Google Traduzido para: Português ▾ Mostrar original Opções ▼ ☒

Tutorials ▾ References ▾ Exercises ▾ Paid Courses Log in

JS Typeof
Conversão de tipo JS
JS Bitwise
JS RegExp
Erros JS
Escopo JS
JS Hoisting
Modo JS Strict
JS esta palavra-chave
Função de seta JS
Classes JS
JS JSON
Depuração JS
Guia de estilo JS
JS Best Practices
JS Erros
JS Performance
Palavras Reservadas JS

Objetos JS
Definições de Objeto

# Guia de estilo de JavaScript

❮ Anterior Próximo ❯

Sempre use as mesmas convenções de codificação para todos os seus projetos JavaScript.

## Convenções de codificação JavaScript

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

- Regras de nomenclatura e declaração para variáveis e funções.
- Regras para o uso de espaço em branco, recuo e comentários.
- Práticas e princípios de programação

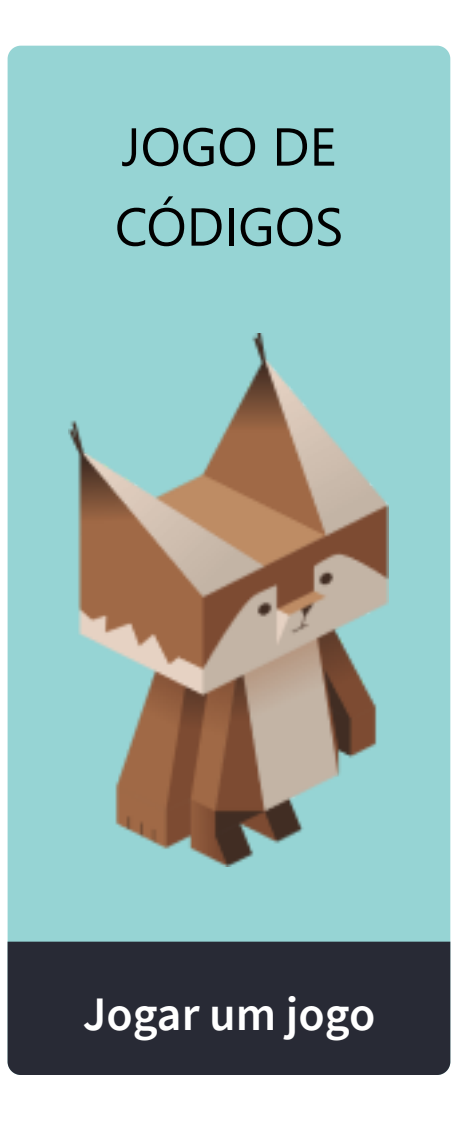
As convenções de codificação **garantem a qualidade** :

- Melhora a legibilidade do código
- Facilite a manutenção do código

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

Esta página descreve as convenções gerais de código JavaScript usadas por W3Schools.
Você também deve ler o próximo capítulo "Melhores práticas" e aprender como evitar armadilhas de codificação.

JOGO DE CÓDIGOS



Jogar um jogo

## Nomes de Variáveis

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

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

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

```
firstName = "John";
lastName = "Doe";

price = 19.90;
tax = 0.20;

fullPrice = price + (price * tax);
```

## Espaços ao redor dos operadores

Sempre coloque espaços entre os operadores (= + - * /) e depois das vírgulas:

### Exemplos:

```
let x = y + z;
const myArray = ["Volvo", "Saab", "Fiat"];
```

## Recuo de código

Sempre use 2 espaços para indentação de blocos de código:

### Funções:

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

Não use tabulações (tabuladores) para recuo. Editores diferentes interpretam as guias de maneira diferente.

## Regras de Declaração

Regras gerais para declarações simples:

- Sempre termine uma instrução simples com um ponto e vírgula.

### Exemplos:

```
const cars = ["Volvo", "Saab", "Fiat"];

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

Regras gerais para declarações complexas (compostas):

- Coloque o colchete de abertura no final da primeira linha.
- Use um espaço antes do colchete de abertura.
- Coloque o colchete de fechamento em uma nova linha, sem espaços à esquerda.
- Não termine uma instrução complexa com um ponto e vírgula.

### Funções:

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

### Rotações:

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

### Condicionais:

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

## Regras de Objeto

Regras gerais para definições de objetos:

- Coloque o colchete de abertura na mesma linha do nome do objeto.
- Use dois pontos 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 colchete de fechamento em uma nova linha, sem espaços à esquerda.
- Always end an object definition with a semicolon.

### Example

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

Short objects can be written compressed, on one line, using spaces only between properties, like this:

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

## Line Length < 80

For readability, avoid lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it, is after an operator or a comma.

### Example

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

Try it Yourself »

## Naming Conventions

Always use the same naming convention for all your code. For example:

- Variable and function names written as **camelCase**
- Global variables written in **UPPERCASE** (We don't, but it's quite common)
- Constants (like PI) written in **UPPERCASE**

Should you use **hyp-hens**, **camelCase**, or **under_scores** in variable names?

This is a question programmers often discuss. The answer depends on who you ask:

**Hyphens in HTML and CSS:**

HTML5 attributes can start with data- (data-quantity, data-price).

CSS uses hyphens in property-names (font-size).

Hyphens can be mistaken as subtraction attempts. Hyphens are not allowed in JavaScript names.

**Underscores:**

Many programmers prefer to use underscores (date_of_birth), especially in SQL databases.

Underscores are often used in PHP documentation.

**PascalCase:**

PascalCase is often preferred by C programmers.

**camelCase:**

camelCase is used by JavaScript itself, by jQuery, and other JavaScript libraries.

Do not start names with a $ sign. It will put you in conflict with many JavaScript library names.

## Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js"></script>
```

## Accessing HTML Elements

A consequence of using "untidy" HTML styles, might result in JavaScript errors.

These two JavaScript statements will produce different results:

```
const obj = getElementById("Demo")

const obj = getElementById("demo")
```

If possible, use the same naming convention (as JavaScript) in HTML.

Visit the HTML Style Guide.

## File Extensions

HTML files should have a **.html** extension (**.htm** is allowed).

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

## Use Lower Case File Names

Most web servers (Apache, Unix) are case sensitive about file names:

london.jpg cannot be accessed as London.jpg.

Other web servers (Microsoft, IIS) are not case sensitive:

london.jpg can be accessed as London.jpg or london.jpg.

Se você usar uma mistura de maiúsculas e minúsculas, será necessário ser extremamente consistente.

Se você mudar de um servidor sem distinção entre maiúsculas e minúsculas, para um servidor que diferencia maiúsculas de minúsculas, até mesmo pequenos erros podem corromper seu site.

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

## Desempenho

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

O recuo e os espaços extras não são significativos em scripts pequenos.

Para código em desenvolvimento, deve-se dar preferência à legibilidade. Roteiros de produção maiores devem ser minimizados.

❮ Anterior Próximo ❯

Reportar erro Fórum Cerca de Comprar

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

Top References
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 »