Lesson 1 Learn NodeJS

Intro to NodeJS and the server environment

What is NodeJS?

- Node.js is a JavaScript runtime built on the V8 engine that allows you to run JavaScript code outside of a web browser.
- The V8 engine is an open-source JavaScript engine developed by Google. It's designed to compile and execute JavaScript code at lightning-fast speeds.
- This means that developers can use JavaScript to build server-side applications, command-line tools, and even desktop applications, in addition to web applications that run in the browser.
- Running JavaScript outside of the browser also means that developers can access and interact with system-level resources, such as the file system, network sockets, and other devices, which are not available in a web browser environment.

NodeJS unlocks system-level resources

Server environments vs Web environments

- Server environments execute server-side code, while web environments execute client-side code in a web browser.
- With web development most of the code that we write gets sent to the end users web browser where it is run in the browser environment (vs run on a server)
- Writing code in a server environment our project can be self-hosted, secured, and can serve end users without them having any kind of access
- There are a ton of usecases unlocked when building in a server / system level environment but we will be focussing exclusively on the fundamentals (Building and API and a database)

Let's get our hands dirty!

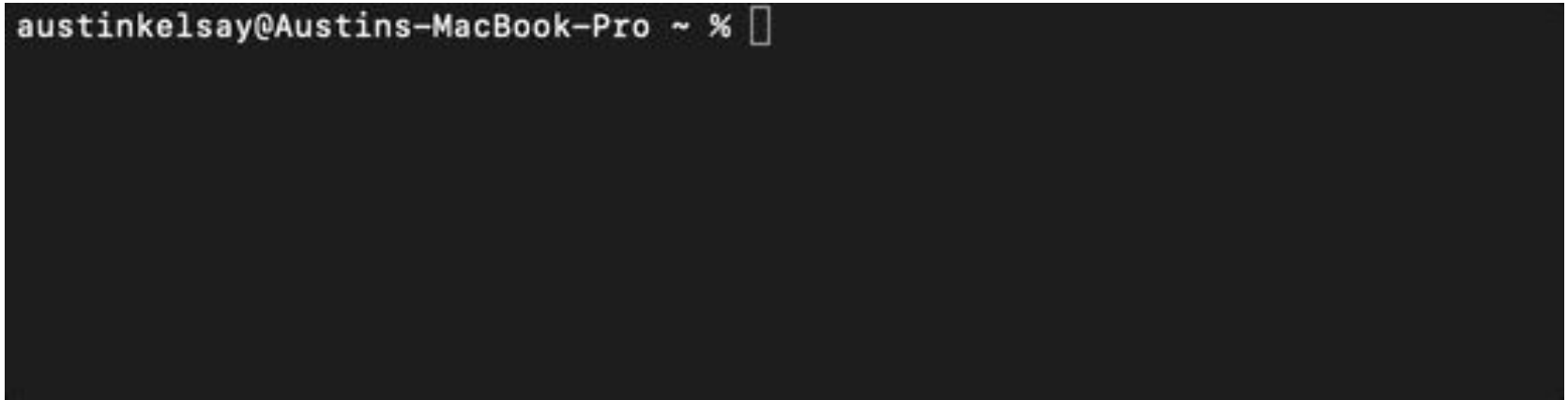
What do we need to know to get started writing some NodeJS

NodeJS Modules

- Node.js uses the CommonJS module system, which allows you to split your code into separate modules that can be imported and exported.
- Using modules helps keep your code organized and manageable, especially as your project grows in size and complexity.
- Modules are self-contained units of code that have their own variables and functions, which can be accessed and used by other parts of your code through exports and imports.
- With CommonJS modules, you can easily share code between different files in your project, and also reuse code from third-party libraries and packages.

Node package manager (npm)

- NPM is a package manager for JavaScript that allows developers to easily install and manage third-party packages (i.e., libraries, frameworks, tools, etc.)
- NPM provides a centralized repository of over 1 million open-source packages that can be installed using a simple command-line interface, making it easy to find and use pre-built code
- NPM also allows developers to easily share their own packages with the community making it globally available and open source



Working with modules in NodeJS

- Node.js has a built-in **require** function that allows you to import modules from other files or libraries.

```
const library = require('library');
```

- Node.js provides many built-in modules that you can use to perform common tasks, such as reading and writing files, making HTTP requests, and working with the file system.

Using your own modules

```
// greeting.js
const greeting = "Hello, World!";

module.exports = greeting;




// index.js
const greeting = require('./greeting');

console.log(greeting);
```

Using Node's built in modules

```
const fs = require('fs');


fs.readFile('example.txt', 'utf8', (err, data) => {
 if (err) throw err;
 console.log(data);
});
```

Using someone else's module!

```
const bitcoin = require('bitcoinjs-lib');


const keyPair = bitcoin.ECPair.makeRandom();
const address = keyPair.getAddress();


console.log(`Bitcoin address: ${address}`);
```

Mini Project

A command-line application that greets the user and displays the current date and time

Setup a NodeJS project locally

To set up a new Node.js project, follow these steps:

- Create a new directory for your project by running the following command in the terminal: `mkdir hello-node`
- Navigate into the new directory by running: `cd hello-node`
- Initialize a new Node.js project by running: `npm init -y`
- Create an index.js file either in your code editor or by running `touch index.js`

- You can now put `console.log("Hello, world!")` in index.js

- In your terminal run `node index.js` to run your code!

Reading input from the console

```
const readline = require('readline');

// Create an interface for reading input from the console
const rl = readline.createInterface({
 input: process.stdin,
 output: process.stdout
});

// Get the current date and time
const now = new Date();

// Ask the user for their name
rl.question('What is your name? ', (name) => {
 // Display the greeting and current date and time
 console.log(`Hello, ${name}! The current date and time is ${now.toString()}.`);

 // Close the readline interface
 rl.close();
});
```

Overview of the code

1. First, we import the **readline** module, which allows us to read input from the console.
2. We create a readline interface using **readline.createInterface**, specifying **process.stdin** as the input stream and **process.stdout** as the output stream.
3. We create a new **Date** object to get the current date and time.
4. We use the **rl.question** method to ask the user for their name. This method takes two arguments: the prompt to display to the user, and a callback function to execute when the user enters a response.
5. In the callback function, we use **console.log** to display a greeting and the current date and time, using string interpolation to include the user's name and the date and time.
6. Finally, we call **rl.close()** to close the readline interface and allow the program to exit.

Review

- Node.js is a JavaScript runtime that allows you to run JavaScript code outside of a web browser. It provides a range of built-in modules and tools that allow you to build server-side applications and command-line tools.
- Server environments and web environments have different use cases and involve different types of code execution. By understanding the differences between them, you can choose the appropriate environment for your project and build applications that are optimized for their intended use case.
- When building a project with Node.js, it's important to take advantage of its built-in modules and tools, as well as external libraries and dependencies available through npm. This will help you streamline your development process and create applications that are efficient, scalable, and maintainable.

Resources

1. Node.js Official Documentation - The official Node.js documentation provides a comprehensive guide to getting started with Node.js, including installation, basic usage, and more advanced topics.
2. Node.js Crash Course on YouTube - This 1-hour crash course video provides a beginner-friendly introduction to Node.js and covers the basics of building a simple web server.
3. Node.js Best Practices - a GitHub repository with a collection of best practices for Node.js development, covering topics such as code organization, error handling, and testing.
4. Learn Node.js on Codecademy - Codecademy's Node.js course is a hands-on tutorial that guides you through building a complete web application with Node.js and Express.
5. Node.js Tutorials on TutorialsPoint - TutorialsPoint provides a collection of Node.js tutorials that cover the basics of Node.js, as well as more advanced topics like Node.js with MongoDB and Node.js with MySQL.
6. Node.js Basics Cheat Sheet - This cheat sheet provides a quick reference to the basics of Node.js, including installing Node.js, running Node.js programs, and using the built-in modules.