

- JS Tutorial
- JS HOME

JS Introduction

JS Where To

JS Output

JS Syntax

JS Statements

JS Comments

JS Variables

JS Operators

JS Arithmetic

JS Assignment

JS Data Types

JS Functions

JS Objects

JS Scope

JS Events

JS Strings

JS String Methods

JS Numbers

JS Number Methods

JS Math

JS Dates

JS Date Formats

JS Date Methods

JS Arrays

JS Array Methods

JS Booleans

JS Comparisons

JS Conditions

JS Switch

JS Loop For

JS Loop While

JS Break

JS Type Conversion

JS RegExp

JS Errors

JS Debugging

JS Hoisting

JS Strict Mode

JS Style Guide

JS Best Practices

JS Mistakes

JS Performance

JS Reserved Words

JS JSON

JS Forms

Forms Validation

Forms API

JS Objects

Object Definitions

Object Properties

Object Methods

Object Prototypes

JS Functions

Function Definitions

Function Parameters

Function Invocation

Function Closures

JS HTML DOM

DOM Intro

DOM Methods

DOM Document

DOM Elements

DOM HTML

DOM CSS

DOM Animations

DOM Events

DOM EventListener

DOM Navigation

DOM Nodes

DOM Nodelist

JS Browser BOM

JS Window

JS Screen

JS Location

JS History

JS Navigator

JS Popup Alert

JS Timing

JS Cookies

JS Examples

JS Examples

JS HTML DOM

JS HTML Input

JS HTML Objects

JS HTML Events

JS Browser

JS Quiz

JS Certificate

JS Summary

JS References

JavaScript Objects

HTML DOM Objects



JavaScript Function Definitions

[<< Previous](#)

[Next Chapter >>](#)

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".

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.

[<< Previous](#)

[Next Chapter >>](#)



W3SCHOOLS EXAMS

HTML, CSS, JavaScript, PHP, jQuery, Bootstrap and XML Certifications

COLOR PICKER



LEARN MORE:

- Color Converter
- Google Maps
- Animated Buttons
- Modal Boxes
- Modal Images
- Tooltips
- Loaders
- JS Animations
- Progress Bars
- Dropdowns
- Slideshow
- Side Navigation
- HTML Includes
- Color Palettes

SHARE THIS PAGE

