Melhores Práticas 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_conventions.asp&usg=ALkJrhjZIVQQtruQaT4AAF1sa7yfgevOTA)[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_mistakes.asp&usg=ALkJrhhOZIxtbxVKU10X5aVbrzmfsmvPiw)

Evite variáveis ​​globais, evite novas, evite ==, evite eval ()

Evite as variáveis ​​globais

Minimize o uso de variáveis ​​globais.

Isso inclui todos os tipos de dados, objetos e funções.

As variáveis ​​e funções globais podem ser substituídas por outros scripts.

Use, em vez disso, variáveis ​​locais e aprenda a usar [fechamentos](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_closures.asp&usg=ALkJrhjAW3D5tGKWKnMS0mlMyovoZDtDog) .

Declarar sempre as variáveis ​​locais

Todas as variáveis ​​usadas em uma função devem ser declaradas como variáveis **locais** .

As variáveis ​​locais **devem** ser declaradas com a palavra-chave **var** , caso contrário elas se tornarão variáveis ​​globais.

O modo estrito não permite variáveis ​​não declaradas.

Declarações no topo

É uma boa prática de codificação colocar todas as declarações no topo de cada script ou função.

Isso vai:

* Dê um código mais limpo
* Fornecer um único local para procurar variáveis ​​locais
* Facilite a evitação de variáveis ​​globais indesejadas (implícitas)
* Reduzir a possibilidade de re-declarações indesejadas

// Declare at the beginning  
var firstName, lastName, price, discount, fullPrice;  
  
// Use later  
firstName = "John";  
lastName = "Doe";  
  
price = 19.90;  
discount = 0.10;  
  
fullPrice = price \* 100 / discount;

Isso também vale para variáveis ​​de loop:

// Declare at the beginning  
var i;  
  
// Use later  
for (i = 0; i < 5; i++) {

Por padrão, o JavaScript move todas as declarações para o topo ( [JavaScript Hoisting](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_hoisting.asp&usg=ALkJrhiAE3p-EabddXV8ZJOdZ5SWz2Ecwg) ).

Inicializar variáveis

É uma boa prática de codificação para inicializar variáveis ​​quando as declara.

Isso vai:

* Dê um código mais limpo
* Fornecer um único local para inicializar variáveis
* Evite valores indefinidos

// Declare and initiate at the beginning  
var firstName = "",  
    lastName = "",  
    price = 0,  
    discount = 0,  
    fullPrice = 0,  
    myArray = [],  
    myObject = {};

Inicializar variáveis ​​fornece uma idéia do uso pretendido (e do tipo de dados pretendido).

Nunca Declare Número, String ou Objetos Booleanos

Sempre trate números, cordas ou booleanos como valores primitivos. Não como objetos.

Declarar esses tipos como objetos, retarda a velocidade de execução e produz efeitos colaterais desagradáveis:

Exemplo

var x = "John";               
var y = new String("John");  
(x === y) // is false because x is a string and y is an object.

[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_best_object_string1&usg=ALkJrhiHKQModlfLV04FkHKaSgpj3eRX6A)

Ou pior ainda:

Exemplo

var x = new String("John");               
var y = new String("John");  
(x == y) // is false because you cannot compare objects.

[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_best_object_string2&usg=ALkJrhhf7TGy0lVfQHVEjOy61AUcbZmqUg)

Não use novo objeto ()

* Use {} em vez de Object novo ()
* Use "" em vez de novo String ()
* Use 0 em vez do novo Número ()
* Use false em vez de novo booleano ()
* Use [] em vez de novo Array ()
* Use / () / em vez do novo RegExp ()
* Use a função () {} em vez da nova Função ()

Exemplo

var x1 = {};           // new object  
var x2 = "";           // new primitive string  
var x3 = 0;            // new primitive number  
var x4 = false;        // new primitive boolean  
var x5 = [];           // new array object  
var x6 = /()/;         // new regexp object  
var x7 = function(){}; // new function object

[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_best_constructors&usg=ALkJrhhLkZQR3cU3IlFXrlItNTDkVTtB3Q)

Cuidado com conversões automáticas de tipos

Cuidado com o fato de que os números podem ser convertidos acidentalmente em strings ou NaN (Não é um Número).

O JavaScript está vagamente digitado. Uma variável pode conter diferentes tipos de dados e uma variável pode alterar seu tipo de dados:

Exemplo

var x = "Hello";     // typeof x is a string  
x = 5;               // changes typeof x to a number

[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_best_typeof&usg=ALkJrhi2jgmuETDTWGMkEskkuZztDhLEpA)

Ao fazer operações matemáticas, o JavaScript pode converter números em strings:

Exemplo

var x = 5 + 7;       // x.valueOf() is 12,  typeof x is a number  
var x = 5 + "7";     // x.valueOf() is 57,  typeof x is a string  
var x = "5" + 7;     // x.valueOf() is 57,  typeof x is a string  
var x = 5 - 7;       // x.valueOf() is -2,  typeof x is a number  
var x = 5 - "7";     // x.valueOf() is -2,  typeof x is a number  
var x = "5" - 7;     // x.valueOf() is -2,  typeof x is a number  
var x = 5 - "x";     // x.valueOf() is NaN, typeof x is a number

[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_best_valueof&usg=ALkJrhgPKaLLJTFzaD408duEuMPMEp8b4w)

Subtrair uma string de uma string, não gera um erro, mas retorna NaN (Não é um Número):

Exemplo

"Hello" - "Dolly"    // returns NaN

[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_best_string_op1&usg=ALkJrhgjkTkPei2UThw94dSEr8bH4wI9vg)

Use === Comparação

O operador de comparação == sempre converte (para tipos correspondentes) antes da comparação.

O === operador força comparação de valores e tipo:

Exemplo

0 == "";        // true  
1 == "1";       // true  
1 == true;      // true  
  
0 === "";       // false  
1 === "1";      // false  
1 === true;     // false

[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_best_comparison&usg=ALkJrhhcympJU_uuYlKskDAsYhNliJoYmQ)

Usar Padrões de Parâmetros

Se uma função é chamada com um argumento ausente, o valor do argumento faltante é definido como **indefinido** .

Valores indefinidos podem quebrar seu código. É um bom hábito atribuir valores padrão aos argumentos.

Exemplo

function myFunction(x, y) {  
    if (y === undefined) {  
        y = 0;  
    }  
}

[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_best_parameter_default&usg=ALkJrhhtg-_Dthvcn-I49UlFBNoExCRRJw)

Leia mais sobre parâmetros de função e argumentos em [Parâmetros de Função](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)

Encerre seus interruptores com padrões

Sempre termine suas declarações de troca com um padrão. Mesmo se você acha que não há necessidade disso.

Exemplo

switch (new Date().getDay()) {  
    case 0:  
        day = "Sunday";  
        break;  
    case 1:  
        day = "Monday";  
        break;  
    case 2:  
        day = "Tuesday";  
        break;  
    case 3:  
        day = "Wednesday";  
        break;  
    case 4:  
        day = "Thursday";  
        break;  
    case 5:  
        day = "Friday";  
        break;  
    case 6:  
        day = "Saturday";  
        break;  
    default:  
        day = "Unknown";  
}