Chapter 1. Our Development Environment

John Wooden, the late coach of the UCLA men's basketball team is one of the most successful coaches of all time, winning 10 national championships in a twelve year period. His teams consisted of top recruits, including hall-of-fame players such as Lew Alcindor (Kareem Abdul-Jabbar) and Bill Walton On the first day of practice, John Wooden would sit down each of his new recruits, players who had been the best in the country in high school, and teach them to put on their socks properly. When asked about this, Wooden stated that "it's the little details that make the big things come about."

Chefs use the term mise en place, meaning "everything in its place," to describe the practice of preparing the tools and ingredients required for the menu prior to cooking. This preparation enables the kitchen's cooks to successfully prepare meals during busy rushes, as the small details have already been considered. Much like Coach Wooden's players and chefs preparing for a dinner rush, it is worth dedicating time to setting up our development environment.

A useful development environment does not require expensive software or top of the line hardware. In fact, I'd encourage you to start simple, use open source software, and grow your tools with you. Though a runner prefers a specific brand of sneakers and a carpenter may always reach for her favorite hammer, it took time and experience to establish these preferences. Experiment with tools, observe others, and over time you will create the environment that works best for you.

In this chapter we'll install a text editor, Node.js, Git, MongoDB, and several helpful JavaScript packages as well as locate our terminal application. It's possible that you already have a development environment that works well for you, however we will also be installing several required tools that will be used throughout the book. If you're like me and typically skip over the instruction manual, I'd still encourage you to read through this guide.

If you find yourself stuck at any point, please reach out to the JavaScript Everywhere community, via our Spectrum channel at spectrum.chat/jseverywhere.

Your text editor

Text editors are a lot like pants. We all need them, but our preferences may vary wildly. Some like simple, well-constructed, and timeless. Some prefer the flashy paisley pattern. There's no wrong decision and you should use whatever makes you most comfortable.

If you don't already have a favorite, I highly recommend Visual Studio(VS) Code. It's an open source editor that is available for Mac, Windows, and Linux. Additionally, it offers built-in features to simplify development and is easily modified with community extensions. It's even built using JavaScript!

The terminal

If you're using Visual Studio Code, it comes with an integrated terminal. For most development tasks, this may be all you need. Personally, I find using a dedicated terminal client preferable as I find it easier to manage multiple tabs and use more dedicated window space on my machine. I'd suggest trying both out and find what works best for you.

Using VSCode

To access the terminal in VSCode, click `View > Integrated Terminal`. This will present you with a terminal window. The prompt will be be present in the same directory as the current project.

Using the built in terminal

All operating systems come with a built in terminal application and this is a great place to get started. On OS X it is called, fittingly enough, Terminal. In Windows, the program is PowerShell. The name of the terminal for Linux distributions may vary, but often include "Terminal."

Navigating the file system

Once you've found your terminal, the most critical ability you will need is the ability to navigate the file system. This can be done using the `cd` command, which stands for "change directory."

COMMAND LINE PROMPTS

When looking at terminal instructions they will often include a `$` or `>` at the start of the line. These are used to designate the prompt and should not be copied. In this book, I'll be designating the terminal prompt with a dollar sign (`$`). When entering instructions into your terminal application, do not copy the `$`.

When we open our Terminal application, we'll be presented with a blinking cursor prompt. By default, we are in our computer's home directory. If you haven't already, I'd recommend making a `Projects` folder that is a sub-directory within your home directory. This folder can house all of your development projects. To navigate into that folder you would type:

```
$ cd Projects
```

Now let's say that we can have a folder called `jseverywhere` in our Projects directory. We can type `cd jseverywhere` from the Projects directory to navigate into there. To navigate backwards a directory (in this case, to Projects), we would type `cd ..`

All together, this would look something like:

```
> $ cd Projects # navigate from Home dir to Projects dir
/Projects > $ cd jseverywhere # navigate from Projects dir to
jsevewehre dir
/Projects/jseverwhere > $ cd .. # navigate back from jseverwhere to
Projects
/Projects > $ # Prompt is currently in the Projects dir
```

If this is new to you, spend some time navigating through your files until you're comfortable. I've found that file system issues are a common tripping point for budding developers. Having a solid grasp of this will provide you with a solid basis for establishing your workflows.

Command Line Tools and Homebrew (Mac Only)

Certain command line utilities are only available to macOS users once Xcode is installed. You can jump through this hoop, without installing Xcode by installing `xcode-select` via your terminal. To do so, run the following command and click through the install prompts:

```
$ xcode-select --install
```

Homebrew is a package manager for macOS. It makes installing development dependencies, like programming languages and databases as simple as running a command line prompt. If you use a Mac, it will dramatically simplify your development environment. To install Homebrew, either head over to brew.sh to copy and paste the install command, or type the following:

```
$ /usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/in
stall)"
```

Node.js and npm

Node.js is "a JavaScript runtime, built on Chrome's V8 JavaScript Engine." In practical terms this means that Node is a platform that allows developers to write JavaScript outside of a browser environment. Node.js enables us to write server-side applications in JavaScript. Bundled along with Node.js, comes npm, the default package manager for Node.js. npm enables us to install thousands of libraries and JavaScript tools within our projects.

MANAGING NODE.JS VERSIONS

If you plan on managing a large number of Node projects you may find that you also need to manage multiple versions of Node on your machine. If that's the case, I recommend using nvm to install Node. nvm is a script that enables you to manage multiple active Node versions. For Windows users, I recommend nvm-windows. I won't be covering Node versioning, but it is a helpful tool. If this is your first time working with

Node, I recommend proceeding with the following instructions for your system.

Installing Node.js and npm for macOS

macOS users can install Node.js and npm using Homebrew. To install Node.js, type the following command into your terminal:

```
$ brew update
$ brew install node@10.16
```

With Node installed, open your terminal application to verify it is working.

```
$ node --version
## Expected output v10.16.3
$ npm --version
## Expected output 6.2.0
```

If you see a version number after typing those commands, congratulations you've successfully installed Node and npm for macOS!

Installing Node.js and npm for Windows

For Windows, the most straightforward way to install Node.js is to visit Node.org and download the installer for your operating system.

First, visit Node.organd install the LTS version (10.15.3 at the time of writing), following the installation steps for your operating system. With Node installed, open your terminal application to verify it is working.

```
$ node --version
## Expected output v10.15.3
$ npm --version
## Expected output 6.2.0
```

If you see a version number after typing those commands, congratulations you've successfully installed Node and npm for Windows!

MongoDB

MongoDB is the database that we will be using in the development of our API. Mongo is a popular choice when working with Node.js,

because it treats our data as JSON documents. This means that it's comfortable for JavaScript developers to work with from the get-go.

Installing MongDB for macOS

To install MongoDB for macOS, first install with Homebrew:

```
$ brew update
$ brew install mongodb
```

Now, we will create a directory to which Mongo will write our data and give it the proper permissions. Within your terminal, run the following commands:

```
$ cd
$ sudo mkdir -p /data/db
# If prompted, enter your password when prompted and press `return`
sudo chown -R `id -un` /data/db
# If prompted, enter your password when prompted and press `return`
```

To verify that Mongo has installed and start the Mongo Daemon, type `mongod` into your terminal. This should start the Mongo server. To stop the server and exit the Mongo process, hold `ctrl+c`.

Installing MongoDB for Windows

To install MongoDB for Windows, first download the installer from the MongoDB Download Center at https://www.mongodb.com/download-center/community. Once the file has downloaded, run the installer.

Once installation is complete, we will need to create a directory in which Mongo will write our data. Within your terminal, run the following commands:

```
$ cd C:\
$ md "\data\db"
```

To verify that Mongo has installed and start the Mongo Daemon, type `C:\mongodb\bin\mongod.exe` into your terminal. This should start the Mongo server.

TROUBLE INSTALLING MONGO ON WINDOWS?

Note that, the location of `mongod.exe` may be different depending on your system preferences. If that prompt doesn't work, you may need to track it down. The MongoDB installation documentation offers a more extensive Windows installation guide.

Git

Git is the most popular version control software, allowing you to do things like copy code repositories, merge code with others, and create branches of your own code that do not impact one another. Git will be helpful for "cloning" this book's sample code repositories, meaning it will allow you to directly copy a folder of sample code. Depending on your operating system, Git may already be installed. Type the following into your Terminal window:

```
$ git --version
```

If a number is returned, congrats you're all set! If not, visit git-scm.com to install Git, or use Homebrew for macOS. Once you've completed the installation steps, once again type `git --version` into your terminal to verify that it has worked.

Expo

Expo is a toolchain that simplifies the bootstrapping and development iOS and Android projects with React Native. We will need to install the Expo command line tool and, optionally (though recommended), the Expo app for iOS or Android. We'll cover this in more detail in the mobile application portion of the book, but if you're interested in getting a head start visit expo.io to learn more. To install the command line tools, type the following into your terminal:

```
npm install -g expo-cli
```

To install the Expo application, visit the Apple App Store of Google Play Store on your device.

Prettier

Prettier is a code formatting tool with support for a number of languages including JavaScript, HTML, CSS, GraphQL, and Markdown. It makes it easy to follow basic formatting rules, meaning that when you run the Prettier command, your code is automatically formatted to follow a standard set of best practices. Even better, you can configure your editor to do this automatically every time you save a file. This means that you'll never again have a project with things like inconsistent spaces and mixed quotes.

I recommend installing Prettier globally on your machine and configuring a plugin for your editor. To install Prettier globally, go to your command line and type

```
npm install --global prettier
```

Once you've installed Prettier, visit Prettier.io to find the plugin for your text editor. With the editor plugin installed, I recommend adding the following settings, within your editor's settings file:

```
"editor.formatOnSave": true,
"prettier.requireConfig": true
```

These settings will automatically format files on save, whenever a `.prettierrc` configuration file is within the project. The `.prettierrc` file specifie specific options for prettier to follow. Now whenever that file is present, your editor will automatically reformat your code to meet the conventions of the project. Each project within this book will include a `.prettierrc` file.

ESLint

ESLint is a code linter for JavaScript. A linter differs from a formatter, such as Prettier, in that a linter also checks for code quality rules, such as unused variables, infinite loops, and unreachable code that falls after a return. Much like Prettier, I recommend installing the ESLint plugin for your favorite text editor. This will alert you to errors in real time as you

write your code. You can find a list of editor plugins on the ESLint Website, at eslint.org/docs/user-guide/integrations.

Similar to Prettier, projects can specify the ESLint rules they would like to follow within an `.eslintrc` file. This allows project maintainers fine grained control over their code preferences and a means to automatically enforce coding standards. Each of the projects within this book will include a helpful, but permissive set of ESLint rules, aimed at helping you to avoid common pitfalls.

Making things look nice

This is optional, but I've found that I enjoy programming just a bit more when I find my setup aesthetically pleasing. I can't help it, I have a degree in the arts. Take some time and test out different color themes and typefaces. Personally, I've grown to love the Dracula Theme, which is a color theme available for nearly every text editor and terminal, along with Adobe's Source Code Pro typeface.

Conclusion

In this chapter we've set up a working and flexible JavaScript development environment on our computer. One of the great joys of programming is personalizing your environment. I encourage you to experiment with the themes, colors, and tools that you use to make this environment your own. In the next section of the book, we will put this environment to work by developing our API application.