# Definições de Função de JavaScript

[❮ Anterior](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_object_prototypes.asp&usg=ALkJrhiSYheN0uMQBuP1DUI84udxl6-iyw)[Próximo ❯](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_function_parameters.asp&usg=ALkJrhhzuzz8YYe20pQ2DXPHBjosjXulfw)

As funções de JavaScript são **definidas** com a palavra-chave da **função** .

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

## Declarações de função

No início deste 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 são invocados (chamados).

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_return&usg=ALkJrhgd8oQWBAfPcQVTFQhtjPIjnEVdtA)

Semicolons são usados ​​para separar declarações de JavaScript executáveis.   
Uma vez que uma **declaração de** função não é uma declaração executável, não é comum terminar com um ponto e vírgula.

## Expressões de funções

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

Uma expressão de função pode ser armazenada em uma variável:

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_expression&usg=ALkJrhge17Ui-QvSHYdDDRS9NRV8wAzf9g)

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

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_expression_variable&usg=ALkJrhhyA59IOaZH8TOg1i8u5T9nrMwPBg)

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

As funções armazenadas em variáveis ​​não precisam de nomes de funções. Eles são sempre invocados (chamado) usando o nome da variável.

A função acima termina com um ponto-e-vírgula porque é parte de uma declaração executável.

## O Construtor Function ()

Como você viu nos exemplos anteriores, as funções JavaScript são definidas com a palavra-chave da **função** .

As funções também podem ser definidas com um construtor de funções JavaScript chamado Function ().

### Exemplo

var myFunction = new Function("a", "b", "return a \* b");  
  
var x = myFunction(4, 3);

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_constructor&usg=ALkJrhjwy_tuv2qlS5k9dI2QGblrqMm9rQ)

Na verdade, você não precisa usar o construtor de funções. O exemplo acima é o mesmo que escrever:

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_constructor2&usg=ALkJrhgFqhvQomyTTtSwVtxHUzoVr8lCnw)

Na maioria das vezes, você pode evitar usar a **nova** palavra-chave em JavaScript.

## Funcionamento da função

No início deste tutorial, você aprendeu sobre "hoisting".

Hoisting é o comportamento padrão do JavaScript de **declarações** móveis para o topo do escopo atual.

A elevação aplica-se a declarações variáveis ​​e a declarações de função.

Por isso, as funções JavaScript podem ser chamadas antes de serem declaradas:

myFunction(5);  
  
function myFunction(y) {  
    return y \* y;  
}

As funções definidas usando uma expressão não são içadas.

## Funções de auto-invocação

As expressões de função podem ser feitas "auto-invocando".

Uma expressão de auto-invocação é invocada (iniciada) automaticamente, sem ser chamada.

As expressões de função serão executadas automaticamente se a expressão for seguida por ().

Você não pode auto-invocar uma declaração de função.

Você deve adicionar parênteses em torno da função para indicar que é uma expressão de função:

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_expression_self&usg=ALkJrhhxegghH3a-b7B7wLYAUHpTz0fBRg)

A função acima é, na verdade, uma **função de auto-invocação anônima** (função sem nome).

## Funções podem ser usadas como valores

As funções de JavaScript podem ser usadas como valores:

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_value&usg=ALkJrhh4CsYhzDEGy41-Vx0jWsKAFwIS6Q)

As funções de JavaScript podem ser usadas em expressões:

### Exemplo

function myFunction(a, b) {  
    return a \* b;  
}  
  
var x = myFunction(4, 3) \* 2;

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_value2&usg=ALkJrhhaB53Fj1NzHm1Rp8GqDtG6WtEwng)

## As funções são objetos

O **tipo de** operador em JavaScript retorna a "função" para funções.

Mas, as funções JavaScript podem ser descritas como objetos.

As funções de JavaScript possuem **propriedades** e **métodos** .

A propriedade arguments.length retorna o número de argumentos recebidos quando a função foi invocada:

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_length&usg=ALkJrhi9fDw8mk4aU97VFLdkFC738PsajA)

O método toString () retorna a função como uma seqüência de caracteres:

### Exemplo

function myFunction(a, b) {  
    return a \* b;  
}  
  
var txt = myFunction.toString();