Client Side Scripting Language

Prepared as per MSBTE I Schema syllabus.

Author:

"Prof. Vishal Jadhav"

[BE in Computer Engineering and Having 5 years of IT industry experience (Worked in Capgemini, Amdocs & currently working in TIBCO Software]

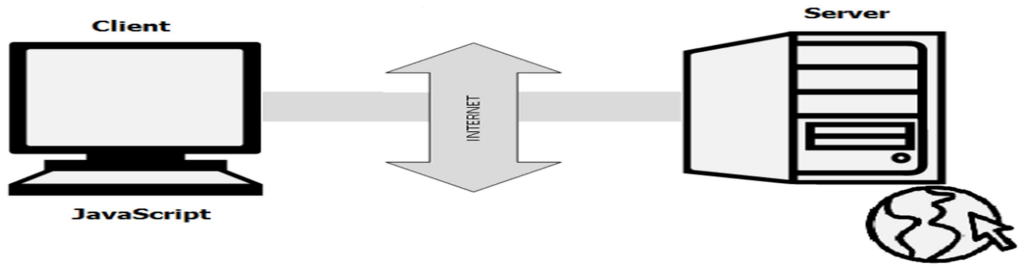
Address:

VJTech Computer Academy, College – Awasari road, opposite of Sahyadri girls hostel, Awasari(KH),Pune.

Contact No: +91-9730087674, Email id: vjtechacademy@gmail.com)]

❖ What is JavaScript?

- JavaScript is an open source & most popular client side scripting language supported by all browsers.
- JavaScript is a dynamic computer programming language.
- JavaScript is a lightweight, interpreted programming language.
- It is an interpreted programming language with object-oriented capabilities.
- JavaScript is a very powerful client-side scripting language.
- JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- You can make your webpage more lively and interactive, with the help of JavaScript.
- JavaScript is also being used widely in game development and Mobile application development.



❖ Javascript History:

- JavaScript was developed by Brendan Eich.
- It was developed in 1995.
- JavaScript language was initially called as LiveScript. Later its name got changed to JavaScript.

❖ Features of JavaSript:

1. JavaScript is an object-based scripting language.
2. Giving the user more control over the browser.
3. It Handling dates and time.
4. It detecting the user's browser and OS.
5. It is light weighted.
6. JavaScript is a scripting language and it is not java.
7. JavaScript is interpreter based scripting language.
8. JavaScript is case sensitive.
9. JavaScript is object based language as it provides predefined objects.
10. Every statement in JavaScript must be terminated with semicolon (;).
11. Most of the JavaScript control statements syntax is same as syntax of control statements in C language.

❖ Advantages of JavaScript:

1. **Less server interaction –** you can validate user input before sending the web page to the server. This saves server traffic, which means less loads on your server.
2. **Immediate feedback to the visitors** − they don't have to wait for a page reload to see if they have forgotten to enter something.
3. **Increased interactivity** − you can create interfaces that more interactive.
4. **Richer interfaces** − you can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

❖ Limitations of JavaScript:

1. Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
2. JavaScript cannot be used for networking applications because there is no such support available.
3. JavaScript doesn't have any multi-threading or multiprocessor capabilities.

JavaScript Tag:

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.
- You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.
- A simple syntax of your JavaScript will appear as follows.

```
<script ...>

   JavaScript code

</script>
```

- The script tag takes two important attributes :

1. **Language** − this attribute specifies what scripting language you are using. Typically, its value will be JavaScript

2. **Type** – this attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

- Example:

```
<script language = "javascript" type = "text/javascript">

   JavaScript code

</script>
```

- First JavaScript code:

```
<html>
   <body>
      <script language = "javascript" type = "text/javascript">
            document.write("Hello World!")
      </script>
   </body>
</html>
```

❖ Comments in JavaScript:

- JavaScript supports both C-style and C++-style comments.
  1. Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
  2. Any text between the characters /* and */ is treated as a comment. This will cover multiple lines.
  3. JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment.
  4. The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.
- Example: The following example shows how to use comments in JavaScript.

```
<script language = "javascript" type = "text/javascript">
   <!--
      // This is a comment. It is similar to comments in C++
```

```
      /*
      * This is a multi-line comment in JavaScript
      * It is very similar to comments in C Programming
      */
   //-->
</script>
```

❖ JavaScript Data Types:

- JavaScript provides different data types to hold different types of values.
- There are two types of data types in JavaScript.
  1. Primitive data type
  2. Non-primitive data type

Primitive data type:

- There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

Non-Primitive data types:

- The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

❖ JavaScript Variable:

- JavaScript variable is a name of storage location.
- It is used to store the value.
- Before using a variable, you first need to declare it.
- You have to use the keyword **var** to declare a variable.
- **Syntax:**

```
var name;
```

- **Example:**

```
1) var name = "John";
2) var a,b,c;
3) var sum;
   sum=0;
```

- **The general rules for constructing names for variables:**
  1. Names can contain letters, digits, underscores, and dollar signs.
  2. Names must begin with a letter
  3. Names can also begin with $ and _
  4. Names are case sensitive (y and Y are different variables)
  5. Reserved words (like JavaScript keywords) cannot be used as names

- **There are two types of variables in JavaScript.**
  1. Local variable - A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.
  2. Global variable - A global variable has global scope which means it can be defined / accessed anywhere in your JavaScript code.

- Example

```
<html>
  <body>
          <script>
                  var no=4;        //global variable
                  function CalcSquare()
                  {
                          var square;      //local variable
                          square=no*no;
                          document.writeln("Square of Number="+square);
                  }
                  CalcSquare();
          </script>
  </body>
</html>
```

❖ JavaScript Reserved Words:

- A list of all the reserved words in JavaScript is given in the following table.
- They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

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

❖ Operators and Expressions:

- Operators are the symbols which indicate operations to be performed.
- Operands are the variable on which operation to be performed.
- The systematic arrangement of operators and operands is known as Expression.
- There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Conditional Operators

Arithmetic Operators:

- Arithmetic operators are used to perform arithmetic operations on the operands.
- The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

- Example

```
<html>
   <body>
      <script type = "text/javascript">
            var a = 33;
```

```
            var b = 10;
            var c = "Test";
            var linebreak = "<br>";

            document.write("a + b = ");
            result = a + b;
            document.write(result);
            document.write(linebreak);

            document.write("a - b = ");
            result = a - b;
            document.write(result);
            document.write(linebreak);

            document.write("a / b = ");
            result = a / b;
            document.write(result);
            document.write(linebreak);

            document.write("a % b = ");
            result = a % b;
            document.write(result);
            document.write(linebreak);

            document.write("a + b + c = ");
            result = a + b + c;
            document.write(result);
            document.write(linebreak);

            a = ++a;
            document.write("++a = ");
            result = ++a;
            document.write(result);
            document.write(linebreak);

            b = --b;
            document.write("--b = ");
            result = --b;
            document.write(result);
            document.write(linebreak);
      </script>
   </body>
</html>
```

Comparison Operators:

- The JavaScript comparison operator compares the two operands.
- The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

- Example

```
<html>
   <body>
      <script type = "text/javascript">
            var a = 10;
            var b = 20;
            var linebreak = "<br>";

            document.write("(a == b) => ");
            result = (a == b);
            document.write(result);
            document.write(linebreak);

            document.write("(a < b) => ");
            result = (a < b);
            document.write(result);
            document.write(linebreak);

            document.write("(a > b) => ");
            result = (a > b);
            document.write(result);
            document.write(linebreak);
```

```
            document.write("(a != b) => ");
            result = (a != b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >= b) => ");
            result = (a >= b);
            document.write(result);
            document.write(linebreak);

            document.write("(a <= b) => ");
            result = (a <= b);
            document.write(result);
            document.write(linebreak);
      </script>
   </body>
</html>
```

Bitwise Operators:

- The bitwise operators perform bitwise operations on operands.
- The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

- Example

```
<html>
   <body>
      <script type = "text/javascript">
            var a = 2;
            var b = 3;
            var linebreak = "<br>";

            document.write("(a & b) => ");
            result = (a & b);
            document.write(result);
            document.write(linebreak);

            document.write("(a | b) => ");
            result = (a | b);
            document.write(result);
            document.write(linebreak);

            document.write("(a ^ b) => ");
            result = (a ^ b);
            document.write(result);
            document.write(linebreak);

            document.write("(~b) => ");
            result = (~b);
            document.write(result);
            document.write(linebreak);

            document.write("(a << b) => ");
            result = (a << b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >> b) => ");
            result = (a >> b);
            document.write(result);
            document.write(linebreak);
      </script>
   </body>
</html>
```

Logical Operators:

- The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

- Example:

```
<html>
   <body>
      <script type = "text/javascript">
            var a = true;
            var b = false;
            var linebreak = "<br>";

            document.write("(a && b) => ");
            result = (a && b);
            document.write(result);
            document.write(linebreak);

            document.write("(a || b) => ");
            result = (a || b);
            document.write(result);
            document.write(linebreak);

            document.write("!(a && b) => ");
            result = (!(a && b));
            document.write(result);
            document.write(linebreak);
      </script>
   </body>
</html>
```

Assignment Operators:

- The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

- Example:

```
<html>
   <body>
      <script type = "text/javascript">

            var a = 33;
            var b = 10;
            var result;

            var linebreak = "<br>";

            document.write("Value of a => (a = b) => ");
            result = (a = b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a += b) => ");
            result = (a += b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a -= b) => ");
            result = (a -= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a *= b) => ");
```

```
            result = (a *= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a /= b) => ");
            result = (a /= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a %= b) => ");
            result = (a %= b);
            document.write(result);
            document.write(linebreak);

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

Conditional Operator:

- The operator (?:) is known as conditional operator.
- It is also called as Ternary operator.
- Syntax:

  Condition? Expression-1 : Expression-2;

- In above syntax, if condition is true then it will execute Expression-1 otherwise Expression-2.
- Example:

```
<html>
   <body>
      <script type = "text/javascript">

            var a = 100;
            var b = 20;
            var result;

            result = (a > b) ? a : b;
            document.write("Greatest Number="+result);

      </script>

   </body>
</html>
```

❖ If Statement:

- if is a predefined keyword of JavaScript.
- It is a selection statement/decision making statement.
- If condition is true then it will execute the body of if statement.
- If condition is false then it will not execute the body of if statement.
- Syntax:

```
if(condition)
{
     //body of if
}
```

- Example.

```
<html>
   <body>
      <script type = "text/javascript">
            var age = 20;

            if(age > 18)
            {
               document.write("Qualifies for driving");
            }
      </script>
   </body>
</html>
```

❖ If-else Statement:

- if and else both are predefined keyword of JavaScript.
- It is a selection statement/decision making statement.
- If condition is true then it will execute the body of if statement.
- If condition is false then it will execute the body of else statement.
- Syntax:

```
if(condition)
{
     //body of if
}
else
{
     //body of else
}
```

- Example.

```
<html>
   <body>
      <script type = "text/javascript">

            var age = 20;

            if(age > 18)
            {
               document.write("Qualifies for driving");
            }
            else
            {
               document.write("Not Qualifies for driving");

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

❖ If…else…if Statement:

- if and else both are predefined keyword of JavaScript.
- It is a selection statement/decision making statement.
- The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.
- It is a chain of if-else.
- Syntax:

```
if(condition-1)
{
  //statement-1
}
else if(condition-2)
{
  //statement-2
}
else if(condition-3)
{
  //statement-3
}
else
{
  //statement-4
}
```

- Example.

```
<html>
   <body>
      <script type = "text/javascript">

            var marks = 67.78;

            if(marks >= 75)
            {
                  document.write("You got Distinction");
            }
           else if(marks>=60)
           {
                  document.write("You got First Class");
           }
          else if(marks>=40)
           {
```

```
                document.write("You are Pass Only");
          }
          else
          {
                document.write("You are Fail");
          }

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

❖ Nested If-else statement:

- if and else both are predefined keyword of JavaScript.
- It is a selection statement/decision making statement.
- If we use one if-else within another if-else statement then it is called as nested if-else statement.
- This statement is used to take correct decision out of several conditions.
- Syntax:

```
if(condition-1)
{
          if(condition-2)
          {
                //statements
          }
          else
          {
                //statements
          }
}
else
{
          if(condition-3)
          {
                //statements
          }
          else
          {
                //statements
          }

}
```

- Example.

```
<html>
   <body>
      <script type = "text/javascript">

            var a=10;
            var b=20;
            var c=30;

           if(a>b)
           {
                   if(a>c)
                   {
                           document.write("Greatest Number="+a);
                   }
                   else
                   {
                           document.write("Greatest Number="+c);

                   }
           }
           else
           {
                   if(b>c)
                   {
                           document.write("Greatest Number="+b);
                   }
                   else
                   {
                           document.write("Greatest Number="+c);

                   }

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

❖ Switch case statement:

- Switch statement is used to execute one of the statements from many blocks of statements.
- It is a selection statement.
- There are four keywords are used i.e switch, case, break, default.
- Switch case expression value compared with case value and if it is match then corresponding block will be executed.
- If none of case values are matched then default block is executed.
- Syntax:

```
switch(Expression)
{
          case value1:     //statement;
                           break;

          case value2:     //statement;
                           break;

          case value3:     //statement;
                           break;

          default:         //default statement;

}
```

- Example.

```
<html>
   <body>
      <script type = "text/javascript">
            var grade = 'M';
            switch (grade)
            {
               case 'A': document.write("Good job");
                         break;

               case 'B': document.write("Pretty good");
                         break;

               case 'C': document.write("Passed");
                         break;

               case 'D': document.write("Not so good");
```

```
                        break;

                case 'F': document.write("Failed");
                          break;

                default:  document.write("Please enter valid grade");
            }
        </script>
    </body>
</html>
```

❖ JavaScript Loops:

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

For Loop:

- 'for' is a predefined keyword of JavaScript.
- It is a looping statement which executes block of code N times.
- Syntax:

```
for (initialization; condition; increment/decrement)
{

           //block of statements

}
```

- In for loop, program controller first goes to initialize section, after that it will test the condition. if condition is true then it will executes body of for loop and after that goes to increment/decrement section.
- Again it will check the condition and if condition is true then it will again executes the body of for loop. This process will continue till condition becomes false.
- Example

```
<html>
   <body>
      <script type = "text/javascript">
            var i;

            document.write("Output of For Loop=");

            for(i = 1; i <= 10; i++)
            {
               document.write("Current value of i : " + i );
            }

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

While Loop:

- 'while' is a predefined keyword of JavaScript.
- It is called as entry controlled loop.
- It is a looping statement which executes block of code N times.
- Syntax:

```
while(condition)
{

           //block of statements

}
```

- In while loop, program controller first test the condition. if condition is true then it will executes body of for loop.
- Again it will check the condition and if condition is true then it will again executes the body of while loop. This process will continue till condition becomes false.
- Example

```
<html>
   <body>
      <script type = "text/javascript">
            var i=1;

            document.write("Output of For Loop=");

            while(i<=10)
            {
               document.write("Current value of i : " + i );
               i=i+1;
            }

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

Do-While Loop:

- 'do' & 'while' both are predefined keyword of JavaScript.
- It is called as exit controlled loop.
- It is a looping statement which executes block of code N times.
- Syntax:

```
do
{

           //block of statements

}while(condition);
```

- In do- while loop, program controller first executes the body of loop and then test the condition. if condition is true then it will again executes body of loop.
- This process will continue till condition becomes false.
- In do-while loop, program controller executes body of loop at least once even if condition becomes false very first time.
- Example

```
<html>
   <body>
      <script type = "text/javascript">

            var i=1;

            document.write("Output of For Loop=");

            do
            {
               document.write("Current value of i : " + i );
               i=i+1;
            }while(i<=10);

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

For-in Loop:

- 'for' is a predefined keyword of JavaScript.
- The for...in loop is used to loop through an object's properties.
- In each iteration, one property from object is assigned to variablename and this loop continues till all the properties of the object are exhausted.
- Syntax:

```
for (variablename in object)
{
   //block of code
}
```

- Example: following example implement 'for-in' loop. It prints the web browser's Navigator object.

```
<html>
   <body>
      <script type = "text/javascript">

            var aProperty;
            document.write("Navigator Object Properties<br> ");

            for (aProperty in navigator)
            {
               document.write(aProperty);
               document.write("<br>");
            }

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

❖ **Break and continue theory is similar to our C and C++ language concept.**