CLIENT-SIDE SCRIPTING USING JAVASCRIPT

10

Objectives

After completing this Chapter, the student will be able to:

- *define JavaScript,*
- *explain the basics of JavaScript,*
- *embed JavaScript code into a HTML document,*
- *compare declare and use of variables,*
- *use variables and literals in expressions,*
- *describe different data types and values,*
- *appreciate the use of branching and loop statements,*
- *perform iteration with **for** loop,*
- *distinguish between **while** and **do...while** loops,*
- *break and continue the loops,*
- *discuss some object manipulation statements and*
- *consider defining and calling of functions.*

Content creation should not be recondite. It should not be this bizarre* arcana* that only experts and gold-plated computer science gurus can do.*

Brendan Eich
Creator of JavaScript

**recondite- complex, bizarre- strange/ unusual , arcana- deep secret*

Introduction

In Chapter 9 we learnt how to create web pages using HTML. The HTML documents consist of many tags, which tell the browser, how to display the text or graphics. Also, we have learnt to create static web pages; which are not interactive. To make the HTML document interactive and dynamic, there is a need to include some special codes (scripts) along with HTML. One such scripting language is JavaScript.

In this Chapter, we will learn about basics of JavaScript and how to add dynamic effects in web pages using expression, operators, popup boxes, conversion functions, conditional statements, looping statements, object manipulation statements and JavaScript functions.

10.1 ABOUT JAVASCRIPT

JavaScript was created by **Brendan Eich** and it came into existence in September 1995, when Netscape 2.0 (a web browser) was released.

JavaScript was designed with a purpose to make web pages dynamic and more interactive. JavaScript is one of the popular scripting languages having following features :

(a) It can be used for client and server applications.
(b) It is platform independent which means it can run on any operating systems (i.e. Linux, Microsoft Windows, Mac OS X etc.).
(c) JavaScript codes are needed to be embedded or referenced into HTML documents then only it can run on a web browser.
(d) It is an interpreted language.
(e) It is a case-sensitive language and its keywords are in lowercase only.

10.1.1 Difference Between Java and JavaScript

Some people think that Java and JavaScript are same but both are two completely different languages. Java is a general-purpose object-oriented programming language from Sun Microsystems where as JavaScript is an object-based scripting language. Script refers to short programming statements to perform a task.

10.1.2 Versions of JavaScript

Some versions of JavaScript and web browsers are :

JavaScript Version	*Web Browsers*
JavaScript 1.0	Navigator 2.0, Internet Explorer 3.0
JavaScript 1.3	Navigator 4.06-4.7x, Internet Explorer 4.0
JavaScript 1.5	Navigator 6.0, Mozilla, Internet Explorer 5.5 onwards
JavaScript 1.6	Mozilla Firefox 1.5
JavaScript 1.7	Mozilla Firefox 2.0
JavaScript 1.8	Mozilla Firefox 3.0

Microsoft has released several versions of JavaScript, currently JavaScript version 5.7 is used with Internet Explorer 7.0.

10.2 CLIENT SERVER MODEL

Before stepping ahead, we should know about the Node, Client and Server.

10.2.1 Node

Node is a component or terminal connected to a network. The components like laptops, PDAs, Internet enabled mobiles etc., can be considered as node in a computer network.

10.2.2 Client

It is a node computer that establishes connection with the server, collects data from the user, sends it to the server, receives information from the server and presents it to the user.

10.2.3 Server

In the context of client-server model, server is the counter part of client. It is a computer that serves queries from the client. The programs which respond to the request of clients are known as server applications. The computer designed to run server application is known as server machine. Web server, database server and mail server are some examples of servers.

The Client Server Model is an architecture of computer network where client and server interact by means of a network (*Figure 10.1*). Client gathers data from the user as input and sends request to the server. Server processes the request and sends the requested information to the client. Railway reservation system, online banking and online gaming are examples of client-server model.

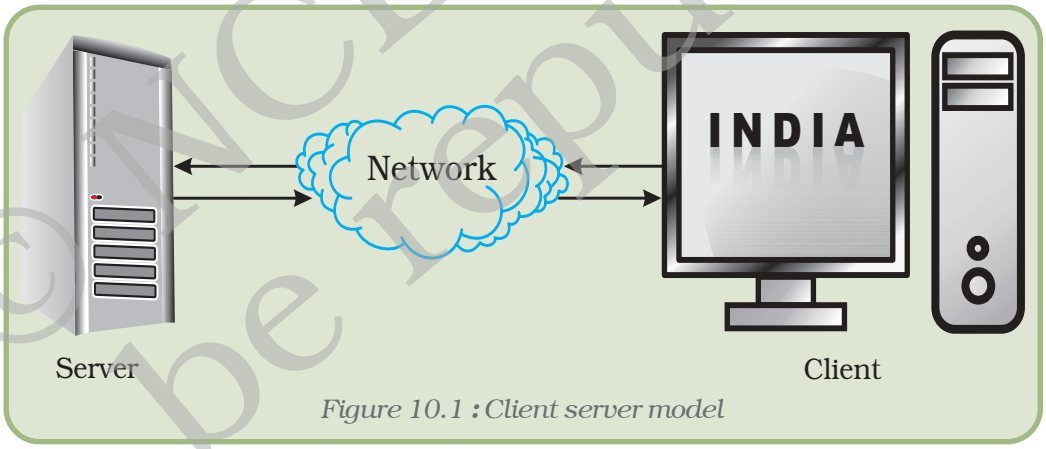


Figure 10.1 : Client server model

10.2.4 Client–Side JavaScript

Client-side JavaScript refers to JavaScript code that gets executed by the web browser on the client machine. JavaScript statements can be embedded in a HTML document that can respond to events such as mouse clicks, form input, and page navigation etc. without any network connection.

10.2.5 Server–Side JavaScript

Server-side JavaScript is also known as "LiveWire". Like client-side JavaScript, server-side JavaScript is also embedded within a HTML document. When a query is made by the client, web server executes the script after interpreting it.

10.3 GETTING STARTED WITH JAVASCRIPT

JavaScript is not a full-fledged language and it needs to be embedded within a HTML document. Otherwise, or to specify an external file that contains the JavaScript code we can use word 'script' or 'program' for the JavaScript code.

The most common way to set off a script is to use the HTML <script> and </script> tags in HTML document. We can place our JavaScript code in either the HEAD or BODY section of a HTML document.

The Syntax (General format) is

<SCRIPT [Attributes = ["Value"] ....]> Indicates starting of JavaScript Code

... JavaScript statement(s);

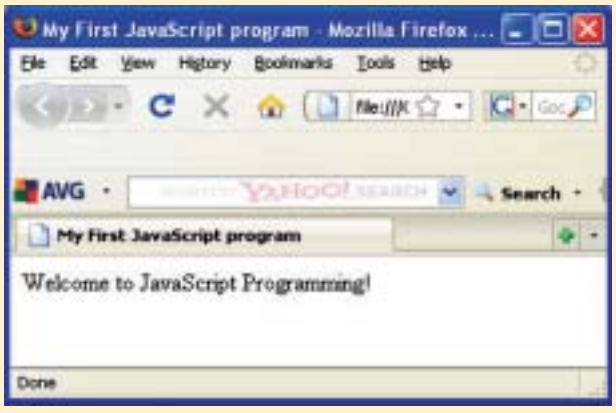
</SCRIPT> Indicates ending of JavaScript Code

The following table contains Script attributes, values and their description.

Attribute	*Value*	*Description*
Type	text/javascript text/ecmascript text/vbscript	the type of script
Language	Javascript vbscript	name of scripting language
Src	URL	a URL to a file that contains the script

Program 10.1 : First simple JavaScript program using *document.write()*.

```
<HTML>
<HEAD>
<TITLE>My First JavaScript program
</TITLE>
</HEAD>
<BODY>
<SCRIPT type="text/javascript">
    document.write("Welcome to ↵
JavaScript Programming!");
</SCRIPT>
</BODY>
</HTML>
```



Note : You notice that the code does not fit into single line. So, we used ↵ to indicate that the code is continued and while inputting it you need not to type it.

To run the above program, type program code given above using any text editor like Notepad, Wordpad and save it as "<file name>.htm" (e.g. abc.htm). Open this file by using any browser application (i.e. Internet Explorer, Mozilla Firefox, Opera etc.).

Tools needed for Writing and Running JavaScript code :

Following tools are needed for working with JavaScript code:

a) **Text Editors:** We can choose any text editor or word processor (i.e. Notepad, Wordpad etc.).
b) **Browser:** Browser interprets JavaScript code and shows the output on browser's document window.

10.3.1 Placing The JavaScript Code

There are two ways to place the JavaScript code :

1. **Embedded/Inline JavaScript :** JavaScript code can be placed either in the HEAD or in the BODY section of a HTML document.
   a. It is advised to place JavaScript code in HEAD section when it is required to be used more than once.
   b. If the JavaScript code is small in size and used only once, it is advisable to put it in the BODY section of the HTML document.
2. **External JavaScript :** In case, same JavaScript code needs to be used in multiple documents then it is the best approach to place JavaScript code in external files having extension as **" .js"**. To do so, we will use ***src*** attribute in <SCRIPT> tag to indicate the link for the source JavaScript file.

Example : Illustration of the use of external JavaScript code.

```
<HTML>
<HEAD>
<TITLE>Using External JavaScript</TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript" src="abc.js">
</SCRIPT>
<P> The actual JavaScript code exists in external file called
"abc.js". </P>
</BODY>
</HTML>
```

Actual JavaScript file "abc.js"

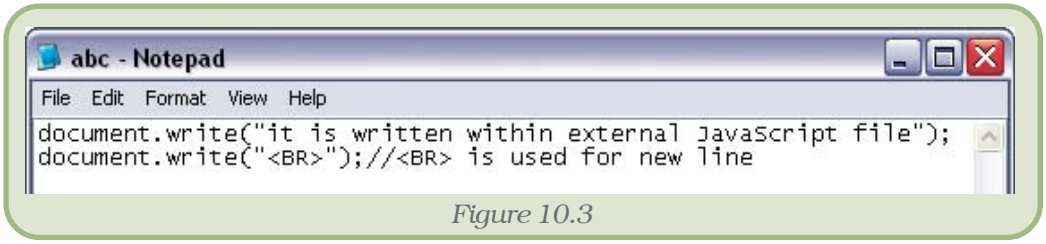


Figure 10.3

Output

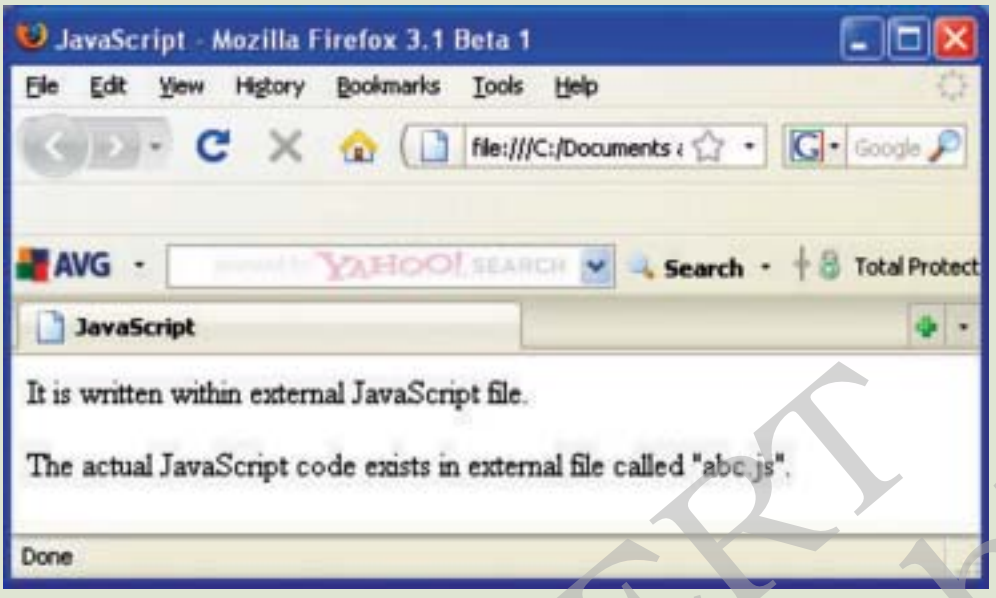


Figure 10.4

10.4 STATEMENTS IN JAVASCRIPT

Statements are the commands or instructions given to the JavaScript interpreter to take some actions as directed. A JavaScript interpreter resides within almost Internet browsers. A collection of statements to accomplish a job, is called as a script or program. The JavaScript statements look like as follow:

```
a = 100;                    // stores value 100 in variable a
b = 200;                    // stores value 200 in variable b
c = a + b;                  // stores the sum of a and b in
                            // variable c

document.write
("Sum of A and B : ");      // displays the string
document.write(c);          // displays the value of c
```

In JavaScript, semicolon(;) is used to end a statement but if two statements are written in separate lines then semicolon can be omitted. Some valid statements are :

(i) p=10
    q=20
(ii) x=12; y=25 // semicolon(;) separating two statements.

Some invalid statements are :

x=12 y=25 // statements within the same line not separated by semicolon (;)

10.4.1 Comments

Comments are the statements that are always ignored by the interpreter. They are used to give remarks to the statement making it more readable and understandable to other programmers. There are two types of comments :

- Single line comment using double-slash (//).
- Multiple lines comment using /* and */ .

For example :

```
// This is a single-line comment.
/* This is a multiple-line comment.
    It can be of any length.    */
```

10.4.2 Literals

Literals refer to the constant values, which are used directly in JavaScript code. For example:

```
a=10;
b=5.7;
document.write("Welcome");
```

In above statements *10*, *5.7*, *"Welcome"* are literals.

10.4.3 Identifiers

Identifiers refer to the name of variables, functions, arrays, etc. created by the programmer. It may be any sequence of characters in uppercase and lowercase letters including numbers or underscore and dollar sign. An identifier must not begin with a number and cannot have same name as any of the keywords of the JavaScript.

Some valid identifiers are :

```
RollNo
bus_fee
_vp
$amt
```

Some invalid identifiers are :

```
to day      // Space is not allowed
17nov       // must not begin with a number
%age        // no special character is allowed
```

10.4.4 Reserved Words or Keywords

Reserved words are used to give instructions to the JavaScript interpreter and every reserved word has a specific meaning. These cannot be used as identifiers in the program. This means, we cannot use reserved words as names for variables, arrays, objects, functions and so on. These words are also known as "Keywords". A list of reserved words in JavaScript is given in Appendix 10.1.

10.4.5 Variables

A variable is an identifier that can store values. These values can be changed during the execution of script. Once a value is stored in a variable it can be accessed using the variable name. Variable declaration is not compulsory, though it is a good practice to use variable declaration. Keyword **var** is used to declare a variable.

Syntax

var *var-name* [= *value*] [..., *var-name* [= *value*]]

Example

```
var name = "Sachin";          // Here 'name' is variable
document.write(name);         // Prints Sachin
```

A JavaScript variable can hold a value of any data type. For example:

```
i = 7;
document.write(i);      // prints 7
i = "seven";            // JavaScript allows to assign string values
document.write(i);      // prints seven
```

Some valid examples of variable declaration:

```
var cost;
var num, cust_no = 0;
var amount = 2000;
```

Naming Conventions

We should use meaningful name for a variable. A variable name must start with a letter, underscore (_), or dollar sign ($). The subsequent characters can be the digits (0-9). JavaScript is *case sensitive*, so the variable name my_school is not the same as My_School.

Some valid variable names

```
f_name
India123
```

_sumof

Some invalid variable names

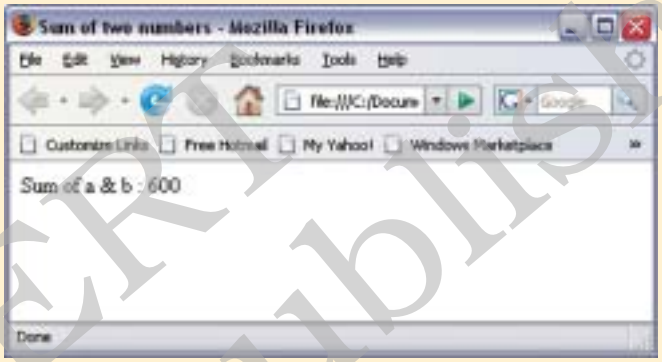
10_numbers - *must not begin with any number.*

rate% - *'%' is not a valid character.*

my name - Space is not allowed.

Program 10.2 : To find the sum of two numbers using ***var***.

```
<HTML>
<HEAD>
<TITLE>Sum of two numbers</TITLE>
</HEAD>
<BODY>
<SCRIPT type="text/javascript">
var a = 100;
var b = 500;
var c = a + b;
document.write ("Sum of a & b ↵
: " + c );
</SCRIPT>
</BODY>
</HTML>
```



10.5 DATA TYPES

JavaScript supports three basic data types – number, string, boolean and two composite data types – arrays and objects.

10.5.1 Number

The number variable holds any type of number, either an integer or a real number. Some examples of numbers are:

29, -43, 3.40, 3.4323

10.5.2 String

A string is a collection of letters, digits, punctuation characters, and so on. A string literal is enclosed within single quotes or double quotes (or "). Examples of string literals are:

'welcome', "7.86" , "wouldn't you exit now", ' country="India" '

JavaScript also allows us to use Escape Sequence within string literals. The escape sequence starts with a backslash (\), followed by another character. This backslash tells browser to represent a special action or character representation. For example, \" is an escape sequence that represents a double quote (").

Escape Sequence	Action/ Character Represented
\b	Backspace
\n	New line
\r	Carriage return
\t	Tab
\'	Single quote (')
\"	Double quote (")
\\	Backslash (\)

Example :

```
document.write ("Abhinav said, \"Earth doesn\'t revolve round ↵
the sun\". But teacher corrected him.");
```

Here, two types of escape characters are used \" and \' in this example.

Output

Abhinav said, "Earth doesn't revolve round the sun". But teacher corrected him.

10.5.3 Boolean Values

A *boolean* variable can store only two possible values either true or false. Internally it is stored as 1 for *true* and 0 for *false*. It is used to get the output of conditions, whether a condition results in true or false.

Example

```
x == 100;     // results true if x=100 otherwise false.
```

10.5.4 Arrays

An array is a collection of data values of same types having a common name. Each data element in array is referenced by its position in the array also called its index number. Individual array elements can be referenced by the array name followed by the pair of square brackets having its index number. The index number starts with zero in JavaScript i.e. the first element in JavaScript has it index value as 0, second has its index value as 1 and so on. An array can be declared in any of the following ways :

```
var a  = new a( );
```

```
var x = [ ];
var m = [2,4,"sun"];
```

An array is initialised with the specified values as its elements, and its length is set to the number of arguments specified.

Example This creates an array name games with three elements.

```
games = ["Hockey", "Cricket", "Football"];
```

We can also store different types of values in an array.

For example :

```
var arr = new Array();          // creation of an array
arr[0] ="JAVASCIPT";            // stores String literal at index 0
arr[1] = 49.5;                  // stores real number at index 1
arr[2] = true;                  // stores Boolean value
```

10.5.5 Null Value

JavaScript supports a special data type known as ***null*** that indicates "no value or blank". Note that ***null*** is not equal to 0.

Example

```
var distance = new object();
distance = null;
```

10.6 OBJECTS

JavaScript is an object based scripting language. It allows us to define our own objects and make our own variable types. It also offers a set of predefined objects. The tables, forms, buttons, images, or links on our web page are examples of objects. The values associated with object are properties and the actions that can perform on objects are methods or behaviour. Property associated to an object can be accessed as follows:

```
ObjectName.PropertyName
```

Now we will study, some of the predefined objects in JavaScript.

10.6.1 Document Object

The Document object is one of the parts of the Window object. It can be accessed through the **window.document** property. The ***document*** object represents a HTML document and it allows one to access all the elements in a HTML document. For example: title of current document can be accessed by **document.title** property.

Some of the common properties of document object are:

Properties	***Purposes***
Title	returns/ sets title of the current document.
bgColor	returns/ sets the background color of the current document.
fgColor	returns/ sets the text color of the current document.
linkColor	returns/ sets the color of hyperlinks in the document.
alinkColor	returns/ sets the color of active links in the document.
vlinkColor	returns/ sets the color of visited hyperlinks.
height	returns the height of the current document.
width	returns the width of the current document.
Forms	returns a list of the FORM elements within the current document.
Images	returns a list of the images in the current document.
URL	returns a string containing the URL of the current document.
Location	to load another URL in current document window.

Methods	***Purposes***
open()	Opens a document for writing.
write()	Writes string/data to a document.
writeln()	Writes string/data followed by a newline character to a document.
close()	Closes a document stream for writing.

Program 10.3: To illustrate the properties and methods of the document object.

```
<HTML>
<HEAD>
<TITLE>Document Properties</TITLE>
</HEAD>
<BODY>
<SCRIPT type="text/javascript">
document.fgColor = "green"; // sets text color
document.bgColor = "yellow";      // background color
document.title = "Chakde India";  // change title
document.linkColor = "navy";      // hyperlinks color
document.alinkColor = "red";      // active links
document.vlinkColor = "lime";     // visited hyperlinks
document.write("<BR>Do you want to play for India?");
document.write("<BR> <A href='yes.html'> Yes </A>");
document.writeln("<BR> <A href='no.html'>No</A><BR>");
document.write("<BR>Title of current document: " + document.title);
```

```
document.write("<BR>Height of current document: " + document.height);
document.write("<BR>Width of current document: " + document.width);
document.write("<BR> URL of current document is: " + document.URL);
//Use of document.links to list of all the hyperlinks
document.write("<BR><B>The List of Links in current document</B>");
var links = document.links;
for(var i = 0; i < links.length; i++)
{
document.write("<BR>"+ document.links[i]);
}
</SCRIPT>
</BODY>
</HTML>
```

10.6.2 Date Object

This object is used to set and manipulate date and time. JavaScript dates are stored as the number of milliseconds since midnight, January 1, 1970. This date is called the epoch. Dates before 1970 are represented by negative numbers. A date object can be created by using the *new* keyword with Date().

Syntax

newDate()

new Date(milliseconds)

new Date(dateString)

new Date(yr_num, mo_num, day_num

[, hr_num, min_num, sec_num, ms_num])

Parameters

Milliseconds	Milliseconds since 1 January 1970 00:00:00.
dateString	Date String. e.g. "October 5, 2007"
yr_num, mo_num,day_num	Year (e.g. 2007)Month (Value 0-11, 0 for January and 11 for December), Day (1-31)
hr_num, min_num,sec_num, ms_num	Values for Hour, Minutes, Second and milliseconds

Different examples of using a date()

```
today = new Date();
dob = new Date("October 5, 2007 12:50:00");
doj = new Date(2007,10,5);
bday = new Date(2007,10,5,12,50,0);
```

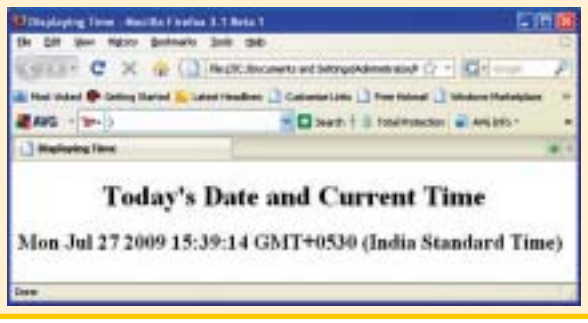
Methods to read date values

We can use the ***get*** methods to get values from a Date object. Here are some get methods that returns some value according to local time:

getDate( )	Returns the day of the month
getDay( )	Returns the day of the week
getFullYear( )	Returns the full year
getHours( )	Returns the hour
getMinutes( )	Returns the minutes
getMonth( )	Returns the month
getSeconds( )	Returns the seconds
getTime( )	Returns the numeric value corresponding to the time
getYear( )	Returns the year

Program 10.4 : A simple JavaScript program that displays today's date and current time.

```
<HTML>
<HEAD>
    <TITLE>Displaying Time</TITLE>
</HEAD>
<BODY>
    <CENTER>
          <H1>Today's Date and Current Time</H1>
    </CENTER>
    <SCRIPT type="text/javascript">
          var today = new Date();
          document.write("<H2>"); // JavaScript allows the use
          document.write(today);  // of HTML formatting tag
          document.write("</H2>"); // with document.write
    </SCRIPT>
</BODY>
</HTML>
```



10.6.3 Math Object

This object contains methods and constants to carry more complex mathematical operations. This object cannot be instantiated like other objects. All properties and methods of ***Math*** are static. We can refer to the constant p as ***Math.PI*** and the sine function as ***Math.sin(x)***, where ***x*** is the method's argument.

Properties	***Description***
Math.PI	Returns the value of p
Math.E	Euler's constant and the base of natural logarithms.
Math.LN2	Natural logarithm of 2.
Math.LN10	Natural logarithm of 10, approximately 2.302.
SQRT1_2	Square root of ½.
SQRT2	Square root of 2.
Methods	***+Description***
pow(x, p)	Returns X^P
abs(x)	Returns absolute value of x.
exp(x)	Returns e^x
log(x)	Returns the natural logarithm of x.
sqrt(x)	Returns the square root of x.
random()	Returns a random number between 0 and 1.
ceil(x)	Returns the smallest integer greater than or equal to x.
floor(x)	Returns the largest integer less than or equal to x.
min(x, y)	Returns the lesser of x and y.
max(x, y)	Returns the larger of x and y.
round(x)	Rounds x up or down to the nearest integer.
sin(x)	Returns the sin of x, where x is in radians.
cos(x)	Returns the cosine of x, where x is in radians.
tan(x)	Returns the tan of x, where x is in radians.

Example : To illustrate the properties and methods of the Math object.

```
<HTML>
<HEAD>
<TITLE>Math Object</TITLE>
</HEAD>
<BODY>
<SCRIPT type="text/JavaScript">
document.write("Value of PI :"+Math.PI + "<BR>");
document.write("Random value:"+Math.random()+"<BR>");
document.write("Rounded value of 0.69 :"+ ↵
Math.round(0.69)+"<br>");
document.write("Value of 5 <SUP>2</SUP> :"+ ↵
```

```
Math.pow(5,2) + "<br>");
document.write("Square root of 2 :"+Math.SQRT2 );
</SCRIPT>
</BODY>
</HTML>
```

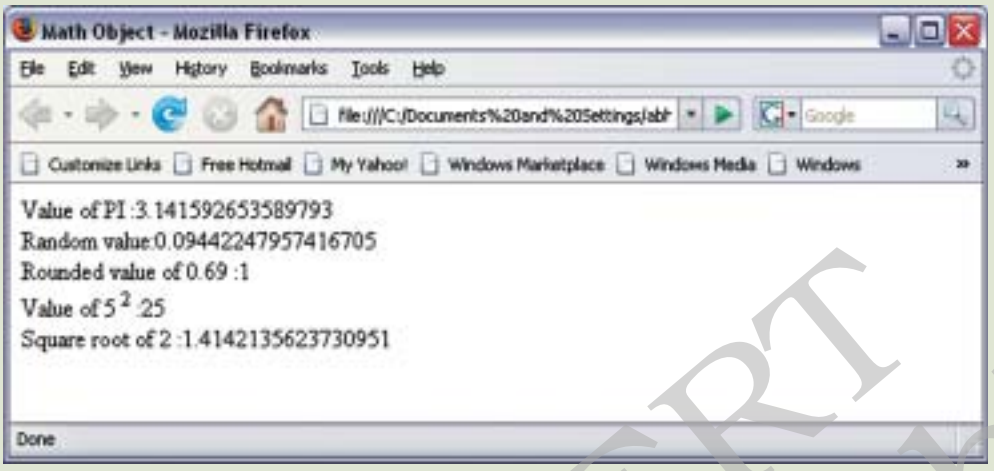


Figure 10.8

10.7 EXPRESSIONS AND OPERATORS

An expression is a combination of operators operands that can be evaluated. It may also include function calls which return values.

Examples

```
x = 7.5                    // a numeric literal
"Hello India!"             // a string literal
false                      // a Boolean literal
{feet:10, inches:5}        // an object literal
[2,5,6,3,5,7]              // an array literal
v= m + n;                  // the variable v
tot                        // the variable tot
```

10.7.1 Arithmetic Operators

These are used to perform arithmetic/mathematical operations like subtraction, division, multiplication etc. Arithmetic operators work on one or more numerical values (either literals or variables) and return a single numerical value. The basic arithmetic operators are:

+	(Addition)	**-**	(Subtraction)
*	(Multiplication)	**/**	(Division)
%	(Modulus)	**++**	(Increment by 1)
—	(Decrement by 1)		

Examples

```
var s = 10 + 20;           // result: s=30
var h = 50 * 4;            // result: h = 200
```

```
var d = 100 / 4;            // result: d = 25
var r = 72 % 14;            // result: r=2
```

Increment and decrement operators

These operators are used for increasing or decreasing the value of a variable by 1. Calculations performed using these operators are very fast.

Example

```
var a = 15;
a++;                        // result: a = 16
var b = 20;
b–;                         // result: b = 19
```

10.7.2 Assignment Operators

It assigns the value of its right operand to its left operand. This operator is represented by equal sign(=).

Example

x = 100; // This statement assigns the value 100 to x.

JavaScript also supports shorthand operator for standard operations. The shorthand operator with example :

Shorthand operator	*Example*	*is equivalent to*
+ =	a + = b	a = a + b
– =	a – = b	a = a – b
* =	a * = b	a = a * b
/ =	a / = b	a = a / b
% =	a % = b	a = a % b

10.7.3 Relational (Comparison) Operators

Relational Operators are some symbols which return a Boolean value true or false after evaluating the condition. For example x > y; returns a value true is value of variable x is greater than variable y.

Basic JavaScript comparison operators are given in the table below :

Operator	*Description*	*Example*
= =	is equal to	4 = = 8 returns false
! =	is not equal to	4 ! = 8 returns true
>	is greater than	8 > 4 returns true
<	is less than	8 > 4 returns false
< =	is less than or equal to	8 < = 4 returns false
> =	is greater than or equal to	8 > = 4 returns true

Relational operators are functional for strings as well. The comparison takes place in alphabetical order. This alphabetical order is based on ASCII number. For example :

Statement	Output
`"zero" < "one"`	`// false`
`"Zero" < "one"`	`// true`
`10 < 5`	`// false, numeric comparison.`
`"10" < "5"`	`// true, string comparison.`
`"10" < 5`	`// false, numeric comparison;`
`"Ten" < 5`	`// Error occurs, "Ten" can not be converted into a number`

10.7.4 Logical Operators

Logical operators are used for combining two or more conditions. JavaScript has following three logical operators :

Operator	Description with Example
&& (AND)	returns *true* if both operands are *true* else it return *false*.
| | (OR)	returns *false* if both operands are *false* else it returns *true*.
! (NOT)	returns *true* if the operand is *false* and false if operand is *true*.

10.7.5 Concatenation Operator

The + operator concatenates two string operands. The + operator gives priority to string operands over numeric operands It works from left to right. The results depend on the order in which operations are performed. For example :

Statement	Output
"Good" + "Morning"	"GoodMorning"
"5" + "10"	"510"
"Lucky" + 7	"Lucky7"
4 + 7 + "Delhi"	"11Delhi"
"Mumbai" + 0 +0+ 7	"Mumbai007"

10.7.6 Special Operators

Conditional Operator (? :)

The conditional operator is a special JavaScript operator that takes three operands. Hence, it is also called ternary operator. A conditional operator assigns a value to a variable based on the condition.

Syntax

```
var_name = (condition) ? v_1 : v_2
```

If (condition) is true, the value v_1 is assigned to the variable, otherwise, it assigns the value v_2 to the variable.

For example

```
status = (age >= 18) ? "adult" : "minor"
```

This statement assigns the value "adult" to the variable status if age is eighteen or more. Otherwise, it assigns the value "minor" to status.

New

new operator is used to create an instance and allocate memory to a user-defined or predefined object types.

Syntax

ObjectName = new objectType (param1 [,param2] ...[,paramN])

Example

```
d = new Date(); // date assigns to object d
r = new rectangle(4, 5, 7, 8);
```

Delete

The *delete* operator de-allocates (releases) the memory space that was allocated using the new operator by deleting an object, an object's property or an element from an array.

The syntax is

```
delete object_name
delete object_name.property
delete array_name[index]
```

delete operator can be used to delete variables declared implicitly but not those declared with the *var* statement. The delete operator returns true if the operation is possible; it returns false if the operation is not possible.

```
a=12
var j= 63
myobj=new Number()
myobj.h=55              // create property h
delete x                /* returns true (x is declared implicitly,
                        without using var)*/
delete y                /* returns false ( y is declared
                        explicitly using var) */
delete Math.PI          /* returns false (cannot delete predefined
                        properties)*/
delete myobj.h          /* returns true (can delete user-defined
                        properties)*/
```

```
delete myobj          /* returns true (can delete if declared
                      implicitly) */
```

When we delete an array element, the array length will not be affected. For example, if we delete a[3], then a[4] still remains a[4] and a[3] will be undefined. When the *delete* operator removes an array element, that element is no longer in the array.

this

JavaScript supports ***this*** operator. The word ***this*** refers to the current object. It is like a pointer to the current object.

The syntax is

```
this[.propertyName]
```

Example

Use of *this* operator to validate the age. Here input is provided through the text box.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    function validate(obj, min_age, max_age)
{
   if ((obj.value < min_age) || (obj.value > max_age))
      alert("Invalid age for the Job!!!");
}
</script>
</HEAD>
<BODY>
<B>Enter the age (between 18 and 40):</B>
<INPUT TYPE = "text" NAME = "age" SIZE = 2
onChange="validate(this, 18, 40)">
</BODY>
</HTML>
```

In this example we called validate() function by onChange event handler. Here, this operator is used to pass current object (viz. text box).

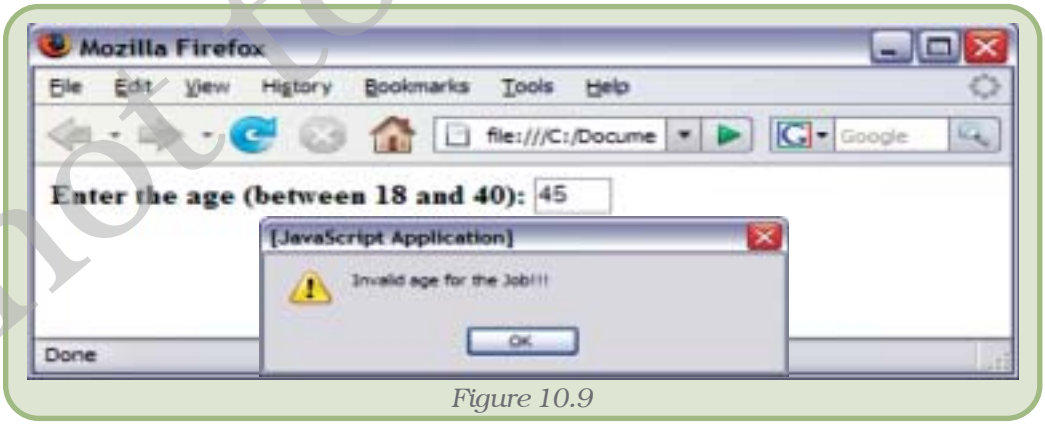


Figure 10.9

10.7.7 Operator Precedence

Operators are evaluated in a predefined order of precedence. The following table shows operators from highest priority to lowest priority:

Table 10.1 : Operator Precedence

Operator	Description	Priority
* / %	Multiplication Division Modulus	Highest
+ –	Addition Subtraction	
< <= > >=	Less than Less than equal to Greater than Greater than equal to	
== !=	Equality Not equality	
&&	Logical AND	
||	Logical OR	
?:	Conditional	
= += –= *= /= %=	Assignment Operators	
,	Comma	Lowest

10.8 JAVASCRIPT POPUP BOXES (DIALOG BOXES)

In JavaScript, three kinds of popup boxes – Alert box, Confirm box, and Prompt box can be created using three methods of window object.

10.8.1 Alert Box

Alert( ) method of window object creates a small dialog box with a short text message and "OK" command button called alert box. Alert box contains an icon indicating a warning.

Syntax

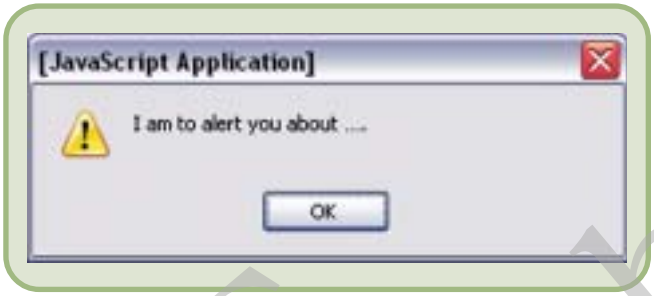
```
[window].alert("Text to be displayed on the popup box");
```

The word *window* is optional.

Example

```
window.alert("I am to alert you about ….");
or
alert("I am to alert you about ….");
```

Output



An alert box is used if we want to display some information to the user. When an alert box appears, the user needs to click "OK" button to proceed.

10.8.2 Confirm Box

Confirm box is used if we want the user to verify and confirm the information. The user will have to click either "OK" or "Cancel" buttons.

Syntax

```
[window].confirm("Text to be confirmed");
```

Example

```
confirm("Do you want to quit now?");
```

Output

Confirm box returns a Boolean value. If the user clicks on "OK", it returns true. If the user clicks on "Cancel", it returns false.

10.8.3 Prompt Box

Prompt box allows getting input from the user. We can specify the default text for the text field. The information submitted by the user from *prompt( )* can be stored in a variable.

Syntax

```
prompt("Message" [, "default value in the text field"]);
```

Example

```
var name = prompt("What's your ↵
name? ", "Your name please…");
```

Output

A prompt box returns input string value when the user clicks "OK". If the user clicks "Cancel", it returns null value.

10.9. BLOCK STATEMENT

Two or more statements can be combined to form a single block statement using a set of curly brackets.

The Syntax is

```
{
   statement_1
   statement_2
   .
   .
   statement_n
}
e.g.  If (z > y)
{    x=10;
y=20;   }
```

10.10 BRANCHING AND LOOPING STATEMENTS

JavaScript allows to select among alternative ways or to repeat the execution of a statement or block of statements. JavaScript supports some conditional statements for the branching. A conditional statement is a statement that we can use to execute a part of code based on a condition or to do something else if the condition is not met.

Looping is repeating execution of a set of statements for a number of times.

10.10.1 Branching (Conditional) Statements

Branching With *If* Statements

An ***if*** statement is used to execute a statement or a block of statements on based of logical expression (condition). There are three different forms :

- if ... statement (simple ***if*** statement).
- if ... else statement.
- If .. else if .. else statement (***else if*** ladder statement)

if ... statement

The '***if***' is the simplest decision making statement. This statement is used to execute statement(s) only if a specified condition is true.

The Syntax is

```
if ( condition )
{
.. statement(s) to be executed if (condition)  is true...
}
```

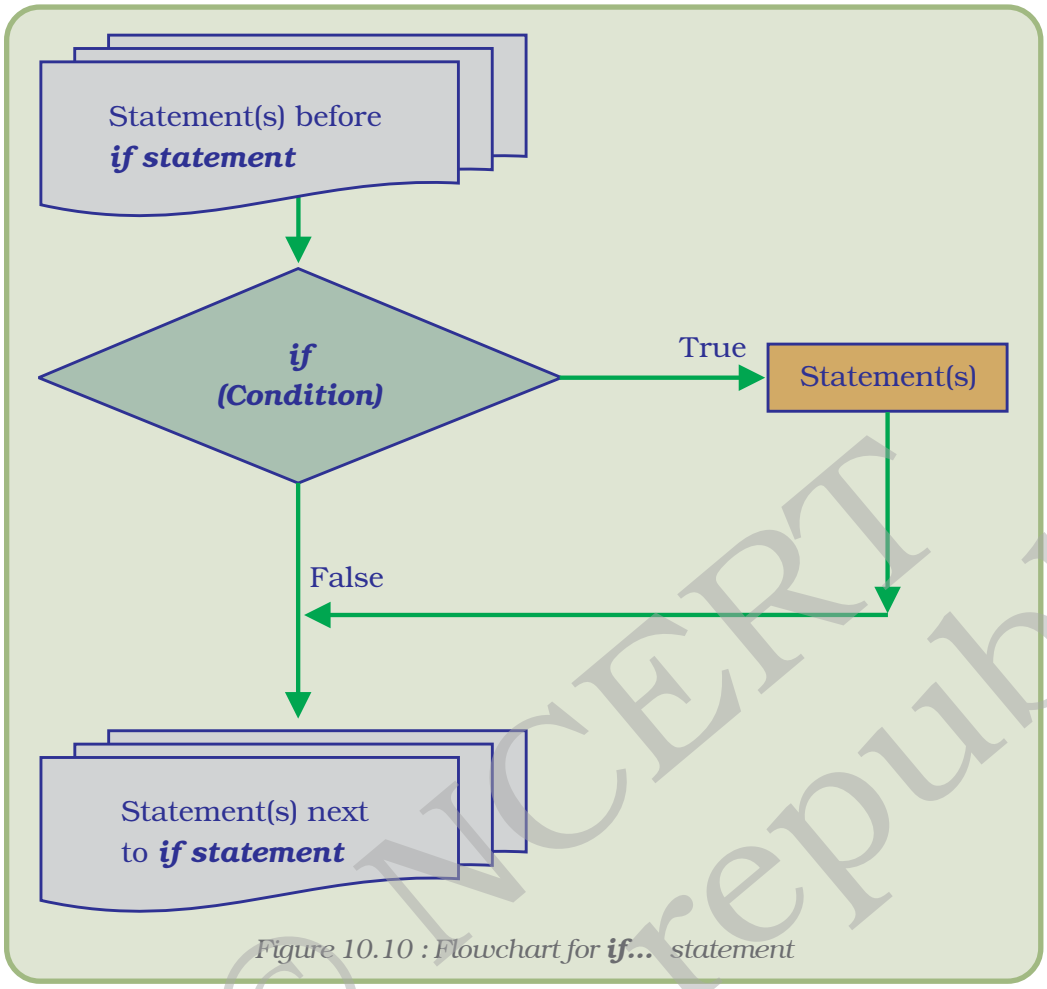


Figure 10.10 : Flowchart for ***if...*** *statement*

In the above flowchart, statement(s) is/are executed only when logical expression is true. Otherwise, the statement following 'if statement' will be executed next.

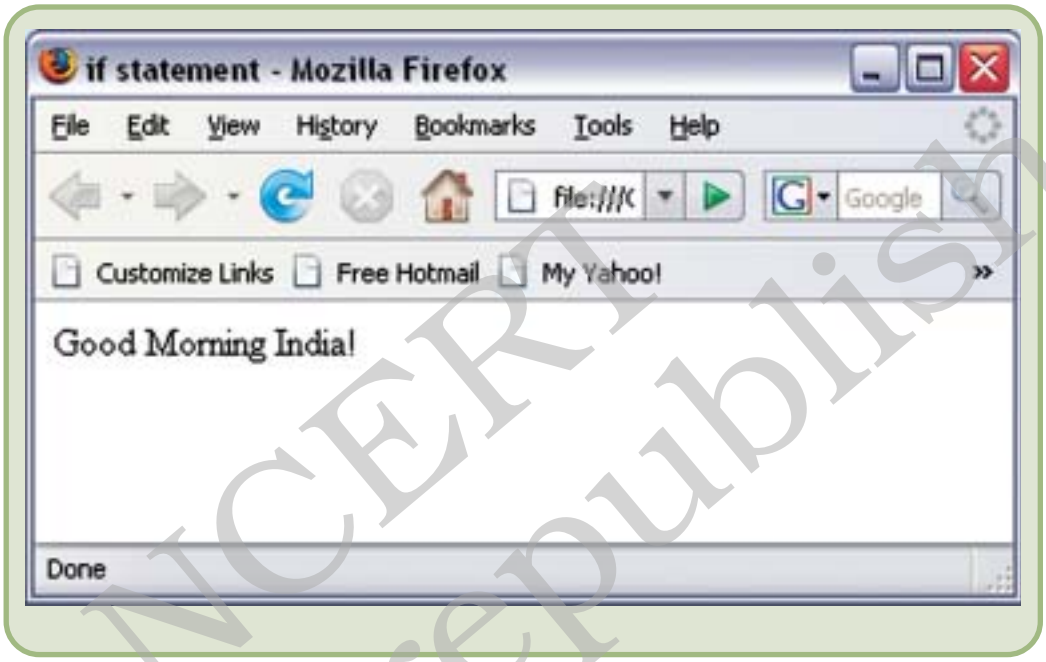
Example : A JavaScript program that displays 'Good Morning India!' if and only if time is less than 12 hours on web page otherwise page will remain blank.

```
<HTML>
<HEAD>
    <TITLE>if statement</TITLE>
</HEAD>
<BODY>
<script type = "text/javascript">
var d = new Date();
var time = d.getHours();  // time stores hours
if (time < 12)
{
    document.write("Good Morning India!");
}
</script>
```

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

This HTML document displays the message "Good Morning India!" if your system time is less then 12 Hrs. Other wise you will find the page blank.

Output



if ... else ... statement

This statement is an extension of the simple if statement. It permits one of two statement or group of statements depending upon the logical test.

The Syntax is

```
if ( condition )
{
    True statement(s)...
}
else
{
    False statement(s)...
}
```

If the logical expression (condition) is true, the true statement(s) will be executed. Otherwise, the false statement(s) will be executed.

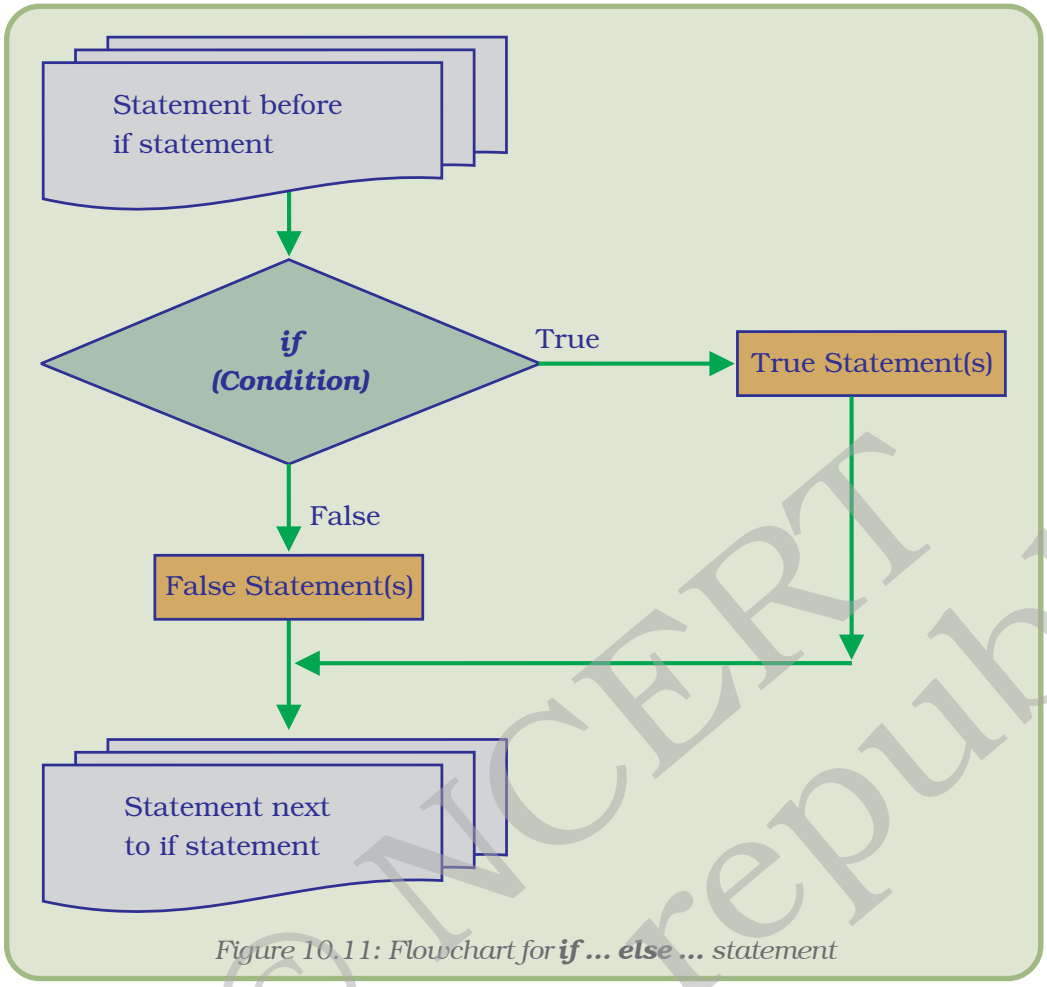


Figure 10.11: Flowchart for ***if ... else ...*** *statement*

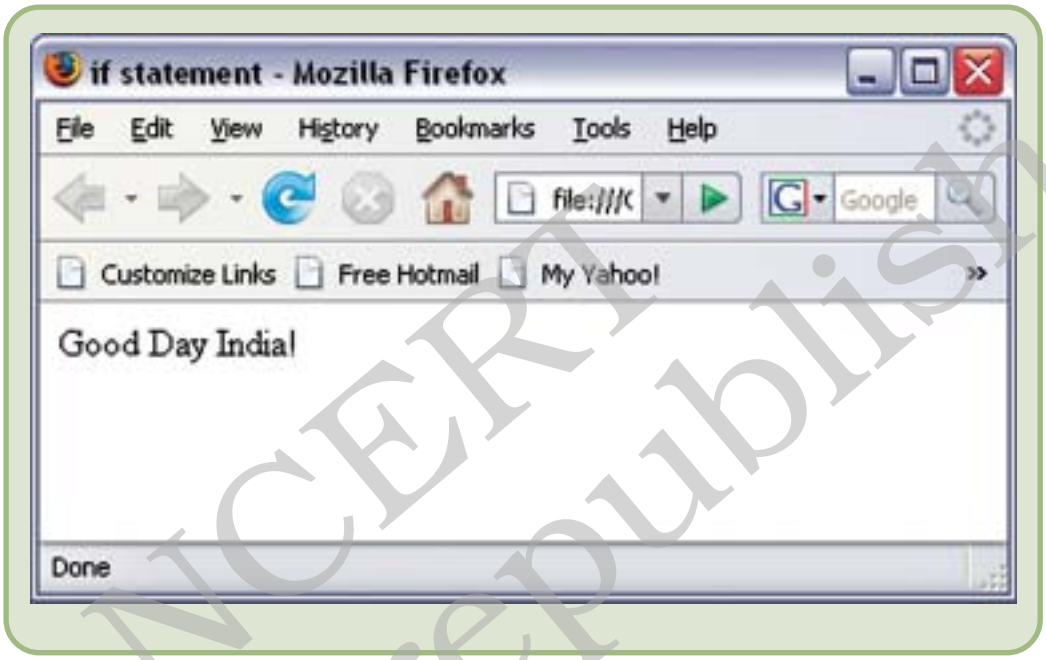
Example : A JavaScript program that displays 'Good Morning India!' if time is less than 12 hours otherwise it shows 'Good Day India!' on the document.

```
<HTML>
<HEAD>
    <TITLE>if else statement</TITLE>
</HEAD>
<BODY>
<script type = "text/JavaScript">
var d = new Date();
var time = d.getHours();
if (time < 12)
{
    document.write("Good Morning India!");
}
else
{
    document.write("Good Day India!");
}
</SCRIPT>
```

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

This HTML document displays the message "Good Morning India!" if the system time is less then 12 hours. Other wise it display the message "Good Day India!".

Output



If...else if...else Statement

The if....else if...else statement is further an extension of the if... else... statement. This statement is useful for selecting one of many sets of statements to execute.

The Syntax is

```
if (condition1)
{
    code to be executed if condition1 is true
}
else if (condition2)
{
    code to be executed if condition2 is true
}
.
.
```

```
else
{
    code to be executed if any of the conditions is not true
}
```

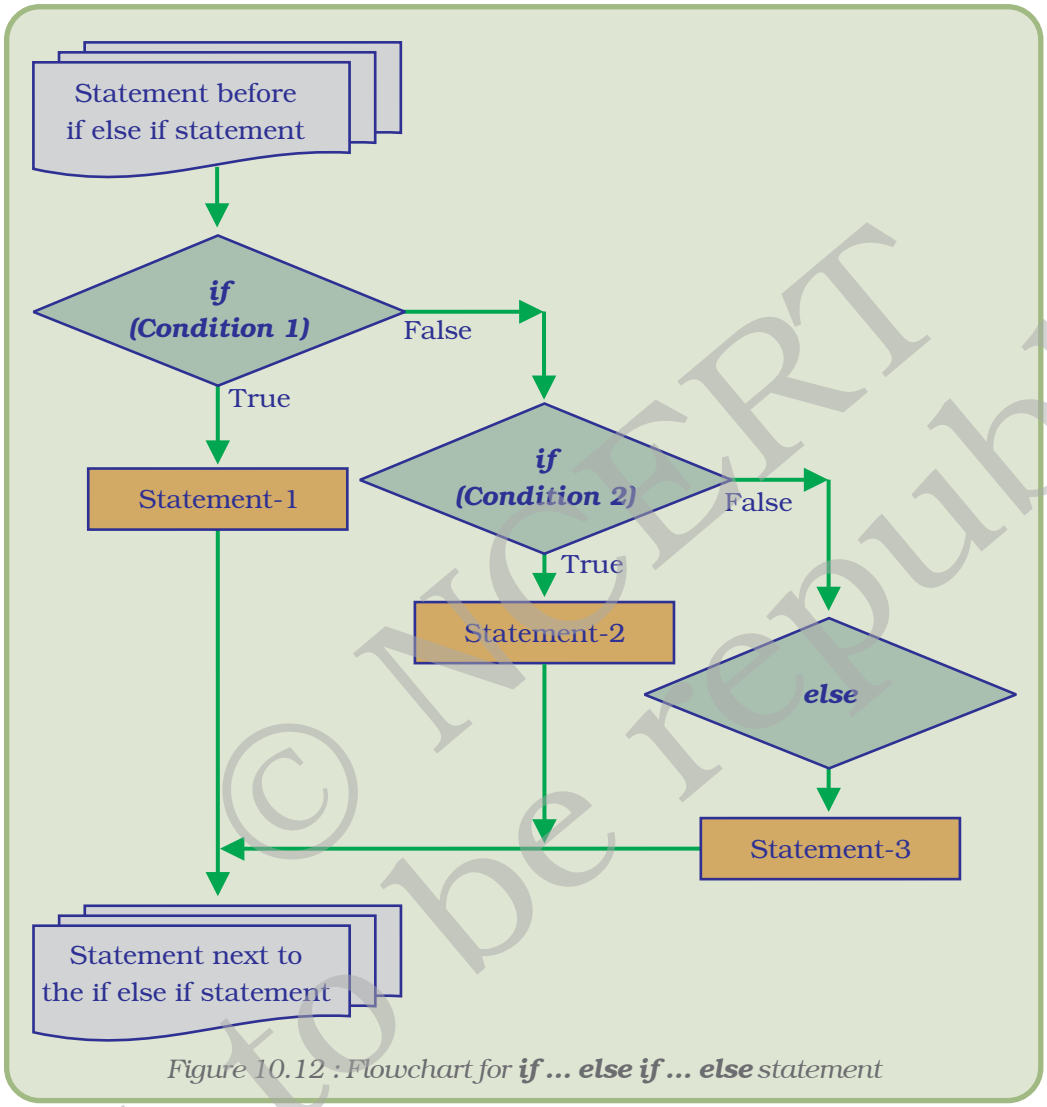
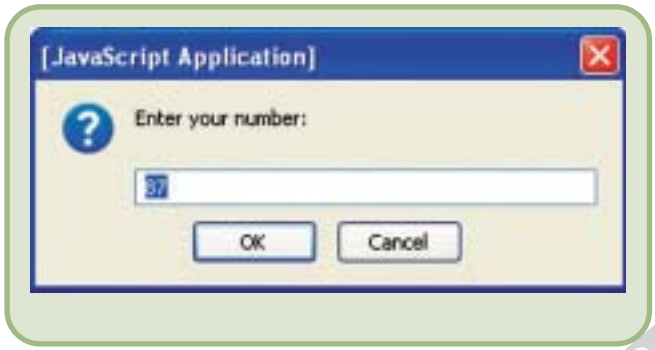


Figure 10.12 : Flowchart for ***if ... else if ... else*** *statement*

Example : Write a program to check whether a number input through prompt box is Zero, Odd or Even.

```
<HTML>
<HEAD>
    <TITLE>Odd, Even or Zero</TITLE>
</HEAD>
<BODY>
<SCRIPT type = "text/JavaScript">
    var n = prompt("Enter your number:","Type your number↵
here");
```

```
n = parseInt(n);        //converts string into number
if (n == 0)
      alert("The number is zero!");
else if (n%2)
      alert("The number is odd!");
else
      alert("The number is even!");
</SCRIPT>
</BODY>
</HTML>
```



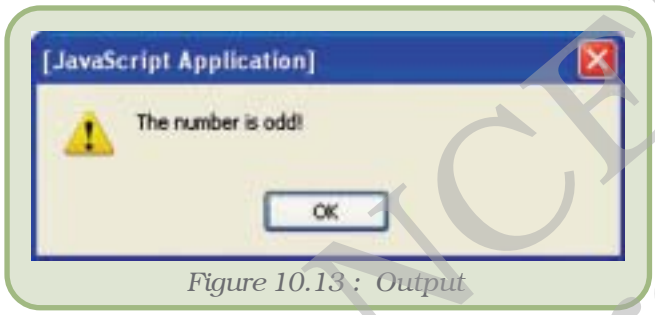


Figure 10.13 : Output

Output

Selection with switch statement

A switch statement is used to execute different statement based on different conditions. It provides a better alternative than a long series of if... else if ... statements.

The Syntax is

```
switch (expression)
{
case label1 :    //executes when value of exp. evaluates to label
statements;
break;
case label2 :
statements;
    break;
...
default : statements;    //executes when non of the above labels
                         //matches the result of expression
}
```

Program 10.5 : To compute the day of week (in words) while you input the date within prompt dialog box.

```
<HTML>
<HEAD>
    <TITLE>Switch statement</TITLE>
</HEAD>
<BODY>
```

```
<script type="text/JavaScript">
// Enter date of birth to know the day of week on that day.
var d=new Date(prompt("Enter your Date of Birth
(e.g. November 17, 2002)", "Month DD, YYYY"))
dy=d.getDay()
switch (dy)
{
case 0:
    document.write("It was <b>Sunday</b> on that day.")
    break
case 1:
    document.write("It was <b>Monday</b> on that day.")
    break
case 2:
    document.write("It was <b>Tuesday</b> on that day.")
    break
case 3:
    document.write("It was <b>Wednesday</b> on that day.")
    break
case 4:
    document.write("It was <b>Thursday</b> on that day.")
    break
case 5:
    document.write("It was <b>Friday</b> on that day.")
    break
case 6:
    document.write("It was <b>Saturday</b> on that day.")
    break
default:
    document.write("Please input a
valid Date in?
prescribed format !!!")
}
</script>
</BODY>
</HTML>
```

The value of this expression is then compared with the values for each case in the switch structure. If there is a match, the block of code associated with that case is executed. If no case is matched, the statement in default will be executed. Use **break** to prevent the code from running into the next case automatically. There is no need to use break within the default case.

When you press OK button, we will get the output (Figure 10.14) :

10.10.2 Loop Statements

Loop statements are the primary mechanism for telling a JavaScript interpreter to execute statements again and again until a specified

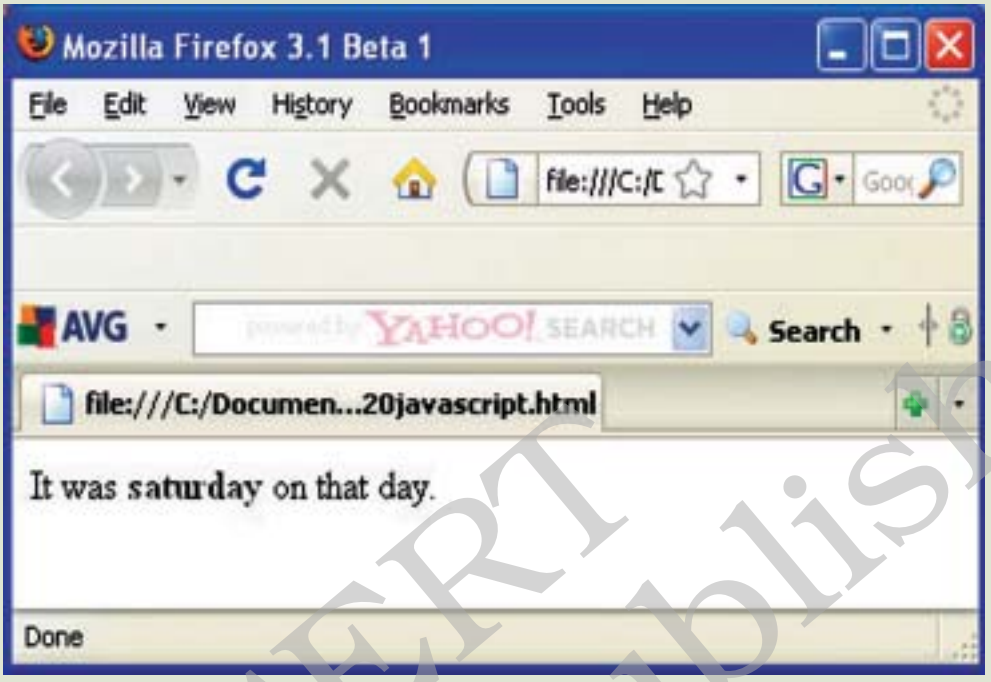


Figure 10.14 : Output

condition is met. JavaScript supports following looping statements :

- for
- do ... while
- while loop

Most loops have a counter variable which is initialised before the loop starts and then it is tested as part of the condition (expression) evaluated before every iteration of the loop. Finally, the counter variable is incremented or updated at the end of the loop body just before the condition is evaluated again.

For

The ***for*** loop consists of three optional expressions separated by semicolon, followed by a block of statements executed in the loop. Loop statements executed repeatedly again and again until the condition is false. The ***for*** loop is used when we know in advance how many times the script code should run.

The Syntax is

for([*initial-expression*]; [*condition*]; [*increment-expression*])

{

statements

}

Parameters

Initial-expression – used to initialise a counter variable.

Condition – If condition evaluates to true, the statements are executed.

Incr.-expression – to increment the counter variable.

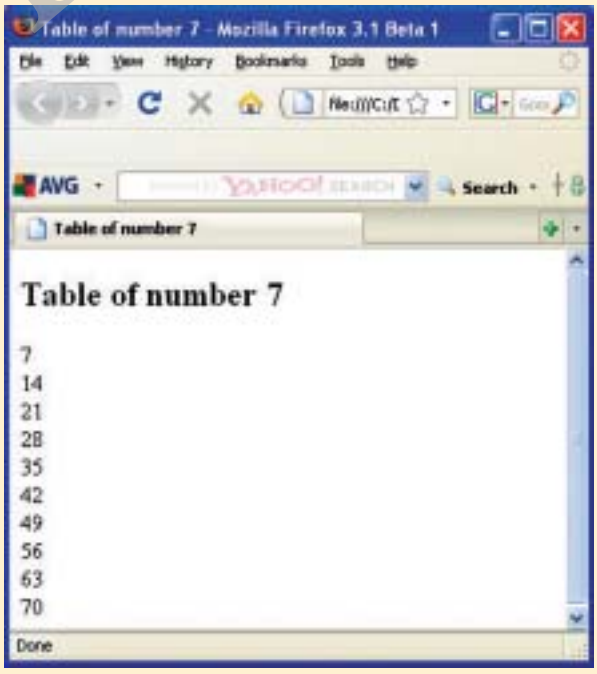
Examples

The following ***for*** statement declares variable i and initialising it to 1. It checks that i is less than 20, performs the two succeeding statements, and increments i by 2 after each pass through the loop.

```
// for loop to display odd numbers between 1 to 20
for (var i = 1; i < 20; i+=2)
{
    document.write(i);
    document.write("<BR>");
}
```

Program 10.6: A JavaScript program to generate the table of number 7 using ***for loop*** statement.

```
<HTML>
<HEAD>
    <TITLE> Table of 7 </title>
</HEAD>
<BODY>
<SCRIPT language="JavaScript" type="text/JavaScript">
    document.write("<H2> Table of number 7 </H2>");
    for(i = 1; i <= 10; i++ )
    {     document.write(7*i);
          document.write("<BR>");
    }
</SCRIPT>
</BODY>
</HTML>
```



While

The ***while*** loop statement is simpler than the ***for*** loop. It consists of a condition and block statement. The condition is evaluated before each pass through the loop. If the condition is true then it executes block statement.

The Syntax is

while (*condition*)

{

statements

}

Example : The following while loop gives same output as for loop in previous example.

```
// While loop to display Odd numbers between 1 to 20
var i = 1;      // Initialization of counter variable
while (i < 20)  // Condition
{
    document.write(i);
    document.write("<BR>");
    i++;        // Updation
}
```

In while loop, we have to maintain a counter variable which controls the execution of statements in the loop.

Program 10.7 : Write a JavaScript program to generate first 10 Fibonacci numbers.

```
<HTML>
<HEAD>
<TITLE>Fibonacci Series</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/JavaScript">
// Program to print the Fibonacci series upto 10 numbers
document.write("Fibonacci Series... upto 10 numbers <BR>".fontsize(4));
//.fontsize to increase the font size of the string
i=0;
document.write(i + "  ");
j=1;
document.writeln(j + "  ");
var x = 3;
while (x <= 10)
{
    t = i + j;
    document.write(t + "  ");
    i = j;
    j = t;
```

```
    x++;
}
</SCRIPT>
</BODY>
</HTML>
```

Do...While

The do...while loop is much like a while loop. It will repeat the loop until the specified condition is false. This loop will always be executed at least once, even if the condition is false, because the block of statements is executed before the condition is tested. In this loop statement, curly braces are optional.

The Syntax is

do

{

statements

}

while (*condition*);

Example : The following do...while loop gives same output as while loop in previous example.

```
// do…while loop to display Odd numbers between 1 to 20
var i = 1;  // Initialization of counter variable
do
{
    document.write(i);
    document.write("<BR>");
    i++;  // Updation
}
while (i < 20); // Condition
```

Condition lies between the parentheses after the block of statements with while keyword.

10.10.3 Label

A label is an identifier followed by a colon that can be helpful in directing the flow of program.

The Syntax is

label: *statement*

The value of label may be any JavaScript identifier. The statement that you identify with a label may be any statement.

Example

In this example, the label "whileloop" identifies a while loop.

```
x=1;
whileloop:              // Label
while(x<=10)
{
document.write(x);
x++;
}
```

10.10.4 Break

Break statement is used to exit from the innermost loop, switch statement, or from the statement named by label. It terminates the current loop and transfers control to the statement following the terminated loop.

The Syntax is

break [*label*]

The break statement includes an optional label that allows the control to exit out of a labeled statement.

Example : The following program segment has a break statement that terminates the while loop when it is equal to 3.

```
var i = 0;
while (i < 6)
{
    if (i == 3)
break;              //the control moves out of loop in first iteration
i++;
}
document.write(i);
```

10.10.5 Continue

The continue statement skips the statement following it and executes the loop with next iteration. It is used along with an *if* statement inside while, do-while, for, or label statements.

The Syntax is

continue [*label*]

The continue statement does not terminate the loop. Instead, in a ***while*** loop, it jumps back to the condition and in a ***for*** loop, it jumps to the update expression. The ***continue*** statement can include an optional label that allows the program to terminate a labeled statement and continue to the specified labeled statement.

Example

A program to input 50 elements using prompt() and then compute sum of marks more than 40 using continue statement.

```
var marks = new Array();
var i = 0, sum=0;
while (i < 50)
{
    i++;
    // parseInt converts string value into a number.
    marks[i]=parseInt(prompt("Enter marks"));
    if (marks[i] <= 40)     // when the condition is true then
    continue;   // control goes to while condition expression.
    sum = sum + marks[i];
}
document.write(sum+"\n");
```

10.11 OBJECT HANDLING STATEMENTS

JavaScript provides some special statements to handle the objects. Two type of statements are ***for...in*** and ***with***

10.11.1 For...In

The ***for...in*** statement iterates a specified variable over all the properties of an object.

The Syntax is

for (<*variable*> in <*object*>)

{

statements

}

The body of the for...in statement is executed once for each property of an object. Before the loop statement is executed, the name of one of the object's property is assigned to variable, as a string. We can use this variable to look up the value of the object's property with the [ ] operator.

Example : To print the name and value of each property of a Book object.

```
<HTML>
<HEAD>
<TITLE> for... in Example </TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
var Book = new Object();          // Object creation
// Properties and values of Book
Book =    {     Title:"The Discovery of India",↵
          Author:"Jawahar Lal Nehru", ↵
          Publisher: "Penguin Books", ↵
          Price:Rs 399//- ↵
    };
var result = "";
// Name of distinct property of Book assign to b in each loop
execution
for (var b in Book)
{
    // Book[b] is used to get the values.
    result += "Book." +  b + " = " + Book[b] + "<br>";
}
// To print names and values of each property of Book object.
document.write(result);
</SCRIPT>
</BODY>
</HTML>
```

Output

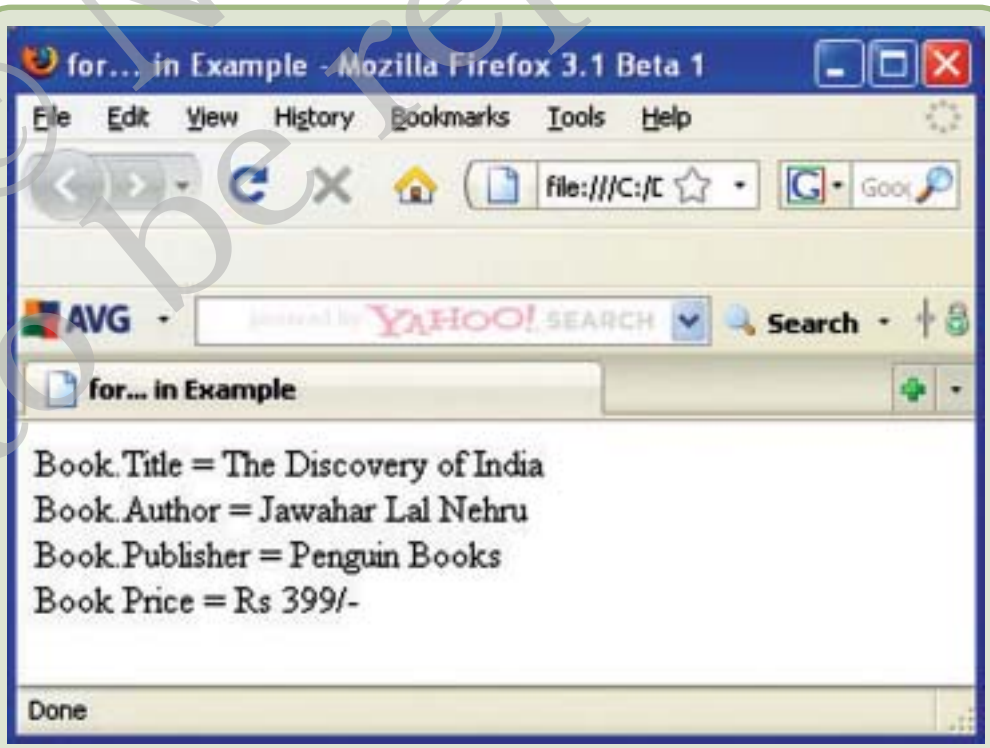


Figure 10.17 : Use of for…in statement

10.11.2 With

With statement saves a lot of typing when properties of same object have to be accessed. For example, it is common to work with deeply nested object hierarchies. Sometimes we have to type expressions like this one to access elements of a HTML form :

Frames[1].document.forms[0].fname.value

Examples

The following script illustrates the use of ***with*** statement. Here two object Math and document are default objects. The statements following the ***with*** statement along with *Math* object refer to the PI property and the cos and sin methods, without specifying an object. Same way the statement following the ***with*** statement along with *document* object refer to the write method, without specifying the object. JavaScript assumes the Math and document object for these references.

```
var area, circumference
var r=10
with (Math)
{
   area = PI * r * r
   circumference  = 2*PI*r
}
with (document)
{
    write("Area of the Circle: "+area+"<br>");
    write("Circumference of the Circle: "+circumference);
}
```

Output

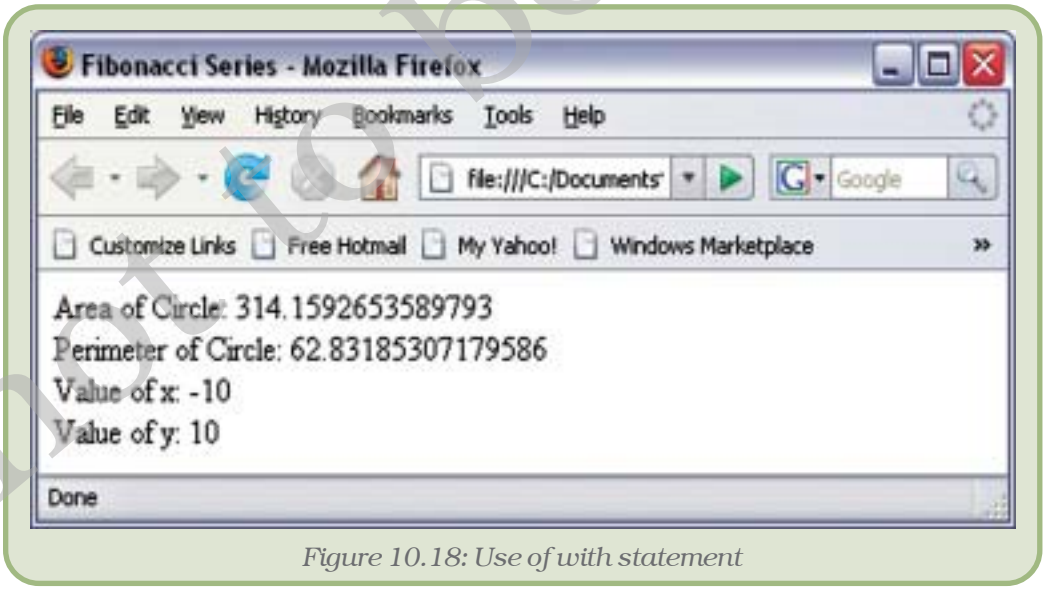


Figure 10.18: Use of with statement

10.12 JAVASCRIPT FUNCTIONS

Function is a named block of statements which can be executed again and again simply by writing its name and can return some value. It is useful in making a program modular and understandable.

10.12.1 Defining A Function

Function can be defined using the following syntax:

The Syntax is

```
function <function-name>([<parameter list>])
{
    ... body of the function ..
}
```

The function definition begins with keyword function, followed by function's name and optional parameter-list within parenthesis.. Braces are used ({ and }) to enclose all of the statement in a function. Let's an example of function definition.

```
function Welcome()
{
    alert("Welcome to NCERT ");
}
```

These statements define a function Welcome that displays an alert message to the user.

10.12.2 Using Parameters With Function

The arguments received by the function in a list of corresponding values are called parameters. These values can be assigned to local variables within the function. Let's try an example of function using parameter:

```
function Welcome(name)
{
    alert("Welcome to NCERT , " + name);
}
```

We have learnt how to create simple functions. The best place for a function definition is within the <HEAD> section of the HTML document, because the statements in this section are executed first, this ensures that function is defined before it is used.

Example : Using a function within the HEAD section of a HTML document.

```
<HTML>
<HEAD>
<TITLE> Define a Function </title>
```

```
<script language="JavaScript" type="text/JavaScript">
function Welcome(name)
{
    alert("Welcome to NCERT, " + name+"!");
}
</script>
</HEAD>
<BODY>
This is the body of a HTML document. You will observe that I
have used script code in small case and other HTML tags in
Capital to enhance the readability of JavaScript Code.
</BODY>
</HTML>
```

10.12.3 Calling The Function

A function can be called by writing the name of the function along with the list of arguments. A function call can also be used in an event handler code also.

The Syntax is

<function name> ([<parameter list>])

Example

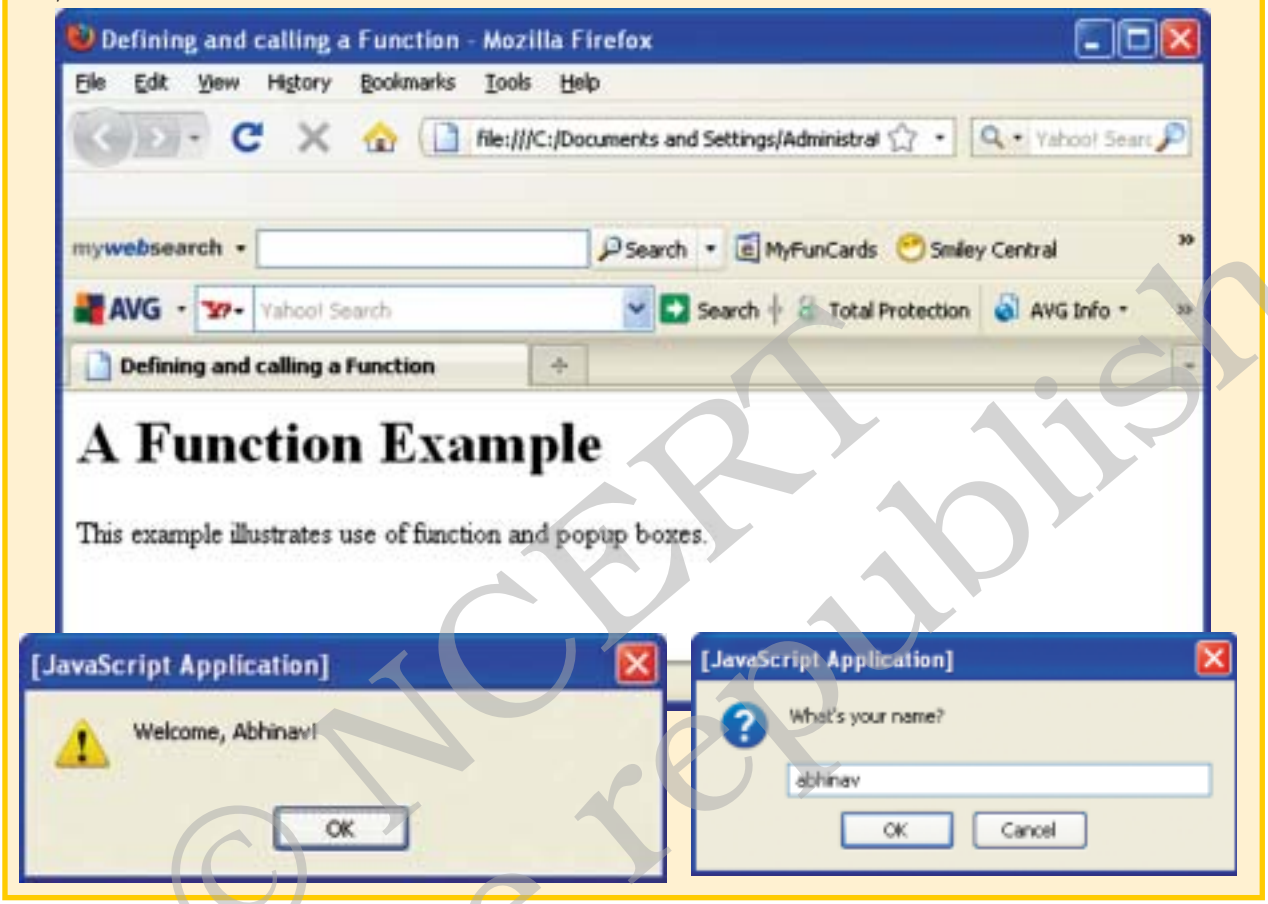
```
Welcome("MANYA");
```

This tells the JavaScript interpreter to transfer control to the first statement of function 'Welcome'. It also passes the parameter "MANYA" to the function. The value will be assigned to the ***name*** variable inside the function.

Program 10.8 : A HTML document that illustrates function definition and calling of Welcome() function.

```
<HTML>
<HEAD>
<TITLE> Defining and calling a Function </title>
<script language="JavaScript" type="text/JavaScript">
// function definition
function Welcome(name)
{
    alert("Welcome, " + name);
}
</script>
</HEAD>
<BODY>
<H1> A Function Example</H1>
This example illustrates use of function and popup boxes.
<br>
<script language="JavaScript" type="text/JavaScript">
    var nm = prompt("What's your name? ", "Your name please…");
    // Calling of function Welcome().
```

```
    Welcome(nm);
</script>
</BODY>
</HTML>
```



10.12.4 Calling A Function From An Event

Once a function is defined, it may be used with events like on Click event. Here, the function simply becomes another JavaScript command. For example:

```
<INPUT type = "button"
   value = "Calculate"
   onClick = calc() >
```

When the user clicks the button, the program automatically calls the calc() function.

10.12.5 Returning Value From Function

The *return* statement is used to return a value from a function. A variable using assignment operator can hold the returned value.

For example, a program to calculate simple interest using function is :

```
<HTML>
```

```
<HEAD>
<TITLE>A Simple JavaScript Function returning Value </TITLE>
<script language="JavaScript" type="text/JavaScript">
    function si( p, r, t )
    {     var s = (p * r * t)/ 100
          return s;   // function returning value s
    }
</script>
</HEAD>
<BODY>
<script language="JavaScript" type="text/JavaScript">
    var result = si( 1000, 5, 7)   // returned value assigned
to result
    document.write ( "Result = " + result);
</script>
</BODY>
</HTML>
```

Output

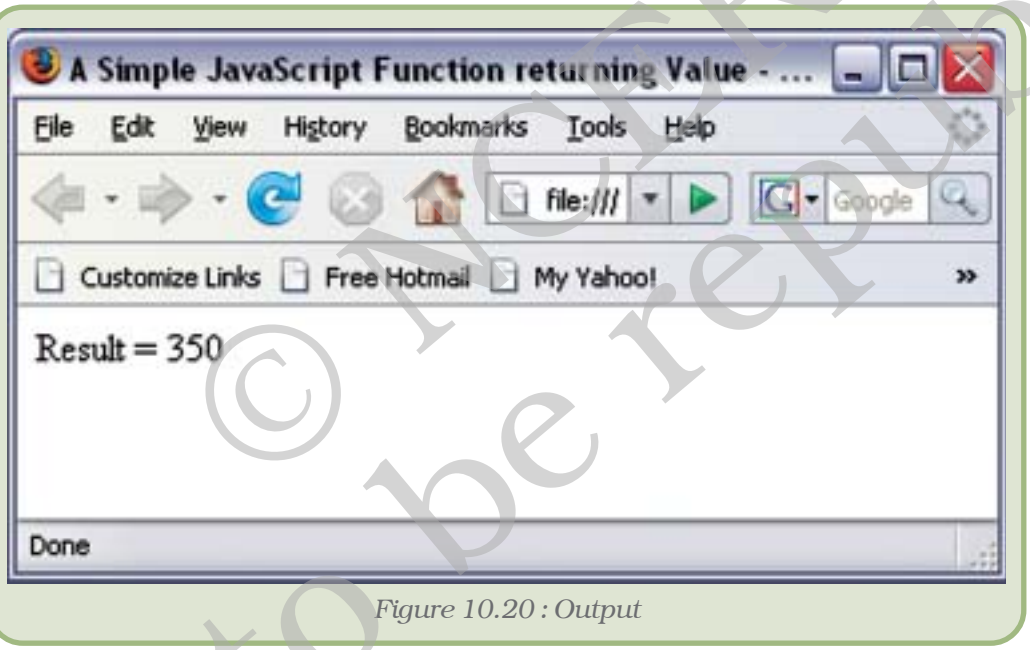


Figure 10.20 : Output

10.12.6 Nested Function

A function may be nested inside another function definition. They may not be defined within statement blocks, such as the body of an if statement or while loop etc.

Suppose we want to calculate the area of the circle, which is equal to 3.14 x r^2 . The nested functions would be written as follows:

```
function Area(r)
{
    function Square(x)
    {
```

```
            return x*x;
        }
        return 3.14*Square(r);
    }
```

In this script, Square() function is nested inside the Area() function.

Reserved/Key Words in JavaScript

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

Summary

- JavaScript is a platform independent object-based scripting language.
- Client-side JavaScript is the name given to JavaScript code that is executed by a web browser on client machine.
- JavaScript is a case-sensitive language and all the statements are written in lower case.
- JavaScript allows omitting semicolon when statements are placed in separate lines. If we combine statements into a single line, we must use semicolon to separate them.
- ***document.write*** is a standard JavaScript command for producing output to a document window.
- Literals refer to the constant values that are used directly in a program code.
- A variable is a container of values or string. The values stored in a variable can be accessed using the variable name.
- JavaScript supports three primitive data types: number, string and boolean. JavaScript allows two composite data types: objects and arrays.
- Expression is combination of operators and literals or variable names.

- The increment and decrement operators act on only one operand. These are used for increasing or decreasing the value of a variable by 1.
- An assignment is a basic operator, which assigns the value of its right operand to its left operand. This operator is represented by an equal sign(=).
- Relational operators are used to compare the values of operands and it returns Boolean value based on the condition.
- JavaScript has three logical operators: **&&** (AND) , **||** (OR) , and **!** - (NOT)
- The + operator is also used to concatenate two string operands. It gives priority to string operands over numeric operands.
- Conditional operator (? :) is also called ternary operator.
- ***new*** operator is used to create an instance of a user-defined object type or one of the predefined object types.
- The ***delete*** operator is used to de-allocate the memory space.
- The *in* operator returns true, if the specified property/index exists in the specified object.
- Alert box is a dialog box with a text message and "OK" button.
- Confirm box is a box meant to verify or accept some information, the user have to click either "OK" or "Cancel" buttons to proceed.
- Prompt box allows us to gather user's input with the help of text field.
- Block statement combines two or more statements into a one statement. Block statements are commonly used with conditional and looping statements.
- An ***if*** statement is used to execute a statement or a block of statements on the basis of condition.
- A switch statement in JavaScript is used to perform different actions based on different conditions. It can be a replacement for multiple if... else if... statement.
- Loop statements tell JavaScript interpreter to execute same statements again and again until a specified condition is met.
- The ***for*** loop consists of three optional expressions. It executes block statement repeatedly again and again until the condition is false.
- The ***while* loop statement** is simpler than ***for*** loop. It repeats block statement again and again until the specified condition is false.
- Unlike ***while*** loop, ***do...while*** loop always executes a block of code at least once.
- Instead of exiting a loop, continue statement skips the statement following it and restarts a loop in a new iteration.
- The ***for...in*** statement iterates a specified variable over all the properties of an object.
- The ***with*** statement establishes the default object for a set of statements.
- Function is a named unit for the group of JavaScript statements. If a user needs to send values to a function, the values are enclosed in the parentheses after the function name and sent as a comma-separated list of arguments when function is called.
- The ***return*** statement is used to return a value from a function. A variable using assignment operator can hold the returned value.
- A function may be nested inside another function definition.

EXERCISES

Short Answer Type Questions

1. Is it possible to write programs for standalone applications in JavaScript?
2. Where should we use semicolon in the statements? Is it mandatory?
3. Differentiate between client-side and server-side JavaScript?
4. What is the purpose of ***document.write()*** in JavaScript code?
5. What are the invalid variable names in following? Explain with reasons.
   My_Name, number10, $100, father's name, marks%
6. What are reserved words? Can we use reserved words as identifiers?
7. What do you mean by literals?
8. What are the data types in JavaScript?
9. What is the purpose of ***var*** statement?
10. Why strings are enclosed within the single or double quotes?
11. Write JavaScript code that displays the following:

    He said, "Mahatma Gandhi was a non-violent soldier of India."
12. Can we use different data type values in a single array?
13. Write a statement to assign numeric values 65, 75, 80, 87, 90 to an array named ***stumarks***.
14. Are the null and zero have similar values?
15. What are operators? What is the significance of an operator?
16. What type of a value a *prompt()method returns*?
17. Write the equivalent JavaScript statements for the following:

    (a)Area = $3.14r^2$
    (b)KE = $\frac{1}{2}$ mv^2
18. What will be the value of variable **r**?
    var r = 45 % 7;
19. What will be the value of variable r?
    var a = 4;
    var b = 7;
    var c = "NCERT";
    r = a + b + c;
20. The '==' operator is not the same as the assignment operator '='. (True or False).
21. Find the value of variable *total*?
    total = (360 * 5) + ((40 / 8) - 9) - ((14 * 6) / 2);
22. What is the value of variable b?
    a = 25;
    b = (++a) + 7;

Long Answer Type Questions

1. Write a JavaScript program to find out whether the given year is leap or not. Use prompt box for input the year.
2. Write a JavaScript program which welcomes the user by addressing
   Good Morning, <User_name> during hour in time is 4 to 11
   Good Noon, <User_name> when hour value is 12.
   Good Afternoon, <User_name> when hour value is 13 to 16
   Good Evening, <User_name> when hour value is 17 to 23
   Good Midnight, <User_name> when hour value is 24 or 1 to 3.
3. Write a program to find the greatest number among three given numbers?
4. What happens when the following JavaScript code is executed?
   var get_res = confirm("Did you like this chapter?");
   if (get_res == true)
   alert("Okay! Let's go to the next!");
5. What will be the output if variable **marks** is 75?

```
if (marks >= 80)
    {
alert("Excellent!");
    }
else if (marks >= 60 && marks < 80)
    {
alert("Good!");
    }
else if (marks > 50 && marks < 60)
    {
alert("Average!");
    }
else
    {
alert("Improve yourself !");
    }
```

6. Write a program using switch statement to print word equivalent of a number from 0 to 9, e.g. 3 should be displayed as "Three".
7. Write a statement that displays an alert box which looks like this :

8. Write a program to print the following output using ***for*** loops.
   1
   22
   333
   4444
9. How many times will the following *for* loop be executed?

```
for (a = 0; a <= 10; a = a + 2)
{
... statements ...
}
```

10. What will be the final value of the variable **sum**?

```
var sum = 0;
function add()
```

```
{
var sum = sum + 20;
}
add();
sum = sum + 1;
```

11. What will be displayed in the alert box at the end of script execution?

```
var y = 0;
for (x = 0; x <= 5; x++, y = y + 50)
{
y = y + 10;
}
alert("The value of y is :" + y);
```

12. Write a program to calculate the average of 5 numbers entered by the visitor through a prompt?

13. If the value of variable **num** is 30, how many times will the following *while* loop be executed?

```
while (num <= 30)
{
... statements ...
num = num + 3;
}
```

14. Write a program to find the reverse of a number (i.e. reverse of 123 is 321)

15. Write a program to convert a decimal number into a binary number.

16. Write a program to check whether a number is palindrome or not. A number is palindrome if it is equal to its reverse number.

17. Using continue display the odd numbers between 1 to 20.

18. Write a program to find sum of digits of a number, e.g. 453 results 4+5+3=12.

19. Write a program to generate prime numbers up to a specific limit.

20. What will be the output of the following JavaScript code?

```
for(i=1; i<=5; i++)
{
document.write("<BR>")
for(j=1; j<=i; j++)
document.write("*")
}
```

21. Identify the errors in the following code segment:

```
function 3_alert_box
{
alert("Hi!, I am from a function.');
}
```

22. Write a program using *while* statement to find out the sum of first n numbers.

23. Identify the errors in the following code segment:

```
fun_alert_box
    {
```

```
document.write("Hi!, I am from a function);+1
    }
```

24. What do you understand by nested function?

Multiple Choice Questions

1. The file extension for external JavaScript file is —
   (i) .jscript
   (ii) .js
   (iii) .jav
   (iv) .java
2. Why comments are used within JavaScript programs?
   (i) To tell the browser there is HTML in our JavaScript code.
   (ii) Because HTML does not have its own command for comments.
   (iii) To explain what a script does.
   (iv) None of the above.
3. The original name of JavaScript was —
   (i) JavaScript
   (ii) LiveScript
   (iii) WireScript
   (iv) ECMAScript
4. Which of following is not a valid expression?
   (i) 7.5
   (ii) b + a = c
   (iii) {feet:10, inches:5}
   (iv) [2,3,6,9,5,7]
5. What will be the value of ***res*** in the following expression?
   var res = "70" + 25
   (i) 95
   (ii) 7025
   (iii) 25
   (iv) No output due to error.
6. What will be the value returned by expression "India" < "bharat"?
   (i) 0
   (ii) true
   (iii) false
   (iv) Invalid expression
7. ***delete*** operator is used to delete the
   (i) .js files.
   (ii) cookies.
   (iii) created objects.
   (iv) functions.
8. To which object does the confirm() method belongs to?
   (i) document
   (ii) window
   (iii) frame
   (iv) date
9. Which of the following statement is used to repeat execution of block of the statements?
   (i) if...else...
   (ii) continue

(iii) while
(iv) switch

10. What will be value of sum after execution of the statement?
for(i=1, sum=0; i<=5; i++) sum+=i;
(i) 13
(ii) 15
(iii) 25
(iv) Error in statement.

11. Which of the loop executes a block of code at least once?
(i) while
(ii) for
(iii) do...while
(iv) for/in

12. Which of the following loop statement is used to iterate a specified variable over all the properties of an object?
(i) for
(ii) do...while
(iii) for/in
(iv) none of these

Activities

- Write an HTML document using JavaScript code, which displays your name, class and name of the school.
- Write an HTML document using JavaScript to change background's colour in your web page randomly. (Hint: use document.bgcolor property and Math.random() function.)
- Write a program to check whether the user is eligible to cast the vote. The user must check following two conditions for casting the vote:
  (a) the age should be equal to or greater than 18 years and
  (b) the nationality should be Indian.
- Write a simple Quiz program that asks the user 5 questions. Alert about answers of the questions and give the user a score at the end of the Quiz.
- Write an HTML document using JavaScript code to validate the form in your website.

References

JavaScript – The definitive Guide - David Flanagan, O'REILLY

SAMS Teach Yourself JavaScript in 24 hours – Michael Moncur, Sams Publishing

JavaScript – A Beginner's Guide – John Pollock, McGraw-Hill.

http://www.webreference.com/programming/javascript

http:// www.webdevelopersnotes.com/tutorials/javascript

http://www.javascriptkit.com

http://www.w3schools.com/js

http://sun.com

http://www.mozilla.org