JS Tutorial

JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

JavaScript Style Guide and Coding Conventions

‹ Previous Next ›

Always use the same coding conventions for all your JavaScript projects.

JavaScript Coding Conventions

Coding conventions are **style guidelines for programming**. They typically cover:

- Naming and declaration rules for variables and functions.
- Rules for the use of white space, indentation, and comments.
- Programming practices and principles

Coding conventions **secure quality**:

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

Coding conventions can be documented rules for teams to follow, or just be your individual coding practice.

> This page describes the general JavaScript code conventions used by W3Schools.
> You should also read the next chapter "Best Practices", and learn how to avoid coding pitfalls.

Variable Names

At W3schools we use **camelCase** for identifier names (variables and functions).

All names start with a **letter**.

At the bottom of this page, you will find a wider discussion about naming rules.

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

price = 19.90;
tax = 0.20;

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

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

Always put spaces around operators (= + - * /), and after commas:

Examples:

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

Code Indentation

Always use 4 spaces for indentation of code blocks:

Functions:

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

Do not use tabs (tabulators) for indentation. Different editors interpret tabs

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

Statement Rules

General rules for simple statements:

- Always end a simple statement with a semicolon.

Examples:

```
var values = ["Volvo", "Saab", "Fiat"];

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

General rules for complex (compound) statements:

- Put the opening bracket at the end of the first line.
- Use one space before the opening bracket.
- Put the closing bracket on a new line, without leading spaces.
- Do not end a complex statement with a semicolon.

Functions:

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

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

Loops:

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

Conditionals:

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

Object Rules

General rules for object definitions:

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

- Use quotes around string values, not around numeric values.
- Do not add a comma after the last property-value pair.
- Place the closing bracket on a new line, without leading spaces.
- Always end an object definition with a semicolon.

Example

```
var 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:

```
var 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

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

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.

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

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.

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

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

var 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 (not **.htm**).

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.

If you use a mix of upper and lower case, you have to be extremely consistent.

JS Tutorial
JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays

To avoid these problems, always use lower case file names (if possible).

Performance

Coding conventions are not used by computers. Most rules have little impact on the execution of programs.

Indentation and extra spaces are not significant in small scripts.

For code in development, readability should be preferred. Larger production scripts should be minified.

‹ Previous Next ›

JS Tutorial

JS HOME
JS Introduction
JS Where To
JS Output
JS Syntax
JS Statements
JS Comments
JS Variables
JS Operators
JS Arithmetic
JS Assignment
JS Data Types
JS Functions
JS Objects
JS Scope
JS Events
JS Strings
JS String Methods
JS Numbers
JS Number Methods
JS Math
JS Random
JS Dates
JS Date Formats
JS Date Methods
JS Arrays