

# JavaScript I

Overview, Syntax, Placement, Variables, Operators

(Based on Tutorialpoints.com)

# 1. OVERVIEW

## What is JavaScript?

---

JavaScript is a dynamic computer programming language. 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.

JavaScript was first known as **LiveScript**, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The [ECMA-262 Specification](#) defined a standard version of the core JavaScript language.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.

## Client-Side JavaScript

---

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

## Advantages of JavaScript

The merits of using JavaScript are:

- **Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **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

We cannot treat JavaScript as a full programming language. It lacks the following important features:

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

Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

## Where is JavaScript Today?

The ECMAScript Edition 5 standard will be the first update to be released in over four years. JavaScript 2.0 conforms to Edition 5 of the ECMAScript standard, and the difference between the two is extremely minor.

The specification for JavaScript 2.0 can be found on the following site:  
<http://www.ecmascript.org/>

Today, Netscape's JavaScript and Microsoft's JScript conform to the ECMAScript standard, although both the languages still support the features that are not a part of the standard.

## 2. SYNTAX

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:

- **Language:** This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript syntax will look as follows.

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

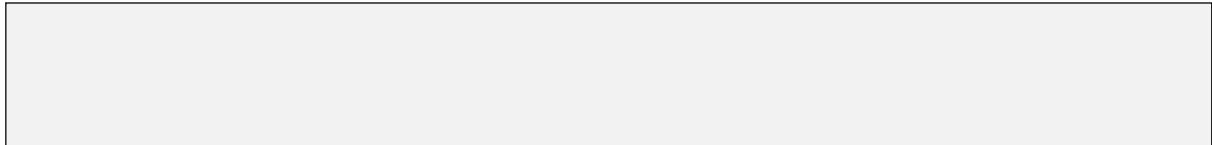
### Your First JavaScript Code

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a **"//-->"**. Here **"//"** signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function **document.write** which writes a string into our HTML document.

This function can be used to write text, HTML, or both. Take a look at the following code.

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

This code will produce the following result:



## **Whitespace and Line Breaks**

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

## **Semicolons are Optional**

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="javascript" type="text/javascript">
<!--
    var1 = 10
    var2 = 20
//-->
</script>
```

But when formatted in a single line as follows, you must use semicolons:

```
<script language="javascript" type="text/javascript">
<!--
    var1 = 10; var2 = 20;
//-->
</script>
```

**Note:** It is a good programming practice to use semicolons.

## **Case Sensitivity**

---

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

**NOTE:** Care should be taken while writing variable and function names in JavaScript.

## **Comments in JavaScript**

---

JavaScript supports both C-style and C++-style comments. Thus:

- Any text between a `//` and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters `/*` and `*/` is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence `<!--`. JavaScript treats this as a single-line comment, just as it does the `//` comment.
- 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 multiline comment in JavaScript
 * It is very similar to comments in C Programming
 */
//-->
</script>
```

# 3. PLACEMENT

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

In the following section, we will see how we can place JavaScript in an HTML file in different ways.

## 1. JavaScript in <head>...</head> Section

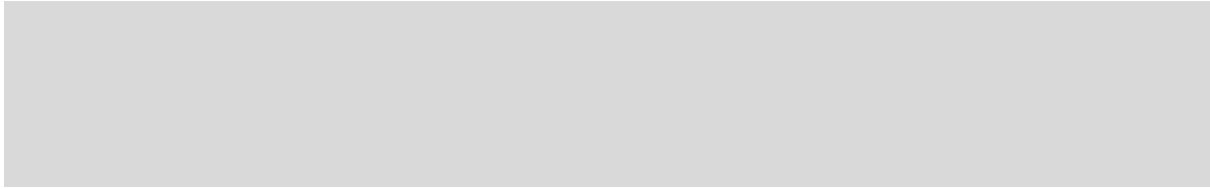
---

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>
Click here for the result
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```



This code will produce the following results:

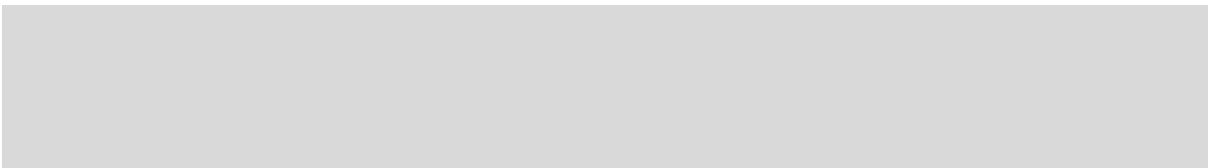


## **2. JavaScript in <body>...</body> Section**

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("Hello World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```

This code will produce the following results:



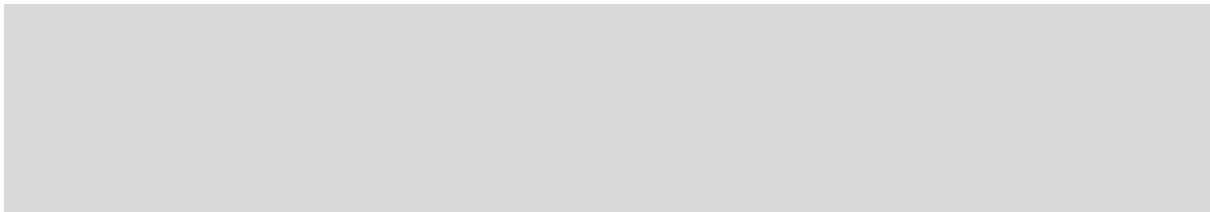
## **3. JavaScript in <body> and <head> Sections**

You can put your JavaScript code in <head> and <body> section altogether as follows.

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

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

This code will produce the following result.



## 4. 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>
```

To use JavaScript from an external file source, you need to write all your JavaScript source code in a simple text file with the extension ".js" and then include that file as shown above.

For example, you can keep the following content in **filename.js** file and then you can use **sayHello** function in your HTML file after including the filename.js file.

```
function sayHello() {
    alert("Hello World")
}
```

# 4. VARIABLES

## JavaScript Datatypes

---

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types:

- **Numbers**, e.g., 123, 120.50 etc.
- **Strings** of text, e.g. "This text string" etc.
- **Boolean**, e.g. true or false.

JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as **object**. We will cover objects in detail in a separate chapter.

**Note:** Java does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values. JavaScript represents numbers using the 64-bit floating-point format defined by the IEEE 754 standard.

## JavaScript Variables

---

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
<!--
var money;
var name;
//-->
</script>
```

You can also declare multiple variables with the same **var** keyword as follows:

```
<script type="text/javascript">
<!--
var money, name;
//-->
</script>
```

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
<!--
var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

**Note:** Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

## **JavaScript Variable Scope**

---

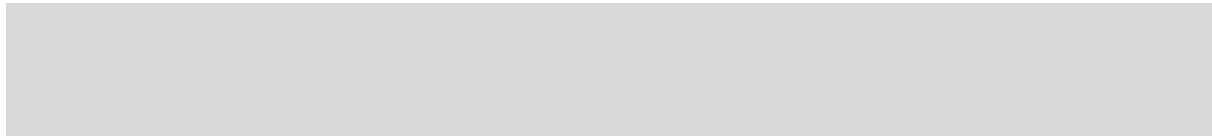
The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **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.

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

It will produce the following result:



## JavaScript Variable Names

---

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, **break** or **boolean** variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **\_123test** is a valid one.
- JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

## **JavaScript Reserved Words**

---

A list of all the reserved words in JavaScript are 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	

# 5. OPERATORS

## What is an Operator?

---

Let us take a simple expression **4 + 5 is equal to 9**. Here 4 and 5 are called **operands** and '+' is called the **operator**. JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Let's have a look at all the operators one by one.

## Arithmetic Operators

---

JavaScript supports the following arithmetic operators:

Assume variable A holds 10 and variable B holds 20, then:

S. No.	Operator and Description
1	<b>+ (Addition)</b> Adds two operands <b>Ex:</b> A + B will give 30
2	<b>- (Subtraction)</b> Subtracts the second operand from the first <b>Ex:</b> A - B will give -10
3	<b>* (Multiplication)</b> Multiply both operands <b>Ex:</b> A * B will give 200
4	<b>/ (Division)</b>



	Divide the numerator by the denominator <b>Ex:</b> B / A will give 2
5	<b>% (Modulus)</b> Outputs the remainder of an integer division <b>Ex:</b> B % A will give 0
6	<b>++ (Increment)</b> Increases an integer value by one <b>Ex:</b> A++ will give 11
7	<b>-- (Decrement)</b> Decreases an integer value by one <b>Ex:</b> A-- will give 9

**Note:** Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

### Example

The following code shows how to use arithmetic operators in JavaScript.

```
<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>

<p>Set the variables to different values and then try...</p>
</body>
</html>

```

## Output



## Comparison Operators

---

JavaScript supports the following comparison operators:

Assume variable A holds 10 and variable B holds 20, then:

S.No	Operator and Description
1	<b>== (Equal)</b> Checks if the value of two operands are equal or not, if yes, then the condition becomes true. <b>Ex:</b> (A == B) is not true.
2	<b>!= (Not Equal)</b> Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. <b>Ex:</b> (A != B) is true.
3	<b>&gt; (Greater than)</b> Checks if the value of the left operand is greater than the value of

	<p>the right operand, if yes, then the condition becomes true.</p> <p><b>Ex:</b> (A &gt; B) is not true.</p>
4	<p><b>&lt; (Less than)</b></p> <p>Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true.</p> <p><b>Ex:</b> (A &lt; B) is true.</p>
5	<p><b>&gt;= (Greater than or Equal to)</b></p> <p>Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.</p> <p><b>Ex:</b> (A &gt;= B) is not true.</p>
6	<p><b>&lt;= (Less than or Equal to)</b></p> <p>Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.</p> <p><b>Ex:</b> (A &lt;= B) is true.</p>

### Example

The following code shows how to use comparison operators in JavaScript.

```
<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>
```

```
<p>Set the variables to different values and different operators and  
then try...</p>  
</body>  
</html>
```

## Output



## Logical Operators

---

JavaScript supports the following logical operators:

Assume variable A holds 10 and variable B holds 20, then:

S.No	Operator and Description
1	<b>&amp;&amp; (Logical AND)</b> If both the operands are non-zero, then the condition becomes true. <b>Ex:</b> (A && B) is true.
2	<b>   (Logical OR)</b> If any of the two operands are non-zero, then the condition becomes true. <b>Ex:</b> (A    B) is true.
3	<b>! (Logical NOT)</b> Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. <b>Ex:</b> ! (A && B) is false.

## Example

Try the following code to learn how to implement Logical Operators in JavaScript.

```
<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);

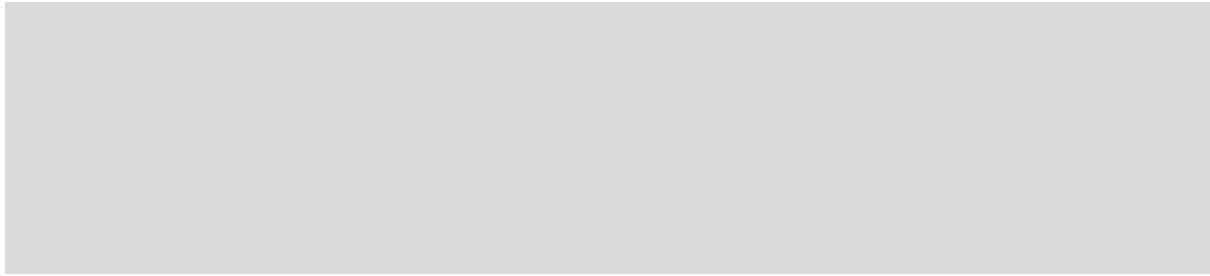
document.write("(!(a && b) => ");
result = (!(a && b));
document.write(result);
document.write(linebreak);

//-->
</script>

<p>Set the variables to different values and different operators and
then try...</p>

</body>
</html>
```

## Output



## Assignment Operators

---

JavaScript supports the following assignment operators:

S.No	Operator and Description
1	<b>= (Simple Assignment )</b> Assigns values from the right side operand to the left side operand <b>Ex:</b> $C = A + B$ will assign the value of $A + B$ into $C$
2	<b>+= (Add and Assignment)</b> It adds the right operand to the left operand and assigns the result to the left operand. <b>Ex:</b> $C += A$ is equivalent to $C = C + A$
3	<b>-= (Subtract and Assignment)</b> It subtracts the right operand from the left operand and assigns the result to the left operand. <b>Ex:</b> $C -= A$ is equivalent to $C = C - A$
4	<b>*= (Multiply and Assignment)</b> It multiplies the right operand with the left operand and assigns the result to the left operand. <b>Ex:</b> $C *= A$ is equivalent to $C = C * A$
5	<b>/= (Divide and Assignment)</b> It divides the left operand with the right operand and assigns the result to the left operand. <b>Ex:</b> $C /= A$ is equivalent to $C = C / A$



6	<p><b>%= (Modules and Assignment)</b></p> <p>It takes modulus using two operands and assigns the result to the left operand.</p> <p><b>Ex:</b> C %= A is equivalent to C = C % A</p>
---	--

**Note:** Same logic applies to Bitwise operators, so they will become <<=, >>=, >>=, &=, |= and ^=.

### Example

Try the following code to implement assignment operator in JavaScript.

```
<html>
<body>

<script type="text/javascript">
<!--
var a = 33;
var b = 10;
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);

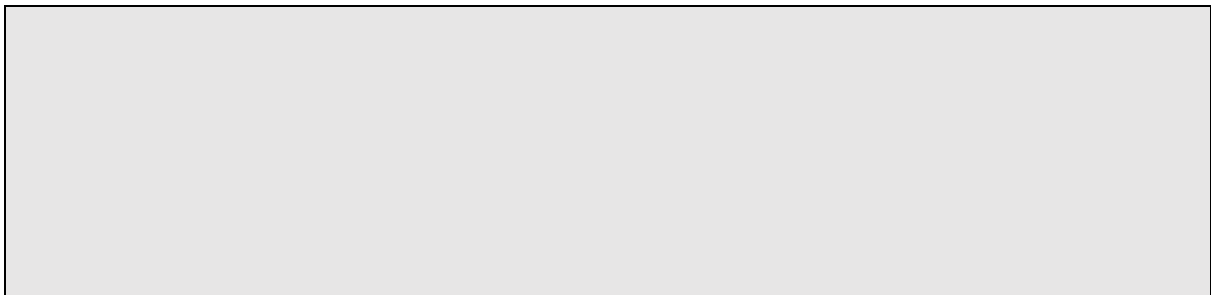
document.write("Value of a => (a /= b) => ");
result = (a /= b);
document.write(result);
document.write(linebreak);

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

//-->
</script>

<p>Set the variables to different values and different operators and
then try...</p>
</body>
</html>
```

## Output



## Miscellaneous Operators

---

We will discuss two operators here that are quite useful in JavaScript: the **conditional operator** (**? :**) and the **typeof operator**.

### Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

S.No	Operator and Description
1	<b>? : (Conditional )</b> If Condition is true? Then value X : Otherwise value Y

#### Example

Try the following code to understand how the Conditional Operator works in JavaScript.

```
<html>
<body>

<script type="text/javascript">
<!--
var a = 10;
var b = 20;
var linebreak = "<br />";

document.write ("((a > b) ? 100 : 200) => ");
result = (a > b) ? 100 : 200;
document.write(result);
document.write(linebreak);

document.write ("((a < b) ? 100 : 200) => ");
```

```

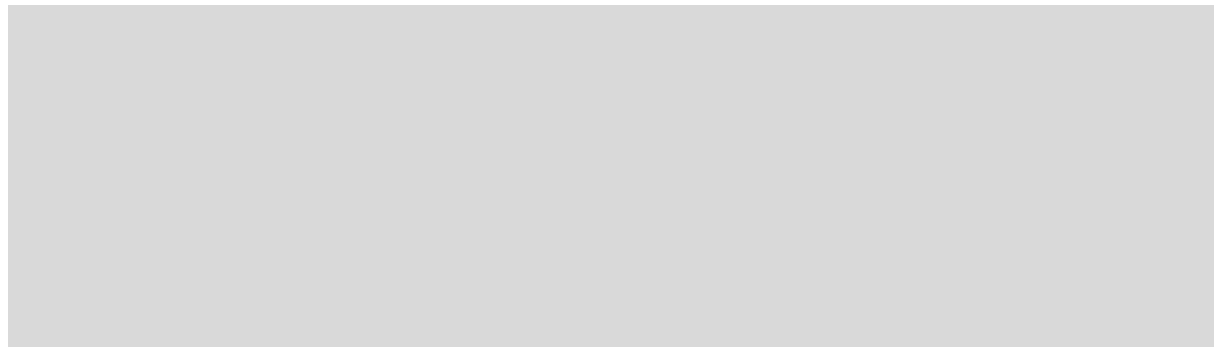
result = (a < b) ? 100 : 200;
document.write(result);
document.write(linebreak);

//-->
</script>

<p>Set the variables to different values and different operators and
then try...</p>
</body>
</html>

```

## Output



## 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"

### Example

The following code shows how to implement **typeof** operator.

```
<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);
document.write(linebreak);

//-->
</script>

<p>Set the variables to different values and different operators and
then try...</p>
</body>
</html>
```

### Output



