An introduction to JavaScript



# What is JavaScript?

*JavaScript* 在各種瀏覽器上都支援的一種程式語言。可以大致上看出來重點是java ,script。

但是和java 又有不同。

##### Why JavaScript?

 各種瀏覽器都有自己的內建java引擎，或者稱為“JavaScript virtual machine”.例如，

* [V8](https://en.wikipedia.org/wiki/V8_%28JavaScript_engine%29) – in Chrome and Opera.
* [Gecko](https://en.wikipedia.org/wiki/Gecko_%28software%29) – in Firefox.
* “Trident”, “Chakra” 是IE上的不同版本。

##### How engines work?

當網頁載入時，這些javascript 隨即被解譯成中間碼的形式，然後執行。相關課題包關了安全性，速度等考量。

##### [What can in-browser JavaScript do?](https://javascript.info/intro#what-can-in-browser-javascript-do)

javascript 到底能做什麼，大部分和其所在的環境相關，例如，在安全性上的考量來說，存取本地檔案是一種不安全的行為，但是 [Node.JS](https://wikipedia.org/wiki/Node.js) 提供javascript 存取任意檔案的函數，甚至允許網路功能的互動。

理論上，javascript 的能力是網頁處理，包括使用者和伺服器的互動。例如:

* Add new HTML to the page, change the existing content, modify styles.
* React to user actions, run on mouse clicks, pointer movements, key presses.
* Send requests over the network to remote servers, download and upload files (so-called [AJAX](https://en.wikipedia.org/wiki/Ajax_%28programming%29) and [COMET](https://en.wikipedia.org/wiki/Comet_%28programming%29) technologies).
* Get and set cookies, ask questions to the visitor, show messages.
* Remember the data on the client-side (“local storage”).

##### [What in-browser JavaScript can NOT do?](https://javascript.info/intro#what-in-browser-javascript-can-not-do)

這方面主要是考量安全性（例如個人資訊），且角度分為是否在客戶端或者伺服器端。限制包括是否能存取不同於html來源的其他伺服器，和本地檔案。但是由於特殊需求，隨著演進，也不再是完全無法存取，例如也可以dropping的形式寫入本地檔案。但是一個重要的基本原則是不允許直接利用作業系統本身的系統函數。

##### Languages “over” JavaScript

由於程式員熟悉的程式語言有限，因此將各種不同的程式語言轉成javascript 的功能出現，例如

* [CoffeeScript](http://coffeescript.org/) is a “syntax sugar” for JavaScript, it introduces shorter syntax, allowing to write more precise and clear code. Usually Ruby devs like it.
* [TypeScript](http://www.typescriptlang.org/) is concentrated on adding “strict data typing”, to simplify development and support of complex systems. It is developed by Microsoft.
* [Dart](https://www.dartlang.org/) is a standalone language that has its own engine that runs in non-browser environments (like mobile apps). It was initially offered by Google as a replacement for JavaScript, but as of now, browsers require it to be transpiled to JavaScript just like the ones above.

# Code editors

節省時間的話，可以考慮Integrated Development Environment(IDE)，但是要節省空間的話，可以使用lightweight editors

##### IDE

The term [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment) (means a powerful editor with many features that usually operates on a “whole project”. As said, that’s not just an editor, but a full-scale “development environment”.

An IDE loads the project (can be many files), and then allows navigation between files, provides autocompletion based on the whole project, integrates with a version management system (like [git](https://git-scm.com/)), a testing environment and other “project-level” stuff.

If you haven’t considered selecting an IDE yet, look at the following variants:

* IntelliJ editors: [WebStorm](http://www.jetbrains.com/webstorm/) for frontend development and [PHPStorm (PHP)](http://www.jetbrains.com/phpstorm/), [IDEA (Java)](http://www.jetbrains.com/idea/), [RubyMine (Ruby)](http://www.jetbrains.com/ruby/) and other if you need additional languages.
* Visual Studio is fine if you’re a .NET developer, and a free version is available ([Visual Studio Community](https://www.visualstudio.com/vs/community/))
* Eclipse-based products, like [Aptana](http://www.aptana.com/) and Zend Studio.
* [Komodo IDE](http://www.activestate.com/komodo-ide) and its lightweight free version [Komodo Edit](http://www.activestate.com/komodo-edit).
* [Netbeans](http://netbeans.org/).

All of the IDEs listed above are available on both Windows and Mac, and the IDEs other than Visual Studio are also available on Linux.

Most IDEs are paid, but have a trial period. Their cost is usually negligible compared to a qualified developer’s salary, so just choose the best one for you.

##### Lightweight editors

“Lightweight editors” are not as powerful as IDEs, but they’re fast, elegant and simple.

They are mainly used to instantly open and edit a file.

The main difference between a “lightweight editor” and an “IDE” is that IDE works on a project-level, so it loads much more data on start, analyzes the project structure if needed and so on. A lightweight editor is much faster if we need only one file.

In practice, lightweight editors may have a lot of plugins including directory-level syntax analyzers and autocompleters, so there’s no strict border between a lightweight editor and an IDE.

The following options deserve your attention:

* [Visual Studio Code](https://code.visualstudio.com/) (cross-platform, free).
* [Atom](https://atom.io/) (cross-platform, free).
* [Sublime Text](http://www.sublimetext.com/) (cross-platform, shareware).
* [Notepad++](https://notepad-plus-plus.org/) (Windows, free).
* Vim and Emacs are also cool, if you know how to use them.
* [WebStorm](http://www.jetbrains.com/webstorm/) for JS, and if there is one more language in the project, then I switch to other Jetbrains editors like [PHPStorm](http://www.jetbrains.com/phpstorm/)(PHP), [IDEA](http://www.jetbrains.com/idea/) (Java), [RubyMine](http://www.jetbrains.com/ruby/) (Ruby). There are editors for other languages too, but I haven’t used them.
* As a lightweight editor – [Sublime Text](http://www.sublimetext.com/) or [Atom](https://atom.io/).