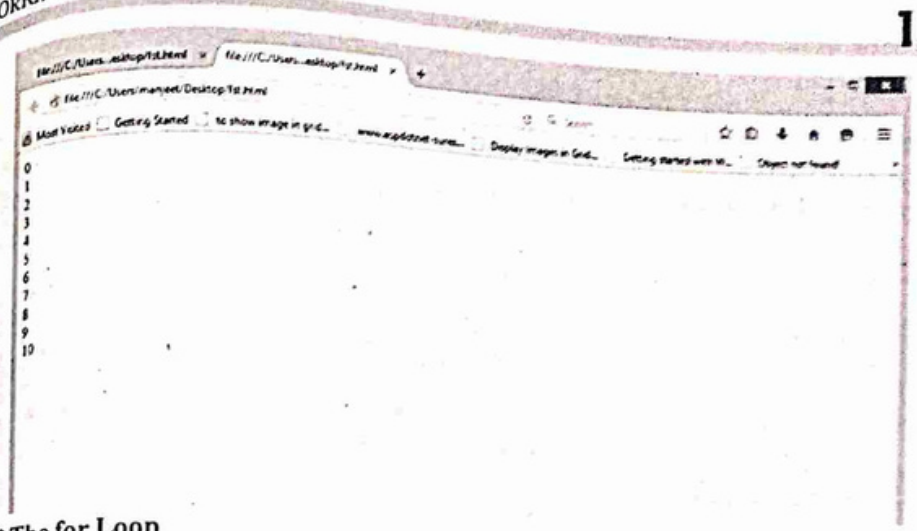
The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false. It is exit control loop as first enter the loop and then check condition.

Syntax:

```
do
{
Statements.
}while(condition);
```

Example:

```
<html><head><script type="text/javascript">
var i=0;
do//Output the values from 0 to 10
{
 document.write(i + "<br>")
 i++;
} while (i<=10);
</script></head>
</body></html>
```

8.2.3 The for Loop

It is also an entry control loop, means first check condition then enter the loop. The For loop is executed till a specified condition returns false. It has basically the same syntax then in other languages. It takes 3 arguments.

When the For loop executes, the following occurs:

1. The initializing expression is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
2. The condition expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the for loop terminates.
3. The update expression increment executes.
4. The statements execute, and control returns to step 2.

The following example generates a multiplication table 2 through 9. Outer loop is responsible for generating a list of dividends, and inner loop will be responsible for generating lists of dividers for each individual number:

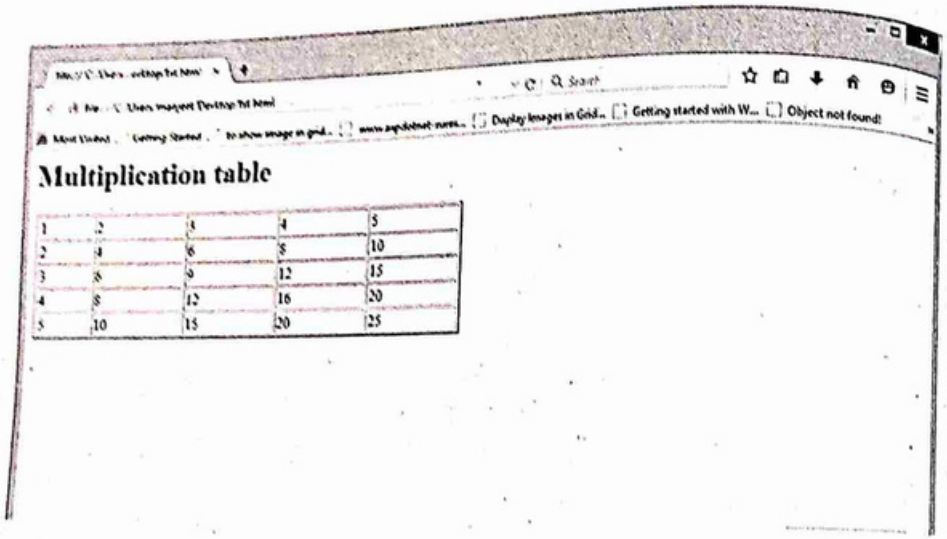
Syntax:

```
For(initialization; condition; increament/decreament)
{
Statement(s) to be executed if test condition is true
}
```

Example:-

```
<html><head><script type="text/javascript">
document.write("<h1>Multiplication table</h1>");
document.write("<table border=2 width=50%");
for (var i = 1; i <= 5; i++ ) {   //this is the outer loop
 document.write("<tr>");
```

```
 document.write("<td>" + i + "</td>");
  for ( var j = 2; j <= 5; j++ ) { // inner loop
     document.write("<td>" + i * j + "</td>");
   }document.write("</tr>");
} document.write("</table>");
</script></head>
<body></html>
```



8.3 JUMP STATEMENTS

8.3.1 The break Statement:

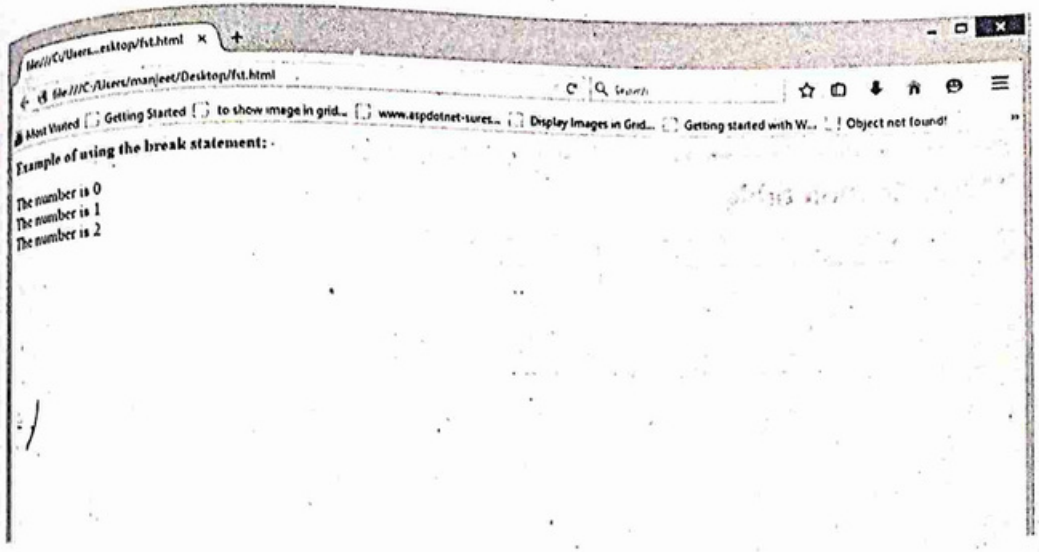
The break statement, is used to exit a loop early, breaking out of the enclosing curly braces. When break statement is encountered control goes out of loop without checking the condition of loop and without executed further statements of loop.

Example:

```
<html>
<head>
<script type="text/javascript">
document.write("<p><b>Example of using the break statement:</b></p>");
var i = 0;
for (i=0; i<=10; i++) {
 if (i==3){break}
```



```
document.write("The number is " + i);
document.write("<br />");
}</script>
</head>
<body>
</body>
</html>
```



8.3.2 The continue Statement:

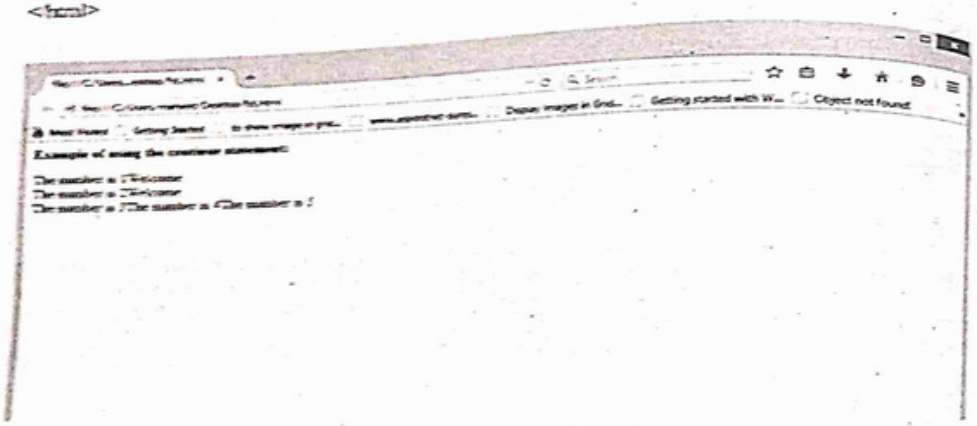
The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block.

When a continue statement is encountered, program flow will move to the loop check expression immediately and if condition remain true then it start next iteration otherwise control comes out of the loop.

Example:-

```
<html>
<head><script type="text/javascript">
document.write("<p><b>Example of using the continue statement:</b><p>");
for (i=1; i<=5; i++) {
 document.write("The number is " + i);
 if (i>=3)
   continue;
```

```
  document.write("Welcome");
  document.write("<br />")
}
</script>
</head>
<body>
</body>
</html>
```



9. FUNCTION:

Defining a Function starts by using the function Keyword, which is followed by the name of the Function and then a comma-separated list of Arguments that are enclosed within parentheses ({ }). You can define a Function that takes no Arguments, but still have to include the parentheses. Next comes a sequence of Statements that are semicolon separated and enclosed within curly braces {}.

Before we call a function we need to define that function. To define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Using functions, give a name to a whole block of code, allowing you to reference it from anywhere in your program. The basic syntax is shown here:

```
<script type="text/javascript">
function functionName(argument1, argument2, ..., argumentN) {
```



```
statement1;
statement2;
statementN;
}</script>
```

Calling a Function:

To invoke a function somewhere later in the script, you would simple need to write the name of that function as follows:

```
<script type="text/javascript">
functionname(parameter-list)
</script>
```

Type of Functions

9.1 PRE-DEFINE FUNCTION

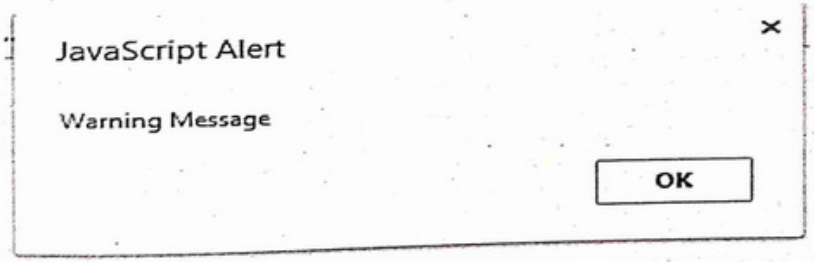
Those function which are not need to define by user, user just call the functions.

Here are some functions.

- alert("message")
- confirm("message")
- prompt("message")

Alert Dialog Box: It is alerts the user that something has happened. An alert dialog box is mostly used to give a warning message to the users. Like if one input field requires to enter some text but user does not enter that field then as a part of validation you can use alert box to give warning message as follows:

```
<html><head><script type="text/javascript">
alert("Warning Message");
</script></head>
<title>first</title>
<body></html>
```

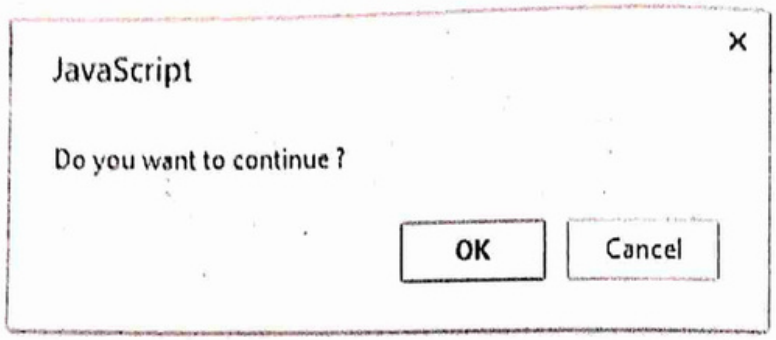


Confirmation Dialog Box: It is asks the user to confirm (or cancel) something. A confirmation

dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: **OK** and **Cancel**.

You can use confirmation dialog box as follows:

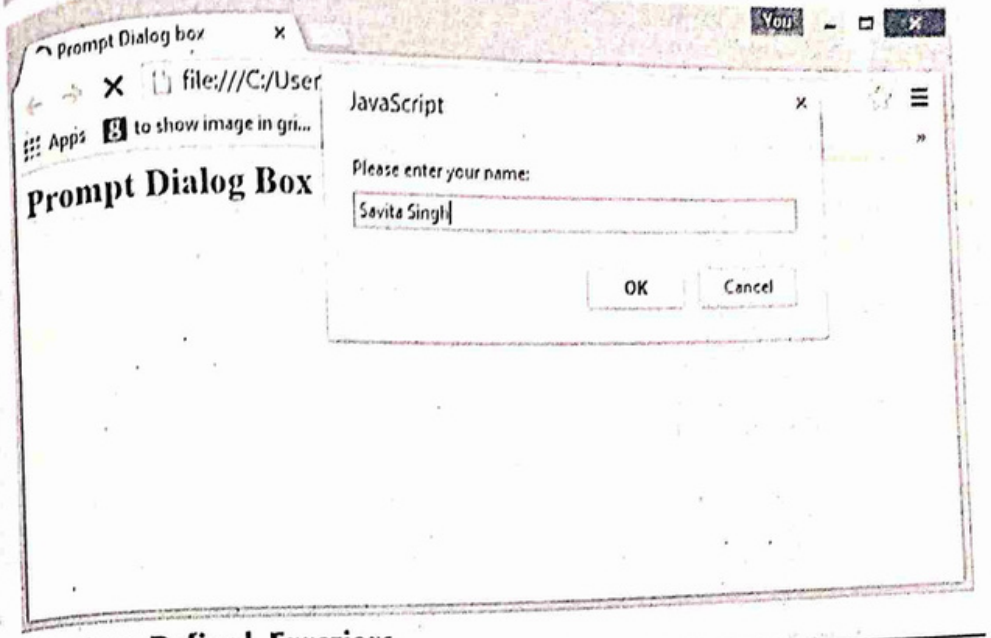
```
<html><head><script type="text/javascript">
 confirm("Do you want to continue ?");
</script></head>
<title>first</title>
</body></html>
```



Prompt dialog box: It is asks the user to enter some text.Prompt dialog box is used to take any input from user. This dialog box contains two buttons **Ok** and **Cancel**.

Example:

```
<HTML>
<HEAD>
<TITLE>Prompt Dialog box </TITLE>
</HEAD>
<BODY>
<h2>Prompt Dialog Box</h2>
<SCRIPT language="JavaScript">
var userName = prompt("Please enter your name:", "")
document.write("<H1>Welcome " + userName + "</H1><BR>")
document.write("<H2>to your new home page.</H2>")
</SCRIPT>
</BODY>
</HTML>
```

9.2 USER-DEFINED FUNCTIONS

With pre-define function, user can also define their own function for any particular purpose. With user-defined functions, name a block of code and call it when according to need. You define a function in the HEAD section of a web page. It is defined with the function keyword, followed by the function name and any number of arguments.

Syntax:

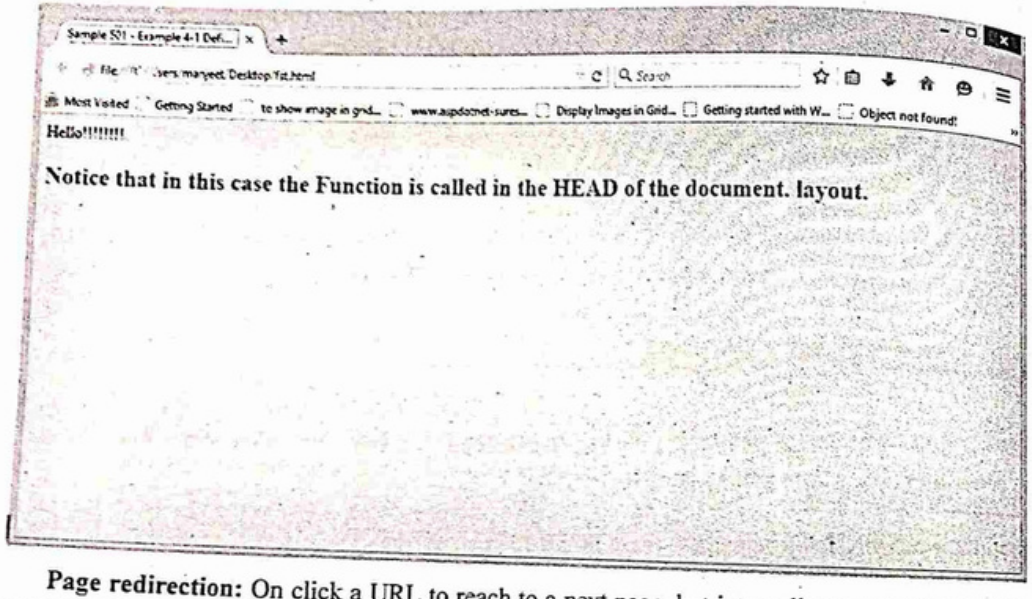
```
function functionName(argument)
{
statements
}
```

Example:-

```
<HTML>
<HEAD>
<BASEFONT SIZE="7">
<TITLE> Define and Call a Function </TITLE>
<SCRIPT LANGUAGE="JavaScript">
//Begning
function sayHello()
{
document.write("Hello!!!!!!!!.");
document.bgColor= "pink";
}
```

```
sayHello();//End
</SCRIPT></HEAD>
<BODY>
<H2><FONT COLOR="#FF0000">
Notice that in this case the Function is called in the HEAD of the document.
</FONT></H2>
</BODY>
</HTML>
```

Here document.write() is in-built function used to display the output, we will discuss in detail in later.



Page redirection: On click a URL to reach to a next page, but internally you are directed to another page that simply happens because of page re-direction. This concept is different from JavaScript Page Refresh.

There could be various reasons why you would like to redirect from original page.

- Same time we want to direct your all visitors to new site. In such case you can maintain your old domain but put a single page with a page re-direction so that all old domain visitors can come to your new domain.
- You have build-up various pages based on browser versions or their names or may be based on different countries, then instead of using your server side page redirection you can use client side page redirection to land your users on appropriate page.
- The Search Engines may have already indexed your pages. But while moving to another domain then you would not like to lose your visitors coming through search engines. So you can use client side page redirection. But keep in mind this should not be done to make search engine a fool otherwise this could get your web site banned.


```
<head>
<script type="text/javascript">
function Redirect()

{
window.location="http://www.newlocation.com";

}
document.write("You will be redirected to main page in 10 sec.");
setTimeout('Redirect()', 5000);
</script>
</head>
```

setTimeout() is a built-in JavaScript function which can be used to execute another function after a given time interval. Here Redirect() function is called after 5 seconds.

10. JAVASCRIPT OBJECTS

Objects enable you to gather together the related bundles of functionality you define as functions, and bind them into a coherent package that you can pass around and refer to as a single item. This ability has very practical impacts upon the code as you are able to write, even if it sounds a little abstract at the moment.

An object is a collection of data and behaviors, or information. In JavaScript, any information about an object is considered a property.

In a class-based language, the classes are typically defined when the class is compiled. Once an instance is created within an application, adding or removing properties or methods is impossible. JavaScript, on the other hand, is a prototype-based language. In a prototype-based language, the objects that are based on a prototype remain connected to it. New properties and methods can be added both to an individual instance and to the prototype on which it is based. If the definition of the prototype changes, all objects based on that prototype change as well. Note the format: object name, dot (.), method name, and arguments in parentheses.

objectname. property property= variables or values belonging an object objectname.method() methods= functions the object performs i.e. function that is part of an object object-variable-name.property-name = value.

Example:

```
<html>
<body>
<script>
var person=new Object();
person.firstname="Savita";
person.lastname="Singh";
person.age=28;
person.course="Javascript";
```

```
var message="Hello Javascript Object";
var x=message.toUpperCase();
document.write(person.firstname + " is " + person.age + " years old."+ person.course+" "+x);
</script>
</body>
</html>
```

Some basic objects are built-in to JavaScript

1. document
2. String
3. Date
4. Array
5. Boolean
6. Math

10.1 DOCUMENT- A IN-BUILT OBJECT

The most common and most basic object used in JavaScript is the document itself. The most common method is the write() method, which enables the output of information to the browser page. This information may be static text, or it may be variables or other object properties or methods.

The Document object represents the Web page that is loaded in the browser window, and the content displayed on that page, including text and form elements. Document represents the current page.

In JavaScript, any information about an object is considered a property, and information on how to do something, is methods. Methods are also considered properties of an object, because they provide information to do.

An object property can also contain another object. For example, the document object is actually a property of the window object, and in turn has many properties that are objects.

Only the round braces () distinguishes property from a method.

E.g. document.write() is an object with a method that outputs string to browser window.

Document Methods

You can use the methods of the document object to work on a Web page.

Here are the most common document methods:

- write() - write a string to the Web page
- open() - opens a new document
- close() - closes the document

Example

```
<HTML>
<HEAD>
<BASEFONT SIZE="7">
<TITLE>welcome </TITLE>
```



```
<SCRIPT language="JavaScript">
function newPage() {
var userName = prompt("Please enter your name:", "")
document.write("<H1>Welcome " + userName + "</H1><BR>")
document.write("<H2>to your new home page.</H2>")
}
</SCRIPT>
</HEAD>
<BODY>
<A HREF="JavaScript:newPage()">Create-a-Page!</A>
</BODY>
</HTML>
```

Document Properties

Use the properties of the document object to set the colors of the page, the title and display the date the document was last modified. JavaScript has about 150 defined color words you can use or you can provide the hexidecimal RGB codes. Here are the most common document properties:

- bgColor
- fgColor
- linkColor
- vlinkColor
- title
- lastModified

Example:

```
<HTML>
<HEAD>
<BASEFONT SIZE="7">
<TITLE>welcome </TITLE>
<SCRIPT language="JavaScript">
document.bgColor="pink"
</SCRIPT>
</HEAD>
<BODY>
<A HREF="JavaScript:newPage()">Create-a-Page!</A>
</BODY>s
</HTML>
```

10.2 Window Object

Window represents the actual browser viewport.The window object represents the browser window. You can use it to open a Web page in a new window and to set the attributes for the window. There are only two main window properties. They are:

- status - set the status bar message
- self - stores the name of the current window

Window Methods

The window methods are mainly for opening and closing new windows.

The following are the main window methods. They are:

- alert() - to display a message box
- confirm() - to display a confirmation box
- prompt() - to display a prompt box
- open() - to open a new window
- close() - to close a window

Example:

```
<HTML>
<HEAD>
<TITLE>welcome </TITLE>
<SCRIPT language = "JavaScript">
function openWin() {
window.open("windowtoo.html")
}
window.open("windowtoo.html", "newWindow",height=200,width=200,") //to open new //window of particular size
</SCRIPT>
<BODY>
<A HREF="JavaScript:openWin()">New Window!</A></HEAD>
</BODY>
</HTML>
```

Window Attributes

If the default new window does not suit your needs, you can specify different features of the window when you open it. The complete syntax of the "window.open" is as follow: window.open(URL, windowName, featureList) By default, if you do not specify any features, then a window will have all of them. If you specify any one feature, then the window will have only those you set equal to 1. The following are the window attributes:

- toolbar
- menubar
- scrollbars
- resizeable

- status
- location
- directories

10.3 The String Object

The String object let's you work with a series of characters and wraps Javascript's string primitive data type with a number of helper methods.

Because Javascript automatically converts between string primitives and String objects. you can call any of the helper methods of the String object on a string primitive.

- String object is creat using declarations, the new keyword can be used.
- Strings have one propterty - length, myString.length will return the number of characters in the string var myString and 28 methods (JS 1.2).
- Creating a new string object

  var myString="Welcome to String" // assign string to variable or

  var new myString=(" Welcome to String ")
- new is keyword and operator also.new operator is optional when using built in objects, it creates a specific string that has properties and methods common to all strings.
- Another Syntax

  objectname = new objectName(params);.

Syntax:

Creating a String object:

```
var val = new String(string);
```

String Methods

Here is a list of each method and its description.

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns a number indicating the Unlcode value of the character at the given index.
concat()	Combines the text of two string and return a new string.
Index()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
last Index Of ()	Returns the index within the calling string object of the last occurrence of the specified value, or -1 if not found.
LocaleCompare()	Returns a number indicating whether a reference string comes efore or after or is the same as the given string in sort order.
match()	Used to match a regular expression against a string.

replace ()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
search()	Executes the search for a match between a regular expression and a specified string.
Slice()	Extracts a section of a string and returns a new string.
split()	splits a string object into an array of strings y separating the string into substrings.
substr()	Returns the characters in a string beginning at the specfied location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
to localeLowerCase()	The characters within a string are conveted to lower case while respecting the current locale.
ToLocaleUpper Case()	The characters within a string are converted to upper case will respecting the current locale
ToLowerCase()	Returns the calling tring value converted to lower case.
ToString()	Returns a string representing the specified object.
toUpperCae()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

10.4 sARRAYS OBJECT

The Array object let's you store multiple values in a single variable.

Syntax:

Creating a Array object:

var fruits=new Array("Apple","Mango","Orange");

The Array parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

You can create array by simply assigning values as follows:

var fruits=["Apple","Mango","Orange"];

We will access the elements like

fruits[0] is the first element

fruits[1] is the second element

fruits[2] is the third element

Array Properties:

Length: Reflects the number of elements in an array.

Array methods

Array methods are given below.

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Create a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
map()	Creates a new array with the result of calling a provided function on every element in thi array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more element to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two value of the array (from left to right) at to reduce it to a single value.
reduceRight()	Apply a function simultaneously agains two values of the array (from right to left) as to reduce it to a single value.
reverse()	Reverses the order of the elements of an array -- the first become the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that elemen.
slice()	Extracts a section of an array and return a new array.
some()	Returns true if at least one element in thi array satisfies the provided testing function.

tosource()	Represents the source code of an object
sort()	Sorts the element of an array.
splice()	Adds and or removes elements from an array.
toString()	Rturns a string representing the array and its elements.

10.5 DATE Object

The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ).

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal) time.

The Date object to be able to represent any date and time, to millisecond precision, within 100 million days before or after 1/1/1970. This is a range of plus or minus 273,785 years, so the JavaScript is able to represent date and time till year 275755.

Syntax:

Here are different variant of Date() constructor:

new Date( )

new Date(milliseconds)

new Date(datestring)

new Date(year,month,date[,hour,minute,second,millisecond)

Note: Paramters in the brackets are always optional

Here is the description of the parameters:

No Argument: With no arguments, the Date( ) constructor creates a Date object set to the current date and time.

milliseconds: When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the getTime( ) method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70.

datestring:When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse( ) method.

7 agruments: To use the last form of constructor given above, Here is the description of each argument:

1. **year:** Integer value representing the year. For compatibility (in order to avoid the Y2K problem), you should always specify the year in full; use 1998, rather than 98.
2. **month:** Integer value representing the month, beginning with 0 for January to 11 for December.
3. **date:** Integer value representing the day of the month.
4. **hour:** Integer value representing the hour of the day (24-hour scale).
5. **minute:** Integer value representing the minute segment of a time reading.
6. **second:** Integer value representing the second segment of a time reading.

7. **millisecond:** Integer value representing the millisecond segment of a time reading.

Date methods

Method	Description
Date()	Returns today' date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getFullYear()	Returns the year of the specified date according to locale time.
getHours()	Return the hour in the specified date according to local time.
getMilliseconds()	Returns the milliseconds in the specified data according to local time.
getMinutes()	Returns the minutes in the specified date according to local time.
getMonth()	Returns the month in the specified date according to local time.
getSeconds()	Returns the seconds in the specified date according to local time.
getTime()	Returns the numeric value of the specified date as the number of milliseconds since January 1,1970, 00 : 00 : 00 UTC.
getYear()	**Deprectated-** Returns the year in the specified date according to local time. Use getFullYar intead.
setDate()	Sets the day of the month for a specified date according to local time.
setFullYear()	Sets the full year for a specified date according to local time.
etHours()	Sets the hours for a specified date according to local time.
setMilliseconds()	Sets the milliseconds for a specified date according to local time.
setMinutes()	Sets the minutes for a specified date according to local time.
setMonth()	Sets the month for a specified date according to local time.
set econds()	Sets the seconds for a specified date according to local time.
SetTime()	Set the Date object to the time represented b a number of milliseconds since January 1, 1970, 00 : 00 : 00 UTC.
setYear()	**Deprecated-** Sets the year for a specified date according to local time. Ue setFullYear intead.
toDateString()	Returns the "date" portion of the Date as a human-readable string.

toGMTString()	**Deprecated** - Converts a data to a string, using the Internet GMT conventions. Use toUTCString instead.
toLocalDateString()	Returns the "date" portion of the Date as a string, using the current locale's conventions.
toLocaleFormat()	Converts a date to a string, using a format string.
toLocalString()	Converts a date to a string, using the current locale's conventions.
toLocaleTimeString()	Returns the "time" portion of the Date as a string, using the current locale's conventions.
toSource()	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.
toString()	Returns a string representing the pecified Date object.
toTime String()	Returns the "time" portion of the Date as a human-readable string.
toUTCString()	Converts a date to a string, using the unversal time convention.
valueOf()	Returns the primitive value of a Date object.

10.6 MATH OBJECT

The math object provides you properties and methods for mathematical constants and functions. Unlike the other globel objects, Math is not a constructor. All properties and methods of Math are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant pi as Math.PI and you call the sine function as Math.sin(x), where x is the method's argument.

Syntax:

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

Math Method

Method	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
asin()	Returns thearcsine (in radians) of a number
atan()	Returns the arctangent (in radians) of a number.
atanI()	Returns the arctangent of the quotient of its arguments.
ceil()	Returns the smallest integer greater than or equal to a number.

cos()	Returns the cosine of a number.
exp()	Returns E^N, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
Max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Return base to the exponent power, that i, base exponent.
random()	Returns a pseudo-random number between 0 and 1.
round()	Returns the value of a number rounded to the nearest integer.
sin()	Returns the sine of a number.
sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.
toSource()	Returns the string "Math".

11. JAVASCRIPT OBJECT MODEL -DOM (DOCUMENT OBJECT MODEL)

The Document Object Model originated as a specification to allow JavaScript scripts and Java programs to be portable among web browsers. Dynamic HTML was the immediate ancestor of the Document Object Model.

Defination:- The Document Object Model (DOM) is a specification that determines a mapping between programming language objects and the elements of an HTML document. When a web page is loaded, the browser creates a Document Object Model of the page. The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.

The Document Object Model, or DOM, is an interface to allow programs and scripts to update content, structure, and style of documents dynamically. It is platform- and language-neutral. The DOM is not HTML nor is it JavaScript. It is something like the glue that binds them together.

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page

- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

Every web page resides inside a browser window which can be considered as an object.

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

The way that document content is accessed and modified is called the Document Object Model, or DOM. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- Window object: Top of the hierarchy. It is the outmost element of the object hierarchy.
- Document object: Each HTML document that gets loaded into a window becomes a document object. The document contains the content of the page.
- Form object: Everything enclosed in the <form>...</form> tags sets the form object.
- Form control elements: The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

DOM is a platform and language neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page."

The DOM specifies an API (application programming interface) and provides a structural view of the document. DOM lists required interface objects and the methods and fields (data entries in the object) each object must support.

As an interface, each DOM object exposes a set of fields and methods for JavaScript to access and manipulate the underlying data structure that actually implements the document structure.

The DOM tree represents the logical structure of a document. Each tree node is a Node object. There are different types of nodes that all inherit the basic Node interface.

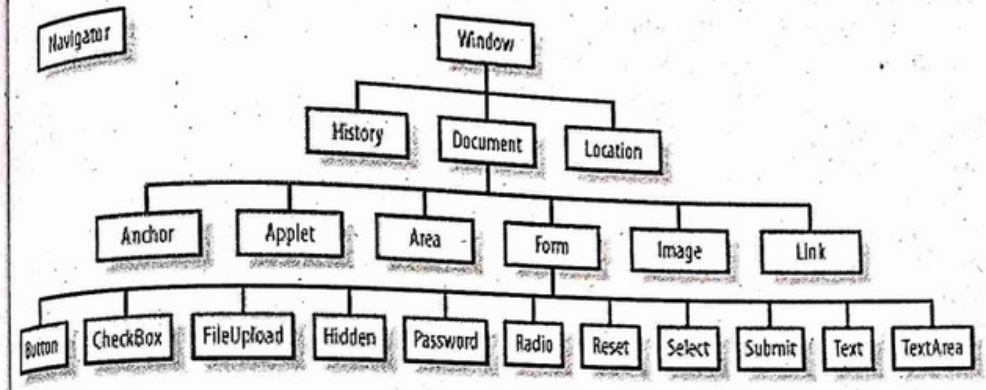
In DOM, the Node object sits at the top of the interface hierarchy and many types of DOM tree nodes are directly or indirectly derived from Node. This means all DOM tree node types must support the properties and methods required by Node.

On the DOM tree, some types of nodes are internal nodes that may have child nodes ofvarious types. Leaf nodes, on the other hand, have no child nodes. For any Web page, the root of the DOM

tree is an HTMLDocument node and it is usually available directly from JavaScript as document or window.document.

The document object implements the HTMLDocument interface which gives you access to all the quantities associated with a Web page such as URL, stylesheets, title, characterSet, and many others.

Here is a simple hierarchy of few important objects:



In other words, the nodes in the above diagram do not represent a data structure, they represent objects, which have functions and identity. As an object model, the Document Object Model identifies:

- the interfaces and objects used to represent and manipulate a document
- the semantics of these interfaces and objects - including both behavior and attributes
- the relationships and collaborations among these interfaces and objects

The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

The DOM architecture consists of modules covering different domains of the document object model:

DOM HTML : inherits from the Core and provides specialized and convenient ways to access and manipulate HTML/XHTML documents.

DOM XML:inherits from the Core and provides support for XML specific needs. DOM Events specifies events and event handling for user interfaces and the DOM tree. With DOM Events, drag and drop programs, for example, can be standardized.

DOM CSS : defines easy to use ways to manipulate Cascading Style Sheets for the formatting and presentation of documents.

document Object: The base structure of the DOM is the document master container. This refers to the entire document, and all elements within it (as discuss earlier). It is referenced as: document

Because the DOM root is very basic, it only allows the developer to manipulate items on the page.

documentAlways refers to the document root. Properties of documents objects are described in previous section.

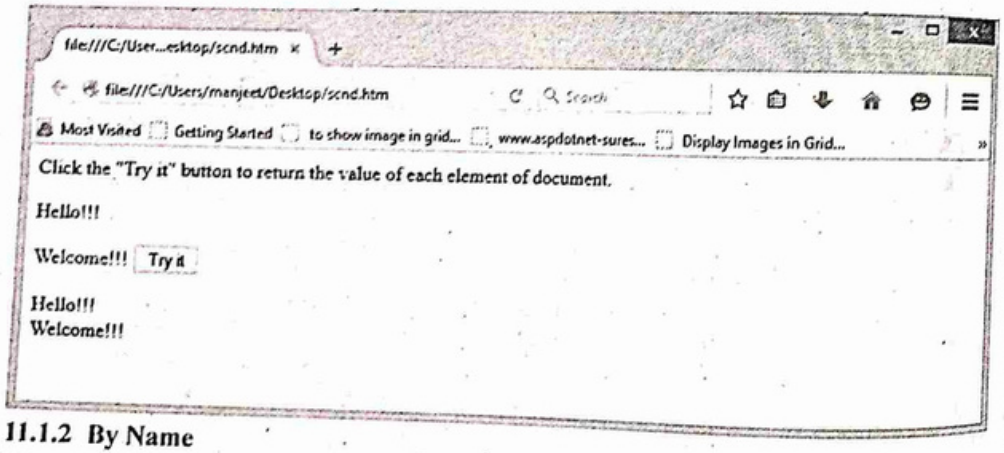
11.1 VARIOUS REFERENCING FORMS

There are two ways to reference a HTML element control using DOM.

11.1.1 By id

This is the easiest way to reference a HTML control. For example.

```
<html>
<body>
<p>Click the "Try it" button to return the value of each element of document.</p>
<p id="demo">Hello!!!</p>
<span id="spn" style="color:blue;">Welcome!!!</span>
<button onclick="myFunction()">Try it</button>
<p id="ans"></p>
<script>
function myFunction(){
var x = document.getElementById("demo").innerHTML;
var txt = document.getElementById("spn").innerHTML;
document.getElementById("ans").innerHTML=x+"<br>"+txt;
}</script>
</body>
</html>
```



11.1.2 By Name

Then if you wanted to reference an element within the form, you would name it as well:

```
<html>
<body>
<form>
<input name="to" type="text" value="Welcome!!!!" />
<input type="button" onclick="getValue()" value="Get Value!" />
</form/>
<script type="text/javascript">
function getValue()
 {
 alert(document.getElementsByName("to").value);
 }
</script>
</body>
</html>
```


11.1.3 By Tag Name

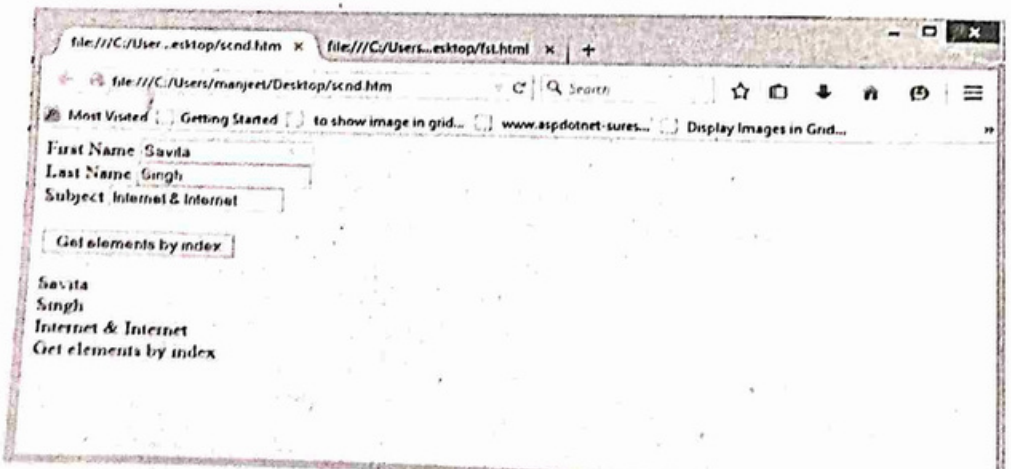
Example using DOM by tag name

```
<!DOCTYPE html>
<html>
<body>
<p>An unordered list:</p>
<ul>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>
<p>Click the button to display the innerHTML of the second li element (index 1).</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
   var x = document.getElementsByTagName("LI");
   document.getElementById("demo").innerHTML = x[1].innerHTML;
}
</script>
</body>
</html>
```

11.1.4 By Number(Index Number)

Each form, and element within the form, is given a number in an array, starting with 0. They are numbered starting with the first <form> element found in the flow of the document.

```
<html>
<head>
<script>
function getElements()
 {
 var y=document.getElementById("f1");
 var val = "";
 for (var i=0;i<y.length;i++)
 {
 val = val + y.elements[i].value + "<br>";
 }
 document.getElementById("demo").innerHTML=val;
 }
</script>
</head>
<body>
<form id="f1">
First Name <input type="text"  value=""><br>
Last Name <input type="text"  value=""><br>
Subject <input type="text"  value=""><br><br>
<input type="button" onclick="getElements()" value="Get elements by index">
</form>
<p id="demo"></p>
</body>
</html>
```



12. EVENTS HANDLING IN JAVASCRIPT

Events are methods which are on user interaction. User actions cause events. An event is some notable action occurring inside the browser to which a script can respond. An event model with intersect with JavaScript language, HTML, and the DOM model. Events can be simple like mouse movement or complex like copy-paste procedures. Event handler is JavaScript code associated with a particular part event usually in relationship to a particular part of the document such as form button. Events are bound often in HTML using various HTML attributes prefixed with the word on. for example onclick, ondblclick,onmouseover, etc.

The objects in a Web pages are organized in a hierarchy. All objects have properties and methods. In addition, some objects also have "events".

Events are things that happen, actions, that are associated with an object. Events are things that happen usually user actions,that are associated withan object. JavaScript event handlers, are JavaScript code that are not added inside the <script> tags, but rather inside the html tags that execute JavaScript when something happens such as pressing a button, moving your mouse over a link, submitting a form etc. The syntax of these event handlers is:

name_of_handler="JavaScript code here"

example: <bodyonload="alert('Hello world!');">

An event occurs when something happens in a browser window. The kinds of events that might occur are due to:

- A document loading
- The user clicking a mouse button
- The browser screen changing size

When a function is assigned to an event handler, that function is run when that event occurs.

Events are actions that can be detected by JavaScript, and the event object gives information about the event that has occurred. Sometimes we want to execute a JavaScript when an event occurs, such as when a user clicks a button. Events are normally used in combination with functions and the function will not be executed before the event occurs! Javascript event handlers are divided into 2 types:

1. Interactive event handlers- depends on user interaction with the HTML page ex. Clicking a button
2. Non-Interactive event handlers-does not need user interaction. There are also some events that aren't directly caused by the user. Ex. Onload().

As you can, this is certainly unlike a regular JavaScript code in that here we're inserting it directly inside a HTML tag, via the onClick event handler. When the above link is clicked, the user will first see an alert message before being taken to Google.

Different event handlers with different HTML tags. For example, while "onclick" can be inserted into most HTML tags to respond to that tag's onclick action, something like "onload" (see below) only works inside the <body> and <img> tags.

The "event handler" is a command that is used to specify actionsin response to an event. For example:

- onLoad - occurs when a page loads in a browser
- onUnload - occurs just before the user exits a page
- onMouseOver - occurs when you point to an object
- onMouseOut - occurs when you point away from an object
- onSubmit - occurs when you submit a form
- onClick - occurs when an object is clicked

Below are some of the most common events:

Attribute	Trigger
onabort	Loading of image was interrupted
onblur	Element loses focus
onchange	Element gets modified
onclick	Element gets clicked
ondblclick	Element gets double clicked
onerror	An error occurred loading an element
onfocus	An element received focus
onkeydown	A key was pressed when an element has focus
onkeypress	A keystroke was received by the element
onkeyup	A key was released when the element has focus
onload	An element was loaded
onmousedown	The mouse button was pressed on the element
onmousemove	The mouse pointer moves while inside the element
onmouseout	The mouse pointer was moved outside the element
onmouseover	The mouse pointer was moved onto the element
onmouseup	The mouse button was released on the element.
onreset	The form's reset button was clicked
onresize	The containing window or frame was resized
onselect	Text within the element was selected
onsubmit	A form is being submitted
onunload	The content is being unloaded (e.g. window being closed)
onscroll	The user scrolls (in any direction and with any means).

The names of event handlers are directly connected to the events introduced previous in table(1.1).

for instance, the click event is associated with the onClick event handler, and the load event with the onLoad event handler.

Table 1.2 outlines which window and form elements have event handlers available to them.

Object	Event Handlers Available
Selection list	onBlur, onChange, onFocus
Text element	onBlur, onChange, onFocus, onSelect
Textarea element	onBlur, onChange, onFocus, onSelect
Button element	onClick
Checkbox	onClick
Radio button	onClick
Hypertext link	onClick, onMouseOver, onMouseOut
Clickable Imagemap area	onMouseOver, onMouseOut
Reset button	onClick
Submit button	onClick
Document	onLoad, onUnload, onError
Window	onLoad, onUnload, onBlur, onFocus
Framesets	onBlur, onFocus
Form	onSubmit, onReset
Image	onLoad, onError, onAbort

Table: 1.2

In HTML, JavaScript events can be included within any specified attribute - for example, a body tag can have an onChange() event:

```
<INPUT TYPE="text" onChange="
alert('Thanks for the entry.');
confirm('Do you want to continue?');
>
```

The content of the HTML event attributes is JavaScript code that is interpreted when the event is triggered, and works very similarly to the blocks of JavaScript. This form of code is used when you want to have the JavaScript attached directly to the tag in question.

This type of technique is called inline JavaScript. The use of inline JavaScript can be considered to be similar in nature to that of using inline css, where HTML is styled by putting css in style attributes. This is a practice that is best avoided in favour of more versatile techniques.

The advantage of using functions as event handlers, however, is that you can use the same event handler code for multiple items in your document and functions make your code easier to read and understand.

this keyword: The this keyword refers to the current object. In the case of

<INPUT TYPE="text" onChange="checkField(this)">

this refers to the current field object. Here checkFields(obj) is a javascript function which is called when text of textbox is changed, and takes the reference of current object field.

There are different ways to apply events. For this, DOM concept must be clear. To assign an event handler to a document element using JavaScript, simply set the event handler property to the desired function. For example, consider the following HTML form:

```
<form name="f1">
<input name="b1" type="button" value="Press Me">
</form>
```

The button in this form can be referred to as document.f1.b1, which means that an event handler can be assigned with a line of JavaScript like this one:

```
document.f1.b1.onclick=function( ) { alert('Thanks!'); };
```

An event handler can also be assigned like this:

```
function plead( ) { window.status = "Please Press Me!"; }
document.f1.b1.onmouseover = plead;
```

Pay attention to that last line: there are no parentheses after the name of the function.

12.1 Event Handler Return Values

In many cases, an event takes its return value to indicate the action of the event. For example, if the onsubmit event handler of a Form object to perform form validation and discover that the user has not filled in all the fields, then return false from the handler to prevent the form from actually being submitted. This ensure that a form is not submitted with an empty text field like this:

```
<form action="search.cgi"
   onsubmit="if (this.elements[0].value.length<= 0) return false;">
<input type="text">
</form>
```

Generally, if the web browser performs some kind of default action in response to an event, then return false to prevent the browser from performing that action.

Exercise

Very Short Questions:

Q1. Why JavaScript is known as scripting language?
Q2. Why JavaScript is a client side language?
Q3. Why JavaScript is not a secure language?
Q4. Is there any relation between Java and JavaScript language?
Q5. Write down the name of interpreter or compiler.
Q6. Explain JavaScript comments.

Q7. How break and continue keywords are works?
Q8. What do you mean by page redirection?
Q9. What does DOM stands for?
Q10. What is event in JavaScript?

Short Questions:

Q1. Explain the feature of JavaScript language. How JavaScript is useful in web development?
Q2. What is difference between java and JavaScript?
Q3. Explain the position, where to put JavaScript code.
Q4. Write a note JavaScript variables.
Q5. Explain the controlling of JavaScript programming flow.
Q6. What is function? Write down different type of functions with example. Also explain three in-built functions.
Q7. What do you mean by Event Handling in JavaScript? Explain type Events with example.

Long Questions

Q1. Explain JavaScript syntax with proper code and detail explanation.
Q2. Describe JavaScript operators.
Q3. What do you mean by iteration? What are entry and exit control loop?
Q4 Explain for, while and do-while loop with example.
Q5. What is JavaScript object? Write note on following in-built JavaScript object:

a. Document Object
b. Window object
c. String object
d. Array Object
e. Date Object
f. Math Object

■■■

Unit-IV

Chapter 8

Security with Form Validation and Cookie

1. JavaScript Security

JavaScript is designed as an open scripting language. It is not intended to replace proper security measures, and should never be used in place of proper encryption.

JavaScript has its own security model, but this is not designed to protect the Web site owner or the data passed between the browser and the server. The security model is designed to protect the user from malicious Web sites, and as a result, it enforces strict limits on what the page author is allowed to do. They may have control over their own page inside the browser, but that is where their abilities end.

JavaScripts cannot read or write files on users' computers, though they can cause the browser to load remote pages and resources like scripts or images, which the browser may choose to cache locally. They cannot create files on the server. The only thing they can store on the user's computer are cookies. Cookies are discussed later in this chapter.

JavaScript cannot access the cookies or variables from other sites.

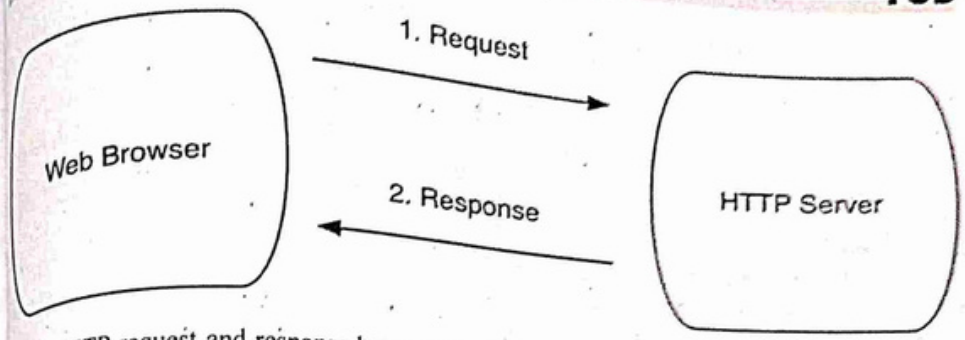
Most people who want to know about security with JavaScript are interested in producing password protected pages or sending encrypted data to or from the user's computer.

2. Client-Server basics

The web is based on the client-server architectural pattern. The client is played by a web browser with limited resources and technologies. Its usual main responsibilities are user interface and interaction as well as data validation.

The server role is fulfilled by a program implemented in a wide variety of technologies, with a controlled set of resources. Its usual responsibilities are to serve pages to web-browsers, enforce business rules, and persist and validate data.

The client and server processes communicate though the HTTP (Hyper Text Transfer protocol), the client makes HTTP requests and, for each, the server answers with an HTTPresponse, as illustrated. When the response arrives, the web browser normally discards the current content and loads the new one, causing a full refresh of the user interface. Since HTTP is a stateless protocol - requests are independent from each other. , as it is state-less, which means that once the server has sent a page to a browser requesting it, it doesn't remember a thing about it. So if you come to the same web page a second, third, hundredth time, the server considers it first time.



An HTTP request and response between a web browser and a web Server.

2.1 Stateless Server

Stateless server are also use full as they increase the speed between server and client communication, and for maintain previous request information on server, there are several mechanisms which also maintain state between a client and a server (e.g., cookies.)

Feature of Stateless Server:-

1). When there is stateless protocol, between a server and the client, the server does not remember anything about client and previous request. It treats any message from a client as the client's first message and responds with the same effects every time.

2). A stateless server does not keeps state between connections.

3). A stateless system can be seen as a box ,where at any point in time the value of the output(s) depends only on the value of the input(s) after a certain processing time.

4). A stateless protocol does not require the server to save session information or status about each communications partner for the duration of multiple requests.

5). Stateless protocol does not maintain the state, cannot maintain the persistence (data base),once sutdown ,we cannot see that data.

Example = UDP(User Datagram Protocol) , HTTP, NFS.

2.2 Statefull Server

Those web server which stores information of client requests and identify client requests.

Feature of Statefull Server:-

1). Stateful protocol means the server remembers what a client has done before.

2). A stateful server keeps state between connections. When you send a request to a stateful server, it tracks what information you request. When you send another request, that request operates on the state from the previous request.

4). A protocol which requires the keeping of internal state is known as a stateful protocol.

5). Statefull Protocol maintain the state, but cannot maintain the persistance, once we shutdown the system the values stored in local hard disk.

Originally there was no way to identify a user on the web that was viewing a website with a user that had previously visited, unless they had an account on the website and logged in each time.

It was possible to identify a user within a visit, without them being logged-in, by storing additional information in URLs.

As computers running web browsers become more and more powerful, there is a tendency to move responsibilities from the server. These new responsibilities are mostly implemented in JavaScript and JavaScript-based languages.

3. Cookie

Cookies were originally invented by Netscape to give 'memory' to web servers and browsers. A cookie is a string that is exchanged between the client and the server. Cookies are small information of client which are stored by server machine on client machine. when request is send to server, then server save cookies on client machine with response object. That information (cookies) is then sent to server with request object. and maintain state between particular client and server.

Cookies are data, stored in small text files, on your computer. A cookie is a named piece of data, created and used by a certain website for a certain viewing user, and sent from the user's web browser to the web server each time a page is viewed.

Cookies are normally stored on the client machines to let web application state survive after the client process is stopped.

Cookies are key-value pairs associated with the equals character '=' and separated from each other with a semi colon (e.g.... "name=savi';expires='date' "). The expiration date tells the browser when to delete it.

A cookie can contain other properties like domain and path, that tells the browser in which requests it should be exchanged. which by default are the current domain and path. The cookie (as mentioned before) can only store up to 4000 characters of data. This is enough to store lots of information about a user.

Cookies can be changed from JavaScript code accessing the cookie property of the current document object. Cookies can contain sensible data (e.g., emails and passwords) and must be studied for a security analysis.

Disadvantage of Cookie

One major security problem with cookies, is that they can easily be read by anyone using the computer. They are just a simple text file, so you should not under any circumstances store passwords in cookies.

Other problem is that if user set the browser setting as disable JavaScript then no information is set on your browser and no information is send to server with request object.

3.1 Use of Cookie

- Many portals and search engines use them to provide customized pages and give results to their users, allowing features as 'My Yahoo' etc.
- Many websites use cookies to log their users in automatically. By storing a few of user's information they can automatically authenticate the user's details and use them to save the user time when they log in.

- Visitor tracking is also done by cookie. By assigning the visitor a cookie, they will not be counted more than once, so accurate unique visitor track can be obtained. Also, if a user has a unique cookie the system can 'follow' them through a website, showing the webmaster exactly where the visitor has been.

3.2 Create cookie

To create the following code is used:

- document.cookie="username=Savita Singh";

Here cookie name is 'username'.

And cookie value is 'Savita Singh'.

- Cookie is also created like following:

document.cookie = cookie_name + "=" + cookie_value + "; " + expires_date;

document object is discussed earlier. cookies is also a in-built and global object of JavaScript. which stores the information related to cookie.

Here expires_date is time period after that cookie will destroyed.

Example:

```
function setCookie(cookie_name, cookie_value, exdays)
{
 var d = new Date();
 d.setTime(d.getTime() + (exdays*24*60*60*1000));
 var expires = "expires="+d.toUTCString();
 document.cookie = cookie_name + "=" + cookie_value + "; " + expires;
}
```

Here getTime() gives current time of system.

3.3 Get Cookie

It is possible to create a function that returns the value of a specified cookie:

Example:

```
function getCookie(cookie_name)
{
 var name = cookie_name + "=";
 var all_cookie = document.cookie.split(';');
 for(var i=0; i< all_cookie.length; i++) {
   var c = all_cookie [i];
   while (c.charAt(0)==' ') c = c.substring(1);
   if (c.indexOf(name) == 0) return c.substring(name.length,c.length);
 }
 return "";
}
```

Here we have to pass cookie name to function and we get all cookies values which are stored on computer.

split(;) function splits the string where semicolons are present.

document.cookie.split(;) gives cookies values array which is stored in var all_cookie.

3.4 Delete Cookie

To remove cookie there are several options. If we assign any expiry date then after that date cookie is automatically delete. Otherwise we can delete cookie by coding. Best way to delete cookie is following:

document.cookie("cookie_name".null);

Another way is: document.cookie = name+"="+value+"-1";

4. Interactive Forms and JavaScript - Form Validation

Interactive forms are created by applying validation on Form. Form validation used to occur at the server, after the client had entered all necessary data and then pressed the Submit button. If some of the data that had been entered by the client had been in the wrong form or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process and over burdening server.

Form validation is the process of checking that a form has been filled in correctly before it is processed. 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.

A user input validation function is stops falsified user information from being submitted. The validation function is easy to add to any Web form by creating a list of form objects, and registering the function as the onsubmit() event handler to the form. The programming logic allows for relations to be expressed between associated form fields when performing user input validation. For added flexibility, the validation process allows the form creator to permit specific data entry fields to contain unverified data at the time the form is successfully validated and submitted.

JavaScript, provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- Basic Validation - First of all, the form must be checked to make sure data was entered into each form field that required it. This would need just loop through each field in the form and check for data.
- Data Format Validation - Secondly, the data that is entered must be checked for correct form and value. This would need to put more logic to test correctness of data.

4.1 Basic Form Validation:

First we will show how to do a basic form validation. In the above form we are calling validate() function to validate data when onsubmit event is occurring. Following is the implementation of this validate() function:


```
function validate()
{
if( document.myForm.Name.value == "" )
{
alert( "Please provide your name!" );
document.myForm.Name.focus() ;
return false;
}
if( document.myForm.EMail.value == "" )
{
alert( "Please provide your Email!" );
document.myForm.EMail.focus() ;
return false;
}
```

4.2 Data Format Validation:

Now we will see how we can validate entered form data before submitting it to the web server. This example shows how to validate an entered email address which means email address must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign:

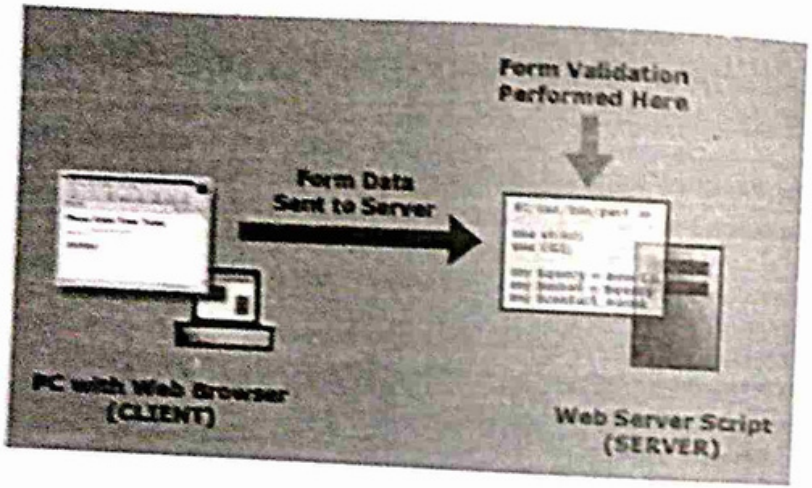
```
function validateEmail()
{
var emailID = document.myForm.EMail.value;
atpos = emailID.indexOf("@");
dotpos = emailID.lastIndexOf(".");
if (atpos < 1 || ( dotpos - atpos < 2 ))
{
alert("Please enter correct email ID")
document.myForm.EMail.focus() ;
return false;
}
return( true );
}
```

When you create forms, providing form validation is useful to ensure that your customers enter valid and complete data.

There are two main methods for validating forms: server-side (using CGI scripts, ASP, J2EE, php etc), and client-side (usually done using JavaScript).

4.3 SERVER-SIDE VALIDATION

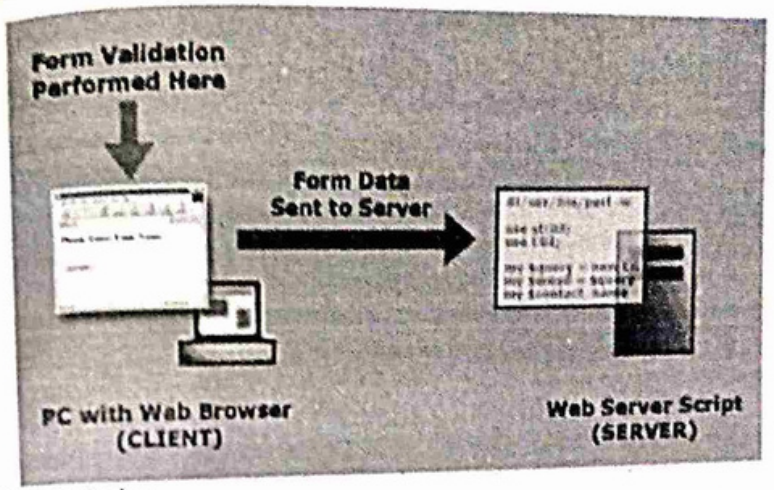
Server-side validation is more secure but often more tricky to code.Server-side validation incorporates code validation into a form handler. Server-side -validation means validation apply on server-side like ASP or ASP.NET application on the server that provides the functionality that your form needs for processing after a customer has submitted it. Because the code is stored on the server, server-side validation requires a roundtrip to the server. Sending data back and forth between the client and server is time consuming due to traffic load on server. However for some form functions, server-side validation is necessary. For example, if you have a form that populates a drop-down list based on the value of another form field, server-side processing may be needed to pull data from a database and populate the drop-down list.



Server-side Validation

4.4 CLIENT-SIDE VALIDATION

Client-side (JavaScript) validation is easier to do and quicker too as the browser doesn't have to connect to the server to validate the form, so the user finds out instantly if they've missed out that required field etc.Client-side validation provides validation within the browser(client-side) on client computers through JavaScript. Because the code is stored within the page or within a linked file, it is downloaded into the browser when a user accesses the page and therefore, doesn't require a roundtrip to the server. For this reason, client form validation can be faster than server-side validation. However client-side validation may not work in all instances. A user may have a browser that doesn't support client-side scripting or may have scripting disabled in the browser. Knowing the limits of client-side scripting helps you decide whether to use client-side form validation



Client-side Validation

JavaScript can use to validate important types of form data, including names, addresses, URLs, email addresses, phone numbers, expiration dates etc.

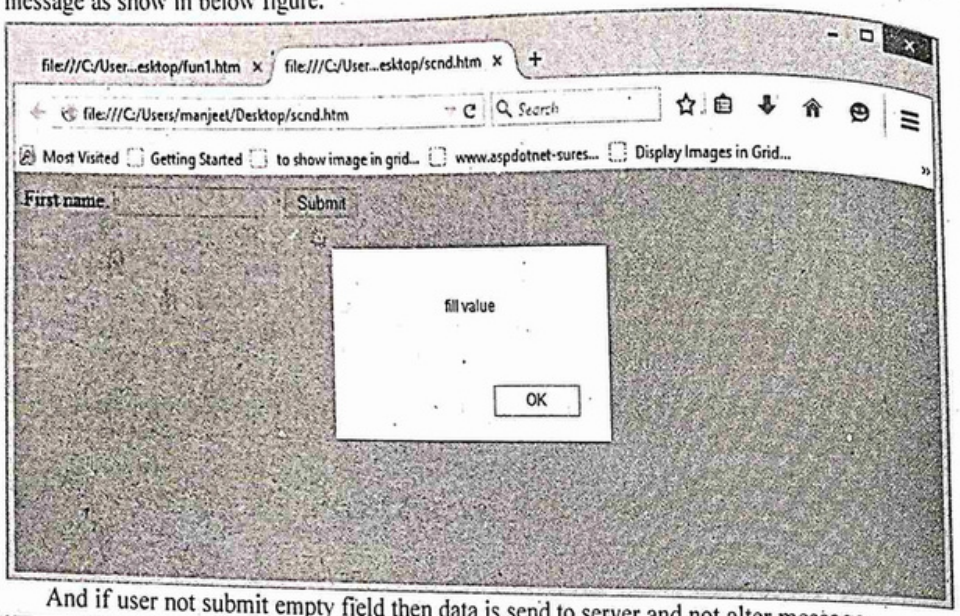
4.5 VALIDATION ON TEXTBOX:-

- Validation for Empty textbox

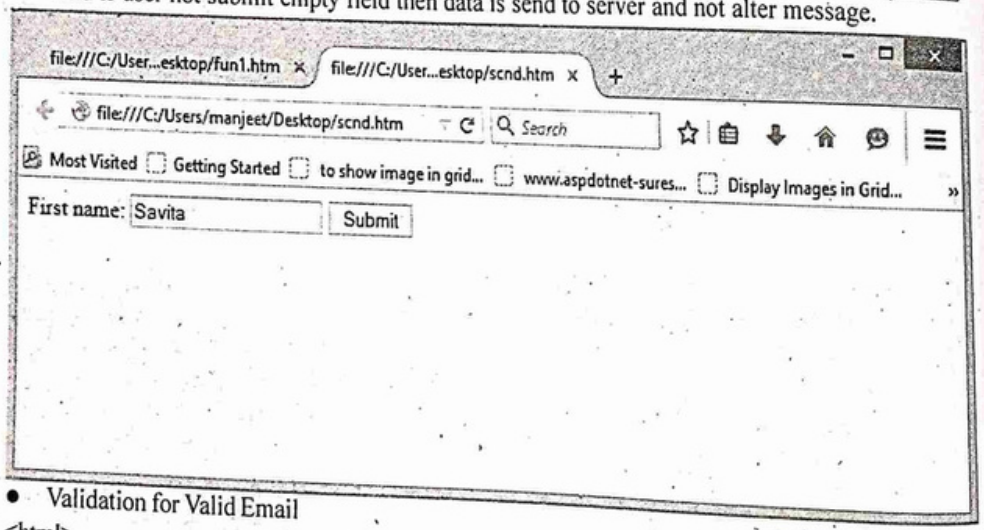
```
<html>
<head>
<script>
function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if(x==null || x=="")
 {
 alert("fill value");
 return false;
 }
}
</script>
</head>
<body>
<form name="myForm" onsubmit="validateForm()" method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
</body>
```

```
</html>
```

If user makes the Field empty, and submit the form having required field empty then there is a alter message as show in below figure.



And if user not submit empty field then data is send to server and not alter message.



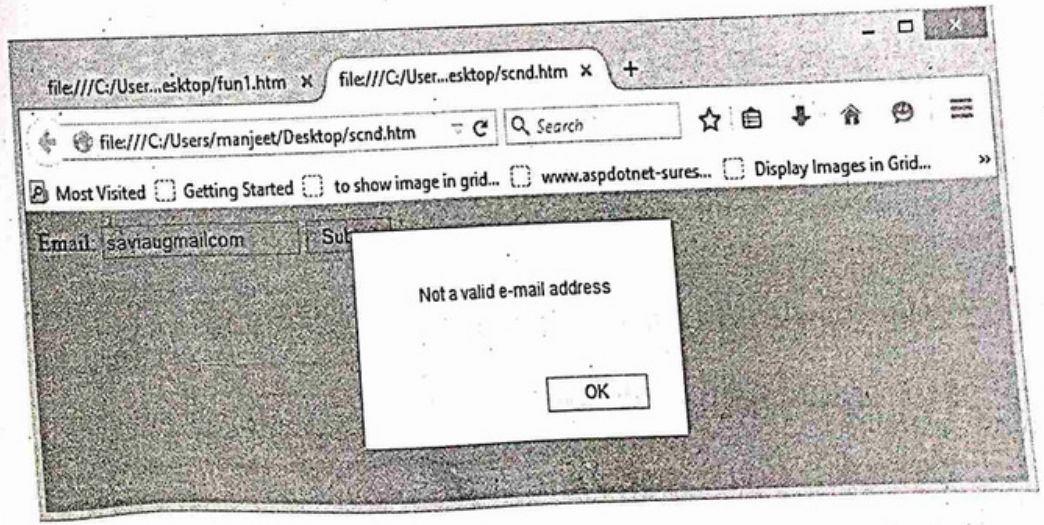
- Validation for Valid Email

```
<html>
<head>
<script>
function validateForm()
```



```
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
 {
alert("Not a valid e-mail address");
return false;
 }
document.write(x.length);
}
</script>
</head>
<body>
<form name="myForm"  onsubmit="validateForm();" method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

If user Enter wrong email address like email address without @ or dot(.) then there is a alret message as shown in figure.



4.6 Validating radio buttons

Validations are apply on radio button to check if any radio button is checked or not.

Example:

```
if ( ( document.form.gender[0].checked == false )  &&
(  document.form.gender[1].checked == false ) )
{
    alert ( "Please choose your Gender: Male or Female" );
}
```

Here gender is name of radio button.

This code checks to see whether either of the radio buttons ("Male" or "Female") have been clicked. If neither have been clicked (checked == false), the user is alerted and valid is set to false.

4.7 Validating drop-down lists

Drop-down list is checked to see if the user has selected an option. In the form, we named the first option in the drop-down list, "Please Select an Option". Our JavaScript can then check which option was selected when the user submitted the form. If the first option is selected, we know the user has not selected a "real" option and can alert them.

Here age is name of drop-down list.

Example:

```
if ( document.form.age.selectedIndex == 0 )
{
alert ( "Please select your Age." );
}
```

Note that the values for selectedIndex start at zero (for the first option).

4.8 Validating checkboxes

The checkbox validation is applies weather the check box is checked or not.

Example:

```
if ( document.contact_form.terms.checked == false )
{       alert ( "Please check the Terms & Conditions box." );
}
```

5. Controlling Frame in JavaScript

A page is divided into frames using the FRAMESET tag as discussed earlier. The tag is used in the top-level document defining a window containing frames and is used to specify how to divide the document window.

Because windows divided into frames are created from multiple HTML files, it is important to keep the hierarchical relationship of documents. In a document window divided into frames, the top-level document is the HTML document that defines the frames and files that will load into those frames. Inside a FRAMESET container, the FRAME tag is used to specify which files should be

displayed in each frame. The URLs of the files-which can be relative or absolute-should be specified using the SRC attribute.

JavaScript provides the frames property of the window object for working with different frames from a script.The frames property is an array of objects with an entry for each child frame in a parent frameset. The number of frames is provided by the length property.

For instance, in a given window or frameset with two frames, you could reference the frames as parent.frames[0] and parent.frames[1]. The index of the last frame could be parent.frames.length.

By using the frames array, we can access the functions and variables in another frame, as well as objects, such as forms and links, contained in another frame. This is useful when building an application that spans multiple frames but that also must be able to communicate between the frames.

Each frame has a different document, location, and history object associated with it. This is because each frame contains a separate HTML document and has a separate history list.

For example, if there are two frames, you can create a form in the first frame to provide the user with a field to enter an expression. Then you could display the results in a form in the other frame.

This cross-frame communication is achieved by referencing the document object's forms[] array in the second frame with parent.frames[1].document.forms[0].

```
<HTML>
<HEAD>
<TITLE>Listing</TITLE>
</HEAD>
<FRAMESET COLS="50%,*">
  <FRAME SRC="input.html">
  <FRAME SRC="output.html">
</FRAMESET>
</HTML>
<!-- HTML FOR INPUT FRAME (this is a separate file called input.html-->
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
 HIDE FROM OTHER BROWSERS
function update(field)
{
  var result = field.value;
  var output = "" + result + " = " + eval(result);

  parent.frames[1].document.forms[0].result.value = output;
}
```

```
// STOP HIDING FROM OTHER BROWSERS
</SCRIPT>
</HEAD>
<BODY>
<FORM METHOD=POST>
<INPUT TYPE=text NAME="input" onChange="update(this);">
</FORM>
</BODY>
</HTML>
<!-- HTML FOR OUTPUT FRAME (this is a separate file called output.html)-->
<HTML>
<BODY>
<FORM METHOD=POST>
<TEXTAREA NAME=result ROWS=2 COLS=20
WRAP=SOFT></TEXTAREA>
</FORM>
</BODY>
</HTML>
```

In addition to specifying frames using the frames array, if you name the frames, you can specify certain frames using the form parent.framename. In the example you just saw, if you name the frames input and output, you could rewrite the update() function:

In any given window or frame, one of the primary objects is the document object. The document object provides the properties and methods to work with numerous aspects of the current document, including information about anchors, forms, links, the title, the current location and URL, and the current colors. We already discussed document object earlier.

The document object is defined when the BODY tag is evaluated in an HTML page and the object remains in existence as long as the page is loaded. Because many of the properties of the document object are reflections of attributes of the BODY tag.

6. INTRODUCTION TO JQUERY?

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that requires many lines of JavaScript code to accomplish, and wraps it into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax (discussed in later) much simpler with an easy-to-use API that works across a browsers. With a combination of versatility and

extensibility, jQuery has changed the way that millions of people write JavaScript.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

To download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

To use jQuery from Google, use one of the following:

<head><script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script></head>

6.1 JQUERY SYNTAX

The jQuery syntax is tailor made for selecting HTML elements and perform some action on the element(s).

Basic syntax is: $(selector).action()

- A $ sign to define/access jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

6.2 JQUERY SELECTORS

- jQuery selectors are one of the most important parts of the jQuery library.
- jQuery selectors allow you to select and manipulate HTML element(s).
- With jQuery selectors you can find elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
- All type of selectors in jQuery, start with the dollar sign and parentheses: $().

The element Selector: The jQuery element selector selects elements based on their tag names.

Example:

```
$(document).ready(function(){ $("button").click(function(){ $("p").hide(); }); });
```

"p" is for <p>tag

The #id Selector: The jQuery #id selector uses the id attribute of an HTML tag to find the

specific element. An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the element:

$("#test")

The .class Selector: The jQuery class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

$(".test")

6.3 jQuery as Event Methods

jQuery is tailor-made to handle HTML/DOM events. Event handlers are methods that are called when "something happens" in HTML. The term "triggered (or "fired") by an event" is often used. It is common to put jQuery code into event handler methods in the <head> section. In the example below, a function is called when the click event for the button is triggered:

Example

```
<html>
<head>
<script src="jquery.js">
</script>
<script>
$(document).ready(function(){ $("button").click(function(){ $("p").hide(); }); });
</script>
</head>
<body><h2>This is a heading</h2><p>This is a paragraph.</p><p>This is another paragraph.</p><button>Click me</button>
</body>
</html>
```

7. jQuery Effects

7.1 Hide and Show Methods

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

Example

```
$("#hide").click(function(){ $("p").hide(); }); $("#show").click(function(){ $("p").show(); });
```

Syntax:

$(selector).hide(speed,callback); $(selector).show(speed,callback);

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after hide (or show) completes.

7.2 jQuery Fading Methods

jQuery can fade elements in and out of visibility.

jQuery has the following fade methods:

- fadeIn() : used to fade in a hidden element.
- fadeOut() : used to fade out a visible element.
- fadeToggle() : toggles between the fadeIn() and fadeOut() methods.
- fadeTo() : allows fading to a given opacity (value between 0 and 1).

Syntax:

$(selector).fadeIn(speed,callback);

$(selector).fadeOut(speed,callback);

$(selector).fadeToggle(speed,callback);

$(selector).fadeTo(speed,opacity,callback);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the fading completes.

7.3 jQuery animate() Method

The jQuery animate() method is used to create any custom animations like using CSS properties

.

Syntax:

$(selector).animate({params},speed,callback);

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is the name of a function to be executed after the animation completes.

Example:

```
$("button").click(function()
{
$("div").animate(
{ left:'250px', opacity:'0.5', height:'150px', width:'150px'
});
 });
```

7.4 jQuery -Get and Set CSS Classes

jQuery has several methods for CSS manipulation. Look at the following methods:

- addClass() - Adds one or more classes to the selected elements

Example :

```
$("button").click(function()
{ $("#div1").addClass("important blue");
});
```

Here important and blue are class of CSS. And #div is id of HTML tag.

- removeClass() - Removes one or more classes from the selected elements

Example

```
$("button").click(function()
{ $("h1.h2.p").removeClass("blue"); });
```

- css() - Sets or returns the style attribute.

To return the value of a specified CSS property, use the following syntax:

```
css("propertyname");
```

Example:

```
$("p").css("background-color");
```

To set a specified CSS property, use the following syntax:

```
css("propertyname","value");
```

Example

```
$("p").css("background-color","yellow");
```

8. Client-side JavaScript custom

Client-side scripting mean those scripting language which are execute on client machine, like JavaScript, can be embedded into the page on the client's browser. This script will allow the client's browser to decrease the burden of web server when running a web application. Client-side scripting is source code that is executed on the client's browser instead of the web-server, and allows for the creation of faster and more responsive web applications.

Window applications are those applications which are access only on those machines on which these application are installed. Example Microsoft office, which is installed on a client's workstation, other than calls to the database server to save and retrieve data, all the source code and assemblies are run locally. In fact, it is a requirement that a windows application be installed on the client's workstation.

Web applications, on the other hand, don't need to be installed on individualworkstations. Web applications are those application which are installed on any machine and can access through word by using internet. Example gmail application. An internet connection and a browser is need. However, the majority of the processing will not take place on the client's browser, but on a web server at some other location.

8.1 The Need for Client-Side Scripting

As web applications become more and more powerful, it is expected that they works and act just like a windows application. The basic architecture for a web application is that most of the source code and assemblies reside and are processed on a web server. The sole task of a web server is to accept incoming HTTP requests and to return the requested resource in an HTTP response. There is never a continuous live connection between the client's browser and the web server. Web pages will always be in the form of HTML.

PostBack is the name given to the process of submitting an page to the server for processing. Every time a PostBack happens, the HTML page is sent to the web server. The server loads the page, processes events and renders the new HTML back to the client. On a PostBack, the entire HTML is refreshed.

For large and complex web-applications that store a large amount of data, this can be very time consuming for the client. This will become a huge problem for a web server with limited resources like memory and bandwidth.

Web page to persist changes to the state of a Web Form across postbacks. View state allows the state of objects to be stored in a hidden field on the page. View state is transported to the client and back to the server, and is not stored on the server or any other external source. View state is used to retain the state of server-side objects between postbacks and will become very large for large applications and pages.

Client-side scripting (embedded scripts) is code that exists inside the client's HTML page. This code will be processed on the client machine and the HTML page will NOT perform a PostBack to the web-server. Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser.

The two main benefits of client-side scripting are:

1. The user's actions will result in an immediate response because they don't require a trip to the server.
2. Fewer resources are used and needed on the web-server.

9. Introduction to AJAX

Ajax or Asynchronous JavaScript and XML enables the programmer to execute a server-side script without refreshing the page.

An AJAX-enabled Web page includes a JavaScript library on the client side that guards calls to the Web server. It does this when it is possible to send a request and get a response for just part of the page and when using script, the client library updates that part of the page without updating the entire page.

An entire page is a lot of code to send down to the browser to process each and every time. By processing just part of the page, which makes the page more responsive. The amount of code required to update just a portion of a page is less than AJAX makes it fast.

This library can make asynchronous page requests and update just the portion of the page that needs updating.

EXERCISE

VERY SHORT QUESTIONS

1. Difference between Web application and Window application.
2. What do by HTTP?
3. What is full form of AJAX?
4. Write syntax if external script.
5. Explain jQuery in one line.

SHORT QUESTIONS

1. What is client side and server side scripting language?
2. What do you mean by client-side validation and server-side validation?
3. Explain need of client side scripting language.
4. Explain about jQuery selectors.
5. What is Cookie? Write down advantage and disadvantage of cookies.
6. What do you mean by stateless and statefull servers?
7. How frames are controlled by JavaScript? Explain with example.

LONG QUESTIONS

1. What is form validation? Create an interactive form applying validation on some html controls using JavaScript.
2. Explain following effects of jQuery:
   a. Hide and Show effect
   b. Fading effect
   c. Animation effect
3. Expalin cookies in brief. Explain how to create, get and delete the cookie with proper JavaScript code.
4. Write JavaScript code for email validation on textbox and also apply some validation radio button.