Learning Outcome

After completing this module, a student will be able to:

* Summarize variable naming rules and JavaScript data types.
* Identify expressions and operators.
* Summarize flow control.
* Demonstrate objects and arrays usage
* Define functions and methods.

# Introduction to JavaScript

## What is JavaScript?

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage livelier and more interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

JavaScript is a scripting language that is used to create and manage dynamic web pages, basically anything that moves on your screen without requiring you to refresh your browser. It can be anything from animated graphics to an automatically generated Facebook timeline.

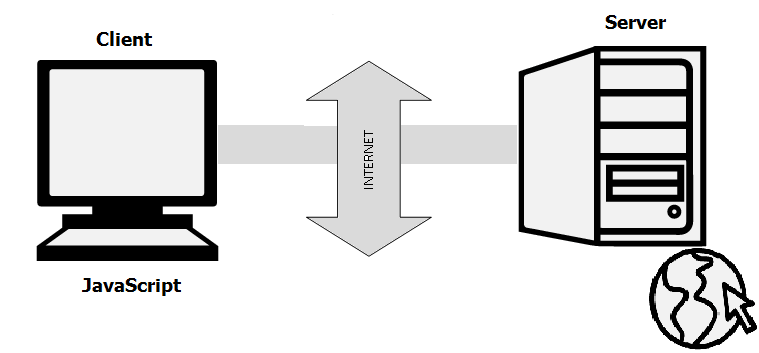


Image 1: what is java script

Reference: https://www.guru99.com/images/JavaScript/javascript1\_1.png

When most people get interested in web development, they start with good old HTML and CSS. From there, they move on to JavaScript, which makes sense, because, these three elements together form the backbone of web development.

**HTML** is the structure of your page like the headers, the body text, any images you want to include. It basically defines the contents of a web page.

**CSS** controls how that page looks (it’s what you’ll use to customize fonts, background colors, etc.).

**JavaScript** is the third element. Once you’ve created your structure (HTML) and your aesthetic vibe (CSS), JavaScript makes your site dynamic (automatically updateable).

## Why JavaScript?

JavaScript is an essential programming language, almost compulsory to learn for students or software developers that are gravitated towards web development. Wondering why? Here’s the answer:

* JavaScript is the most popular programming language in the world and that makes it a default choice for web development. There are many frameworks available which you can use to create web applications once you have learned JavaScript.
* JavaScript offers lots of flexibility. You can create stunning and fast web applications with tons of customizations to provide users with the most relevant graphical user interface.
* JavaScript is now also used in mobile app development, desktop app development, and game development. This opens many possibilities for you as a JavaScript developer.
* Due to the high demand in the industry, there are tons of job growth opportunities and high pay for those who know JavaScript.
* The incredible thing about JavaScript is that you can find tons of frameworks and libraries already developed, which can be used directly in web development. That reduces the development time and enhances the graphical user interface.

## What is JavaScript Used For?

JavaScript is used in various fields from the web to servers, and here’s a quick list of the significant areas it’s used in:



Image 2: JavaScript Application

Reference: https://www.simplilearn.com/ice9/free\_resources\_article\_thumb/js-app.JPG

* **Web Applications:** JavaScript is used for adding interactivity and automation to websites. So, if you want your web application to be anything more than just a static page of contents, you’ll probably need to do some “JavaScripting.”
* **Mobile Applications:** JavaScript isn’t just for developing web applications; it is also used for developing applications for phones and tablets. With frameworks like React Native, you can develop full-fledged mobile applications with all those fancy animations.
* **Web-based Games:** If you’ve ever played a game directly on the web browser, JavaScript was probably used to make that happen.
* **Back-end Web Development:** JavaScript has traditionally been used for developing the front-end parts of a web application. However, with the introduction of NodeJS, a prevalent back-end JavaScript framework, things have changed. And now, JavaScript is used for developing the back-end structure also.

## Features of JavaScript

There are following features of JavaScript:

* All popular web browsers support JavaScript as they provide built-in execution environments.
* JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
* JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
* JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
* It is a light-weighted and interpreted language.
* It is a case-sensitive language.
* JavaScript is supportable in several operating systems including, Windows, macOS, etc.
* It provides good control to the users over the web browsers.

## 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

<script>

document.write("Hello JavaScript by JavaScript");

</script>

JavaScript example is easy to code. JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.

Let’s create the first JavaScript example.

<script type="text/javascript">

document.write("JavaScript is a simple language for javatpoint learners");

</script>

The script tag specifies that we are using JavaScript.

The text/javascript is the content type that provides information to the browser about the data.

The document.write() function is used to display dynamic content through JavaScript. We will learn about document object in detail later.

**3 Places to put JavaScript code**

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)
4. **JavaScript Example: code between the body tag**

In the above example, we have displayed the dynamic content using JavaScript. Let’s see the simple example of JavaScript that displays alert dialog box.

<script type="text/javascript">

alert("Hello Javatpoint");

</script>

1. **JavaScript Example: code between the head tag**

Let’s see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function\_name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

<html>

<head>

<script type="text/javascript">

function msg(){

alert("Hello Javatpoint");

}

</script>

</head>

<body>

<p>Welcome to JavaScript</p>

<form>

<input type="button" value="click" onclick="msg()"/>

</form>

</body>

</html>

1. **External JavaScript file**

We can create external JavaScript file and embed it in many html page.It provides code re usability because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let's create an external JavaScript file that prints Hello Javatpoint in a alert dialog box.

message.js

function msg(){

alert("Hello Javatpoint");

}

Let's include the JavaScript file into html page. It calls the JavaScript function on button click.

index.html

<html>

<head>

<script type="text/javascript" src="message.js"></script>

</head>

<body>

<p>Welcome to JavaScript</p>

<form>

<input type="button" value="click" onclick="msg()"/>

</form>

</body>

</html>

## JavaScript Syntax

The syntax of JavaScript is the set of rules that define a correctly structured JavaScript program.

A JavaScript consists of JavaScript statements that are placed within the <script></script> HTML tags in a web page, or within the external JavaScript file having .js extension.

The following example shows how JavaScript statements look like:

var x = 5;

var y = 10;

var sum = x + y;

document.write(sum); // Prints variable value

## JavaScript Variables

### **What is Variable?**

Variables are fundamental to all programming languages. Variables are used to store data, like string of text, numbers, etc. The data or value stored in the variables can be set, updated, and retrieved whenever needed. In general, variables are symbolic names for values.

You can create a variable with the var keyword, whereas the assignment operator (=) is used to assign value to a variable, like this: var varName = value;

**Example:**

var name = "Peter Parker";

var age = 21;

var isMarried = false;

In the above example we have created three variables, first one has assigned with a string value, the second one has assigned with a number, whereas the last one assigned with a boolean value. Variables can hold different types of data, we'll learn about them in later chapter.

In JavaScript, variables can also be declared without having any initial values assigned to them. This is useful for variables which are supposed to hold values like user inputs.

**Example:**

// Declaring Variable

var userName;

// Assigning value

userName = "Clark Kent";

### **Declaring Multiple Variables at Once**

In addition, you can also declare multiple variables and set their initial values in a single statement. Each variable are separated by commas, as demonstrated in the following example:

**Example**

// Declaring multiple Variables

var name = "Peter Parker", age = 21, isMarried = false;

/\* Longer declarations can be written to span

multiple lines to improve the readability \*/

var name = "Peter Parker",

age = 21,

isMarried = false;

### **Naming Conventions for JavaScript Variables**

These are the following rules for naming a JavaScript variable:

* A variable name must start with a letter, underscore (\_), or dollar sign ($).
* A variable name cannot start with a number.
* A variable name can only contain alpha-numeric characters (A-z, 0-9) and underscores.
* A variable name cannot contain spaces.
* A variable name cannot be a JavaScript keyword or a JavaScript reserved word.

**Note:** Variable names in JavaScript are case sensitive, it means $myvar and $myVar are two different variables. So be careful while defining variable names.

# Javascript Datatype

JavaScript includes data types similar to other programming languages like Java or C#. JavaScript is dynamic and loosely typed language. It means you don't require to specify a type of a variable. A variable in JavaScript can be assigned any type of value, as shown in the following example.

Example: Loosely Typed Variables

var myvariable = 1; // numeric value

myvariable = 'one'; // string value

myvariable = 1.1; // decimal value

myvariable = true; // Boolean value

myvariable = null; // null value

In the above example, different types of values are assigned to the same variable to demonstrate loosely typed characteristics of JavaScript. Different values 1, 'one', 1.1, true are examples of different data types.

JavaScript includes primitive and non-primitive data types.

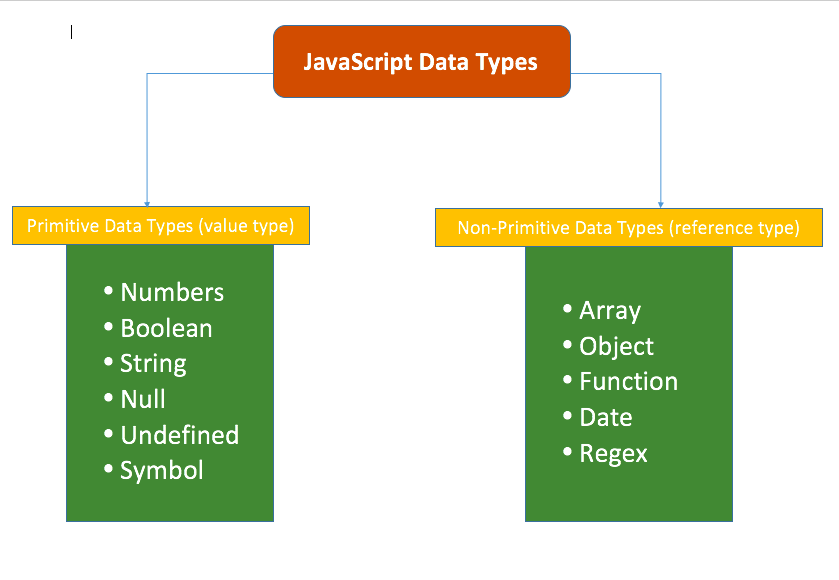


Image 3: JS Datatype

Reference: https://miro.medium.com/max/839/1\*EJVSM8Mtm6kPmPYMTP5gug.png

### **Primitive Data Types**

The primitive data types are the lowest level of the data value in JavaScript. The typeof operator can be used with primitive data types to know the type of a value.

The followings are primitive data types in JavaScript:

|  |  |
| --- | --- |
| Data Type | Description |
| String | String is a textual content wrapped inside ' ' or " " or ` ` (tick sign).  Example: 'Hello World!', "This is a string", etc. |
| Number | Number is a numeric value.  Example: 100, 4521983, etc. |
| BigInt | BigInt is a numeric value in the arbitrary precision format.  Example: 453889879865131n, 200n, etc. |
| Boolean | Boolean is a logical data type that has only two values, true or false. |
| Null | A null value denotes an absence of value.  Example: var str = null; |
| Undefined | undefined is the default value of a variable that has not been assigned any value.  Example: In the variable declaration, var str;, there is no value assigned to str. So, the type of str can be check using typeof(str) which will return undefined. |

### **Non-primitive data types**

The non-primitive data types contain some kind of structure with primitive data.

|  |  |
| --- | --- |
| Data Type | Description |
| Object | An object holds multiple values in terms of properties and methods.  Example:  var person = {  firstName: "James",  lastName: "Bond",  age: 15  }; |
| Date | Date object represents date & time including days, months, years, hours, minutes, seconds and milliseconds.  Example: var today = new Date("25 July 2021"); |
| Array | An array stores multiple values using special syntax.  Example: var nums = [1, 2, 3, 4]; |

# Javascript Operators

JavaScript includes operators same as other languages. An operator performs some operation on single or multiple operands (data value) and produces a result. For example, in 1 + 2, the + sign is an operator and 1 is left side operand and 2 is right side operand. The + operator performs the addition of two numeric values and returns a result.

**Syntax:**

<Left operand> operator <right operand>

<Left operand> operator

JavaScript includes following categories of operators:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators
7. **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 |

1. **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 |

1. **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 |

1. **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 |

1. **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 |

1. **Special Operators**

The following operators are known as JavaScript 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. |

# Javascript Condition

## JavaScript If-else

The JavaScript if-else statement is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement
4. **If statement**

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

**Syntax:**

if(expression){

//content to be evaluated

}

**Flowchart of JavaScript If statement**

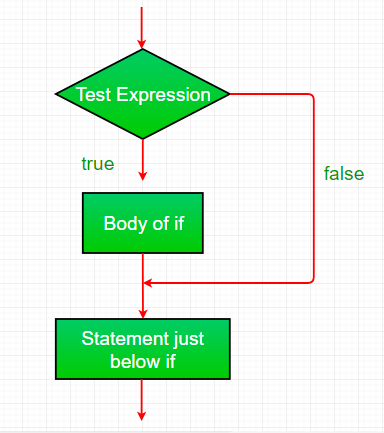
****

Image 4: if statement

Reference: https://media.geeksforgeeks.org/wp-content/uploads/if.png

Let’s see the simple example of if statement in javascript.

<script>

var a=20;

if(a>10){

document.write("value of a is greater than 10");

}

</script>

Output of the above example

value of a is greater than 10

1. **If...else Statement**

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

**Syntax**

if(expression){

//content to be evaluated if condition is true

}

else{

//content to be evaluated if condition is false

}

**Flowchart of JavaScript If...else statement**

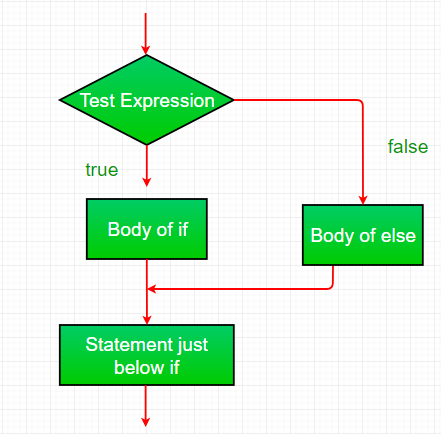
****

Image 5: if-else

Reference: https://media.geeksforgeeks.org/wp-content/uploads/if-else.png

**Let’s see the example of if-else statement in JavaScript to find out the even or odd number.**

<script>

var a=20;

if(a%2==0){

document.write("a is even number");

}

else{

document.write("a is odd number");

}

</script>

Output of the above example

a is even number

1. **If...else if statement**

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

**Syntax:**

if(expression1){

//content to be evaluated if expression1 is true

}

else if(expression2){

//content to be evaluated if expression2 is true

}

else if(expression3){

//content to be evaluated if expression3 is true

}

else{

//content to be evaluated if no expression is true

}

**Flowchart of JavaScript If...else statement**

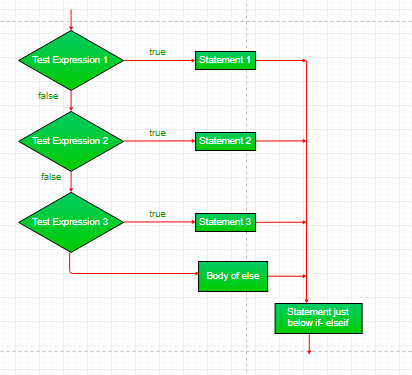
****

Image 5: if-else--if

Reference: https://media.geeksforgeeks.org/wp-content/uploads/if-elseif.png

Let’s see the simple example of if else if statement in javascript.

<script>

var a=20;

if(a==10){

document.write("a is equal to 10");

}

else if(a==15){

document.write("a is equal to 15");

}

else if(a==20){

document.write("a is equal to 20");

}

else{

document.write("a is not equal to 10, 15 or 20");

}

</script>

Output of the above example

a is equal to 20

## JavaScript Switch

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement that we have learned in previous page. But it is convenient than if..else..if because it can be used with numbers, characters etc.

**Syntax:**

switch(expression){

case value1:

code to be executed;

break;

case value2:

code to be executed;

break;

......

default:

code to be executed if above values are not matched;



Image: Switch

Reference: https://media.geeksforgeeks.org/wp-content/uploads/switch.png

Let’s understand the behavior of switch statement in JavaScript.

<script>

var grade='B';

var result;

switch(grade){

case 'A':

result+=" A Grade";

case 'B':

result+=" B Grade";

case 'C':

result+=" C Grade";

default:

result+=" No Grade";

}

document.write(result);

</script>

Output of the above example

undefined B Grade C Grade No Grade

# Loop Control statement

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

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

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iterations is known. The syntax of for loop is given below.

**Syntax:**

for (initialization; condition; increment)

{

code to be executed

}

Let’s see the simple example of for loop in javascript.

<script>

for (i=1; i<=5; i++)

{

document.write(i + "<br/>")

}

</script>

Output:

1

2

3

4

5

1. **While loop**

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below.

**Syntax:**

while (condition)

{

code to be executed

}

Let’s see the simple example of while loop in javascript.

<script>

var i=11;

while (i<=15)

{

document.write(i + "<br/>");

i++;

}

</script>

**Output:**

11

12

13

14

15

1. **do while loop**

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. but, code is executed at least once whether condition is true or false. The syntax of do while loop is given below.

**Syntax:**

do{

code to be executed

}while (condition);

Let’s see the simple example of do while loop in javascript.

<script>

var i=21;

do{

document.write(i + "<br/>");

i++;

}while (i<=25);

</script>

**Output:**

21

22

23

24

25

1. **for in loop**

JavaScript includes for loop like Java or C#. Use for loop to execute code repeatedly.

Syntax:

for(initializer; condition; iteration)

{

// Code to be executed

}

The for loop requires following three parts.

**Initializer:** Initialize a counter variable to start with

**Condition:** specify a condition that must evaluate to true for next iteration

**Iteration:** increase or decrease counter

Let’s see the simple example of do for loop in javascript.

<script>

// JavaScript program to illustrate for loop

var x;

// for loop begins when x=2

// and runs till x <=4

for (x = 2; x <= 4; x++)

{

document.write("Value of x:" + x + "<br />");

}

< /script>

**Output:**

Value of x:2

Value of x:3

Value of x:4

# 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

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)
4. **JavaScript array literal**

The syntax of creating array using array literal is given below:

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

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

<script>

var emp=["Sonoo","Vimal","Ratan"];

for (i=0;i<emp.length;i++){

document.write(emp[i] + "<br/>");

}

</script>

The .length property returns the length of an array.

**Output**

Sonoo

Vimal

Ratan

1. **JavaScript Array directly (new keyword)**

The syntax of creating array directly is given below:

var arrayname=new Array();

Here, new keyword is used to create instance of array.

Let's see the example of creating array directly.

<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>

**Output**

Arun

Varun

John

1. **JavaScript array constructor (new keyword)**

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

The example of creating object by array constructor is given below.

<script>

var emp=new Array("Jai","Vijay","Smith");

for (i=0;i<emp.length;i++){

document.write(emp[i] + "<br>");

}

</script>

**Output**

Jai

Vijay

Smith

### **JavaScript Array Methods**

Let's see the list of JavaScript array methods with their description.

|  |  |
| --- | --- |
| Methods | Description |
| [concat()](https://www.javatpoint.com/javascript-array-concat-method) | It returns a new array object that contains two or more merged arrays. |
| [copywithin()](https://www.javatpoint.com/javascript-array-copywithin-method) | It copies the part of the given array with its own elements and returns the modified array. |
| [entries()](https://www.javatpoint.com/javascript-array-entries-method) | It creates an iterator object and a loop that iterates over each key/value pair. |
| [every()](https://www.javatpoint.com/javascript-array-every-method) | It determines whether all the elements of an array are satisfying the provided function conditions. |
| [flat()](https://www.javatpoint.com/javascript-array-flat-method) | It creates a new array carrying sub-array elements concatenated recursively till the specified depth. |
| [flatMap()](https://www.javatpoint.com/javascript-array-flatmap-method) | It maps all array elements via mapping function, then flattens the result into a new array. |
| [fill()](https://www.javatpoint.com/javascript-array-fill-method) | It fills elements into an array with static values. |
| [from()](https://www.javatpoint.com/javascript-array-from-method) | It creates a new array carrying the exact copy of another array element. |
| [filter()](https://www.javatpoint.com/javascript-array-filter-method) | It returns the new array containing the elements that pass the provided function conditions. |
| [find()](https://www.javatpoint.com/javascript-array-find-method) | It returns the value of the first element in the given array that satisfies the specified condition. |
| [findIndex()](https://www.javatpoint.com/javascript-array-findindex-method) | It returns the index value of the first element in the given array that satisfies the specified condition. |
| [forEach()](https://www.javatpoint.com/javascript-array-foreach-method) | It invokes the provided function once for each element of an array. |
| [includes()](https://www.javatpoint.com/javascript-array-includes-method) | It checks whether the given array contains the specified element. |
| [indexOf()](https://www.javatpoint.com/javascript-array-indexof-method) | It searches the specified element in the given array and returns the index of the first match. |
| [isArray()](https://www.javatpoint.com/javascript-array-isarray-method) | It tests if the passed value ia an array. |
| [join()](https://www.javatpoint.com/javascript-array-join-method) | It joins the elements of an array as a string. |
| [keys()](https://www.javatpoint.com/javascript-array-keys-method) | It creates an iterator object that contains only the keys of the array, then loops through these keys. |
| [lastIndexOf()](https://www.javatpoint.com/javascript-array-lastindexof-method) | It searches the specified element in the given array and returns the index of the last match. |
| [map()](https://www.javatpoint.com/javascript-array-map-method) | It calls the specified function for every array element and returns the new array |
| [of()](https://www.javatpoint.com/javascript-array-of-method) | It creates a new array from a variable number of arguments, holding any type of argument. |
| [pop()](https://www.javatpoint.com/javascript-array-pop-method) | It removes and returns the last element of an array. |
| [push()](https://www.javatpoint.com/javascript-array-push-method) | It adds one or more elements to the end of an array. |
| [reverse()](https://www.javatpoint.com/javascript-array-reverse-method) | It reverses the elements of given array. |
| [reduce(function, initial)](https://www.javatpoint.com/javascript-array-reduce-method) | It executes a provided function for each value from left to right and reduces the array to a single value. |
| [reduceRight()](https://www.javatpoint.com/javascript-array-reduceright-method) | It executes a provided function for each value from right to left and reduces the array to a single value. |
| [some()](https://www.javatpoint.com/javascript-array-some-method) | It determines if any element of the array passes the test of the implemented function. |
| [shift()](https://www.javatpoint.com/javascript-array-shift-method) | It removes and returns the first element of an array. |
| [slice()](https://www.javatpoint.com/javascript-array-slice-method) | It returns a new array containing the copy of the part of the given array. |
| [sort()](https://www.javatpoint.com/javascript-array-sort-method) | It returns the element of the given array in a sorted order. |
| [splice()](https://www.javatpoint.com/javascript-array-splice-method) | It add/remove elements to/from the given array. |
| [toLocaleString()](https://www.javatpoint.com/javascript-array-tolocalestring-method) | It returns a string containing all the elements of a specified array. |
| [toString ()](https://www.javatpoint.com/javascript-array-tostring-method) | It converts the elements of a specified array into string form, without affecting the original array. |
| [unshift()](https://www.javatpoint.com/javascript-array-unshift-method) | It adds one or more elements in the beginning of the given array. |
| [values()](https://www.javatpoint.com/javascript-array-values-method) | It creates a new iterator object carrying values for each index in the array. |

# Javascript Object

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)
4. **JavaScript Object by object literal**

The syntax of creating object using object literal is given below:

object={property1:value1,property2:value2.....propertyN:valueN}

As you can see, property and value is separated by : (colon).

Let’s see the simple example of creating object in JavaScript.

<script>

emp={id:102,name:"Shyam Kumar",salary:40000}

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

**Output**

102 Shyam Kumar 40000

1. **By creating instance of Object**

The syntax of creating object directly is given below:

var objectname=new Object();

Here, new keyword is used to create object.

Let’s see the example of creating object directly.

<script>

var emp=new Object();

emp.id=101;

emp.name="Ravi Malik";

emp.salary=50000;

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

**Output**

101 Ravi 50000

1. **By using an Object constructor**

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

This keyword refers to the current object.

The example of creating object by object constructor is given below.

<script>

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

}

e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);

</script>

Output

103 Vimal Jaiswal 30000

**Defining method in JavaScript Object**

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

The example of defining method in object is given below.

<script>

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

this.changeSalary=changeSalary;

function changeSalary(otherSalary){

this.salary=otherSalary;

}

}

e=new emp(103,"Sonoo Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);

e.changeSalary(45000);

document.write("<br>"+e.id+" "+e.name+" "+e.salary);

</script>

Output:

103 Sonoo Jaiswal 30000

103 Sonoo Jaiswal 45000

**JavaScript Object Methods**

The various methods of Object are as follows:

|  |  |  |
| --- | --- | --- |
| Sr.No | Methods | Description |
| 1 | [Object.assign()](https://www.javatpoint.com/javascript-object-assign-method) | This method is used to copy enumerable and own properties from a source object to a target object |
| 2 | [Object.create()](https://www.javatpoint.com/javascript-object-create-method) | This method is used to create a new object with the specified prototype object and properties. |
| 3 | [Object.defineProperty()](https://www.javatpoint.com/javascript-object-defineproperty-method) | This method is used to describe some behavioral attributes of the property. |
| 4 | [Object.defineProperties()](https://www.javatpoint.com/javascript-object-defineproperties-method) | This method is used to create or configure multiple object properties. |
| 5 | [Object.entries()](https://www.javatpoint.com/javascript-object-entries-method) | This method returns an array with arrays of the key, value pairs. |
| 6 | [Object.freeze()](https://www.javatpoint.com/javascript-object-freeze-method) | This method prevents existing properties from being removed. |
| 7 | [Object.getOwnPropertyDescriptor()](https://www.javatpoint.com/javascript-object-getownpropertydescriptor-method) | This method returns a property descriptor for the specified property of the specified object. |
| 8 | [Object.getOwnPropertyDescriptors()](https://www.javatpoint.com/javascript-object-getownpropertydescriptors-method) | This method returns all own property descriptors of a given object. |
| 9 | [Object.getOwnPropertyNames()](https://www.javatpoint.com/javascript-object-getownpropertynames-method) | This method returns an array of all properties (enumerable or not) found. |
| 10 | [Object.getOwnPropertySymbols()](https://www.javatpoint.com/javascript-object-getownpropertysymbols-method) | This method returns an array of all own symbol key properties. |
| 11 | [Object.getPrototypeOf()](https://www.javatpoint.com/javascript-object-getprototypeof-method) | This method returns the prototype of the specified object. |
| 12 | [Object.is()](https://www.javatpoint.com/javascript-object-is-method) | This method determines whether two values are the same value. |
| 13 | [Object.isExtensible()](https://www.javatpoint.com/javascript-objects) | This method determines if an object is extensible |
| 14 | [Object.isFrozen()](https://www.javatpoint.com/javascript-objects) | This method determines if an object was frozen. |
| 15 | [Object.isSealed()](https://www.javatpoint.com/javascript-objects) | This method determines if an object is sealed. |
| 16 | [Object.keys()](https://www.javatpoint.com/javascript-objects) | This method returns an array of a given object's own property names. |
| 17 | [Object.preventExtensions()](https://www.javatpoint.com/javascript-object-preventextensions-method) | This method is used to prevent any extensions of an object. |
| 18 | [Object.seal()](https://www.javatpoint.com/javascript-object-seal-method) | This method prevents new properties from being added and marks all existing properties as non-configurable. |
| 19 | [Object.setPrototypeOf()](https://www.javatpoint.com/javascript-object-setprototypeof-method) | This method sets the prototype of a specified object to another object. |
| 20 | [Object.values()](https://www.javatpoint.com/javascript-object-values-method) | This method returns an array of values. |

# JavaScript Function

A function is a block of code that performs a specific task.

Suppose you need to create a program to create a circle and color it. You can create two functions to solve this problem:

* a function to draw the circle
* a function to color the circle

Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

JavaScript also has a huge number of inbuilt functions. For example, Math.sqrt() is a function to calculate the square root of a number.

## User-defined functions.

### **Declaring a Function**

The syntax to declare a function is:

function nameOfFunction () {

// function body

}

* A function is declared using the function keyword.
* The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for your function. For example, if a function is used to add two numbers, you could name the function add or addNumbers.
* The body of function is written within {}.

For example,

// declaring a function named greet()

function greet() {

console.log("Hello there");

}

### **Calling a Function**

In the above program, we have declared a function named greet(). To use that function, we need to call it.

Here's how you can call the above greet() function.

// function call

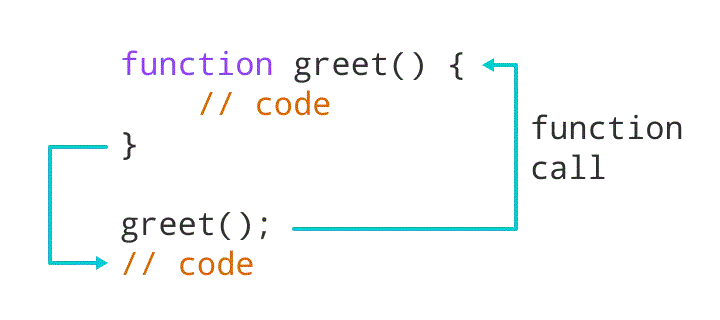
greet();

Image: Working of a Function in JavaScript

Reference:https://cdn.programiz.com/cdn/farfuture/NdxxeWlRfoHMPgdcWPkeVy1wN9MwAgoqoYqZkFQDMFQ/mtime:1591592059/sites/tutorial2program/files/javascript-function-example1.png

Example 1: Display a Text

// program to print a text

// declaring a function

function greet() {

console.log("Hello there!");

}

// calling the function

greet();

Output

Hello there!

### **Function Parameters**

A function can also be declared with parameters. A parameter is a value that is passed when declaring a function.

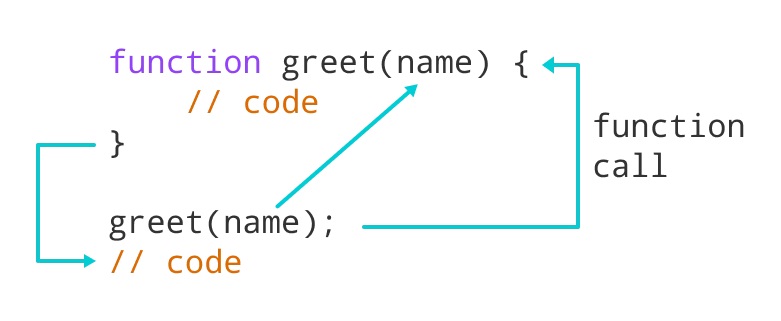


Image: Working of JavaScript Function with parameter

Reference: https://cdn.programiz.com/cdn/farfuture/oAZVf3IqOKOYj\_aJ-IoYQvbJ2CB-B3y4HXSLXBUmYcY/mtime:1591592163/sites/tutorial2program/files/javascript-function-with-parameter.png

Example 2: Function with Parameters

// program to print the text

// declaring a function

function greet(name) {

console.log("Hello " + name + ":)");

}

// variable name can be different

let name = prompt("Enter a name: ");

// calling function

greet(name);

Output

Enter a name: Simon

Hello Simon :)

In the above program, the greet function is declared with a name parameter. The user is prompted to enter a name. Then when the function is called, an argument is passed into the function.

**Note:** When a value is passed when declaring a function, it is called parameter. And when the function is called, the value passed is called argument.

Example 3: Add Two Numbers

// program to add two numbers using a function

// declaring a function

function add(a, b) {

console.log(a + b);

}

// calling functions

add(3,4);

add(2,9);

Output

7

11

In the above program, the add function is used to find the sum of two numbers.

* The function is declared with two parameters a and b.
* The function is called using its name and passing two arguments 3 and 4 in one and 2 and 9 in another.

Notice that you can call a function as many times as you want. You can write one function and then call it multiple times with different arguments.

### **Function Return**

The return statement can be used to return the value to a function call.

The return statement denotes that the function has ended. Any code after return is not executed.

If nothing is returned, the function returns an undefined value.

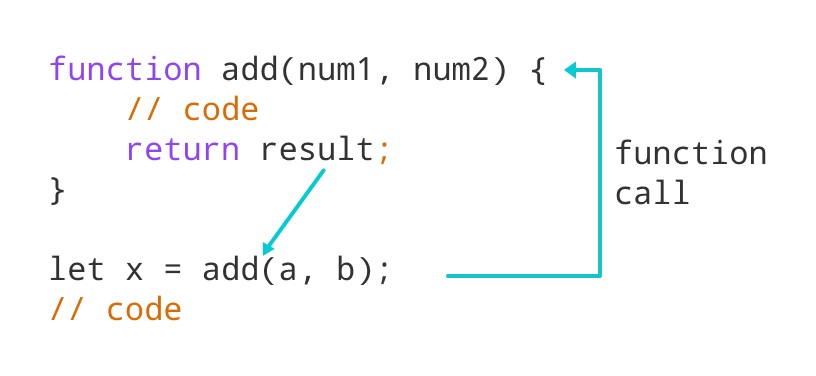


Image: Working of JavaScript Function with return statement

Reference: https://cdn.programiz.com/cdn/farfuture/b4h4Zo5ZYxj-EyfQyao-J5TqbKEefFgqqusPGLWPFS0/mtime:1591786573/sites/tutorial2program/files/javascript-return-statement.png

Example 4: Sum of Two Numbers

// program to add two numbers

// declaring a function

function add(a, b) {

return a + b;

}

// take input from the user

let number1 = parseFloat(prompt("Enter first number: "));

let number2 = parseFloat(prompt("Enter second number: "));

// calling function

let result = add(number1,number2);

// display the result

console.log("The sum is " + result);

Output

Enter first number: 3.4

Enter second number: 4

The sum is 7.4

In the above program, the sum of the numbers is returned by the function using the return statement. And that value is stored in the result variable.

### **Benefits of Using a Function**

* Function makes the code reusable. You can declare it once and use it multiple times.
* Function makes the program easier as each small task is divided into a function.
* Function increases readability.

### **Function Expressions**

In Javascript, functions can also be defined as expressions. For example,

// program to find the square of a number

// function is declared inside the variable

let x = function (num) { return num \* num };

console.log(x(4));

// can be used as variable value for other variables

let y = x(3);

console.log(y);

Output

16

9

In the above program, variable x is used to store the function. Here the function is treated as an expression. And the function is called using the variable name.

The function above is called an anonymous function.

# References

1. <https://www.javatpoint.com/javascript-tutorial>
2. <https://www.guru99.com/introduction-to-javascript.html>
3. <https://www.tutorialrepublic.com/javascript-tutorial/javascript-variables.php>
4. <https://www.tutorialsteacher.com/javascript/javascript-data-types>