# Fechamentos 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_function_apply.asp&usg=ALkJrhhmbJhJuV9K6p5U5VURF2SrrMcvyA)[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_htmldom.asp&usg=ALkJrhjRCSdhjGnCYxuIc0kewxeuJRyitw)

As variáveis ​​de JavaScript podem pertencer ao escopo **local** ou **global** .

Variáveis ​​globais podem ser feitas localmente (privadas) com **fechamentos** .

## Variáveis ​​globais

Uma função pode acessar todas as variáveis ​​definidas **dentro** da função, assim:

### Exemplo

function myFunction() {  
    var a = 4;  
    return a \* a;  
}

[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_scope2&usg=ALkJrhiCRSceQItoZhDtaEyO9mGI-6i62A)

Mas uma função também pode acessar variáveis ​​definidas **fora** da função, como esta:

### Exemplo

var a = 4;  
function myFunction() {  
    return a \* a;  
}

[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_scope1&usg=ALkJrhiI6561T0pbiBOnkgEP2K2VCiuA2w)

No último exemplo, **a** é uma variável **global** .

Em uma página da Web, variáveis ​​globais pertencem ao objeto da janela.

As variáveis ​​globais podem ser usadas (e alteradas) por todos os scripts na página (e na janela).

No primeiro exemplo, **a** é uma variável **local** .

Uma variável local só pode ser usada dentro da função onde ela é definida. Está oculto de outras funções e outros códigos de script.

Variáveis ​​globais e locais com o mesmo nome são variáveis ​​diferentes. Modificando um, não modifica o outro.

As variáveis ​​criadas **sem** a palavra-chave **var** , são sempre globais, mesmo que sejam criadas dentro de uma função.

## Vida variável

As variáveis ​​globais vivem enquanto a sua aplicação (sua janela / sua página da web) reside.

As variáveis ​​locais têm vidas curtas. Eles são criados quando a função é invocada e excluída quando a função está concluída.

## Um Contra-Dilema

Suponha que você queira usar uma variável para contar algo e você quer que este contador esteja disponível para todas as funções.

Você poderia usar uma variável global e uma função para aumentar o contador:

### Exemplo

var counter = 0;  
  
function add() {  
    counter += 1;  
}  
  
add();  
add();  
add();  
  
// the counter is now equal to 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_counter&usg=ALkJrhjs885FJioQ05Gw4-2Qb5B-1kpH3w)

O contador só deve ser alterado pela função add ().

O problema é que qualquer script na página pode mudar o contador, sem ligar para adicionar ().

Se eu declarar o contador dentro da função, ninguém poderá alterá-lo sem chamar add ():

### Exemplo

function add() {  
    var counter = 0;  
    counter += 1;  
}  
  
add();  
add();  
add();  
  
// the counter should now be 3, but it does not work !

[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_counter2&usg=ALkJrhhtyP7fgPNtwWkfzFSA8S1QLCjlgw)

Não funcionou! Toda vez que eu chamo a função add (), o contador é definido como 1.

**Uma função interna de JavaScript pode resolver isso.**

## JavaScript Funções Aninhadas

Todas as funções têm acesso ao escopo global.

Na verdade, em JavaScript, todas as funções têm acesso ao escopo "acima".

O JavaScript suporta funções aninhadas. As funções aninhadas têm acesso ao escopo "acima".

Neste exemplo, a função interna **plus ()** tem acesso à variável do **contador** na função pai:

### Exemplo

function add() {  
    var counter = 0;  
    function plus() {counter += 1;}  
    plus();      
    return counter;   
}

[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_counter1&usg=ALkJrhhTa7URLywdI_aGn7h8YC_4qhLWUw)

Isso poderia ter resolvido o dilema do contador, se pudéssemos alcançar a função **plus ()** do lado de fora.

Também precisamos encontrar uma maneira de executar **counter = 0** apenas uma vez.

**Precisamos de um encerramento.**

## Fechamentos de JavaScript

Lembre-se das funções de auto-invocação? O que essa função faz?

### Exemplo

var add = (function () {  
    var counter = 0;  
    return function () {return counter += 1;}  
})();  
  
add();  
add();  
add();  
  
// the counter is now 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_counter3&usg=ALkJrhhp-qnR1Zwn0FOsYIXM_EOzJZCbtg)

## Exemplo Explicado

O atributo de variável atribui o valor de retorno de uma função de auto-invocação.

A função de auto-invocação é executada apenas uma vez. Ele define o contador como zero (0) e retorna uma expressão de função.

Dessa forma, o add torna-se uma função. A parte "maravilhosa" é que ele pode acessar o contador no escopo dos pais.

Isso é chamado de **encerramento de** JavaScript **.** Isso permite que uma função tenha variáveis ​​" **privadas** ".

O contador é protegido pelo escopo da função anônima e só pode ser alterado usando a função de adicionar.

Um fechamento é uma função que tem acesso ao escopo pai, mesmo após a função principal ter fechado.