[JavaScript Tutorial](http://127.0.0.1:5500/function.html)

* [Home](http://127.0.0.1:5500/function.html)
* [Link](http://127.0.0.1:5500/function.html)
* [Upcoming](http://127.0.0.1:5500/function.html)

Top of Form

Search

Bottom of Form

# **JavaScript Tutorial**

## JavaScript Functions

**JavaScript functions**are used to perform operations. We can call JavaScript function many times to reuse the code.

#### Advantage of JavaScript function

1. Code reusability: We can call a function several times so it save coding.
2. 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 Defination

* A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
* Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
* Function parameters are listed inside the parentheses () in the function definition.
* The code to be executed, by the function, is placed inside curly brackets: {}
* Function arguments are the values received by the function when it is invoked.

#### Type of Functions

###### Simple Function

###### Anonymous Function

###### Arrow Function

###### Call back Function / Self invoke function

##### **1. Simple Function**

##### **Sytax:**

function functionName (Parameter1, Parameter2) //function definition

{

// code to be executed

}

console.log(functionName ()) // function called

##### **Example:**

function square(number)

{

return number \* number; //25

}

console.log(square (5));

##### **2. Anonymous Function**

##### **Sytax:**

variable functionName = function (Parameter1, Parameter2, ..)

{

// code to be executed

}

* An anonymous function is a function without a name. The following shows how to define an anonymous function:
* If you want to create a function and execute it immediately after the declaration, you can declare an anonymous function like this :

##### **Sytax:**

(function() {

console.log('Immediately invoked function execution'); //Immediately invoked function execution

})();

* The following shows an anonymous function that displays a message.

##### **Example:**

let show = function() {

console.log('Anonymous function'); //Anonymous function

};

show();

* Using anonymous passing parameter.

##### **Example:**

var test = function (p1,p2)

{

return p1 \* p2; //45

}

console.log(test (5,9));

* Using anonymous functions as arguments.

##### **Example:**

setTimeout(function() {

console.log('Execute later after 1 second') //Execute later after 1 second

}, 1000);

**In this example, we pass an anonymous function into the setTimeout() function. The setTimeout() function executes this anonymous function one second later.**

##### **3. Arrow Function**

##### **Sytax:**

**variable functionName = (Parameter1, Parameter2, ..) =>**

**{**

**// Function body**

**}**

* **Arrow Functions Return Value by Default:**

##### **Example:**

**test = () => "Hello World!"; //Hello World!**

**test();**

**It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword :**

* **Arrow Function With Parameters:**

##### **Example:**

**test = (p1,p2) => p1\*p2; //45**

**test(5,9);**

##### **4. Call back Function / Self invoke function**

** A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.**

##### **Sytax:**

** In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function.**

##### **Example:**

**// callback function**

**function firstFunc(){**

**console.log("call first...")**

**}**

**// function**

**function secondFunc(para){**

**para()**

**console.log("call second...")**

**}**

**//passing function as an argument**

**secondFunc(firstFunc);**

**// callback function**

**function firstFunc(){**

**console.log("call first...")**

**}**

**// function**

**function secondFunc(para){**

**console.log("call second...")**

**para()**

**}**

**//passing function as an argument**

**secondFunc(firstFunc);**

##### **Output:**

**call first...**

**call second...**

**call second...**

**call first...**

** In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function.**

##### **Example:**

**function secondFunc(para){**

**para()**

**console.log("call second...")**

**}**

**// call back by arrow**

**secondFunc(()=>console.log("call first..."))**

**function secondFunc(para){**

**para()**

**console.log("call second...")**

**}**

**//call back by anonymous**

**secondFunc(function(){**

**console.log("call first...")**

**})**

##### **Output:**

**call first...**

**call second...**

**call first...**

**call second...**

** In the above program, there are two functions. While calling the secondFunc() function, one argument as function are passed. The firstFunc() function is a callback function.**

##### **Example:Program with setTimeout()**

**// program that shows the delay in execution**

**function greet() {**

**console.log('Hello world');**

**}**

**function sayName(name) {**

**console.log('Hello' + ' ' + name);**

**}**

**// calling the function**

**setTimeout(greet, 2000);**

**sayName('John');**

##### **Output:**

**Hello John**

**Hello world**

* **As you know, the setTimeout() method executes a block of code after the specified time.**
* **Here, the greet() function is called after 2000 milliseconds (2 seconds). During this wait, the sayName('John'); is executed. That is why Hello John is printed before Hello world.**
* **The above code is executed asynchronously (the second function; sayName() does not wait for the first function; greet() to complete).**

##### **Example:Using a Callback Function**

**// Callback Function Example**

**function greet(name, myFunction) {**

**console.log('Hello world');**

**// callback function**

**// executed only after the greet() is executed**

**myFunction(name);**

**}**

**// callback function**

**function sayName(name) {**

**console.log('Hello' + ' ' + name);**

**}**

**// calling the function after 2 seconds**

**setTimeout(greet, 2000, 'John', sayName);**

##### **Output:**

**Hello world**

**Hello Jone**

* **In the above program, the code is executed synchronously. The sayName() function is passed as an argument to the greet() function.**
* **The setTimeout() method executes the greet() function only after 2 seconds. However, the sayName() function waits for the execution of the greet() function.**

**Note : The callback function is helpful when you have to wait for a result that takes time. For example, the data coming from a server because it takes time for data to arrive.**

##### **Examle:**

**test = () => "Hello World!"; //Hello World!**

**test();**

**It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword :**

**hello**

**world**