Guia de estilo do JavaScript e convenções de codificação

[❮ 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_strict.asp&usg=ALkJrhjXghOhXawZGq1egZkwH53oTnfVyw)[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_best_practices.asp&usg=ALkJrhiDo8wuZin5yRpQ0vbcM8j6iWn0Rg)

Use sempre as mesmas convenções de codificação para todos os seus projetos de JavaScript.

Convenções de codificação de JavaScript

As convenções de codificação são **diretrizes de estilo para programação** . Eles geralmente cobrem:

* Regras de nomeação e declaração para variáveis ​​e funções.
* Regras para o uso de espaço em branco, indentação e comentários.
* Práticas e princípios de programação

Convenções de codificação de **qualidade segura** :

* Melhora a legibilidade do código
* Facilite a manutenção do código

As convenções de codificação podem ser regras documentadas para as equipes seguir, ou apenas ser sua prática de codificação individual.

Esta página descreve as convenções de códigos JavaScript gerais usadas pelo W3Schools.   
Você também deve ler o próximo capítulo "Melhores Práticas", e aprender como evitar armadilhas de codificação.

Nomes variáveis

Na W3schools, usamos **camelCase** para nomes de identificadores (variáveis ​​e funções).

Todos os nomes começam com uma **carta** .

No final desta página, você encontrará uma discussão mais ampla sobre regras de nomeação.

firstName = "John";  
lastName = "Doe";  
  
price = 19.90;  
tax = 0.20;  
  
fullPrice = price + (price \* tax);

Espaços em torno de operadores

Sempre coloque espaços em torno de operadores (= + - \* /), e depois de vírgulas:

Exemplos:

var x = y + z;  
var values = ["Volvo", "Saab", "Fiat"];

Indentação do código

Use sempre 4 espaços para indentação de blocos de código:

Funções:

function toCelsius(fahrenheit) {  
    return (5 / 9) \* (fahrenheit - 32);  
}

Não use abas (tabuladores) para indentação. Diferentes editores interpretam separadores de forma diferente.

Regras da declaração

Regras gerais para declarações simples:

* Sempre termine uma simples declaração com um ponto-e-vírgula.

Exemplos:

var values = ["Volvo", "Saab", "Fiat"];  
  
var person = {  
    firstName: "John",  
    lastName: "Doe",  
    age: 50,  
    eyeColor: "blue"  
};

Regras gerais para declarações complexas (compostos):

* Coloque o suporte de abertura no final da primeira linha.
* Use um espaço antes do suporte de abertura.
* Coloque o suporte de fechamento em uma nova linha, sem espaços principais.
* Não termine uma indicação complexa com um ponto-e-vírgula.

Funções:

function toCelsius(fahrenheit) {  
    return (5 / 9) \* (fahrenheit - 32);  
}

Rotações:

for (i = 0; i < 5; i++) {  
    x += i;  
}

Condicionais:

if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}

Regras do objeto

Regras gerais para definições de objetos:

* Coloque o suporte de abertura na mesma linha que o nome do objeto.
* Use colon mais um espaço entre cada propriedade e seu valor.
* Use aspas em torno de valores de string, não em torno de valores numéricos.
* Não adicione uma vírgula após o último par de valor de propriedade.
* Coloque o suporte de fechamento em uma nova linha, sem espaços principais.
* Sempre termine uma definição de objeto com um ponto-e-vírgula.

Exemplo

var person = {  
    firstName: "John",  
    lastName: "Doe",  
    age: 50,  
    eyeColor: "blue"  
};

Objetos curtos podem ser escritos comprimidos, em uma linha, usando espaços somente entre propriedades, assim:

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

Comprimento da linha <80

Para legibilidade, evite linhas com mais de 80 caracteres.

Se uma declaração de JavaScript não caber em uma linha, o melhor lugar para quebrar, é depois de um operador ou uma vírgula.

Exemplo

document.getElementById("demo").innerHTML =  
    "Hello Dolly.";

[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_line_break&usg=ALkJrhif7aigU8gNsHYRt8egMHBzPJtGfA)

Convenções de nomeação

Use sempre a mesma convenção de nomeação para todo o seu código. Por exemplo:

* Nomes de variáveis ​​e funções escritos como **camelCase**
* Variáveis ​​globais escritas em **MAYÚSCULAS** (Nós não, mas é bastante comum)
* Constantes (como PI) escritas em **MAYÚSCULAS**

Você deve usar **hyp-** **hens** , **camelCase** ou **under\_scores** em nomes de variáveis?

Esta é uma questão que os programadores discutem frequentemente. A resposta depende de quem você pergunta:

**Hyphens em HTML e CSS:**

Os atributos HTML5 podem começar com dados (quantidade de dados, preço de dados).

O CSS usa hífens em nomes de propriedade (tamanho da fonte).

Hyphens pode ser confundido como tentativas de subtração. Hyphens não são permitidos em nomes de JavaScript.

**Sublinha:**

Muitos programadores preferem usar underscores (date\_of\_birth), especialmente em bancos de dados SQL.

Os sublinhados são freqüentemente usados ​​na documentação do PHP.

**PascalCase:**

PascalCase é frequentemente preferido pelos programadores C.

**camelCase:**

CamelCase é usado pelo próprio JavaScript, pelo jQuery e outras bibliotecas de JavaScript.

Não inicie nomes com um $ sign. Isso o colocará em conflito com muitos nomes de bibliotecas JavaScript.

Carregando JavaScript em HTML

Use sintaxe simples para carregar scripts externos (o atributo de tipo não é necessário):

<script src="myscript.js"></script>

Acessando elementos HTML

Uma conseqüência do uso de estilos HTML "desordenados" pode resultar em erros de JavaScript.

Essas duas instruções de JavaScript produzirão resultados diferentes:

var obj = getElementById("Demo")  
  
var obj = getElementById("demo")

Se possível, use a mesma convenção de nomenclatura (como JavaScript) em HTML.

[Visite o Guia de estilo HTML](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/html/html5_syntax.asp&usg=ALkJrhhSb9Yq_bWL254l7ElulF1pbPfuWA) .

Extensões de arquivo

Arquivos HTML devem ter uma extensão **.html** (não **.htm** ).

Os arquivos CSS devem ter uma extensão **.css** .

Os arquivos JavaScript devem ter uma extensão **.js** .

Use nomes de arquivos de minúsculas

A maioria dos servidores web (Apache, Unix) são sensíveis a maiúsculas e minúsculas sobre nomes de arquivos:

london.jpg não pode ser acessado como London.jpg.

Outros servidores web (Microsoft, IIS) não diferenciam maiúsculas de minúsculas:

london.jpg pode ser acessado como London.jpg ou london.jpg.

Se você usa uma mistura de maiúsculas e minúsculas, você precisa ser extremamente consistente.

Se você muda de um caso insensível, para um servidor sensível a maiúsculas e minúsculas, mesmo pequenos erros podem quebrar seu site.

Para evitar esses problemas, use sempre nomes de arquivos em minúsculas (se possível).

atuação

As convenções de codificação não são usadas pelos computadores. A maioria das regras tem pouco impacto na execução de programas.

A indentação e os espaços extras não são significativos em scripts pequenos.

Para código em desenvolvimento, a legibilidade deve ser preferida. Os scripts de produção maiores devem ser minificados.