Objetos JS
Definições de Objeto
Propriedades do objeto
Métodos de Objeto
Exibição de objeto
Acessores de objeto
Construtores de objetos
Protótipos de objeto
Referência de Objeto
Mapa de Objetos ()
Conjunto de objetos ()

Funções JS
Definições de função
Parâmetros de Função
Invocação de Função
Chamada de Função
Função Aplicar
Fechamentos de funções

Definições de função JavaScript

❮ Anterior Próximo ❯

As funções JavaScript são **definidas** com a `function` palavra - chave.

Você pode usar uma **declaração de** função ou uma **expressão de** função .

Declarações de função

Anteriormente neste tutorial, você aprendeu que as funções são **declaradas** com a seguinte sintaxe:

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

As funções declaradas não são executadas imediatamente. Eles são "salvos para uso posterior" e serão executados mais tarde, quando forem invocados (chamados).

Exemplo

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

Tente você mesmo "

Os pontos-e-vírgulas são usados para separar instruções JavaScript executáveis.
Visto que uma **declaração de** função não é uma instrução executável, não é comum terminá-la com um ponto-e-vírgula.

Expressões de função

Uma função JavaScript também pode ser definida usando uma **expressão** .

Uma expressão de função pode ser armazenada em uma variável:

Exemplo

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

Tente você mesmo "

Depois que uma expressão de função foi armazenada em uma variável, a variável pode ser usada como uma função:

Exemplo

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

Tente você mesmo "

A função acima é na verdade uma **função anônima** (uma função sem um nome).

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

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

let 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

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

let 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 () {
  let 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;
}

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

Try it Yourself »

JavaScript functions can be used in expressions:

Example

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

let 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;
}

let text = 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.

Você só pode omitir a `return` palavra - chave e as chaves se a função for uma única instrução. Por causa disso, pode ser um bom hábito mantê-los sempre:

Exemplo

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

Tente você mesmo "

As funções de seta não são suportadas no IE11 ou anterior.

❮ Anterior Próximo ❯

Reportar erro Fórum Cerca de Comprar

Principais tutoriais
Tutorial HTML Tutorial
CSS Tutorial
JavaScript
Como fazer Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C ++ Tutorial
jQuery Tutorial

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

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

Web Courses
HTML Course
CSS Course
JavaScript Course
Front End Course
SQL Course
Python Course
PHP Course
jQuery Course
Java Course
C++ Course
C# Course
XML Course
Get Certified »