

# History & Syntax

JavaScript



# What is JavaScript?

JS

[JavaScript](#) is a programming language used in the development of many different types of software nowadays, but was originally (and still mostly commonly) intended for use in web-pages. It brings dynamic features to web-pages, and most interactive web-pages do feature JavaScript code.

If you see new content load into the page as you scroll, submit a form without a new page loading, or have scrubbed a YouTube video to skip some of it—you've used JavaScript-coded features in a web-page before!

It was first released in late 1995, and (unfortunately confusingly) gained its name due to the then quickly-rising popularity of the Java programming language to better fit in. Despite the similar names, it is important to note that [JavaScript](#) and [Java](#) are not the same language!

JavaScript is a [weakly typed just-in-time compiled](#) language. Java is a completely separate [strongly typed compiled](#) language, that will not run directly in the browser without additional plugins, extensions, and/or configuration.

# Languages of the World Wide Web

## **HyperText Markup Language (HTML)**

The skeleton of the web-site. This is where the content lives, and where we can mark it up to give it meaning.

## **Cascading StyleSheets (CSS)**

The skin or visual aspect of the website. Controls the layout, spacing, position, and sizing of elements in a web-page.

## **JavaScript (JS)**

The brain of the website. Any interactive or dynamic features on a web-page are the responsibility of JavaScript.

# Tools of the Trade

Equip yourself with popular web browsers.

If you want to try out JavaScript, it is important to use a modern web browser to gain easy access to the latest and greatest stable language features!

Popular modern browsers we recommend include:

- [Google Chrome](#) or [Developer Edition](#)
- [Microsoft Edge](#) or [Developer Edition](#)
- [Mozilla Firefox](#) or [Developer Edition](#)

Developer editions are often a version or two ahead in terms of features—if you're a developer this can help you test for compatibility with upcoming features and stay a step ahead.

We suggest having all the most popular browsers installed, so you can test your website on each of them to make sure it is compatible during development.

# Machine Language vs High Level Language

## High Level Language:

- Easy to interpret and compile
- Easy to debug, maintain and understand
- Cross platform
- Not as memory efficient as machine language
- Requires a compiler/interpreter to convert the code written into machine language

## Machine Language:

- Easy for the machine to understand
- Difficult to debug, maintain and understand
- Not portable
- Requires assembler to interpret code instructions

# Strong vs Weak Typed Languages

Types define the data type stored in a variable. For example, if a variable is intended to store a whole number, it's type would likely be a number or integer (depending on the programming language), if a variable is intended to store a string (a sequence of characters), then it's type would be a string. In essence, the difference between strong and weak typing is how strict the language is when it comes to checking types. In a weak typed language such as Javascript, has looser typing rules which makes it more flexible. However, looser typing rules may produce unpredictable or even erroneous results or may perform implicit type conversion when it is run. Strong typed languages are the opposite, there are stricter typing rules, which makes it inflexible, but also lessens the chance of producing unpredictable or erroneous results.

# Browser compatibility.

The standard for how web browser JavaScript engines interpret code is decided by [Ecma International](#), an organization that helps develop standards and technical reporting for computer and technology systems.

The more browsers all agree to a standard, the easier it is for programmers to do their job! In the 90s and early 00s, we saw many cases where certain code would only run on specific browsers based on how each decided JavaScript should be written. For instance, JavaScript code written with Internet Explorer in mind, may not have run on Mozilla's Firefox browser, making some web-pages unusable.

Modern browsers, thankfully, are essentially on the same-page when it comes to JavaScript. What used to be a concern before, is seldom now, making programmers' lives on the internet much easier!

For specific features in JavaScript, you can explore compatibility via websites like [Can I Use](#). These resources will detail which browsers and versions thereof can understand features or commands.

Parentheses following a name tell us that a method or function should execute. Many methods or functions allow you to pass a value, or values, into them by placing the value(s) inside of the parentheses.

A period indicates that we are entering an object or class and accessing a method or property.

Text inbetween quotation marks is considered a "string." A string is a collection of text characters.

**console.log("Hello, World!");**

Log is a method of console. More specifically it is meant for writing a "log" message into the Web Console.

Each "statement," in JavaScript, should end in a semicolon.

We are accessing console, a class that allows us to interact with the Web Console in our browser.



# Syntax

Just like in languages we use to speak and write each day, programming languages have rules and a grammar to them.

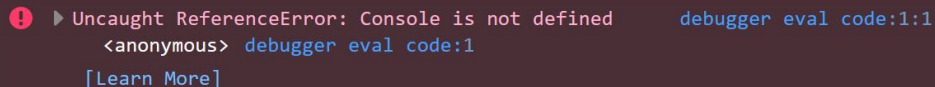
We refer to this as the language's syntax.

Any time you use a programming language, it is important you follow its rules or syntax.

For instance, JavaScript is case-sensitive. If you typed our previous example with a capital like so...

```
Console.log( "Hello, World!" );
```

This would result in an error:

A screenshot of a web browser's developer console showing a red error message. The message reads: 'Uncaught ReferenceError: Console is not defined' followed by 'debugger eval code:1:1' and '<anonymous> debugger eval code:1'. Below the message is a blue link that says '[Learn More]'.

```
! ▶ Uncaught ReferenceError: Console is not defined    debugger eval code:1:1  
    <anonymous> debugger eval code:1  
    [Learn More]
```

When we are writing any code in any language, we have to pay close attention to these rules, or our code will not behave correctly—if at all.

We'll also have to be just as cautious with spelling and punctuation.

# Commenting

When programming it can be helpful to leave notes for yourself and/or your team.

In fact, while you are learning, we highly encourage leaving all sorts of notes in your code! If you come back to a practice you worked on days, weeks, months, or even years later, comments are what will remind you what that code was doing.

In JavaScript there are two ways to leave notes—[comments](#)—in your code.

The simplest comment available is composed of two forward slashes (`//`). This type of comment is intended for single-line notes.

The second is a multi-line comment syntax, composed of a forward slash followed immediately by an asterisk (`/*`) for opening the note; you can close the note by ending with the same characters in the opposite order (`*/`).

# Get comfortable with documentation.

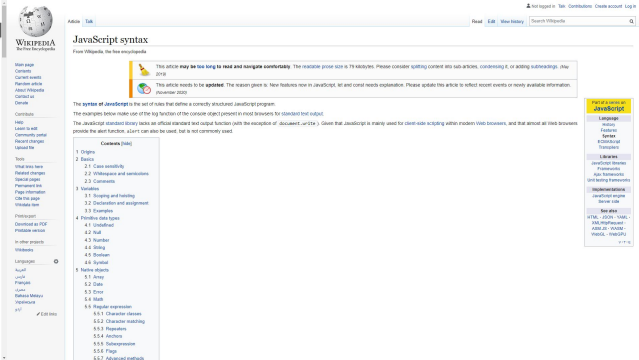
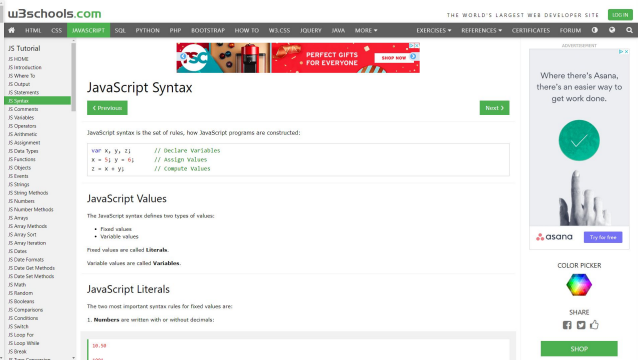
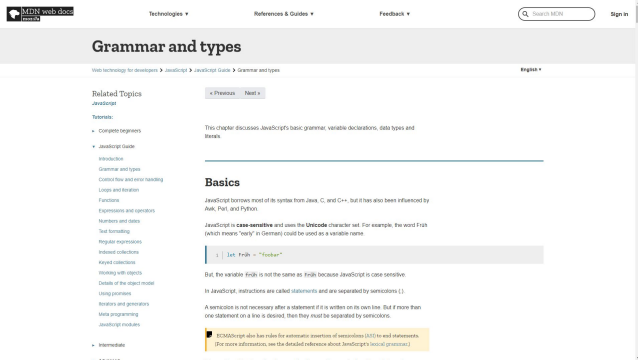
It isn't expected that every developer remember every single aspect of each and every language they use, get familiar with popular and reliable resources that you can go to for answers and reminders.

For example, you can find detailed guides to JavaScript's full syntax from various credible sources; make a note of them:

[Mozilla Developer Network](#)

[W3Schools](#)

[Wikipedia](#)



# Recommended Reading

For more information on how the stack works, check out the following:

- Philip Roberts' "Loupe" Talk: <http://latentflip.com/loupe/>

A great book with extensive coverage on JavaScript in-depth is:

- [Brown, E. \(February, 2016\). \*Learning JavaScript, 3rd Edition\*. O'Reilly Media, Inc.](#)