Java Script Definition

JavaScript (JS) is a lightweight interpreted or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat.

JavaScript is a prototype-based, multi-paradigm, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. The programs in this language are called scripts. They can be written right in a web page's HTML and executed automatically as the page loads.

JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the JavaScript engine. The browser has an embedded engine sometimes called a "JavaScript virtual machine".

Modern JavaScript is a "safe" programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.

Syntax For Java Script

```
<script language = "javascript"  type = "text/javascript">
  Write JavaScript code Here
</script>
```

JavaScript Advantages

The biggest advantages to a JavaScript having a ability to produce the same result on all modern browsers.

1. Client-Side execution: No matter where you host JavaScript, Execute always on client environment to save a bandwidth and make execution process fast.
2. User Interface Interactivity: JavaScript used to fill web page data dynamically such as drop-down list for a Country and State. Base on selected Country, State drop down list dynamically filled. Another one is

Form validation, missing/incorrect fields you can alert to a users using alert box.

3. Rapid Development: JavaScript syntax's are easy and flexible for the developers. JavaScript small bit of code you can test easily on Console Panel (inside Developer Tools) at a time browser interpret return output result. In-short easy language to get pick up in development.
4. Browser Compatible: The biggest advantages to a JavaScript having a ability to support all modern browser and produce the same result.
5. Make XMLHttpRequest() Object: XMLHttpRequest is special JavaScript object that was designed by Microsoft. XMLHttpRequest object call as a asynchronous HTTP request to the Server for transferring data both side without reloading the page.

JavaScript Disadvantages

Biggest disadvantages to a JavaScript, code visible to everyone.

1. Code Always Visible: The biggest disadvantages are code always visible to everyone anyone can view JavaScript code.
2. Bit of Slow execute: No matter how much fast JavaScript interpret, JavaScript DOM (Document Object Model) is slow and will be a never fast rendering with HTML.
3. Stop Render: JavaScript single error can stop to render with entire site. However browsers are extremely tolerant of JavaScript errors.

JavaScript in <head>...</head> section

If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows −

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
              alert("Hello World")
         }
    </script>
  </head>
  <body>
    <input type = "button" onclick = "sayHello()" value = "Say Hello" />
  </body>
</html>
```

JavaScript in <body>...</body> section

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. In this case, you would not have any function defined using JavaScript. Take a look at the following code.

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

  <body>
    <script type = "text/javascript">
      <!--
        document.write("UBTER Semester Examination")
      //-->
    </script>

    <p>This is web page body </p>
  </body>
</html>
```

JavaScript in External File

As you begin to work more extensively with JavaScript, you will be likely to find that there are cases where you are reusing identical JavaScript code on multiple pages of a site.

You are not restricted to be maintaining identical code in multiple HTML files. The script tag provides a mechanism to allow you to store JavaScript in an external file and then include it into your HTML files.

Here is an example to show how you can include an external JavaScript file in your HTML code using script tag and its src attribute.

```
<html>
```

```
  <head>
    <script type = "text/javascript" src = "filename.js" ></script>
  </head>

  <body>
    .......
  </body>
</html>
```

JavaScript Variable Scope

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

Global Variables − A global variable has global scope which means it can be defined anywhere in your JavaScript code.

Local Variables − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

```
<html>
  <body onload = checkscope();>
    <script type = "text/javascript">
        var myVar = "global";      // Declare a global variable
         function checkscope( ) {
            var myVar = "local";    // Declare a local variable
            document.write(myVar);
         }
    </script>
  </body>
</html>
```

The typeof Operator

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

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

Here is a list of the return values for the **typeof** Operator.

Type	String Returned by typeof
• Number	"number"
• String	"string"
• Boolean	"boolean"
• Object	"object"
• Function	"function"
• Undefined	"undefined"
• Null	"object"

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

        result = (typeof b == "string" ? "B is String" : "B is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);

        result = (typeof a == "string" ? "A is String" : "A is Numeric");
        document.write("Result => ");
        document.write(result);
    </script>
  </body>
```

```
</html>
```

JavaScript var statement

JavaScript var statement to declare variables that are scoped to the nearest function block. Another word we can say function scope variable. If you don't assign variable's value at declaration time, JavaScript automatically assigns the undefined value. You can't use variable before the declaration otherwise gives an error.

Syntax

```
var variable_name = value;
```

Parameter

1. var	:	JavaScript reserved keyword.
2. var_name	:	Name of the variable.
3. value	:	Optional, Initial value assign at the time of declaration.

Shorthand style, you can declare a one or more variables in single statement separated by comma.

```
var var1 = val1, var2 = val2, varN = valN;
```

What is local scope?

A variable that you declare within a function called local scope. Variable can be accessed within a function, but outside of function you can't. Inside this function specifies variables with respective values. This all variables have become a local scope.

```
function myfun(){
	var greetings = "Good morning, have a nice day!";
	document.writeln(greetings + '<br />');

	function fun1() {
		document.writeln("Hello world!" + '<br />');
	}

	fun1();

	var myObj = { "commercial": ".com", "network": ".net" };
	document.writeln(myObj.commercial);
}
```

What is global scope?

If you declare variables outside of the function, those variables scopes globally. You should access variable within function as well as outside of the function.

```
var greetings = "Good morning, have a nice day!";    // Global Variable

function fun1() {
   document.writeln("Hello world!" + '<br />');
}

var myObj = { "commercial" : ".com", "network" : ".net" };

function myfun(){
       document.writeln(greetings + '<br />');
       fun1();
       document.writeln(myObj.commercial + '<br />');
}

myfun();

document.writeln(greetings + '<br />');
fun1();
document.writeln (myObj.commercial + '<br />');
```

JavaScript Data Types with Examples

ECMAScript represents JavaScript Data types. This lesson provide JavaScript Data Types With Examples. How to define variable in JavaScript. JavaScript is a loosely data type dynamic language.

JavaScript, also known as ECMAScript specifies six primitive data types and object type.

JavaScript primitives data types

Number
String
Boolean
Symbol
Null
Undefined

Object data types,
Array
JSON
Function, Object and Properties.

JavaScript Variables are declared using var statement. you can declare multiple variables at once. No need to append data type when you declare JavaScript variable. It means any value can store in any variable.

If you not initialize your variable in var statement, It's automatically assume values is undefined.

JavaScript Number Data types

JavaScript has only one Number (numeric) data types. Number data type can store normal integer, floating-point values.

A floating-point represent a decimal integer with either decimal points or fraction expressed (refer to another decimal number).

```
var num1 = 5;            // Numeric integer value
var num2 = 10.5;        // Numeric float value
var num3 = -30.47;      // Negative numeric float value

var num4 = 5E4;         // 5 x 10 Powers of 4 = 50000
var num5 = 5E-4         // 5 x 10 Powers of -4 = -50000

var num6 = 10 / 0;      // Number divide by zero, result is: Infinity
var num7 = 10 / -0;     // Number divide by negative zero, result is: -Infinity

var num8 = 10, num9 = 12.5; // Multiple variable declare and initialize
```

JavaScript String Data types

JavaScript string data type represent textual data surrounding to single/double quotes. Each character is represent as a element that occupies the position of that string. Index value 0, starting from first character of the string.

```
var name = 'Hello, I am run this town.!';    // Single quote
var name = "Hello, I am run this town.!";   // Double quote
```

Whatever quote you use to represent string, but you should take care that quote can't repeat in string statement.

```
var name = "Hello, I'm run this town.!";  // Single quote use inside string
var name1 = "";                           // empty string
```

Note: JavaScript empty string is different from the NULL value.

JavaScript Operators with Example

JavaScript Operators use either value or variable to compute some task. This lesson describes the JavaScript operators with example, and operators precedence. JavaScript has following types operators,

Arithmetic Operators
Assignment Operators
Comparison Operators
Logical Operators
Conditional Operator (Ternary Operator)
Bitwise Operators
Miscellaneous Operators
typeof
delete
instanceof
new
this
in

JavaScript Arithmetic Operators

JavaScript arithmetic operator take operand (as a values or variable) and return the single value.

We are use in our routine life arithmetic operators, addition(+), subtraction(-), multiplication (*), and division (/) and some other arithmetic operator are listed below.

We have numeric variable: x = 10, y = 5 and result.

Operator	Description	Example	Results
+	Addition	result = x + y	result = 15
-	Subtraction	result = x - y	result = 5
*	Multiplication	result = x * y	result = 50
/	Division	result = x / y	result = 2
%	Modulus	result = x % y	result = 0
++	Increment	result = x++	
result = x			
result = ++x		result = 10	
result = 11			
result = 12			
--	Decrement	result = x--	
result = x			
result = --x	result = 12		
result = 11			
result = 10			

```
<script>
   var x = 10, y = 5;
   document.writeln(x + y);        //  Addition: 15
   document.writeln(x - y);        //  Subtraction: 5
   document.writeln(x * y);        //  Multiplication: 50
   document.writeln(x / y);        //  Division: 2
   document.writeln(x % y);        //  Modulus: 0

   document.writeln(x++);          //  x: 10, x become now 11
   document.writeln(x);            //  x: 11
   document.writeln(++x);          //  x become now 12, x: 12

   document.writeln(x--);          //  x: 12, x become now 11
   document.writeln(x);            //  x: 11
   document.writeln(--x);          //  x become now 10, x: 10
</script>
```
Run it...   »

JavaScript Assignment Operators

JavaScript assignment operators assign values to left operand based on right operand. equal (=) operators is used to assign a values.

We have numeric variable: x = 10, y = 5 and result.

Operator	Sign	Description	Example	Equivalent to	Results
Assignment	=	Assign value from one operand to another operand value.	result = x	result = x	result = 17
Addition	+=	Addition of operands and finally assign to left operand.	result += x	result = result + y	result = 22
Subtraction	-=	Subtraction of operands and finally assign to left operand.	result -= y	result = result - y	result = 17
Multiplication	*=	Multiplication of operands and finally assign to left operand.	result *= y	result = result * y	result = 85
Division	/=	Division of operands and finally assign to left operand.	result /= y	result = result / y	result = 17
Modulus	%=	Modulus of operands and finally assign to left operand.	result %= y	result = result % y	result = 2
Bitwise AND	&=	AND operator compare two bits values return a results of 1, If both bits are 1. otherwise return 0.	result &= y	result = result & y = 2 & 5 = 0000 0010 & 0000 0101 = 0000 0000 = 0	result = 0
Bitwise OR	|=	OR operator compare two bits values and return result of 1, If the bits are complementary. Otherwise return 0.	result |= y	result = result | y = 2 | 5 = 0000 0010 | 0000 0101 = 0000 0111 = 7	result = 7

Bitwise XOR ^= EXCLUSIVE OR operator compare two bits values and return a results of 1, If any one bits are 1 or either both bits one. result ^= y result = result ^ y
= 7 ^ 5
= 0000 0111 ^ 0000 0101
= 0000 0010 = 2 result = 2
Shift Left <<= Shift left operator move the bits to a left side. result <<= y result = result <<= y
= 2 <<= 5
= 0000 0010 <<= 0100 0000
= 64 result = 64
Shift Right >>= Shift left operator move the bits to a left side. result >>= y result = result >>= y
= 2 >>= 5
= 0100 0000 >>= 0000 0010
= 2 result = 2

```
<script>
   var x = 17, y = 5;
      var result = x; // Assignment to left operand(result) base on right operand(y).
   document.writeln(result);
   document.writeln(result += x);
   document.writeln(result -= y);
   document.writeln(result *= y);
   document.writeln(result /= y);
   document.writeln(result %= y);

   document.writeln(result &= y);
   result = 2;     // Reassign value
   document.writeln(result |= y);
   document.writeln(result ^= y);

   document.writeln(result <<= y);
   document.writeln(result >>= y);
</script>
```

Run it... »

JavaScript Comparison Operators

JavaScript comparison operator determine the two operands satisfied the given condition. Comparison operator return either true or false.

Operator	Sign	Description
Equal	==	If both operands are equal, returns true.
Identical equal	===	If both operands are equal and/or same data type, returns true.
Not equal	!=	If both operands are not equal, returns true.
Identical not equal	!==	If both operands are not equal and/or same data type, returns true.
Greater than	>	If left operand larger than right operand, return true.
Less then	<	If left operand smaller than right operand, return true.
Greater than, equal	>=	If left operand larger or equal than right operand, return true.
Less than, equal	<=	If left operand smaller or equal than right operand, return true.

```
<script>
   document.writeln(5 == 5);       // true
   document.writeln(5 == '5');   // true
   document.writeln(5 === '5');  // false type not same

   document.writeln(5 != 10);      // true
   document.writeln(5 != '10');  // true
   document.writeln(5 !== '10'); // true

   document.writeln(5 > 10);       // false
   document.writeln(5 < 10);       // true

   document.writeln(5 >= 5);       // true
   document.writeln(5 <= 5);       // true
</script>
```

Run it...   »

JavaScript Logical Operators (Boolean Operators)

JavaScript logical operators return boolean result base on operands.

Operator	Sign	Description
Logical AND	&&	If first operand evaluate and return a true, only that evaluate the second operand otherwise skips. Return true if both are must be true, otherwise return false.
Logical OR	||	Evaluate both operands, Return true if either both or any one operand true, Return false if both are false.
Logical NOT	!	Return the inverse of the given value result true become false, and false become true.

```
<script>
   document.writeln((5 == 5) && (10 == 10));   // true
   document.writeln(true && false);            // false

   document.writeln((5 == 5) || (5 == 10));    // true
   document.writeln(true || false);            // true

   document.writeln(5 && 10);                  // return 10
   document.writeln(5 || 10);                  // return 5

   document.writeln(!5);                       // return false
   document.writeln(!true);                    // return false
   document.writeln(!false);                   // return true
</script>
```

Run it... »

JavaScript Conditional Operator (also call Ternary Operator)

JavaScript conditional operator evaluate the first expression(operand), Base on expression result return either second operand or third operand.

```
answer = expression ? answer1 : answer2;       // condition ? true : false
```

Example

```
document.write((10 == 10) ? "Same value" : "different value");
```

Run it... »

JavaScript Bitwise Operators

JavaScript bitwise operators evaluate and perform specific bitwise (32 bits either zero or one) expression.

Operator	Sign	Description
Bitwise AND	&	Return bitwise AND operation for given two operands.
Bitwise OR	|	Return bitwise OR operation for given two operands.
Bitwise XOR	^	Return bitwise XOR operation for given two operands.
Bitwise NOT	~	Return bitwise NOT operation for given operand.
Bitwise Shift Left	<<	Return left shift of given operands.
Bitwise Shift Right	>>	Return right shift of given operands.
Bitwise Unsigned Shift Right	>>>	Return right shift without consider sign of given operands.

```
<script>
   document.writeln(5 & 10);   // return 0,    calculation: 0000 0101 & 0000 1010 = 0000 0000
   document.writeln(5 | 10);   // return 15,   calculation: 0000 0101 | 0000 1010 = 0000 1111
    document.writeln(5 ^ 10);    // return 15,    calculation: 0000 0101 ^ 0000 1010 = 0000 1111
    document.writeln(~5);        // return -6,    calculation: ~ 0000 0101 = 1111 1010

   document.writeln(10 << 2);  // return 40,   calculation: 0000 1010 << 2 = 0010 1000
   document.writeln(10 >> 2);  // return 2,    calculation: 0000 1010 >> 2 = 0000 0010
   document.writeln(10 >>> 2); // return 2,    calculation: 0000 1010 >>> 2 = 0000 0010
</script>
```

Run it...   »

Miscellaneous Operators

typeof
delete
instanceof
new
this
in

typeof

JavaScript typeof operator return valid data type identifiers as a string of given expression. typeof operator return six possible values: "string", "number", "boolean", "object", "function", and "undefined".

```
typeof expression
typeof(expression)
```

Example

```
var name = 'Opal Kole';
var age = 48;
var married = true;
var experience = [2010, 2011, 2012, 2013, 2014];
var message = function(){ console.log("Hello world!"); }
var address;

typeof name;        // Returns "string"
typeof age;         // Return "number"
typeof married;     // Return "boolean"
typeof experience;  // Return "object"
typeof message;     // Return "function"
typeof address;     // Return "undefined"
```

Run it...   »

delete

JavaScript delete operator deletes object property or remove specific element in array.

If delete is not allow (you can't delete if element not exist, array element undefined etc..) then return false otherwise return true.

```
delete expression;             // delete explicit declare variable

delete object;                 // delete object
delete object.property;
delete object[property];

delete array;                  // delete array
delete array[index];
```

Example

```
var address = "63 street Ct.";
delete address;             // Returns false, Using var keyword you can't delete
add = "63 street Ct.";
delete add;                 // Returns true, explicit declare you can delete

var myObj = new Object();
myObj.name = "Opal Kole";
myObj.age = 48;
myObj.married  = true;

delete myObj.name;                // delete object property
delete myObj["count"];            // delete object property

var experience = [2010, 2011, 2012, 2013, 2014];    // array elements
delete experience[2];             // delete 2nd index from array elements
console.log(experience);          // [2010, 2011, undefined × 1, 2013, 2014]
```

Run it...   »

instanceof

JavaScript instanceof indicate boolean result, Return true, If object is an instance of specific class.

object instanceof class

Example

```
<script>
   var num1 = new Number(15);
   document.writeln(num1 instanceof Number);             // Returns true
   var num2 = 10;
   document.writeln(num2 instanceof Number);             // Return false

   document.writeln(true instanceof Boolean);            // false
   document.writeln(0 instanceof Number);                // false
   document.writeln("" instanceof String);               // false

   document.writeln(new Boolean(true) instanceof Boolean); // true
```

```
    document.writeln(new Number(0) instanceof Number);      // true
    document.writeln(new String("") instanceof String);     // true
</script>
```
Run it...   »

new
JavaScript new operator to create an instance of the object.

```
var myObj = new Object;
var myObj = new Object( );  // or you can write
                            // Object - required, for constructor of the object.

var arr = new Array( [ argument1, argument2, ..., ..., argumentN ] );
                          // argument - optional, pass any number of argument in a
object.
```
Example

```
var myObj = new Object();  // or you can write: var myObj = new Object;
myObj.name = "Opal Kole";
myObj.address = "63 street Ct.";
myObj.age = 48;
myObj.married  = true;

console.log(myObj);
    // Object {name: "Opal Kole", address: "63 street Ct.", age: 48, married:
true}
```
Run it...   »

this
JavaScript this operator represent current object.

```
this["propertyname"]
this.propertyname
```
Example

```
function employee(name, address, age, married) {
   this.name = name;
   this.address = address;
   this.age = age;
   this.married = married;
}
var myObj = new employee("Opal Kole", "63 street Ct.", 48, true);
console.log(myObj);
     // employee {name: "Opal Kole", address: "63 street Ct.", age: 48,
married: true}
```

Run it... »

in

JavaScript in operator return boolean result if specified property exist in object.

property in object

Example

```
<script>
    var myObj = new Object();        // or you can write: var myObj = new
Object;
   myObj.name = "Opal Kole";
   myObj.address = "63 street Ct.";
   myObj.age = 48;
   myObj.married  = true;

   document.writeln("name" in myObj);       // Returns true
   document.writeln("birthdate" in myObj);  // Returns false
                                  // birthdate propery not in myObj
   document.writeln("address" in myObj);    // Returns true
   document.writeln("age" in myObj);        // Returns true
   document.writeln("married" in myObj);    // Returns true
</script>
```