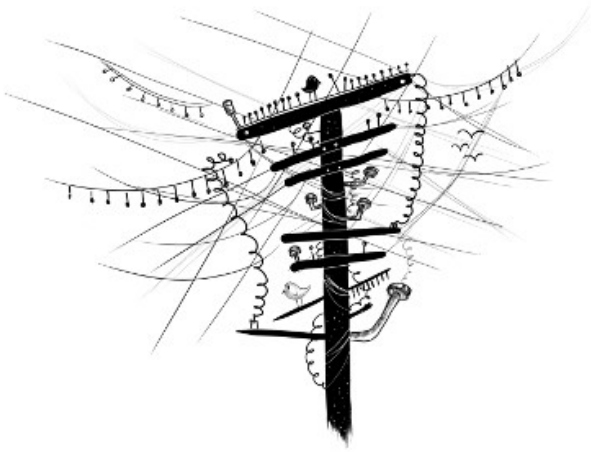
Node.js

A student asked, ‘The programmers of old used only simple machines and no programming languages, yet they made beautiful programs. Why do we use complicated machines and programming languages?’. Fu-Tzu replied, ‘The builders of old used only sticks and clay, yet they made beautiful huts.’

- *Master Yuan-Ma, The Book of Programming*



So far, we have used the JavaScript language in a single environment: the browser. This chapter and the next one will briefly introduce Node.js, a program that allows you to apply your JavaScript skills outside of the browser. With it, you can build anything from small command line tools to HTTP servers that power dynamic websites.

These chapters aim to teach you the main concepts that Node.js uses and to give you enough information to write useful programs for it. They do not try to be a complete, or even a thorough, treatment of the platform.

Whereas you could run the code in previous chapters directly on these pages, because it was either raw JavaScript or written for the browser, the code samples in this chapter are written for Node and often won’t run in the browser.

If you want to follow along and run the code in this chapter, you’ll need to install Node.js version 10.1 or higher. To do so, go to https://nodejs.org and follow the installation instructions for your operating system. You can also find further documentation for Node.js there.

Background

One of the more difficult problems with writing systems that communicate over the network is managing input and output—that is, the reading and writing of data to and from the network and hard drive. Moving data around takes time, and scheduling it cleverly can

make a big difference in how quickly a system responds to the user or to network requests.

In such programs, asynchronous programming is often helpful. It allows the program to send and receive data from and to multiple devices at the same time without complicated thread management and synchronization.

Node was initially conceived for the purpose of making asynchronous programming easy and convenient. JavaScript lends itself well to a system like Node. It is one of the few programming languages that does not have a built-in way to do in- and output. Thus, JavaScript could be fit onto Node's rather eccentric approach to in- and output without ending up with two inconsistent interfaces. In 2009, when Node was being designed, people were already doing callback-based programming in the browser, so the community around the language was used to an asynchronous programming style.

Source - https://eloquentjavascript.net/