JavaScript Function Definitions

‹ Previous Next ›

JavaScript functions are **defined** with the `function` keyword.

You can use a function **declaration** or a function **expression**.

Function Declarations

Earlier in this tutorial, you learned that functions are **declared** with the following syntax:

```
function functionName(parameters) {
  // code to be executed
}
```

Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are invoked (called upon).

Example

```
function myFunction(a, b) {
  return a * b;
}
```

Try it Yourself »

Semicolons are used to separate executable JavaScript statements.
Since a function **declaration** is not an executable statement, it is not common to end it with a semicolon.

Function Expressions

A JavaScript function can also be defined using an **expression**.

A function expression can be stored in a variable:

Example

```
var x = function (a, b) {return a * b};
```

Try it Yourself »

After a function expression has been stored in a variable, the variable can be used as a function:

Example

```
var x = function (a, b) {return a * b};
var z = x(4, 3);
```

Try it Yourself »

The function above is actually an **anonymous function** (a function without a name).

Functions stored in variables do not need function names. They are always invoked (called) using the variable name.

The function above ends with a semicolon because it is a part of an executable statement.

The Function() Constructor

As you have seen in the previous examples, JavaScript functions are defined with the `function` keyword.

Functions can also be defined with a built-in JavaScript function constructor called `Function()`.

Example

```
var myFunction = new Function("a", "b", "return a * b");

var x = myFunction(4, 3);
```

Try it Yourself »

You actually don't have to use the function constructor. The example above is the same as writing:

Example

```
var myFunction = function (a, b) {return a * b};

var x = myFunction(4, 3);
```

Try it Yourself »

Most of the time, you can avoid using the `new` keyword in JavaScript.

Function Hoisting

Earlier in this tutorial, you learned about "hoisting" (JavaScript Hoisting).

Hoisting is JavaScript's default behavior of moving **declarations** to the top of the current scope.

Hoisting applies to variable declarations and to function declarations.

Because of this, JavaScript functions can be called before they are declared:

```
myFunction(5);

function myFunction(y) {
  return y * y;
}
```

Functions defined using an expression are not hoisted.

Self-Invoking Functions

Function expressions can be made "self-invoking".

A self-invoking expression is invoked (started) automatically, without being called.

Function expressions will execute automatically if the expression is followed by ().

You cannot self-invoke a function declaration.

You have to add parentheses around the function to indicate that it is a function expression:

Example

```
(function () {
  var x = "Hello!!";  // I will invoke myself
})();
```

Try it Yourself »

The function above is actually an **anonymous self-invoking function** (function without name).

Functions Can Be Used as Values

JavaScript functions can be used as values:

Example

```
function myFunction(a, b) {
  return a * b;
}

var x = myFunction(4, 3);
```

Try it Yourself »

JavaScript functions can be used in expressions:

Example

```
function myFunction(a, b) {
  return a * b;
}

var x = myFunction(4, 3) * 2;
```

Try it Yourself »

Functions are Objects

The `typeof` operator in JavaScript returns "function" for functions.

But, JavaScript functions can best be described as objects.

JavaScript functions have both **properties** and **methods**.

The `arguments.length` property returns the number of arguments received when the function was invoked:

Example

```
function myFunction(a, b) {
  return arguments.length;
}
```

Try it Yourself »

The `toString()` method returns the function as a string:

Example

```
function myFunction(a, b) {
  return a * b;
}

var txt = myFunction.toString();
```

Try it Yourself »

A function defined as the property of an object, is called a method to the object.
A function designed to create new objects, is called an object constructor.

Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don't need the `function` keyword, the `return` keyword, and the **curly brackets**.

Example

```
// ES5
var x = function(x, y) {
  return x * y;
}

// ES6
const x = (x, y) => x * y;
```

Try it Yourself »

Arrow functions do not have their own `this` . They are not well suited for defining **object methods**.

Arrow functions are not hoisted. They must be defined **before** they are used.

Using `const` is safer than using `var` , because a function expression is always constant value.

You can only omit the `return` keyword and the curly brackets if the function is a single statement. Because of this, it might be a good habit to always keep them:

Example

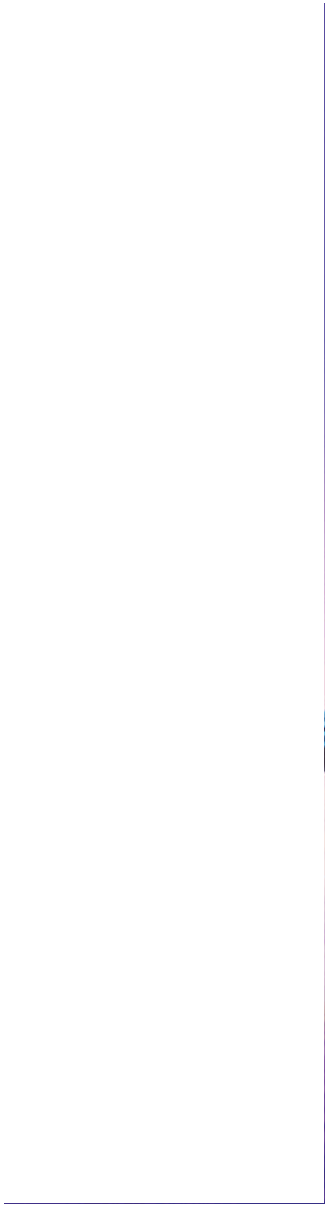
```
const x = (x, y) => { return x * y };
```

Try it Yourself »

Arrow functions are not supported in IE11 or earlier.

‹ Previous

Next ›



COLOR PICKER

HOW TO

Tabs
Dropdowns
Accordions
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes
Google Maps
Range Sliders
Tooltips
Slideshow
Filter List
Sort List

SHARE

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
jQuery Tutorial
Java Tutorial
C++ Tutorial

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
jQuery Reference
Java Reference
Angular Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
jQuery Examples
Java Examples
XML Examples

Web Certificates

HTML Certificate
CSS Certificate
JavaScript Certificate
SQL Certificate
Python Certificate
jQuery Certificate
PHP Certificate
Bootstrap Certificate
XML Certificate

Get Certified »