NPM usage

```
npm init # create a package.json
npm install package-name # install package
# & add --save to also add in package.json
npm install --save package-name
npm install # Install all req's in package.json
node file.js # Run a given file with node
```

Node Terminology

Node.js Run-time spin-off from Chrome that allows us to use node on the "server side" (e.g. in the terminal)

NPM Node Package Manager - used for downloading JavaScript packages, and other JS-dev related tasks.

dependencies Packages that are required to be pre-installed for a given package to be able to run.

`node_modules` Contains your downloaded dependencies.

`package.json` Contains meta-data, dependencies, and NPM configuration about a project.

React Terms

JSX JavaScript variant allowing HTML in JS

Webpack Tool that combines and processes JS, CSS and other static assets

Babel Tool that compiles or translates from one JS variant to another

render React renders templated HTML to the page.

state Keep the variables that change in state, and modify them with setState to rerender your page

JSX Examples

```
// Single HTML element
const paragraph = (
    <p>Lorem ipsum</p>
);

// JSX Templating
const color = "green";
const pWithTemplating = (
    <p>Favorite color: {color}</p>
);

// Loops within JSX
const data = ["alice", "bob", "candice"];
const divOfParagraphs = (
    <div>
        {
            data.map(item => (
                <p>Name: {item} </p>
            ))
        }
    </div>
);
```

Full React Example

```
import React, { Component } from "react";

// CSS & images can be included magically
import "./App.css";
import sendIcon from "./images/envelope.png";

class App extends Component {
    // Define starting state
    state = {
        message: "",
        users: [],
    };

    // Define methods. Must have for forms.
    onMessageChange = (ev) => {
        const value = ev.target.value;
        this.setState({ // Modify state
            message: value,
        });
    }

    // Special method called when page loads
    // useful for fetching initial data
    componentDidMount() {
        fetch("http://some.com/api/")
            .then(response => response.json())
            .then(data => {
                console.log("Data received:", data);
                this.setState({
                    messages: data.messages,
                });
            });
    }

    render() {
        // Any code can go in render() before return
        const messageParagraphs = [];

        // Looping "state" data from API
        for (const m of this.state.messages) {
            // JS and HTML can be mixed, such as for
            // building lists of HTML to be included
            messageParagraphs.push(
                <p>{m}</p>
            );
        }

        // Return what you want visible
        return (
            <div className="App">
                <h1>Messenger</h1>
                {messageParagraphs}
                <input onChange={this.onMessageChange}
                    value={this.state.message} />
                <button onClick={() => alert("Hi")}>
                    <img src={sendIcon} />
                    Send message
                </button>
            </div>

        );
    }
}
```