SNS COLLEGE OF ENGINEERING

An Autonomous Institution

Coimbatore-107

19TS601-FULL STACK DEVELOPMENT

UNIT-1

JAVASCRIPT AND BASICS OF MERN STACK

Java Script-Fundamentals

JavaScript

- JavaScript is used to create client-side, Server side dynamic pages.
- JavaScript is *an object-based scripting language* which is lightweight and cross-platform.
- JavaScript is not a compiled language, but it is a translated language.
- The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

History

- It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.
- It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser.
- Since then, it has been adopted by all other graphical web browsers.

- With JavaScript, users can build modern web applications to interact directly without reloading the page every time.
- JavaScript has no connectivity with Java programming language.
- In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

- In 1993, **Mosaic**, the first popular web browser, came into existence.
- In the **year 1994**, **Netscape** was founded by **Marc Andreessen**.
- He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers.

- Consequently, in 1995, the company recruited **Brendan Eich** intending to implement and embed Scheme programming language to the browser.
- But, before Brendan could start, the company merged with **Sun Microsystems** for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms.

- Finally, in May 1995, Marc Andreessen coined the first code of Javascript named '**Mocha**'.
- Later, the marketing team replaced the name with '**LiveScript**'.
- But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

Application of JavaScript

- JavaScript is used to create interactive websites. It is mainly used for:
- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

JavaScript Example

```
<html>
<body>
<h2>Welcome to JavaScript</h2>
<script>
document.write("Hello JavaScript by JavaScript");
</script>
</body>
</html>
```

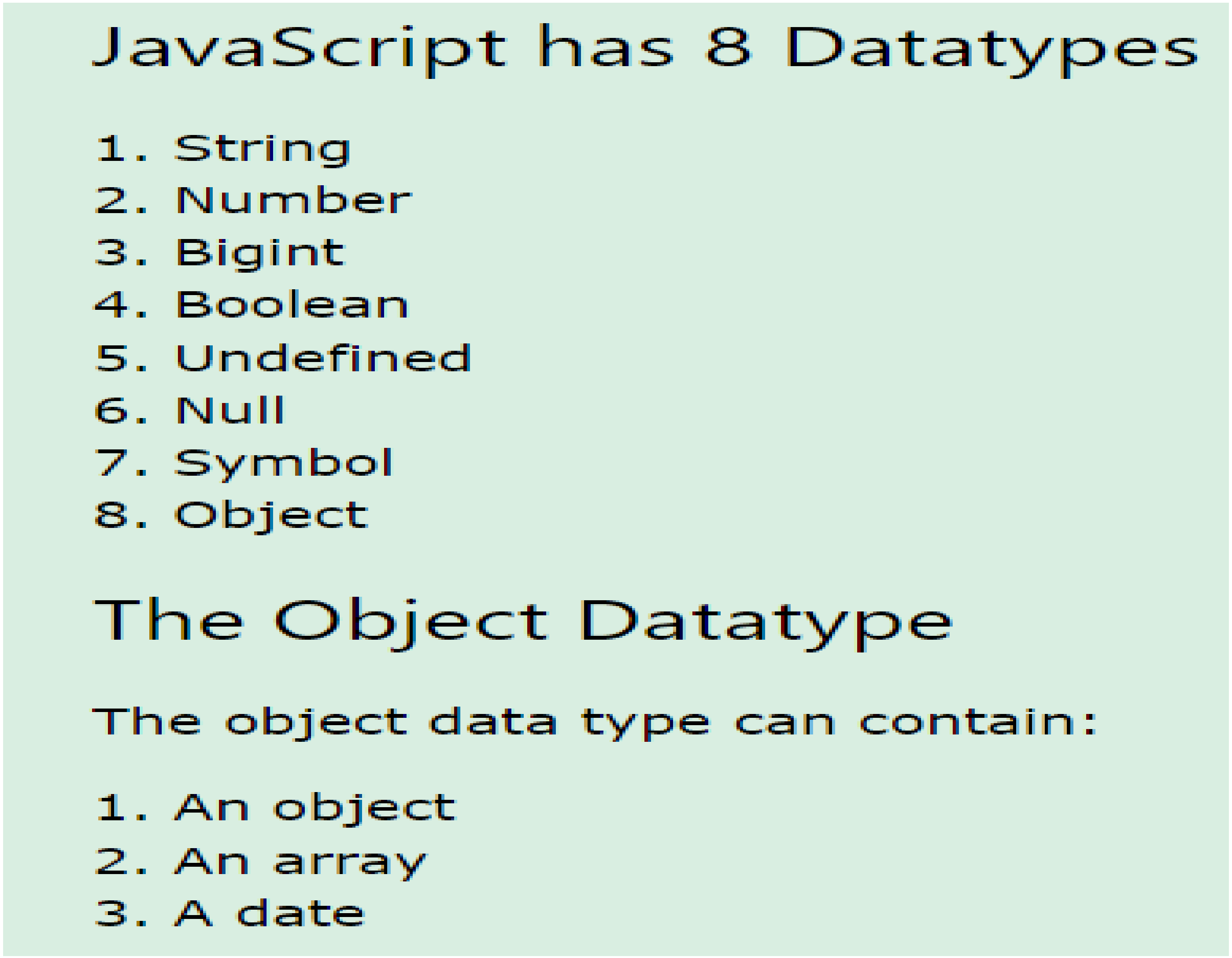
OUTPUT

```
Welcome to JavaScript
Hello JavaScript by JavaScript
```

DataTypes

- JavaScript provides different **data types** to hold different types of values.
- There are two types of data types in JavaScript.
  - Primitive data type
  - Non-primitive (reference) data type
- JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine.

- **var** here to specify the data type.
- It can hold any type of values such as numbers, strings etc.
- For example:
- var a=40;//holding number
- var b="Rahul";//holding string



JavaScript has 8 Datatypes

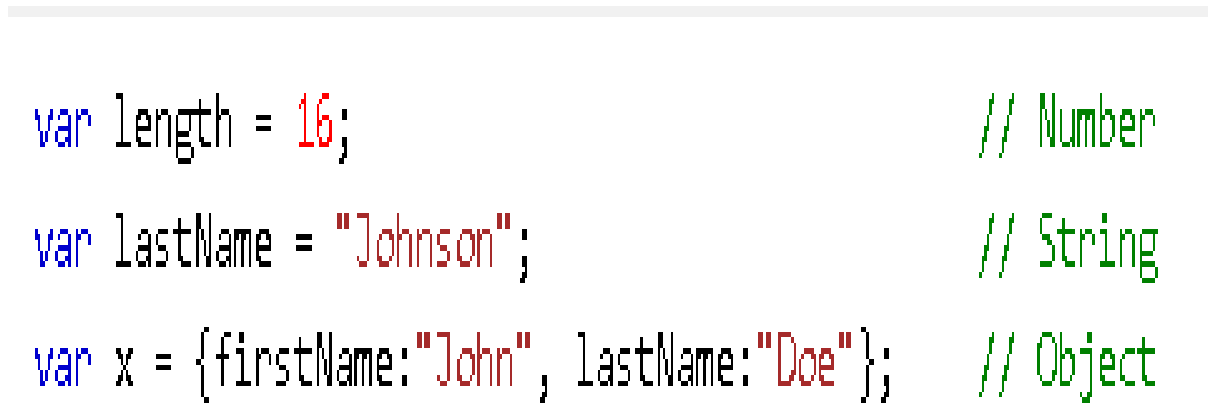
1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

The Object Datatype

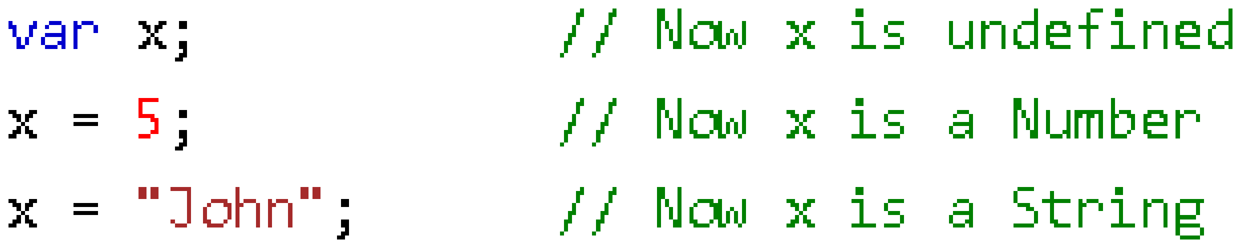
The object data type can contain:

1. An object
2. An array
3. A date

Examples



```
var length = 16;                                    // Number
var lastName = "Johnson";                           // String
var x = {firstName:"John", lastName:"Doe"};         // Object
```



```
var x;              // Now x is undefined
x = 5;              // Now x is a Number
x = "John";         // Now x is a String
```

JavaScript Variable

- A **JavaScript variable** is simply a name of storage location.
- There are two types of variables in JavaScript : local variable and

  global variable.

  There are some rules while declaring a JavaScript variable (also known as identifiers).

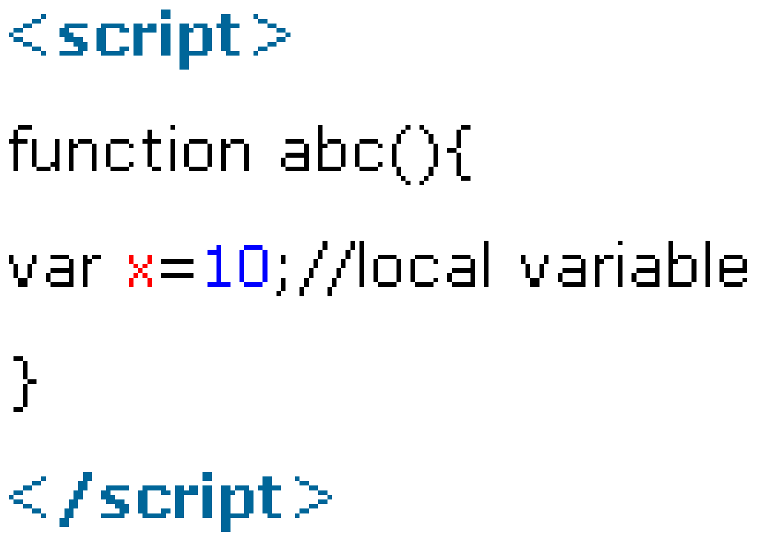
- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

```
<html>
<body>
<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>
</body>
</html>
```

Output: 30

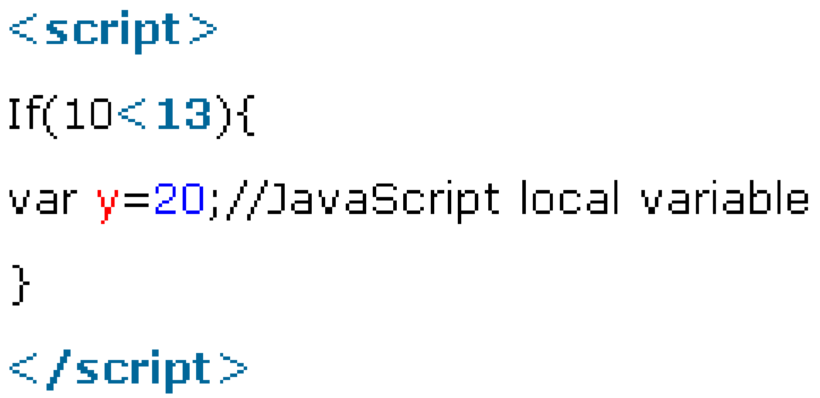
JavaScript local variable

- A JavaScript local variable is declared inside block or function.
- It is accessible within the function or block only. For example:



```
<script>
function abc(){
var x=10;//local variable
}
</script>
```

Or,



```
<script>
If(10<13){
var y=20;//JavaScript local variable
}
</script>
```

JavaScript global variable

- A **JavaScript global variable** is accessible from any function.
- A variable i.e. declared outside the function or declared with window object is known as global variable.
- A **JavaScript global variable** is declared outside the function or declared with window object. It can be accessed from any function.
- For example:

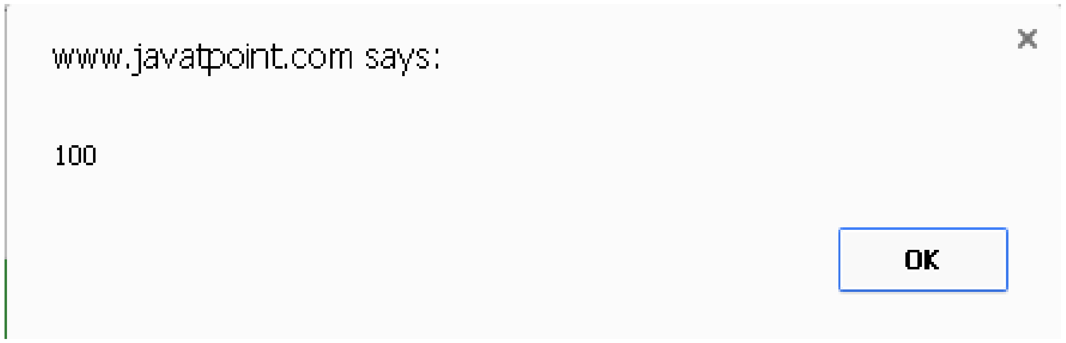
```
<html>
<body>
<script>
var data=200;//gloabal variable
function a(){
document.writeln(data);
}
function b(){
document.writeln(data);
}
a();//calling JavaScript function
b();

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

```
200 200
```

Declaring JavaScript global variable within function

- To declare JavaScript global variables inside function, you need to use **window object**.
- For example: window.value=90;

```
tml>
<body>
<script>
function m(){
window.value=100;//declaring global variable by window object
}
function n(){
alert(window.value);//accessing global variable from other function
}
m();
n();
</script>
</body>
</html>
```


www.javatpoint.com says:

100

JavaScript Operators

- JavaScript operators are symbols that are used to perform operations on operands.
- For example:
- var sum=10+20;
- Here, + is the arithmetic operator and = is the assignment operator.

- Arithmetic Operators
- Comparison (Relational) Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Special 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

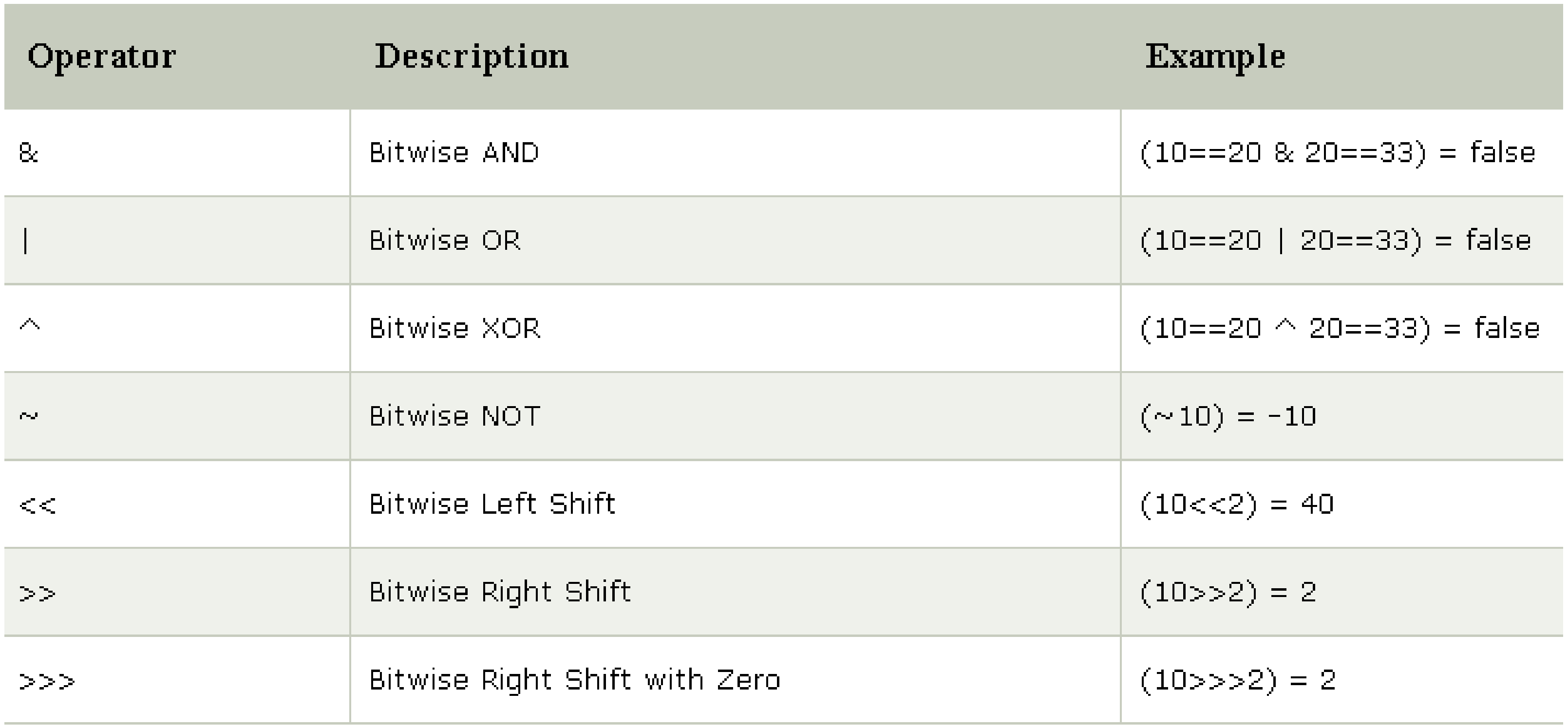
Comparison Operators

- The JavaScript comparison operator compares the two operands.

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

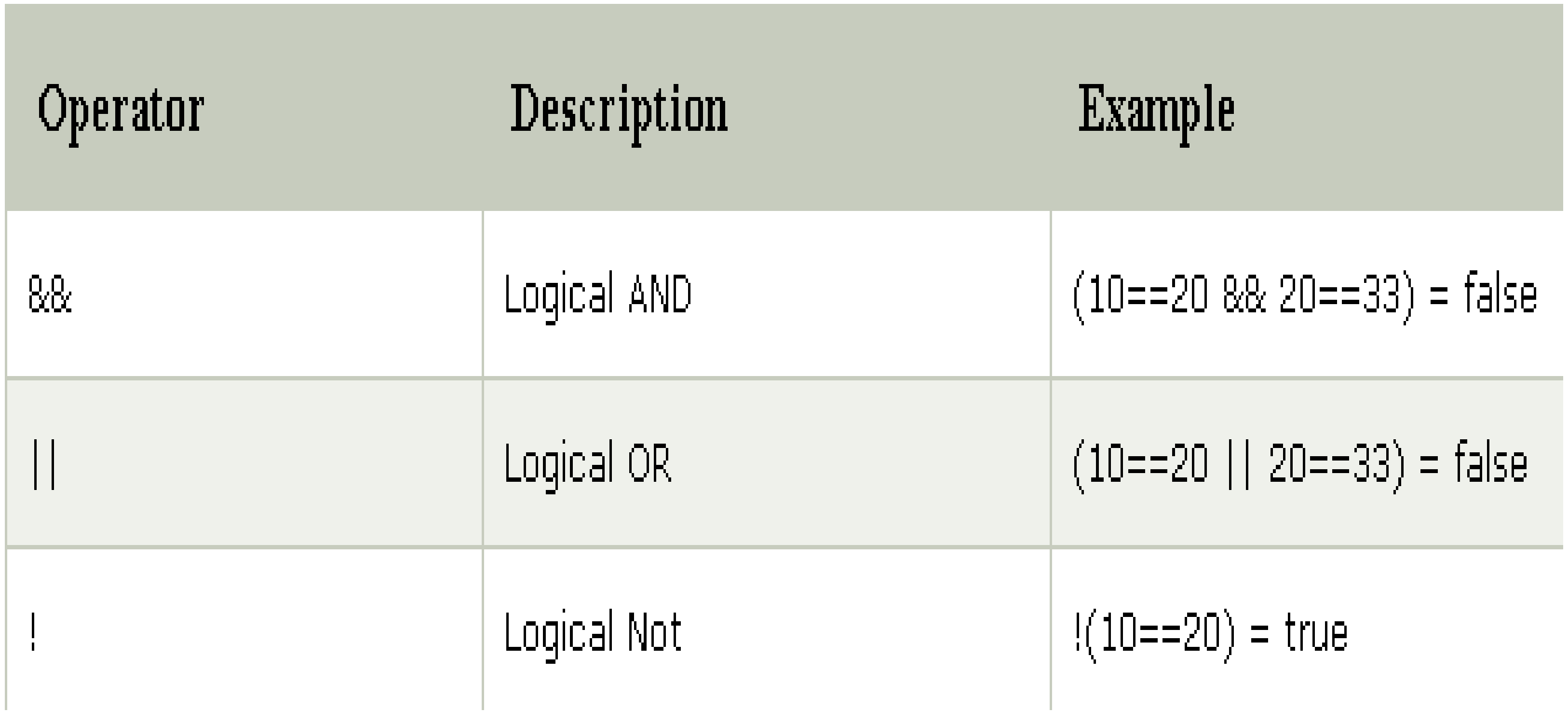
Bitwise Operators

- The bitwise operators perform bitwise operations on operands.



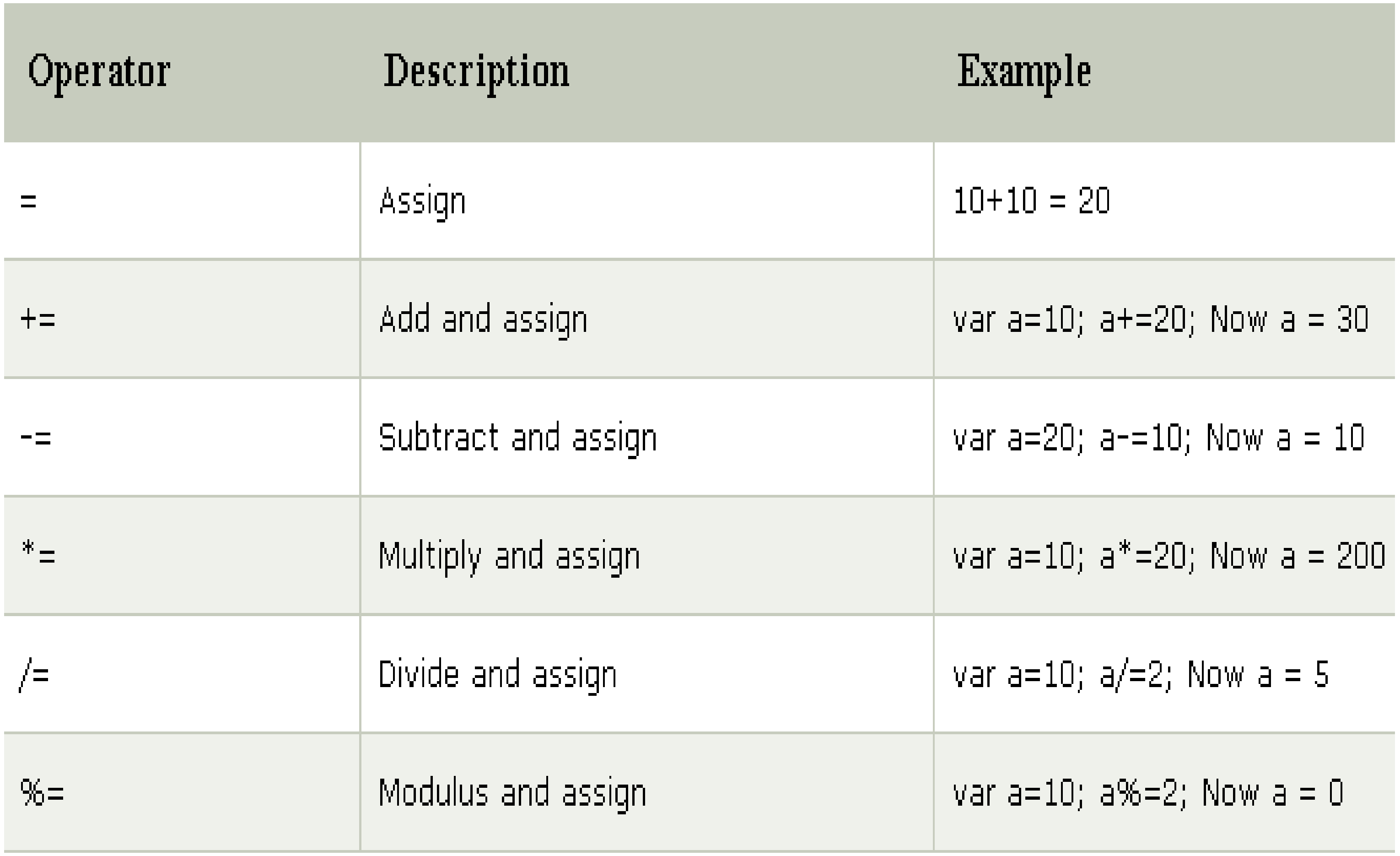
Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

Logical Operators



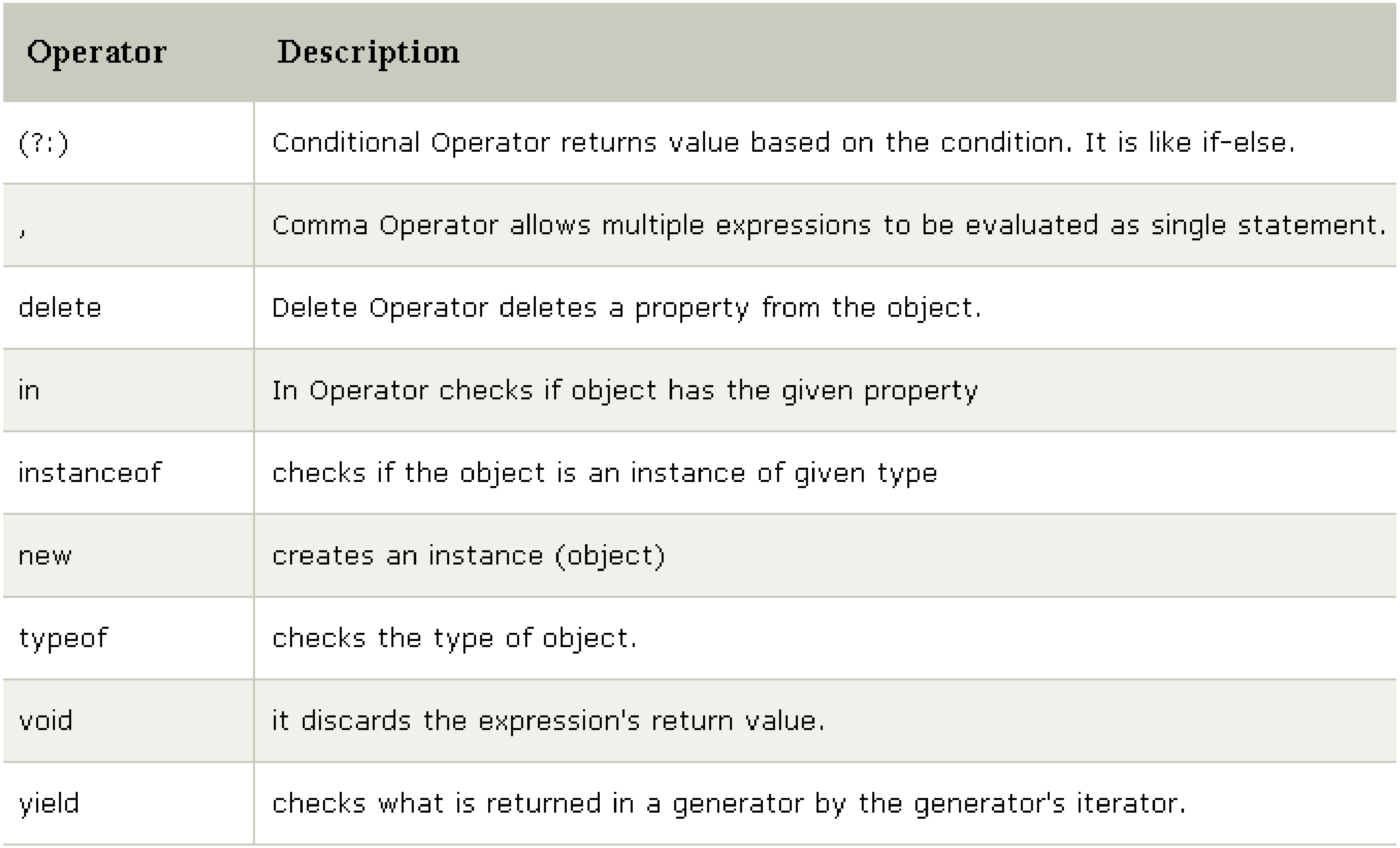
Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

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

Special Operators



Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

Control Structure

- Control structure actually controls the flow of execution of a program.
- if … else
- switch case
- do while loop
- while loop
- for loop

If … else

- The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
- The else statement to specify a block of code to be executed if the condition is false.

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

```
Good day
```

Switch case

- The switch statement is used to perform different actions based on different conditions.
- The switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Switch case

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

```
<script type="text/javascript">
   <!--
      var grade='A';
      document.write("Entering switch block<br/>");
      switch (grade) {
         case 'A': document.write("Good job<br/>");
            break;
         case 'B': document.write("Pretty good<br/>");
            break;
         case 'C': document.write("Passed<br/>");
            break;
         case 'D': document.write("Not so good<br/>");
            break;
         case 'F': document.write("Failed<br/>");
            break;
         default:  document.write("Unknown grade<br/>")
      }
      document.write("Exiting switch block");
   //-->
</script>
```

Do while Loop

- The do loop will always be executed at least once, even if the while condition is false.

```
do{
    Statement(s) to be executed;
} while (expression);
```

Example

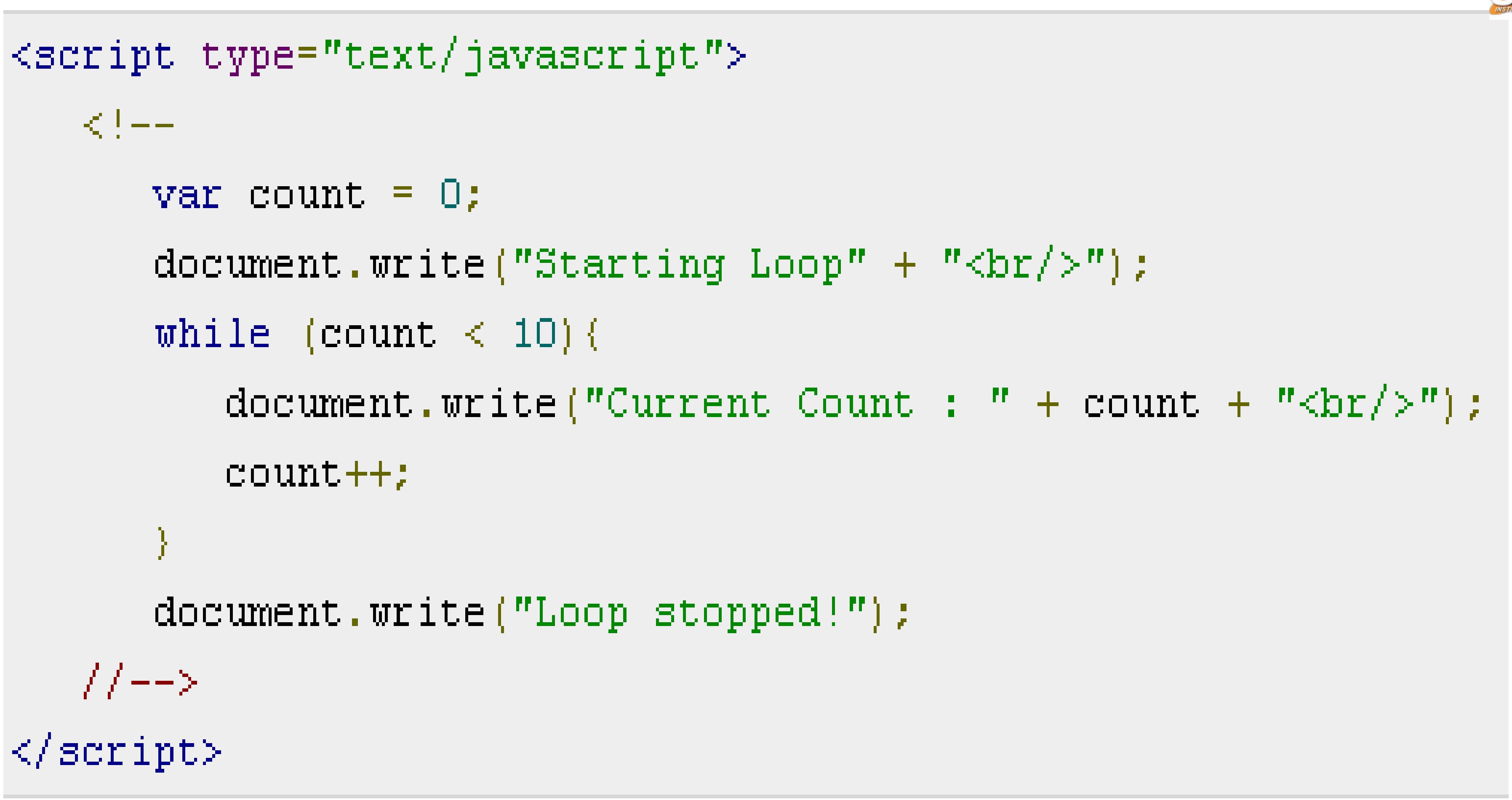
```
<script type="text/javascript">
   <!--
      var count = 0;
      document.write("Starting Loop" + "<br/>");
      do{
         document.write("Current Count : " + count + "<br/>");
         count++;
      }while (count < 0);
      document.write("Loop stopped!");
   //-->
</script>
```

```
Starting Loop
Current Count : 0
Loop stopped!
```

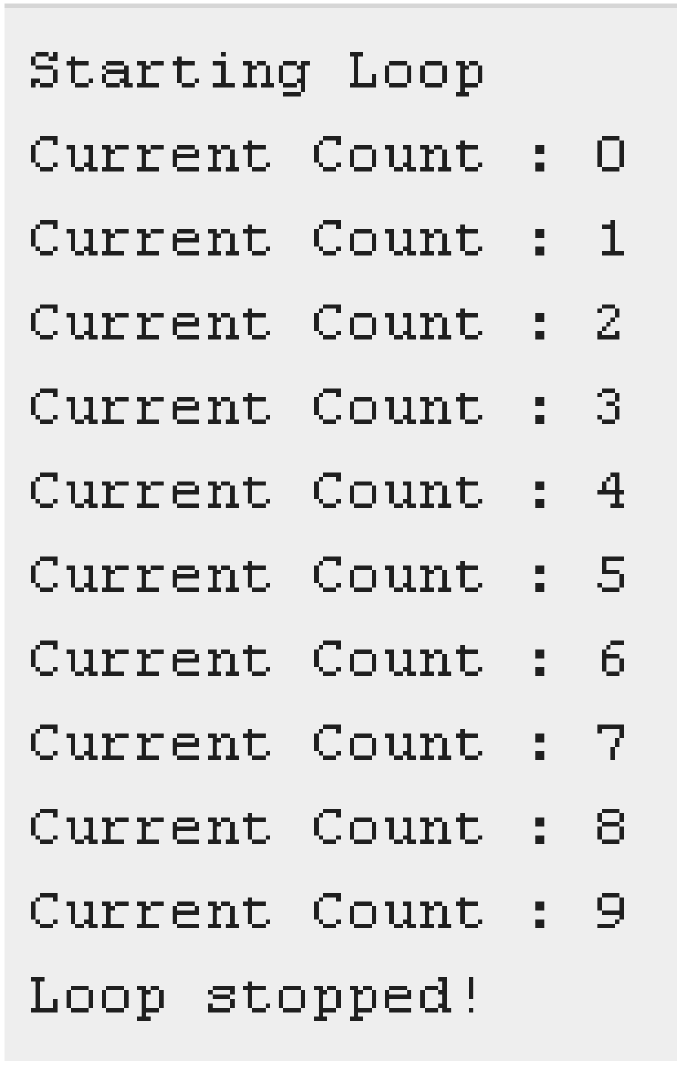
While Loop

- The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true.
- Once expression becomes false, the loop will be exited.

```
while (expression){
   Statement(s) to be executed if expression is true
}
```



```
<script type="text/javascript">
   <!--
      var count = 0;
      document.write("Starting Loop" + "<br/>");
      while (count < 10){
         document.write("Current Count : " + count + "<br/>");
         count++;
      }
      document.write("Loop stopped!");
   //-->
</script>
```



```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

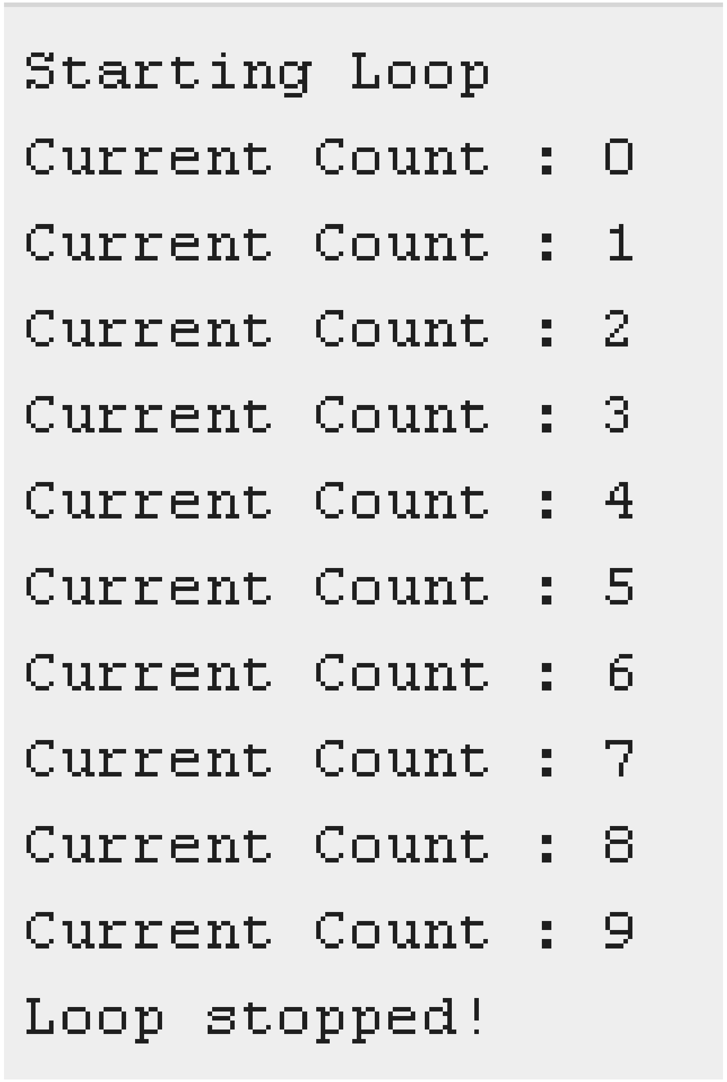
For Loop

- The for loop is the most compact form of looping and includes the following three important parts –
- The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The test statement which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
- The iteration statement where you can increase or decrease your counter.

Syntax and Example

```
for (initialization; test condition; iteration statement){
   Statement(s) to be executed if test condition is true
}
```

```
<script type="text/javascript">
   <!--
      var count;
      document.write("Starting Loop" + "<br/>");
      for(count = 0; count < 10; count++){
         document.write("Current Count : " + count );
         document.write("<br/>");
      }
      document.write("Loop stopped!");
   //-->
</script>
```



```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

JavaScript Functions

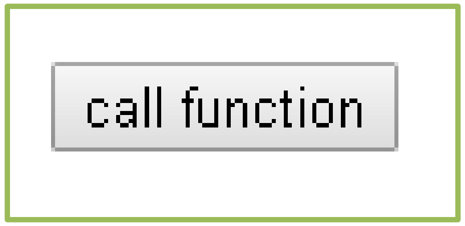
- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- **JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

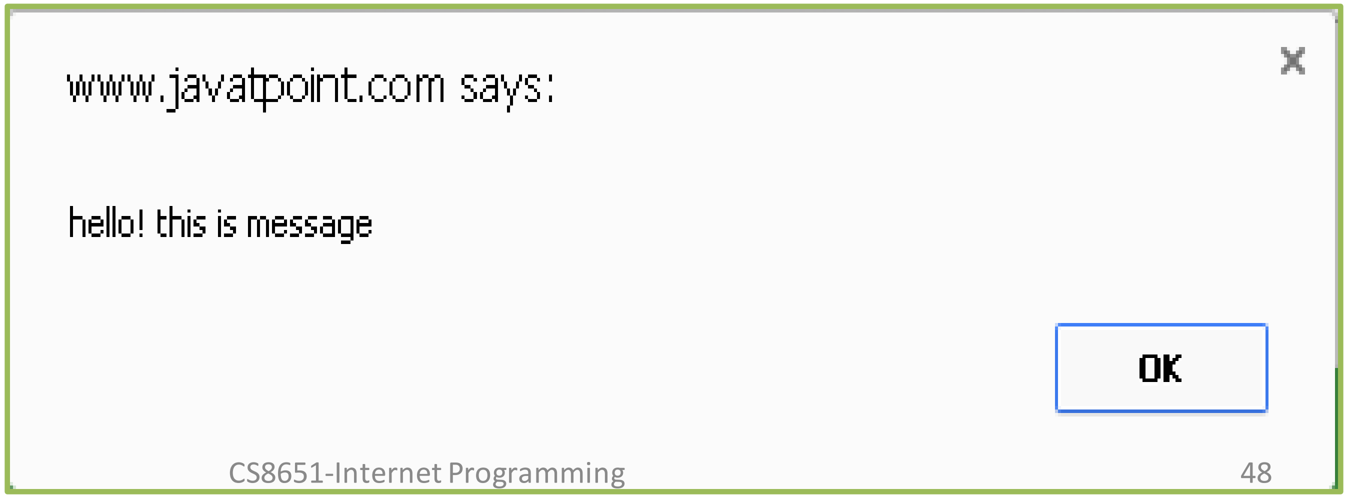
Advantage of JavaScript function

- There are mainly two advantages of JavaScript functions.
- **Code reusability**: We can call a function several times so it save coding.
- **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

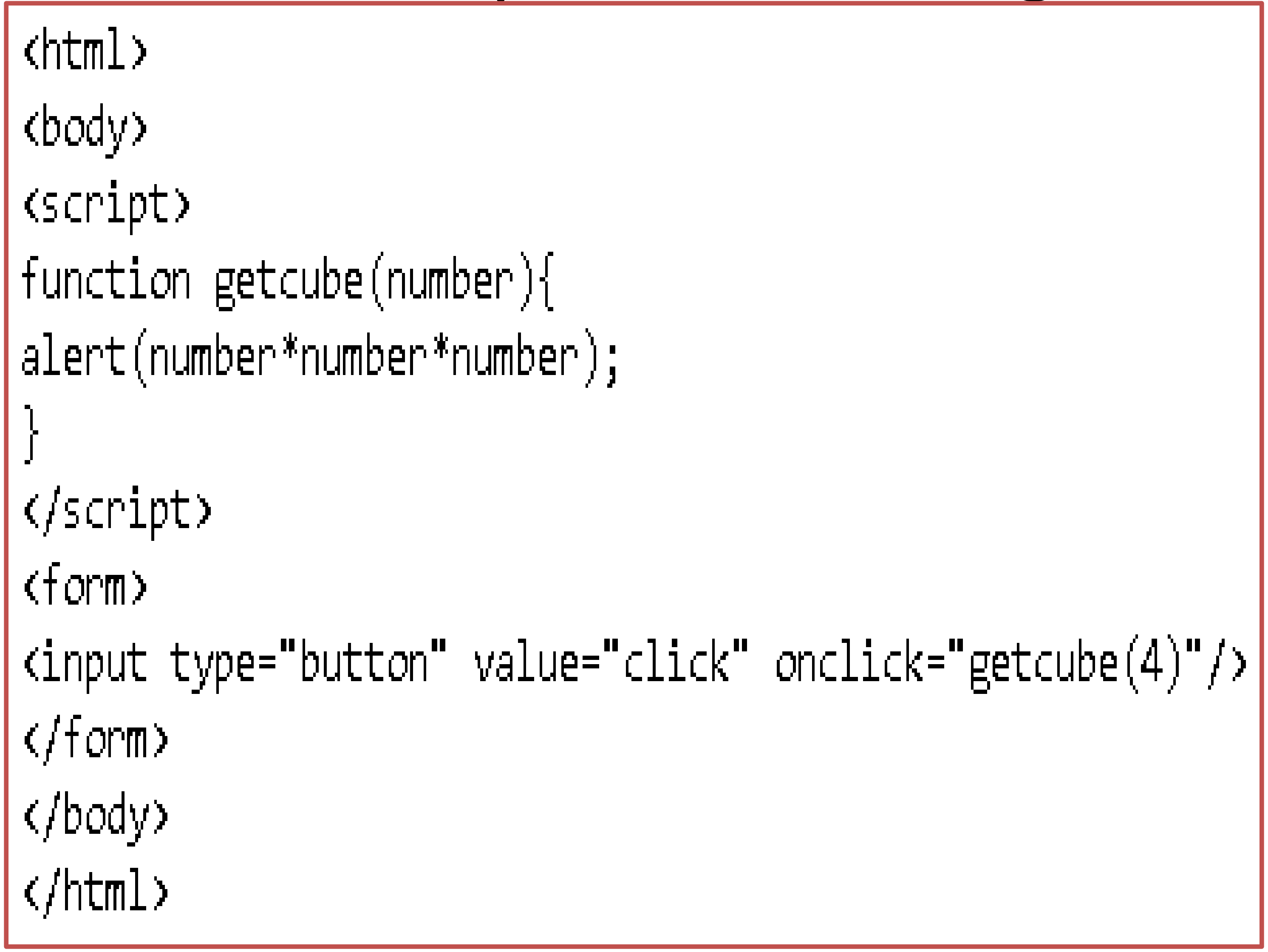
```
<html>
<body>
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
</body>
</html>
```



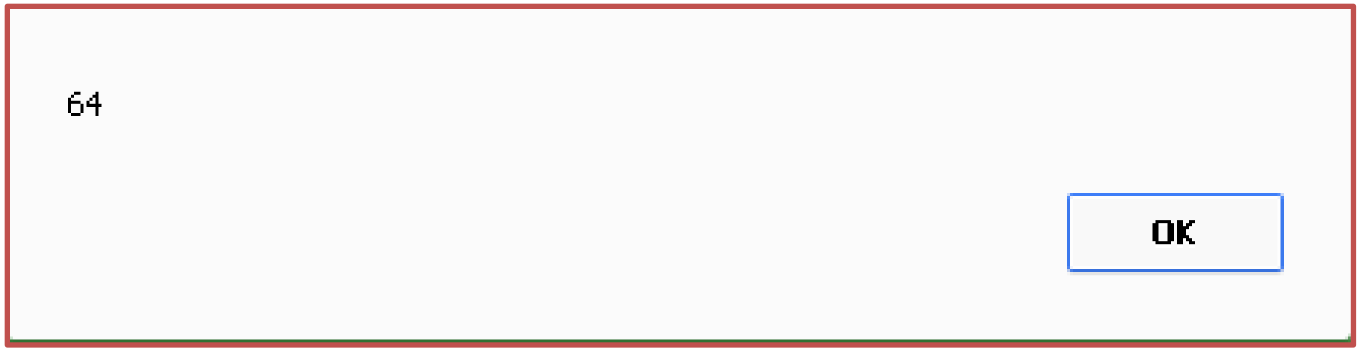

www.javatpoint.com says:

hello! this is message

JavaScript Function Arguments

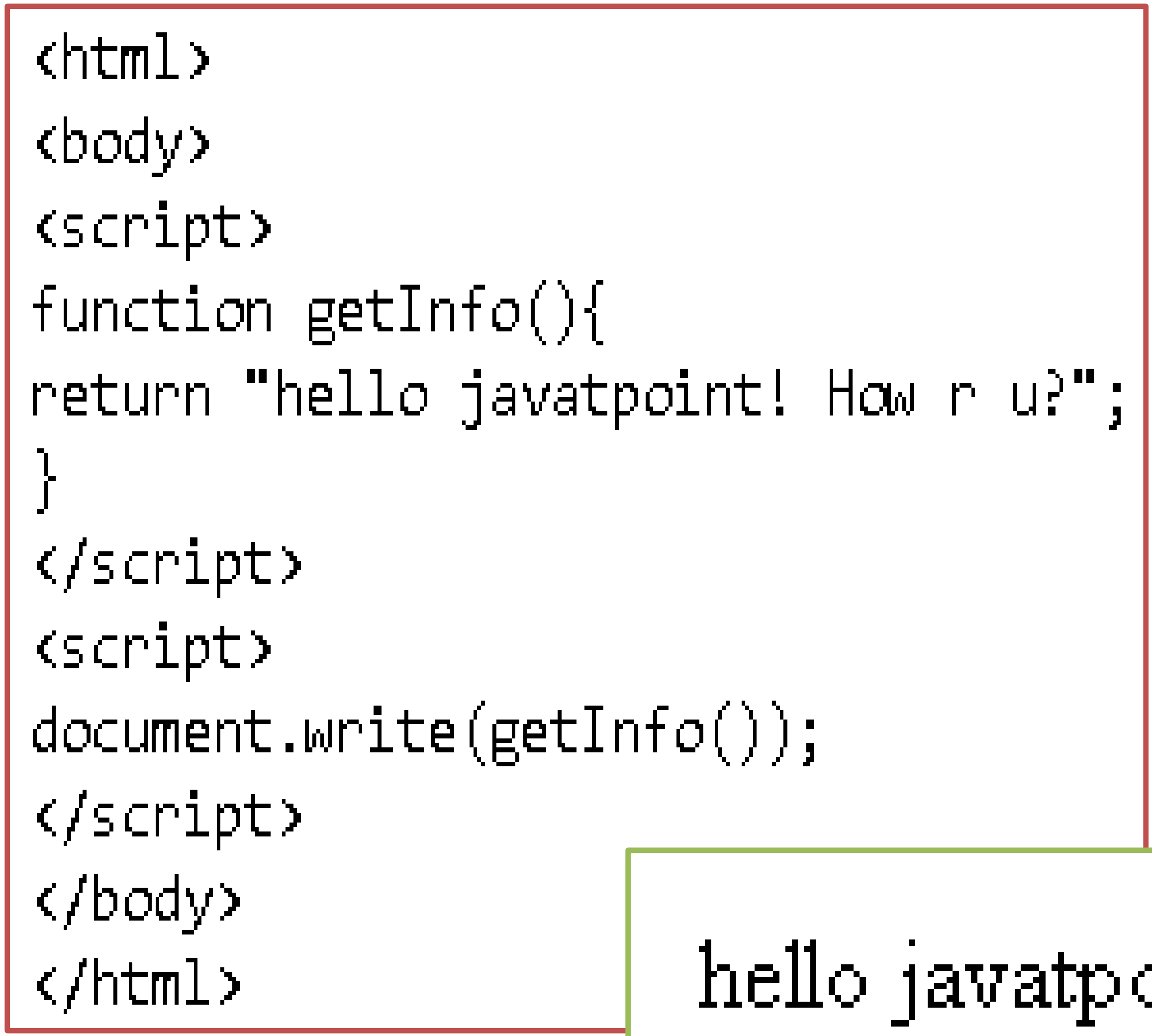


```
<html>
<body>
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
</body>
</html>
```

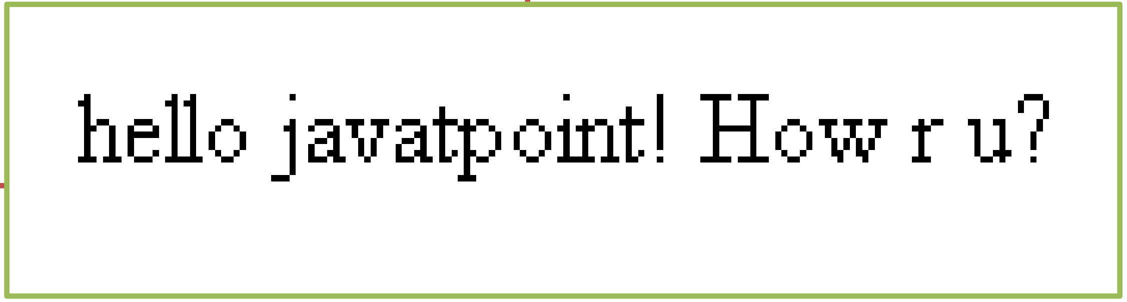


64

Function with Return Value



```
<html>
<body>
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
</body>
</html>
```



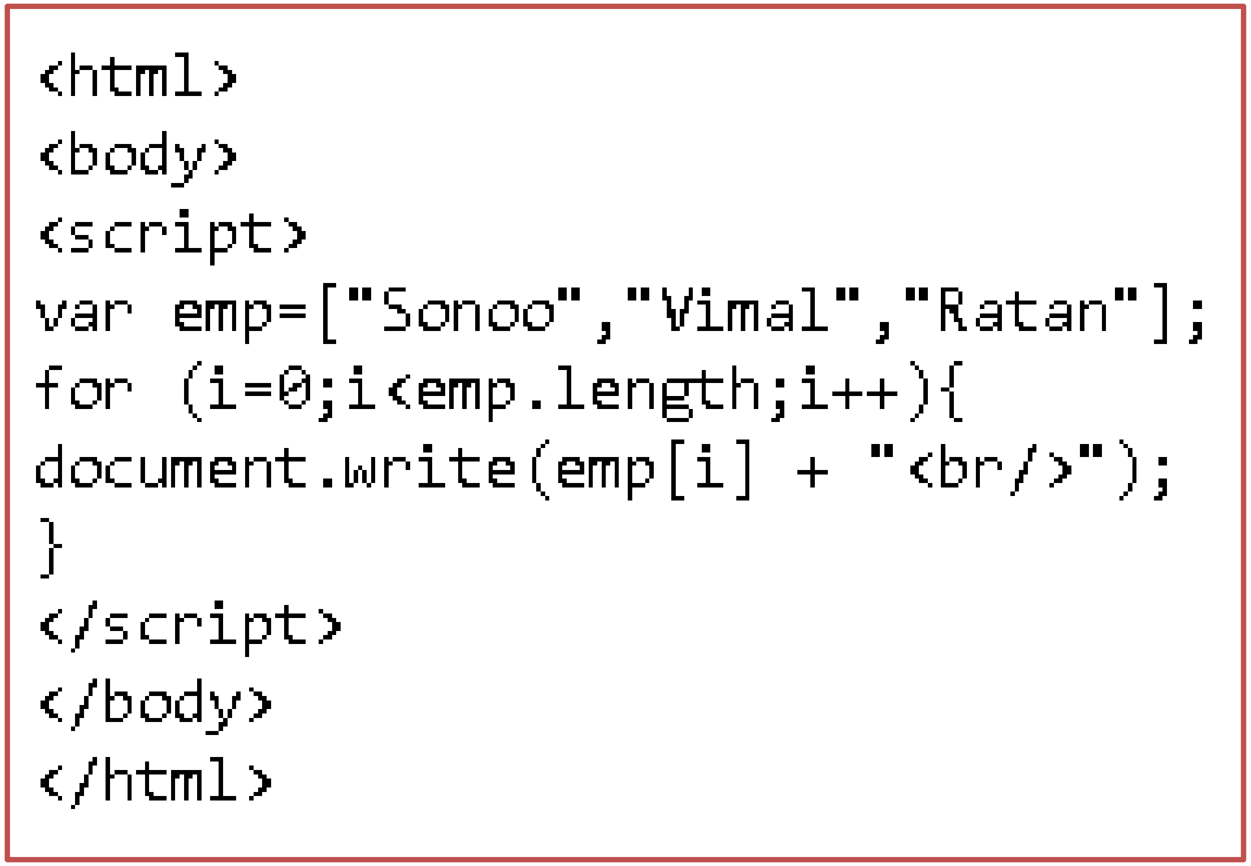
```
hello javatpoint! How r u?
```

JavaScript Array

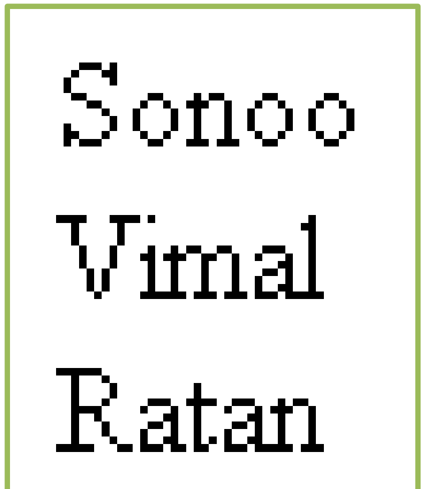
- **JavaScript array** is an object that represents a collection of similar type of elements.
- There are 3 ways to construct array in JavaScript
  - By array literal
  - By creating instance of Array directly (using new keyword)
  - By using an Array constructor (using new keyword)

1) JavaScript array literal:

var arrayname=[value1,value2.....valueN];



```
<html>
<body>
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
</body>
</html>
```

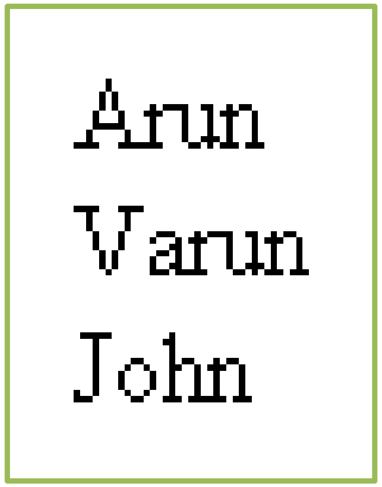


```
Sonoo
Vimal
Ratan
```

JavaScript Array directly (new keyword)

```
var arrayname=new Array();
```

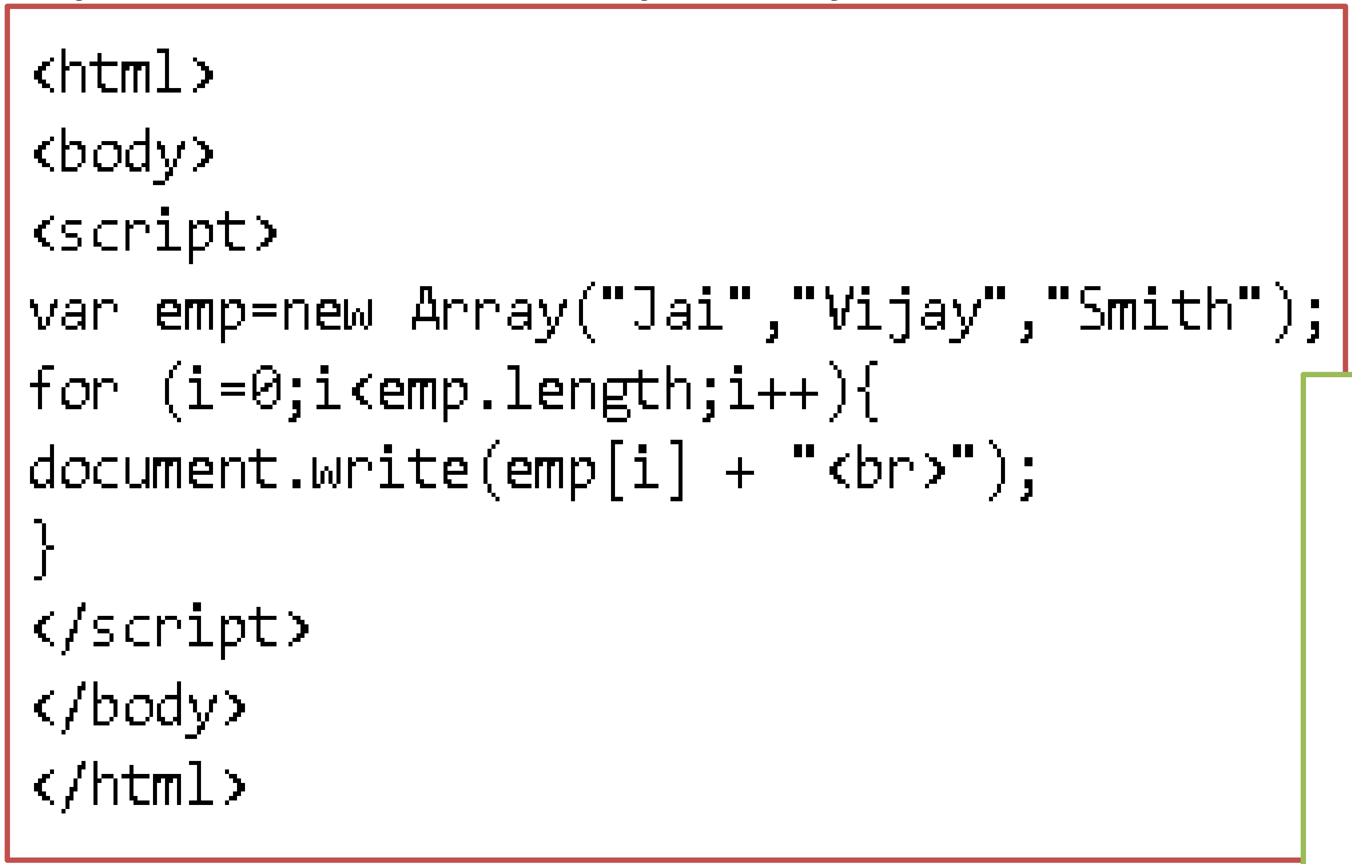
```
<html>
<body>
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```



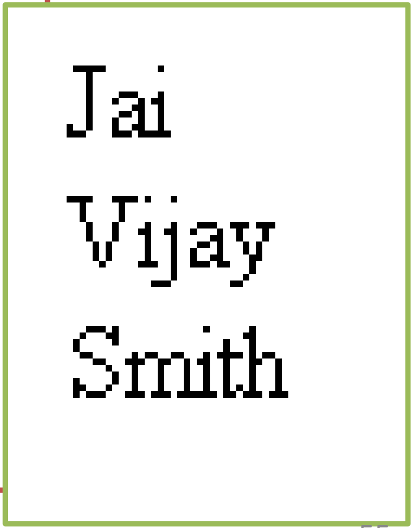
```
Arun
Varun
John
```

JavaScript array constructor (new keyword)

- create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.



```
<html>
<body>
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```



```
Jai
Vijay
Smith
```

Thank You