eCommerce Website Project

After creating your React app (npm create-react-app app-name), you can delete:

* ‘serviceworker’ code from index.js
* Within App.js, remove all content, including the ‘className’ within the div
* Delete the logo import
* Delete everything within App.css

Add SCSS to the project:

* run **npm add node-sass**

SASS/SCSS allows ‘nested’ styling. Using the ‘&; anything sitting between the .main-item element will receive all styling contained within:

.main-item{

width: 80%,

height: 80%

//Nested items

&:first-child {

margin-right: 7px

}

&:last-child {

margin-left: 7px

}

.content {

display: flex;

flex-direction: column;

align-items: center;

padding: 20px 80px;

}

/\* Any menu-item having a property of ‘large’, will attain this larger heightl \*/

&.large {

height: 380px;

}

}

Routing In React

Initially, **single page applications** didn’t have the ‘back button’ functionality. because the navigation within the application’s url didn’t change. This has now been fixed with the browser providing a ‘**History API**’. Libraries like React Router, and other single page application routing libraries, use this API to hijack and mimic urls.

The library we’re going to use is ‘**react-router-dom**’. This is the most popular and sought out library for routing in React.

To install react-router-dom:

* run **npm install react-router-dom**
* Within index.js, add, **import { BrowserRouter } from ‘react-router-dom’**
* Wrap the **<BrowserRouter>** component around the **<App />** component. What this does, is, give our application all of the function of routing:

**<BrowserRouter>**

**<App />**

**</ BrowserRouter>,**

**document.getElementById(“root”)**

Any component that gets rendered by our ‘Route’, gets passed three arguments:

* **history:**
* f push:
* **location:** tells us where we are in our application (ex: ‘pathname’ param)
* **match**:
* IsExact: is only true if the entire url matches the pattern
* params: an object of url parameters **(ex: ‘/:topicId’**. This can dynamically change)
* path: the pattern the router is looking to match
* url: the url of the page

The other ways to navigate is with the ‘**Link**’ component, and the ‘**history**’ and ‘**location**’ prop:

* i**mport { Route, Link } from react-router-dom**
* Syntax:
* **<Link to=”/topics”>TOPICS</ Link>**/\* With ‘Link’, we’re borrowing the url to tell the app what to re-render \*/
* For ‘history’ prop using a button:
  + **<button onClick={ () => props.history.push(‘/topics’) }>TOPICS</ button>**

With Router()

withRouter() is a higher orderl component. A higher orderl component is simply a component, that takes another component as an argument, and returns a modified component containing **match, history,** and **location** props.

* **import { withRouter } from ‘react-router-dom’**
* **export default withRouter(ComponentName);**

To map through an array and pass props to each item in the array:

**render() {**

**return (**

**<div className = 'directory-menu'>**

**{/\* Maps the sections, destructures, and returns the 'id' + the remaining props (...otherSectionProps) into the 'MenuItem' component\*/}**

**{ this.state.sections.map(({ id, ...otherSectionProps }) => (**

**<MenuItem key={ id } { ...otherSectionProps } />**

**))}**

**</div>**

    )

  }

**Also...**

**render() {**

**const { collections } = this.state;**

**return (**

**<div className='shop-page'>**

**{**

**collections.map(({ id, ...otherCollectionsProps}) => (**

**<CollectionPreview key={id} {...otherCollectionsProps} />**

**))**

**}**

**</div>**

**)**

**}**

To filter an array to display a certain amount of items, in this case, 4:

**const CollectionPreview = ({ title, items }) => (**

**<div className='collection-preview'>**

**<h1 className='title'>**

**{ title.toUpperCase() }**

**</h1>**

**<div className='preview'>**

**{**

**/\* Filter the array to display only 4 items \*/**

**items.filter(( item, idx ) => idx < 4).map( item => (**

**<div key={ item.id }>{item.name}</div>**

**))**

**}**

**</div>**

**</div>**

**);**