Basics of Javascript Programming

Syllabus :

1.1 Features of JavaScript

1.2 Object Name, Property, Method, Dot syntax, Main event.

1.3 Values and Variables

1.4 Operators and Expressions- Primary Expressions, Object and Array initializers, function definition expression, property access expressions, invocation expressions.

1.5 if Statement, if. .. else, if..elseif, nested if statement.

1.6 Switch ... case statement

1.7 Loop statement - for loop, for...in loop, while loop, do... while loop, continue statement.

1.8 Querying and setting properties and deleting properties , property getters and setters.

1.1 What is JavaScript Programming ?

- JavaScript is popular, light weight and an Open Source Client Side Language supported by all browsers.
- JavaScript is a dynamic computer programming language. When Javascript is applied with an HTML document, it can provide dynamic interactivity on website.
- JavaScript is also being used in many Game development and Mobile Application development.

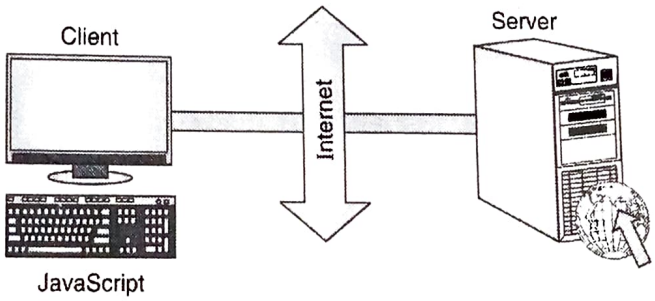


Fig. 1.1.1 : Client side scripting

- It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.
- It is an interpreted programming language with object-oriented capabilities.

1.1.1 Features of JavaScript

Q. Enlist features of JavaScript.

- JavaScript is **an object-based** scripting language.

- Giving the user **more control** over the browser.
- It **Handling** dates and time.
- It Detecting the user's browser and OS,
- It is light weighted.
- JavaScript is a scripting language and it is not java.
- JavaScript is interpreter based scripting language.
- JavaScript is case sensitive.
- JavaScript is object based language as it provides predefined objects.
- Every statement in javascript must be terminated with semicolon (;).
- Most of the javascript control statements syntax is same as syntax of control statements in C language.
- An important part of JavaScript is the ability to create new functions within scripts. Declare a function in JavaScript using **function** keyword.

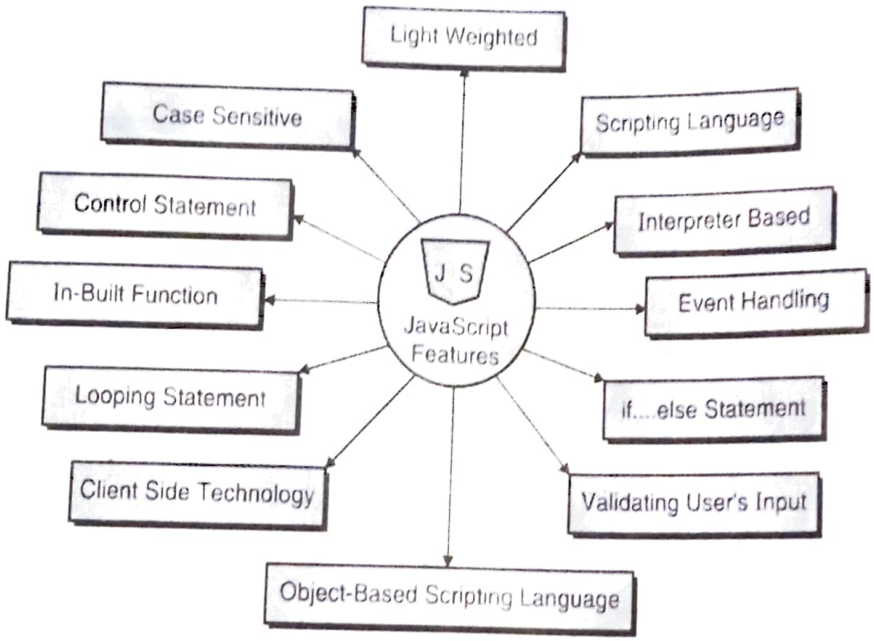


Fig. 1.1.2 : Features of Javascript

1.1.2 Limitations of JavaScript

- Client-side JavaScript does not allow the reading or writing of files.
- It cannot be used for networking applications because there is no such support available.
- It doesn't have any multithreading or multiprocessor capabilities.

1.1.3 How to Write a JavaScript ?

Q. Explain <script>tag with example.

Javascript is implemented by adding <script> ... </script> tag within HTML tag in a web page. The <script> tag tells the browser that a script is coming—not HTML hence browser processes the scripts separately.

<Script> tag can be inserted within web page by following ways :

Script in <head> section -

```
<head>
<script ...>
//Javascript code
</script>
    </head>
```

Script in <body> section -

```
<body>
<script ...>
//Javascript code
</script>
</body>
```

<script> tag has following attributes :

- **Language** – The language attribute is set to value "Javascript", which tells the browser that the scripting language is JavaScript.
- **Type** - The type attribute tells the browser that the script is in plain text and the text is organized in the format of a JavaScript. This is for browser information- how to read the JavaScript code.

```
<script language="Javascript" type="text/javascript">
    document.write("Hello, world!")
</script>
```

1.2 Object Name, Property, Method, Dot Syntax, and Main Event

Q. Explain the terms : (i) Method (ii) Property (iii) Event (iv) Object Name

1.2.1 Object Name

Javascript is object oriented programming language means everything is in the form of objects. A Javascript object is a collection of named values and these named values are properties of object.

Object Name :

- A web page contains various objects; some of them are of same type whereas some of them are of different type of object.
- Each object should be uniquely identified by a name or IDs in web page to distinguished between them.
- Javascript supports various objects like document, form, button, window etc.
- In Javascript, object can be created with curly brackets { } with an optional list of properties.

```
<script language="Javascript" type="text/javascript">
var student = {
  name: "Vijay",
  age: 21,
  year: "TY"
};
</script>
```

- In Javascript all built-in and user defined objects are descendants of an object called Object. The **new** keyword is used to create instance of an object.

```
<script language="Javascript" type="text/javascript">
var student =new Object();
student.name="abc";
student.age=21;
</script>
```

1.2.2 Property

- A property is a value associated with an object.
- Every object has its own set of properties for example window has properties like width, height, background color.
- A property is a **"key: value"** pair,
  - **key** (property name) is always a string,
  - **value** (property value) can be any data type, like strings, numbers, Booleans or data type like arrays, functions, and other objects.

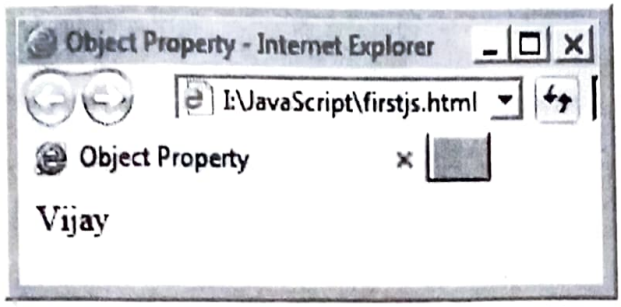
Program 1.2.1 : Write a JavaScript code to declare property.

Solution :

```
<html>
<head>
<title>Object Property</title>
<script language="Javascript" type="text/javascript">
var student= {
  name: "Vijay",
  age: 21,
  year: "TY"
};
document.write(student.name);
</script>
```

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

Output :



In the above code we have declare 3 properties using key:value pairs –

- First property is name with value "Vijay"
- Second property is age with value 21
- Third property year with value "TY"

It is possible to access the properties of objects by two ways – student["age"] or student.age

1.2.3 Method

- A method is a set of instructions performed by an object when it receives a message.
- On the form when you click a Submit button, the form is submitted to the server-side application. It means, clicking the Submit button causes the button to process a some set of instructions (method).
- The kinds of methods that are used differ, depending on the type of object to which they're attached.

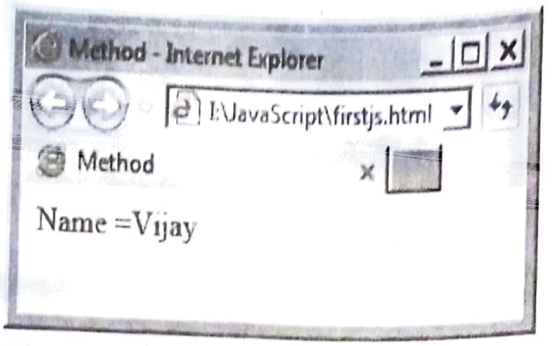
Program 1.2.2 : Write a JavaScript code to illustrate the use of method.

Solution :

```
<html>
<head>
<title>Method</title>
<script language="Javascript" type="text/javascript">
var student= {
   name: "Vijay",
   age: 21,
   year: "TY",
   display: function()
     {
```

```
    return(this.name);
    }
};
document.write("Name =" +student.display());
</script>
</head>
<body>
</body>
</html>
```

Output :



'this' is a special keyword of Javascript, that you can use within a method to refer to the current object.

1.2.4 Dot Syntax

- One can access an object's properties and methods by using the dot syntax along with the object name and its property or method.
- For example, the write method for a document :

  document.write()
- A dot separates the name of the object from the property or method.
- The first part is the name of the object (document) and second part is either a property or method (write) of the object.

1.2.5 Main Event

- In Javascript, an event is the way to start executing your code like om mouse click, button click etc..
- JavaScript reacts to events with the help of event handling.
- **Event handling** is the execution of code on occurrence of event. For example- when user click on Submit button then the event handler for that click event should process the information.
- The functions written for event handling are known as **event handler**.

1.3 Values and Variables

JavaScript is a dynamic or loosely-typed language because a variable can hold value of any data type at any point of time. **var** keyword is used to specify the data type. It can hold any type of values such as numbers, strings etc. Following are the values used by Javascript

- **Number -** A number is a numeric value.

  For example : var a=20;

- **String -** A *string* is a sequence of characters that is enclosed within quotation marks

  For example : var city="Pune";

- **Boolean -** A Boolean is a value - either false or true.

  For example : var b=true;

- **Null –** Null value means no value at all.

  For example : var i= null;

- **Objects -** An object is a instance through which we can access members.

  For example : var person = {firstName:"John", lastName:"Doe"};

1.4 Operators and Expressions

Primary expressions in JavaScript are constant or literal values, certain language keywords, and variable references.

1.4.1 Operators

Q. What is Operator in Javascript? Explain any two operator in javascript.

JavaScript includes following categories of operators.

1.	Arithmetic Operators	2.	Comparison Operators
3.	Logical Operators	4.	Assignment Operators
5.	Conditional Operators		

1. Arithmetic Operators :

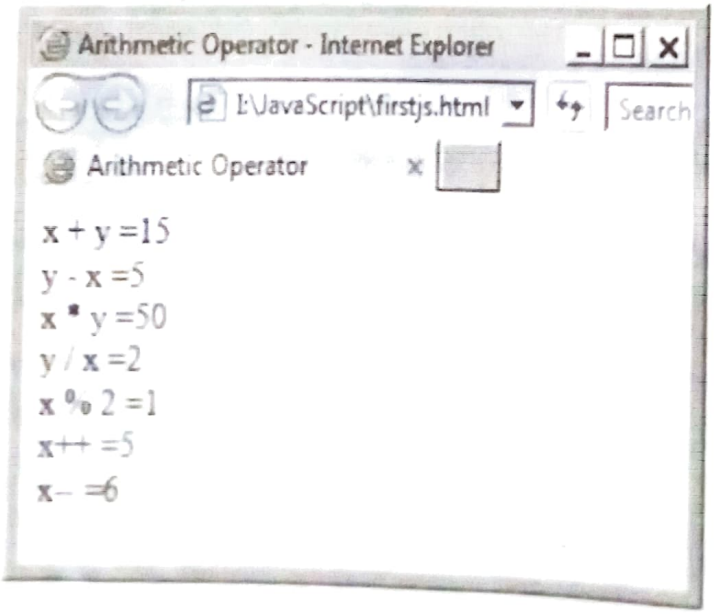
Operator	Description
+	Adds two numeric operands.
-	Subtract right operand from left operand
*	Multiply two numeric operands.
/	Divide left operand by right operand.
%	Modulus operator. Returns remainder of two operands.
++	Increment operator. Increase operand value by one.
--	Decrement operator. Decrease value by one.

Program 1.4.1 : Write a JavaScript code to implement arithmetic operators

Solution :

```
<html>
<head>
<title> Arithmetic Operator </ title>
<script language="javascript" type="text/javascript">
var x = 5, y = 10, z = 15;
document.write("x + y =" + (x + y)); //returns 15
document.write("<br>");
document.write("y - x  =" + (y – x)); //returns 5
document.write("<br>");
document.write("x * y =" + (x * y)); //returns 50
document.write("<br>");
document.write("y / x =" + (y / x)); //returns 2
document.write("<br>");
document.write("x % 2 =" + (x % 2)); //returns 1
document.write("<br>");
document.write("x++ =" + (x++)); //returns 5
document.write("<br>");
document.write("x-- =" + (x--)); //returns 4
</script>
</head>
<body>
</body>
</html>
```

Output :



2. Comparison Operators :

Operators	Description
==	Compares the equality of two operands without considering type.
===	Compares equality of two operands with type.
!=	Compares inequality of two operands.
>	Checks whether left side value is greater than right side value. If yes then returns true otherwise false
Operators	**Description**
<	Checks whether left operand is less than right operand. If yes then returns true otherwise false.
>=	Checks whether left operand is greater than or equal to right operand. If yes then returns true otherwise false.
<=	Checks whether left operand is less than or equal to right operand. If yes then returns true otherwise false.

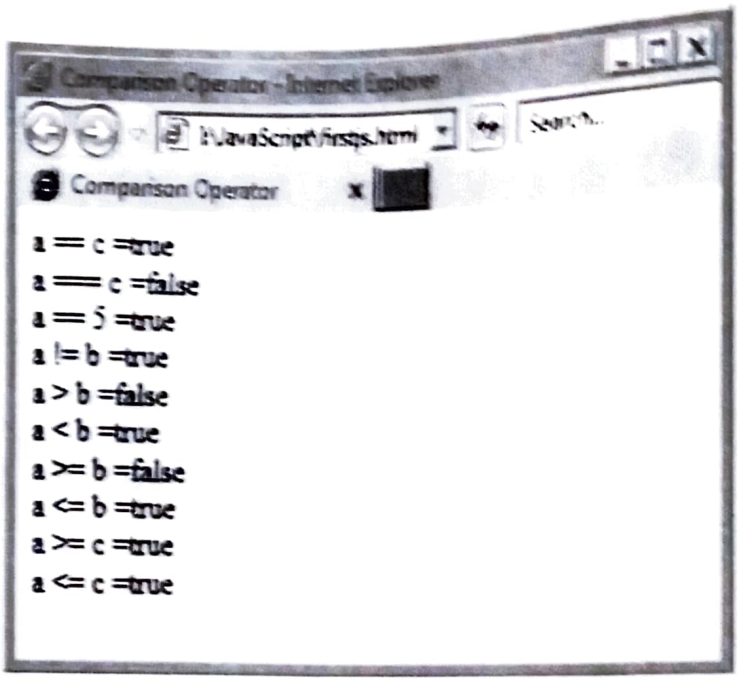
Program 1.4.2 : Write a JavaScript code to illustrate comparison operators.

Solution :

```
<html>
<head>
<title>Comparison Operator</title>
<script language="javascript" type="text/javascript">
var a = 5, b = 10, c = "5";
document.write("a == c ="+(a == c)); // returns true
document.write("<br>");
document.write("a === c ="+(a === c)); // returns false
document.write("<br>");
document.write("a == 5 ="+(a == 5)); // returns true
document.write("<br>");
document.write("a != b ="+(a != b)); // returns true
document.write("<br>");
document.write("a > b ="+(a > b)); // returns false
document.write("<br>");
document.write("a < b ="+(a < b)); // returns true
document.write("<br>");
document.write("a >= b ="+(a >= b)); // returns false
document.write("<br>");
```

```
Client Side Scripting Language ASP.NET
document.write("a <= b = "+(a <= b)); // returns true
document.write("<br>");
document.write("a >= c = "+(a >= c)); // returns true
document.write("<br>");
document.write("a <= c = "+(a <= c)); // returns true
</script>
</head>
<body>
</body>
</html>
```

Output :



3. Logical Operators :

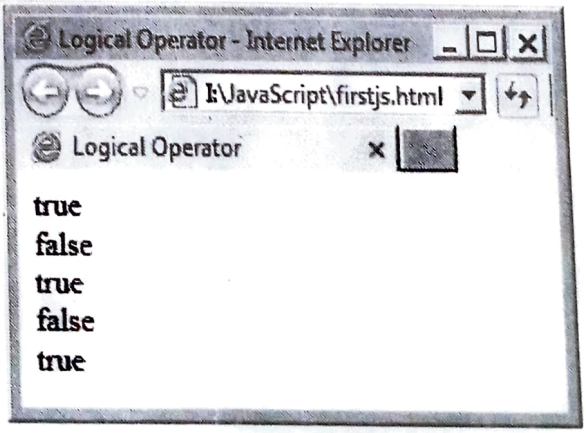
Operator	Description
&&	&& is known as AND operator. It checks whether two operands are non-zero (0, false, undefined, null or "" are considered as zero), if yes then returns 1 otherwise 0.
||	|| is known as OR operator. It checks whether any one of the two operands is non-zero (0, false, undefined, null or "" is considered as zero).
!	! is known as NOT operator. It reverses the boolean result of the operand (or condition)

Program 1.4.3 : Write a JavaScript code to demonstrate logical operators.

Solution :

```
<html>
<head>
<title>Logical Operator</title>
<script language="javascript" type="text/javascript">
var a = 5, b = 10;
document.write((a != b) && (a < b)); // returns true
document.write("<br>");
document.write(((a > b) || (a == b))); // returns false
document.write("<br>");
document.write(((a < b) || (a == b))); // returns true
document.write("<br>");
document.write(!(a < b)); // returns false
document.write("<br>");
document.write(!(a > b)); // returns true
</script>
</head>
<body>
</body>
```

Output :



4. **Assignment Operators :**

Assignment operators	Description
=	Assigns right operand value to left operand
+=	Sums up left and right operand values and assign the result to the left operand.
-=	Subtract right operand value from left operand value and assign the result to the left operand
*=	Multiply left and right operand values and assign the result to the left operand.
/=	Divide left operand value by right operand value and assign the result to the left operand.
%=	Get the modulus of left operand divide by right operand and assign resulted modulus to the left operand

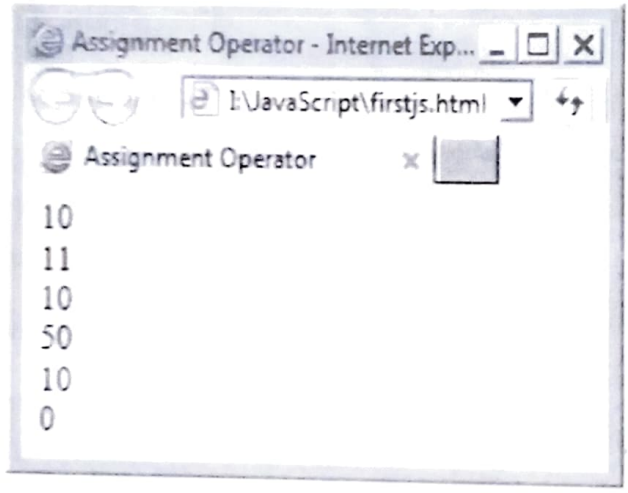
Program 1.4.4 : Write a JavaScript code to implement assignment operators

Solution

```
<html>
<head>
<title> Assignment Operator </title>
<script language="javascript" type="text/javascript">
var x = 5, y = 10, z = 15;

x = y;  // x would be 10
document.write(x);
document.write("<br />");
x += 1;  // x would be 11
document.write(x);
document.write("<br />");
x = 1;  // x would be 10
document.write(x);
document.write("<br />");
x *= y;  // x would be 50
document.write(x);
document.write("<br />");
x /= y;  // x would be 10
document.write(x);
document.write("<br />");
```

```
x %= 2; //x would be 0
document.write(x);
document.write("<br>");
< script>
< /head>
<body>
</body>
</html>
```

Output :



5. Ternary Operator :

Syntax :

```
<condition> ? <value1> : <value2>;
```

Ternary operator starts with conditional expression followed by ? operator. Second part (after ? and before : operator) will be executed if condition turns out to be true. If condition becomes false then third part (after :) will be executed.

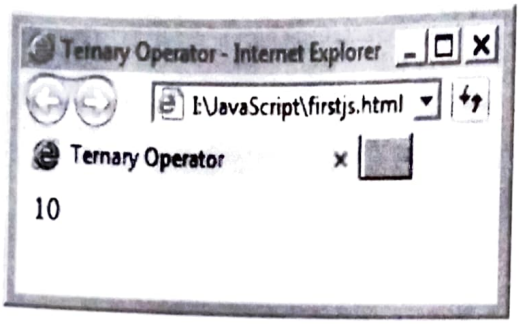
Program 1.4.5 : Write a JavaScript code to illustrate the use of ternary operator

Solution :

```
<html>
<head>
<title>Ternary Operator< title>
<script language="javascript" type="text/javascript">
var a = 10, b = 5;
var c = a > b? a : b;    value of c would be 10
```

```
document.write(c);
</script>
</head>
<body>
</body>
</html>
```

Output :



6. Bitwise Operators :

Bit operators work on 32 bits numbers. Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 7	~0111	1000	8
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

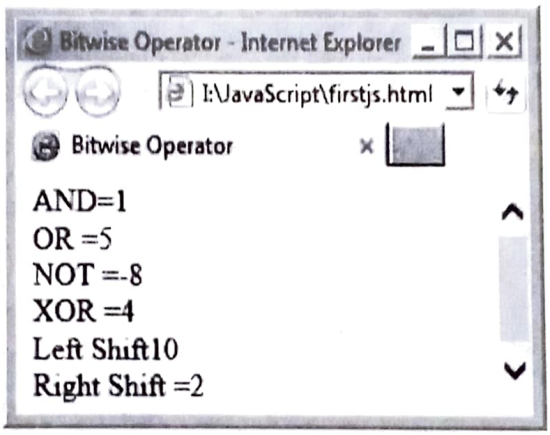
Program 1.4.6 : Write a JavaScript code to implement bitwise operators.

Solution :

```
<html>
<head>
<title>Bitwise Operator</title>
<script language="javascript" type="text/javascript">
document.write("AND="+(x = 5 & 1));
document.write("<br>");
document.write("OR ="+(x = 5 | 1));
```

```
document.write("<br>");
document.write("NOT ="+(x = ~7));
document.write("<br>");
document.write("XOR ="+(x = 5 ^ 1));
document.write("<br>");
document.write("Left Shift"+ (x = 5 << 1));
document.write("<br>");
document.write("Right Shift =" + (x = 5 >> 1));
</script>
</head>
<body>
</body>
</html>
```

Output :



1.4.2 Expressions

Primary expressions :

- Primary expressions are the simplest expressions which do not include any simpler expressions.
- In Javascript, the primary expressions are constant or literal values, keywords, and variable references.

Example :

Literals – these are constant values	
3.2	Number Literal
"Welcome"	String Literal
/pattern/	Regular Expression Literal

Reserve Keywords	
true	Boolean true value
false	Boolean false value
this	Current object
null	Null value

Variable Refernces	
x	Value of variable x
undefined	If no variable with specified name exists, the expression evaluates to the undefined value

Object and Array initializers :

Q. Explain how object and array initializers work.

- Object & array initializers are expressions whose values are created object or array. These are also known as **"object literals"** and **"array literals"**.
- These expressions are not primary expressions, because they include a number of sub-expressions which specifies properties and element values.
- **Array initializer** is a list of expressions separated by comma within square brackets. The value of an array initializer is a newly created array.

[ ]	Empty Array – no elements
[1+2, 3+4]	2-element Array
[[1,2,3], [4,5,6], [7,8,9]]	Nested Array
[1, , , ,5]	Array with five elements including three undefined elements

- **Object initializer** is list of elements separated by comma within curly brackets. These expression contains subexpressions with property name and colon.

var a = { i:2, j:4 };	An object with properties
var a = { };	Empty Object – no properties
a.i=5; a.j =9;	An object a has some properties

Function definition expression :

Q. Explain function definition expression with example.

- It is a **"Functional Literal"** to define a function in Javascript. The value of these expressions is newly created functions.
- **'function'** keyword is used to define a function with list of zero or more identifiers (parameters) separated by comma in the parentheses with Javascript code in curly brackets.

var addition = function(x,y) {return x+y }	This function returns addition of 2 values

- Functions can also be created by using function statement rather than a function expression.

Property access expression :

Q. Explain property access expression and invocation expression.

- A property access expression evaluates to the value of an object property or an array element. Syntax for property access :

expression . identifier	– Expression followed by a period and an identifier – Here, expression specifies an Object and identifier specifies the name of particular property.
expression [expression]	– Follows first expression with another expression in square brackets – It specifies the name of the desired property of the index of the desired array element

Examples :

var o = {x:1,y:{z:3}};	An example object
var a = [o,4,[5,6]];	An example array that contains the object
o.x	property x of expression o
o["x"]	property x of object o
a[2]["1"]	element at index 1 of expression a[2]
a[0].x	property x of expression a[0]

Invocation expression :

Q. Explain property access expression and invocation expression.

- It is a Javascript syntax for executing functions or methods. It starts with a function expression which identifies the function to be called.
- The function expression is followed by an opening parenthesis, a list of zero or more arguments separated by comma, and a closing parenthesis.

f(0)	f is the function expression and 0 is the argument expression.
Math.max(x,y)	Math.max is the function which returns largest of x and y.
a.sort()	a.sort is the function with no arguments

- When these type of expressions are evaluated, the function expression is evaluated first then the argument expressions are evaluated to produce a list of argument values.

1.5 Conditional Statement

Q. What is condional statement? Explain if..else and if...else if statement with example.

Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform another action.

1.5.1 If Statement

There are mainly three types of conditional statements in JavaScript.

Syntax :

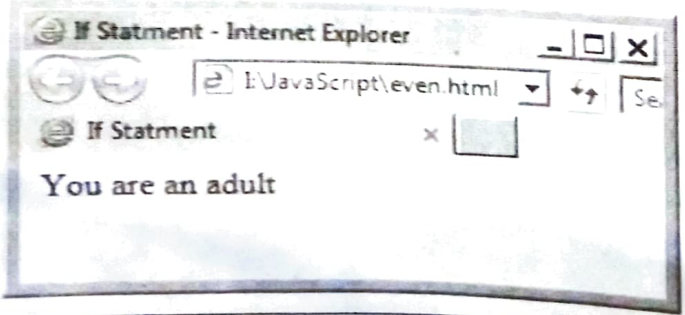
```
if (condition)
{
lines of code to be executed if condition is true
}
```

Program 1.5.1 : Write a Javascript code to demonstrate 'if' statement.

Solution :

```
<html>
<head>
    <title>If Statement</title>
    <script type="text/javascript">
        var age = 19;
        if(age>=18)
        document.write("You are an adult <br />");
        if(age<18)
        document.write("You are NOT an adult <br />");
    </script>
</head>
<body>
</body>
</html>
```

Output :



1.5.2 If...else Statement

Syntax :

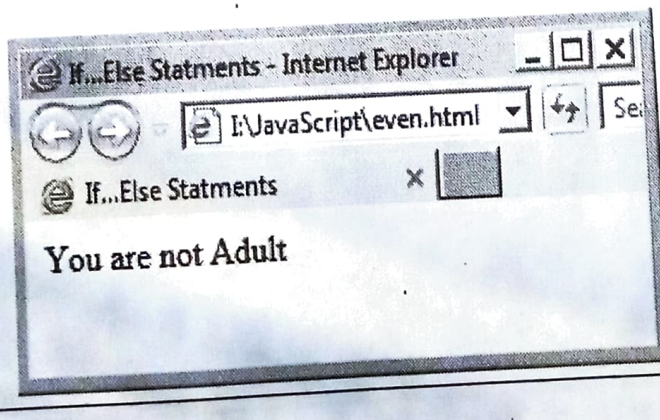
```
if (condition)
{
lines of code to be executed if the condition is true
}
else

{
lines of code to be executed if the condition is false
}
```

Program 1.5.2 : Write a Javascript code to demonstrate 'if .... else' statement.

Solution :

```
<html>
<head>
    <title>If...Else Statements</title>
    <script type="text/javascript">
        // Get the current hours
        var age=17;
        if(age<=18)
        document.write("You are not Adult<br />");
        else
        document.write("You are Adult<br />");
    </script>
</head>
<body>
</body>
</html>
```

Output :



1.5.3 If...else if Statement

Syntax :

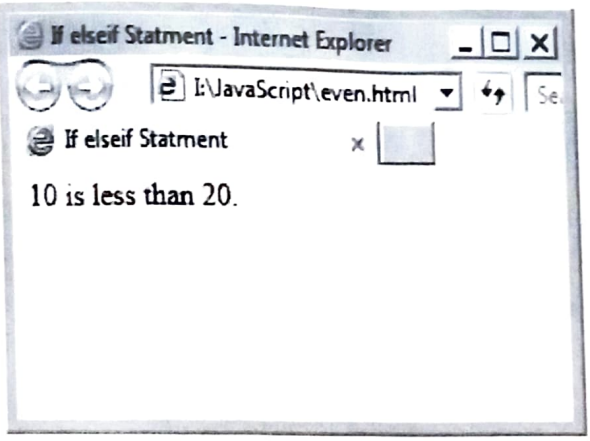
```
if (condition1)
{
    lines of code to be executed if condition1 is true
    }
    else if(condition2)
    {
    lines of code to be executed if condition2 is true
    }
    else
    {
    lines of code to be executed if condition1 is false and condition2 is false
}
```

Program 1.5.3 : Write a Javascript code to demonstrate the use of 'if .... elseif' statement.

Solution :

```
<html>
<head>
<title> If elseif Statement</title>
<script language="javascript" type="text/javascript">
var one =10;
var two = 20;
if (one == two) {
document.write(one + " is equal to " + two + ".");
}
else if (one<two)
{
document.write(one + " is less than " + two + ".");
}
else {
document.write(one + " is greater than " + two + ".");
}
</script>
</head>
<body>
</body>
</html>
```

Output :



1.5.4 Nested if Statement

A nested if is an if statement that is the target of another if or else. Nested if statements means an if statement inside an if statement. Yes, JavaScript allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement.

Syntax :

```
if (condition1)
{
  lines of code to be executed if condition1 is true
  if (condition2)
  {
    lines of code to be executed if condition2 is true
  }
}
```

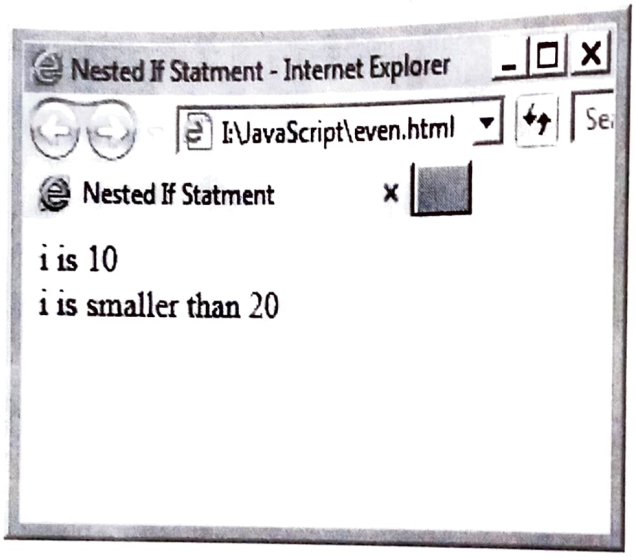
Program 1.5.4 : Write a Javascript code to demonstrate nested if...else statement.

Solution :

```
<html>
<head>
<title>Nested If Statement</title>
<script language="javascript" type="text/javascript">
var i = 10;


if (i == 10) {
    document.write("i is 10<br>");
```

```
if (i < 20) {
  document.write("i is smaller than 20 <br>"); }
else{
  document.write("i is greater than 20"); }
}
</script>
</head>
<body>
</body>
</html>
```

Output :



1.6 Switch Case Statement

Q. Explain switch...case statement in javascript with example.

The switch case statement in JavaScript is used for decision making purposes. The switch case statement is a multiway branch statement. It provides an easy way to execution different parts of code based on the value of the expression.

Syntax :

```
switch (expression)
{
  case value1:
```

```
    statement1;
    break;
  case value2:
    statement2;
    break;
  .
  .
  case valueN:
    statementN;
    break;
  default:
    statementDefault;
}
```

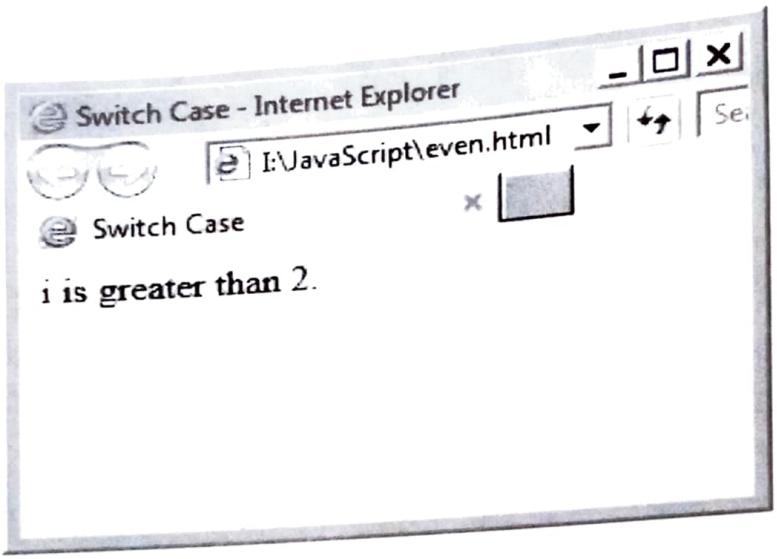
Program 1.6.1 : Write a code to illustrate use of Switch case in Javascript.

Solution :

```
<html>
<head>
<title>Switch Case</title>
<script language="javascript" type="text/javascript">
var i = 3;
   switch (i)
  {
    case 0:
      document.write("i is zero.");
      break;
    case 1:
      document.write("i is one.");
      break;
    case 2:
      document.write("i is two.");
      break;
    default:
```

```
        document.write("i is greater than 2.");
    }
</script>
</head>
<body>
</body>
</html>
```

Output :



1.7 Loop Statement

Q. What is Looping Statement?

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

Following are the four types of loops in JavaScript.

1.7.1 for Loop

Q. Explain for loop with example.

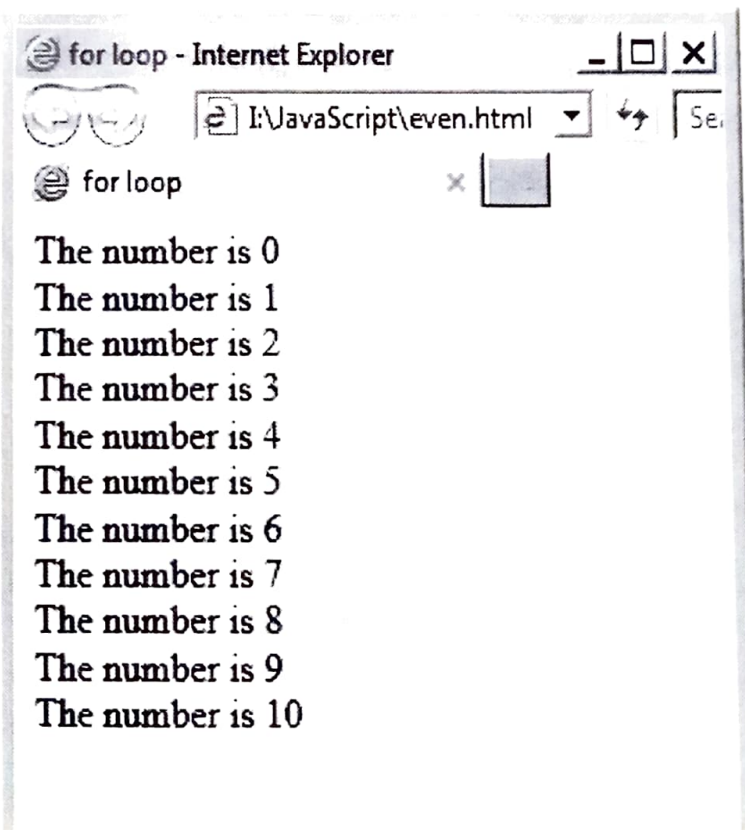
Syntax :

```
for(initial condition; terminating condition; stepping condition)
{
lines of code to be executed
}
```

Program 1.7.1 : Develop a Javascript to implement for... loop.

Solution :

```
<html>
<body>
<head>
<title> for loop </title>
<script type="text/javascript">
var i ;
for (i=0;i<=10;i++)
{
document.write("The number is " + i)
document.write("<br />")
}
</script>
</head>
</body>
</html>
```



1.7.2 while Loop

The "while loop" is executed as long as the specified condition is true.

Syntax :

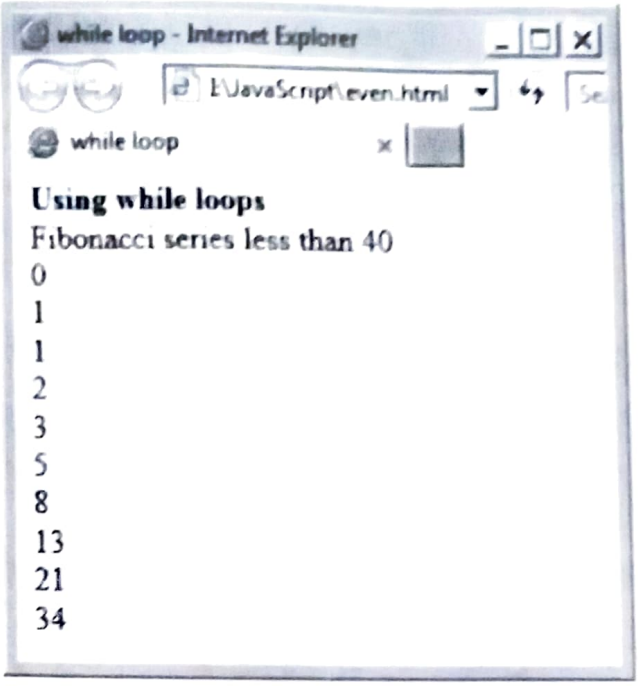
```
while(condition)
{
lines of code to be executed
}
```

Program 1.7.2 : Develop a Javascript to implement while... loop.

Solution :

```
<html>
<head>
<title> while loop </title>
    <script type="text/javascript">
        document.write("<b>Using while loops </b><br />");
        var i = 0, j = 1, k;
        document.write("Fibonacci series less than 40<br />");
        while(i<40)
        {
            document.write(i + "<br />");
            k = i+j;
            i = j;
            j = k;
        }
    </script>
</head>
<body>
</body>
</html>
```

Output :



1.7.3 do...while Loop

The do...while loop is very similar to while loop. The only difference is that in do...while loop, the block of code gets executed once even before checking the condition.

Syntax :

```
do
{
block of code to be executed
} while (condition)
```

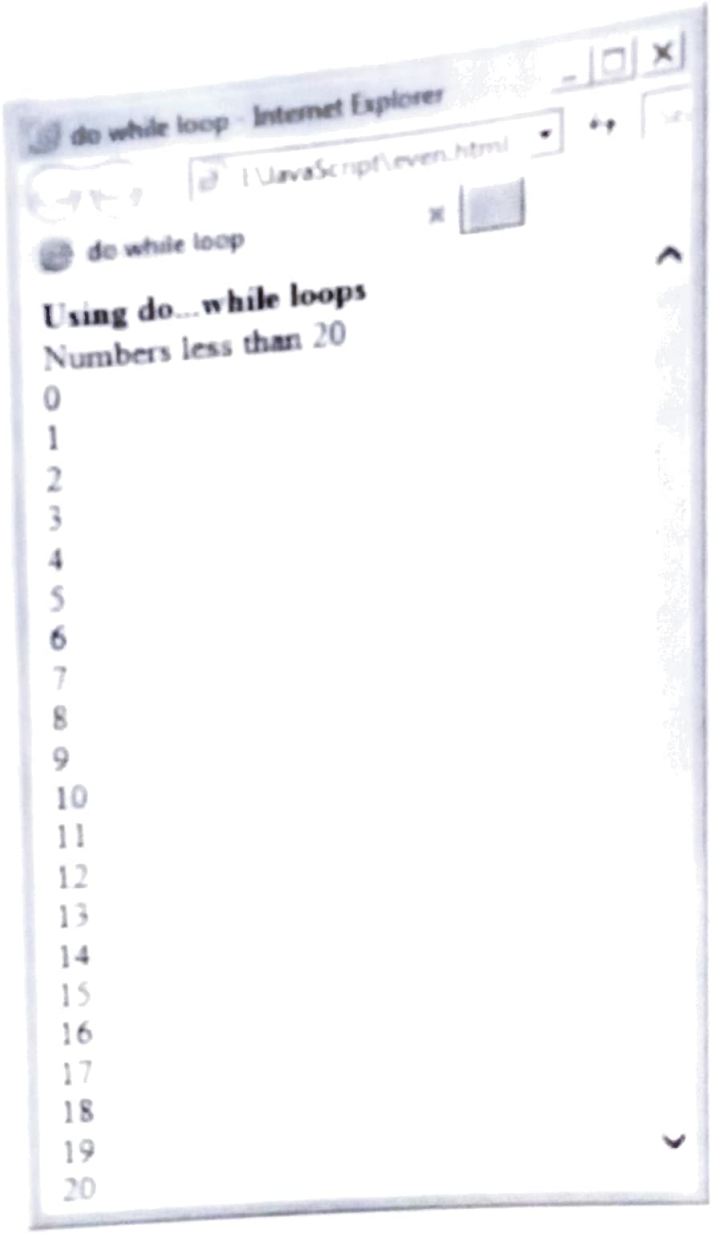
Program 1.7.3 : Develop a Javascript to implement do...while loop.

Solution :

```
<html>
<head>
<title> do while loop </title>
    <script type="text/javascript">
        document.write("<b>Using do...while loops </b><br />");
        var i = 0;
        document.write("Numbers less than 20<br />");
        do
        {
            document.write(i + "<br />");
```

```
    i++;
  }while(i<20);
</script>
</head>
<body>
</body>
</html>
```

Output :



1.7.4 Break and Continue

The **continue** statement terminates execution of the statements in the current loop and continues execution of the loop with the next iteration.

Program 1.7.4 : Develop a Javascript code to illustrate use of continue statement.

Solution :

```
<html>
<head>
<script language="javascript" type="text javascript">
var i=0;
while(i<5)
{
i++;
if(i==4)
{
continue;
}
document.write(i);
}
document.write("<br> Exit from loop...");
</script>
</head>
<body>

</body>
</html>
```

Output :

```
- 1 2 3 5
        Exit from loop ...
```

The **break** statement stops the execution of a loop entirely.

Program 1.7.5 : Develop a Javascript code to illustrate use of break statement.

Solution :

```
<html>
<head>
<script language="javascript" type="text/javascript">
var i=0;
while(i<5)
{
i++;
```

```
if(i==4)
{
break;
}
document.write(i);
}
document.write("<br> Exit from loop...");
</script>
</head>
<body>
</body>
</html>
```

Output :

```
1 2 3
        Exit from loop ...
```

1.8 Querying, Setting Properties and Deleting Properties

1.8.1 Querying and Setting Properties

Q. How to set and delete properties with example.

- The dot (.) operator or square brackets ([ ]) are used to obatined values of properties.

  var price = book.price; // Get the "price" property of the book.

  var name = person.lastname // Get the "lastname" property of the person.

- The left-hand side should be an expression whose value is an object. If using the dot operator, the right-hand must be a simple identifier that names the property.

- If using square brackets, the value within the brackets must be an expression that evaluates to a string that contains the desired property name

  var author = book["author"] // Get the "author" property of the book.

- Use dot or square brackets to create or set a property but place the properties to the left-hand side of an assignment expression:

  book.price = 300; // Create an "price" property of book.

  person["FirstName"] = "Jhon"; // Set the "FirstName" property.

- If the square bracket notation are used 1then the expression inside the square brackets must evaluate to a string.

1.8.2 Deleting Properties

Q. How to set and delete properties with example.

- The delete keyword is used to delete both the value of the property and property itself.

  delete person.age; // The person object now has no age property

  delete person["FirstName"]; // The person object has no FirstName property.

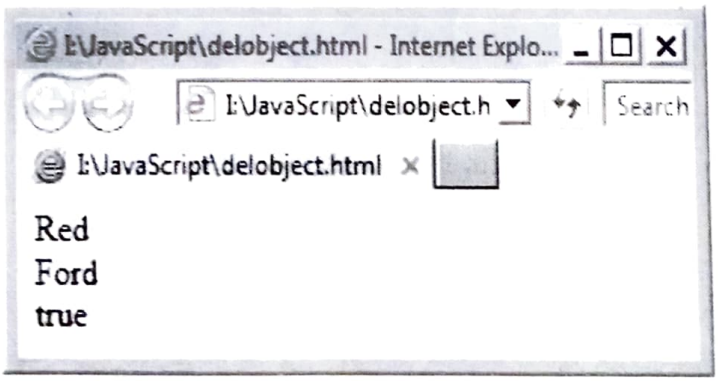
- The delete property returns true if deleted successfully.

Program 1.8.1 : Write a Javascript to delete property of Object.

Solution :

```
<html>
<head>
<script language="javascript" type="text/javascript>
var car ={
color: "Red",
brand: "Ford"
};
document.write(car.color + "<br>" +car.brand + "<br>");
document.write(delete car.brand);
</script>
</head>
<body>
</body>
</html>
```

Output :



1.8.3 Property Getters and Setters

Q. Explain property setter and property getter method with example.

Javascript supports are two kinds of properties - **Data property** and **Accessor properties**.

The Accessor properties are represented by "**getter**" and "**setter**" methods for the fullName property

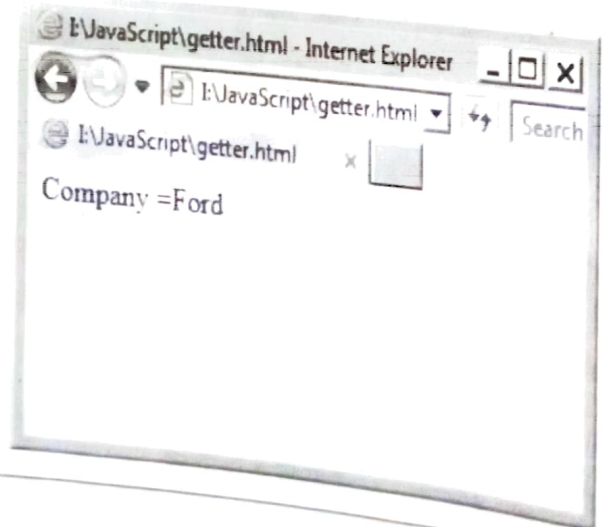
- When the property is accessed, the return value from the getter is used.
  - **get** – a function without arguments, that works when a property is read

Program 1.8.2 : Write a code to illustrate the use of getter method in Javascript.

Solution :

```
<html>
<head>
<script language="javascript" type="text/javascript">
var car = {
 color: "Red",
 brand : "Ford",
  get company() {
   return this.brand;
  }
};
// Display data from the object using a getter:
document.write("Company ="+car.company);
</script>
</head>
<body>
</body>
</html>
```

Output :



- When a value is set, the setter is called and passed the value that was set.
  - **set** – a function with one argument, that is called when the property is set

Program 1.8.3 : Write a code to illustrate the use of setter method in Javascript.

Solution :

```
<html>
<body>
<script language="javascript" type="text/javascript">
// Create an object:
var car = {
 color: "Red",
 brand : "Ford",
 set company(value) {
   this.brand=value;
 }
};
document.write("Company ="+car.brand +"<br>");
car.company="Maruti";
// Display data from the object using a getter:
document.write("Company ="+car.brand);
</script>
</body>
</html>
```

Output :