Javascript Outside of the Browser

Intro to NodeJS

Javascript, despite its quick and dirty beginnings, disparity between browsers, and a number of other factors, is now one of the most popular and important programming languages today.

Despite its popularity, it used to always be exclusive to scripts in the web browser. In order to handle actions such as GET requests or form submissions, developers would need to program in PHP or Java.

Today, developers can use NodeJS to program their entire stack in Javascript. NodeJS is a standalone Javascript runtime that runs outside of the browser. Behaviorally, it basically works like Javascript in the browser.

Besides allowing developers to program the front and back end of their web applications in one language, NodeJS also introduces native event-driven program flow to back-end development. Instead of running a process that waits for a user action, such as requesting a resource, NodeJS will fire an event whenever an action occurs, and handlers can be programmed to respond to that action. It's just like attaching event listeners to HTML elements in the browser. In addition, while the "main thread" of NodeJS is single-threaded just like Javascript, it's capable of running processes concurrently without blocking. For example, NodeJS can write multiple files at the same time without requiring complex setup.

But NodeJS is more than just a web server replacement. It can be used for any number of things, such as writing automated tests, utility scripts, and for our purposes, build scripts for transpiling code.

I'm not going to detail how to download NodeJS, because installation steps may vary. Installation steps can be found on the nodejs.org website.

The version of NodeJS we will be using for these tutorials is 11.11.0. You probably will not have any issues as long as the version you're using is close to this.

Once you install NodeJS, verify that you're able to run it by opening you terminal or command prompt and starting the REPL (Read-Eval-Print-Loop):

```
node
> console.log('Hello, world!')
> 2 + 2
```

Once you're able to do that, you'll be ready for the next session.