COMP6080
Web Front-End Programming

NPM Intro

Problems that we face

When we want to use an external library in NodeJS, we have to "install" it. E.G. To write a script to print out today's date, with a helpful library, we would do this:

```
1 const moment = require('moment');
2
3 console.log(moment().format("MMMM Do YYYY"));
```

And then once we install it, how do we keep track of that so our teammates know to install it too?

Problems that we face

This leads us to having to solve two key problems:

- Sourcing Dependencies
- Package (and verison) management

NPM

NPM (Node Package Manager) is a tool installed alongside NodeJS to manage dependencies/modules for NodeJS projects.

We can install dependencies with **npm install [dependency]** and it's automatically added to our workspace and installed. This updates our package.json and when we push this to git, our team members simply need to run **npm install** after the next pull to install the modules that you've added.

NPM

The key files and folders relevant to NPM are:

package.json: Where we store meta data about our project including a list of dependencies to install

node_modules: Where the installed dependencies are.

package-lock.json: Where we store versioning information about dependencies to ensure everyone has the right versions (oversimplification)

NPM

It's VERY important that:

- **package.json** and **package-lock.json** are in the git repository
- **node_modules** is not in the git repository

Global Installs

By default installs are made in the project directory (inside the **node_modules** folder) of where you run the command. However, if you want the module to be accessible across all modules you can install it globally.

```
1 npm install -g create-react-app
```

This is usually only recommended for utilities, like **create-react-app**.

NPX

NPX is a command line tool that takes a module and installs it before running it as an executable. This is used often when using utility libraries that are maintained by NPM, e.g. **create-react-app**.

```
1  npx create-react-app
```

yarn, pnpm

Besides NPM, there are other package managers, notably

- yarn
- pnpm

They are extremely similar, and for a novice the only differences are the specific commands that are run.

Yarn is used very often with react apps (covered in this course), and the only key difference with yarn is that **package-lock.json** is instead called **yarn.lock.**