Node.js

Learning Objectives

- Understand what Node.js is
- Understand what npm is
- Understand how to use Node.js and npm

Material

What is Node.js?

The code we write is not executed in a vacuum - it needs to interact with other software. JavaScript was originally designed to run in web browsers alongside HTML and CSS. Your JavaScript running in the browser will have access to a set of APIs that allow it to do useful things like make network requests with fetch or modify the page's HTML. These APIs are not strictly part of the JavaScript language itself, they're part of JavaScript's environment.

In 2009, Node.js was created to allow JavaScript to be run outside of a browser. Node.js is therefore a new *runtime environment* - it's the same JavaScript, just in a new context. Rather than having access to APIs for manipulating web pages, Node.js gives you access to APIs useful for server-side tasks such as handling files or performing cryptography. Node.js is now a popular technology for backend development, being particularly suited for handling lots of concurrent requests.

Node.js can be downloaded from https://nodejs.org/en/download/. Once installed, a JavaScript file can be executed by Node.js in the command-line using the `node` keyword before the filename:

```
node myCode.js
```

What is npm?

Npm is Node.js' package manager. It can be used to easily install other 3rd party libraries or to publish your own modules for others to use.

How to setup a new Node.js project

First, create a new directory to house your project. Next run:

```
npm init -y
```

The `init` tells npm that this directory is a new project. The `-y` flag just gives you all the default settings. The command should've created a `package.json` file. This file gives some meta-information about the project, lists the project's dependencies (the modules you install) and can also include scripts.

We can now install a 3rd party module. For example, let's install Express.js - a Node.js framework for creating APIs. To do this we can run:

```
npm install express
```

You should now see your project contains both a `node_modules` directory and a `package-lock.json` file. `node_modules` contains the actual code for all the libraries you install. `package-lock.json` lists your project's dependencies in full along with their accepted versions.

If you're planning to push your code with git, you should include `package-lock.json` and `package.json` but **not** `node_modules`. Another developer cloning your repository can simply run `npm install` and npm will look through the dependencies in the package files and download everything it needs from the npm registry. To avoid including `node_modules`, simply add it to your `.gitignore` file.

Using Node Modules

In order to use a node module we've installed, we need to import it into our .js file. The CommonJS way of doing this is to use the `require` function then the name of our module:

```
//app.js
const express = require("express"); // import express

const app = express();
```

Most modules' npm pages will include a code snippet showing a basic example of importing and using their package.

Core Assignments

Node 101

First, make sure you have Node.js installed. Try and memorise the steps above for creating a new project using Node.js. Then, see if you can create a Node.js project without looking at the notes. After completing the basic setup, you should install the chalk package - have a read on its npm page to see what it does then have a go at using it yourself.

Extension Assignments

Node 101++

Push your code to Github without including node_modules. Clone your repository in a new folder and use the `npm install` command to install all its dependencies.