Bootstrap and React for Web Development

L A Liggett

2019-07-16

Contents

Chapter 1

Introduction

Chapter 2

Bootstrap

2.1 Setup

Create a folder that will contain the webfiles use npm to initialize a `package.json` file. Follow through the prompts to add the desired information. Set the entry point to be `index.html`. It can also be helpful to add the node_modules folder to `.gitignore`.

```
npm init
```

Then just initialize some basic `index.html` file for testing purposes.

```
<body>
    <h1>This is a Header</h1>
    <p>This is a paragraph</p>
</body>
```

Install the lite server, which will serve up the content from the folder. The `save-dev` flag will add the information to the json file that the lite-server should be used to serve the content. This should add lite-server under the devDependencies listing within `package.json` and a node-modules folder.

```
npm install lite-server --save-dev
```

Within the `package.json` file, add the start and lite listings so it looks like the following.

```
"scripts": {
  "start": "npm run lite",
  "test": "echo \"Error: no test specified\" && exit 1",
  "lite": "lite-server"
},
```

The lite-server can then be run using npm start.

```
npm start
```

Chapter 3

Hackernews

3.1 Setup

First make sure create react app is installed. The project here follows this tutorial. There are lots of other good looking tutorials like The React Handbook, and others at gitconnected.

```
npm i -g create-react-app
```

Then create a new directory for the app.

```
create-react-app hacker-news-clone
```

Change into the newly created directory and then create a file to handle environmental variables.

```
cd hacker-news-clone
touch .env
```

Within the `.env` file refer to the `src` folder. This will allow dependencies to be more easily imported. Add the following to the `.env` file.

```
NODE_PATH=src
```

Make a components directory within `src` to hold all of the components for the project.

```
mkdir -p src/components/App
```

Make a services directory within `src` to add additional functionality to the app and reference other site APIs.

```
mkdir src/services
```

Make a styles directory within `src` to add styles that can be used across the app.

```
mkdir -p src/styles
```

Make a store directory within `src` to add styles that will add Redux function.

```
mkdir -p src/store
```

Make a utils directory within `src` for shared functions across the app.

```
mkdir -p src/utils
```

Now move `App.js` to components just to keep the components bundled together. Rename `App.js` to index so that it can be imported from the mycomponents app.

```
mv src/App*js src/components/App/
mv src/components/App/App.js src/components/App/index.js
mv src/logo.svg src/components/App/
```

Delete the css files because style components will be used instead.

```
rm src/*css
```

Remove the imports of the css files in `src/components/App/index.js`.

```
import './App.css';
```

And remove the import within `src/index.js`.

```
import './index.css';
```