Lesson 15

The focus of this lesson will be on tools you should expect to use as part of more advanced web development projects. Two of these tools are:

- Node Package Manager (NPM)
- React

Lesson Recordings:

https://vimeo.com/720588623/ecf828d855

https://vimeo.com/720625689/168d7f1ab7

https://vimeo.com/720659353/c98f9346bb

Node Package Manager (NPM)

NPM is a tool that allows you to distribute your code without needing to include open source dependencies that can significantly increase the storage size of your project. This makes it easier for you to share your code with other developers and easier for them to install using the NPM install tool. Today's exercise covers the steps you need to initiate NPM with your project.

Step 1: Project Initiation

The first step is to initialise your project with NPM. This is achieved using a special JSON file called "package.json" that describes the settings for your projects. Fortunately, NPM provides a command line tool that helps you to easily create this file.

First, open your computer's terminal tool to access the command line, and then navigate to your project directory using the **cd** command:

```
cd <your-project-directory>
```

Once in your project directory, you can use the following command to call the NPM initiation tool to create your package.json file:

```
npm init
```

The NPM initiation tool asks you for the details to populate the package.js file with:

- name - The name you want your project to be referenced as.
- version - The version number of the latest project build in the format x.x.x - e.g. 1.0.1 .
- description - A brief description detailing what the project is about.
- main - The file to execute the project from; equivalent to index.html for a standard website.
- scripts - The scripts used by the project.
- keywords - A list of keywords that can be used for search purposes.
- author - Name of the person who created the project - i.e. you.
- license - The name of the license you want to distribute your code under - e.g. MIT is a popular license for open source software.

Once you've entered the above details, the NPM tool will ask you to confirm whether you would like to create the package.json file - **type yes and press return**. You will then see the package.js file appear in your project directory.

Step 2: Package Installation

The next step is to install the components that your code will use. NPM makes this easy with using the "install" command; all you need to do is type the command with the package name and it will be automatically be downloaded to your project's "node_modules" folder:

```
npm install <package-name>
or
npm i <package-name>
```

The following are two examples that you could use for <package-name>:

- express
- react

Flags can also be used with NPM install to perform additional actions with the package download:

- --save - Updates the "dependencies" field inside the package.json to describe the inclusion of the downloaded package.
- --save-dev - Similar to using --save, but saves the package details under the "devDependencies" field instead. This is useful for configuring package.json to distinguish the packages that are only required for development from those that are required to operate the main application.
- --global - Allows you to install scripts to your Node installation so that it's available to all of your projects - i.e. not just the current project.

Let's look at examples of how these are applied:

```
npm install <package-name> --save
```

```
npm install <package-name> --save-dev
```

```
npm install <package-name> --global
or
npm install <package-name> --g
```

React

The second technology you are being introduced to in this lesson is React - a popular Javascript library that provides several advantages for web development projects.

- Out of the box functionality - i.e. no need to develop them yourself.
- Encourages developers write consistent code - i.e. the React way of code.
- Overcomes multiple quirks of the DOM for complicated web pages and web apps.

Today is only an introduction to using React, so the lesson will be based on using an online version to avoid any need for installation. Go to the following URL in your web browser to access an online code editor that supports React:

https://codesandbox.io/s/new

Q: Why React?

Development teams need to create code that multiple people can work with. The main advantage of using frameworks and libraries like React is to have a standardised way for developers to write complicated code. This allows developers to automatically know where to look for specific functionalities created within a project without the need for training. It also allows teams to become more productive by making use of reusable components that have been created for the framework - i.e. time saved by writing the functionality yourself.

Q: What do I need to know about components?

Think of components as self contained programs that you can call into use anywhere in your app - much like functions.

Also like functions, components return data to wherever they are called from. This data typically consists of XML / JSX content describing what should be displayed on the web page.

Component names must begin with capital letter.

Q: What's the purpose of index.js?

This is the default Javascript file that begins the React app - although you can change this in the "main" field of the project's package.json file.

Consider this main file, index.js by default, as the file to initiate the presentation of your app. You can define how components such as navigation, header, footer and the main app are placed on the page.

Q: How do I send data to a component?

Like standard Javascript functions, components can accept data as input that can be used to affect their behaviour. You can attach data attributes to the XML / JSX descriptions used to place and call a component. These attributes are collected and supplied to the component's execution function as its first parameter. Take note - terminology used by React labels these as props, but they are just attributes that have been covered for standard Javascript.

Here is an example of calling a data with the title attribute / prop:

```
<App title="ABC" />
```

The component function can be written to use the title attribute / prop like the following:

```
export default function App(props) {
  return (
    <div>
      <h1>{props.title}</h1>
      <p>Standard paragraph content...</p>
    </div>
  );

}
```

React Code

src/index.js

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";


import App from "./App";


const rootElement = document.getElementById("root");
const root = createRoot(rootElement);


root.render(
  <StrictMode>
    <h1>Navigation</h1>
    <hr />
    <App title="123" list={[5, 10, 15]} />


    <App />
  </StrictMode>
);
```

src/App.js

```
import "./styles.css";


export default function App(props) {
  console.log(props.list);


  let className = "App";
  if (props.title) {
    className = className + " active";
  }


  return (
    <div className={className}>
      {props.title && <h1>Title: {props.title}</h1>}
      <h1>Hello {props.title}</h1>
      <h2>Start editing to see some magic happen!</h2>
      <button
        onClick={function () {
          message("ABC");
          alert("789");
        }}
      >
        Click Me
      </button>
    </div>
  );


  function message(m) {
    alert(m);
    alert(123);
  }
}
```

src/styles.css

```css
.App {
  font-family: sans-serif;
  text-align: center;
}


.App.active,
.App .active {
  background: yellow;
}
```