What is JavaScript?

- ❑ JavaScript is the world's most popular programming language.
- ❑ It is the language for HTML, for the web, for servers, PCs, laptops, tablets, phones, and more.
- ❑ A scripting language is a lightweight programming language.
- ❑ JavaScript is programming code that can be inserted into HTML pages.
- ❑ JavaScript code can be executed by all modern web browsers.
- ❑ ECMA-262 is the official name of the JavaScript standard.
- ❑ JavaScript was invented by Brendan Eich.
- ❑ It appeared in Netscape (a no longer existing browser) in 1995, and has been adopted by ECMA (a standard association) since 1997.

How to use JavaScript

- ❑ JavaScripts in HTML must be inserted between <script> and </script> tags.
- ❑ JavaScripts can be put in the <body> and in the <head> section of an HTML page.
- ❑ The <script> and </script> tells where the JavaScript starts and ends.
- ❑ It is a common practice to put functions in the <head> section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.
- ❑ Scripts can also be placed in external files.
- ❑ External files often contain code to be used by several different web pages.
- ❑ External JavaScript files have the file extension .js.
- ❑ To use an external script, point to the .js file in the "src" attribute of the <script> tag:

```
<script>
    alert(“Hello World!”);
</script>
<script  src=“ajax.js”></script>
```

We can take some action when JavaScript is not supported by browser by using <noscript> tag.

```
<script>
</script>
<noscript>
       Sorry! JavaScript is not supported!
</noscript>
```

document.write

- ❑ Use document.write() only to write directly into the document output.
- ❑ You can only use document.write in the HTML output. If you use it after the document has loaded, the whole document will be overwritten

```
<script>
     document.write(“Hello!”)
</script>
```

Variables

- ❑ Variable names must begin with a letter
- ❑ Variable names can also begin with $ and _
- ❑ Variable names are case sensitive
- ❑ The recommended naming convention is camel case naming.
- ❑ You declare JavaScript variables with the var keyword.
- ❑ After the declaration, the variable is empty (it has no value). So its value is undefined.
- ❑ JavaScript has dynamic types. This means that the same variable can be used as different types.

```
var  v1 = 10;
var  n1 = 10,  n2=30,  msg="Hello";
var  x  = 5;                 // x is a Number
var  x  = "Test";        //  x is a String
```

Variables have local scope and Global scope. Any variable outside all functions has global scope and variables inside the function have local scope.
Remember, only function create scope and not braces.

Operators

The following are arithmetic operators.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (division remainder)
++	Increment
--	Decrement

***Note**: The + operator can also be used to add string variables or text values together.*

Combined operators like += and -= are also supported.

Comparison Operators

The following are relational and logical operators.

Operator	Description
==	equal to
===	exactly equal to (equal value and equal type)
!=	not equal
!==	not equal (different value or different type)
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
&&	And
||	OR
!	Not

Miscellaneous Operators

Conditional expression (?:) and typeof operators belong to this category.

```
Condtion ? truevalue: falsevalue
```

typeof operator returns a string that indicates type of the variable based on the value of the variable.

```
var s = typeof(v) == "number" ? "Number" : "Non-Number";
```

Objects

- ❑ In JavaScript almost everything is an object.
- ❑ Even primitive data types (except null and undefined) can be treated as objects.
- ❑ Booleans can be objects or primitive data treated as objects
- ❑ Numbers can be objects or primitive data treated as objects
- ❑ Strings are also objects or primitive data treated as objects
- ❑ Dates are always objects
- ❑ Arrays are always objects
- ❑ Even functions are always objects
- ❑ An object is just a special kind of data, with properties and methods.

```
var student=new Object();
student.rollno=10;
student.name="Jhon Resig";
student.mobile="9933993399";
```

```
var  student={rollno:10, name: Jhon Resig", mobile:"9933993399"};
```

For in loop

This loop is used to access each property of the object.

```
var student={rollno:10, name:"Jhon Resig", mobile:"9933993399"};
for (var p in student) {
      document.write( "<p/>" + p  + ":" +  student[p]);
}
```

```
<script>
        // constructor for Employee
        function Employee(id, name, salary) {
            this.id = id;
            this.name = name;
            this.salary = salary;
            this.tostring = tostring;  // function declaration
            // define function
            function tostring() {
                return id + ":" + name + ":" + salary;
            }
        }
       var e1 = new Employee(1, "Steve", 10000); //create an instance of Employee
       document.write(e1.tostring());
    </script>
```

Arrays

The Array object is used to represent an array.
An array is a collection of values accessed using single name using index.
Use following syntax to create an array.

```
var array-name = [item1, item2, ...];
```

```
var langs=[“Java","C#","JavaScript”];
```

You can also create an array using Array constructor.

```
var langs=new Array();
langs[0] = “Java”;
langs[1] = “C#”;
langs[2] = “JavaScript”;
```

```
var langs=new Array("Java","C#","JavaScript”);
```

```
var name=langs[0];
langs[0]="C++";
```

***Note**: Arrays are object so you methods and properties for an array.*

The property length returns number of elements in the array.

```
var langs=[“Java","C#","JavaScript”];
for (var i=0; i < langs.length; i++)
{
}
```

The following are the available method of an array.

Method	**Description**
concat()	Joins two or more arrays, and returns a copy of the joined arrays
indexOf()	Search the array for an element and returns its position
join()	Joins all elements of an array into a string
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

```
  var databases = ["Oracle", "MySQL", "Sql Server", "DB2" ];
  document.write(databases.valueOf() + "<p/>");
  databases.sort();
  document.write(databases.valueOf() + "<p/>");
  var dbs = databases.slice(1,2);
  document.write("<p/>" + dbs.valueOf() + "<p/>");
  databases.splice(0,1);  // remove first element
  document.write(databases.valueOf() + "<p/>");
  var  st = databases.join(" : ");
  document.write(st + "<p/>");
```

Functions

- ❑ A function is a block of code.
- ❑ It may take parameters and return value
- ❑ A function is defined with keyword function
- ❑ The return statement is used to return a value from function
- ❑ More than required parameters are ignored
- ❑ Less than required parameters results in undefined parameters

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

A function expression can be assigned to a variable and then invoked using variable as follows:

```
var add = function(a,b) { return a + b; };

document.write (add(10,20));
```

Function expressions can be made "self-invoking". They are also known as Immediately-invoked function expressions (IIFE).

A self-invoking expression is invoked (started) automatically, without being called.

Function expressions will execute automatically if the expression is followed by ().

You cannot self-invoke a function declaration.

```
(function(){ /* code */ }());
```

```
( function() { document.write("IIFE");})();
```

Default parameters

In JavaScript, parameters of functions default to undefined. However, in some situations it might be useful to set a different default value. This is where default parameters can help.

```
function multiply(a, b = 1) {
  return a*b;
}

multiply(5); // 5
```

Using the arguments object

The arguments of a function are maintained in an array-like object. Within a function, you can address the arguments passed to it as follows:

```
arguments[0] // first argument
arguments.length // returns no. of arguments
```

```
function total(){
        var total = 0;
        for (var i = 0; i < arguments.length ; i ++)
        {
            total += arguments[i];
        }
        return total;
     }
     document.write( total(10,20,30));
```

We can create functions in objects as follows:

```
var person = { firstName : "Srikanth" ,  lastName : "Pragada",
           fullname : function() {
              return  this.firstName + " " + this.lastName;
           }
     };

     document.write(person.fullname());
```

OBJECT MODEL

- JavaScript is an object-based language based on prototypes, rather than being class-based.
- A prototype-based language has the notion of a *prototypical object*, an object used as a template from which to get the initial properties for a new object.
- We define a constructor function to create objects with a particular initial set of properties and values. Any JavaScript function can be used as a constructor. You use the `new` operator with a constructor function to create a new object.

```
  function Course(name, fee, duration) {
            this.name = name;
            this.fee = fee;
            this.duration = duration;

            this.toString = function() {
              return name + ":" + fee + ":" + duration + ":" + this.courseFee();
            };

            this.courseFee = function () {
                return this.fee + this.fee *  Course.prototype.tax / 100;
            };


        }

        var c1 = new Course("Java", 4000, 40);


        document.write(c1.toString() + "<br/>");
        document.write(c1.courseFee() + "<br/>");

        var c2 = new Course("JavaEE", 5000, 40);
        var c3 = new Course("Oracle", 3000, 40);
        document.write(c2.toString() );
        document.write(c3.toString());
```