Module 1 Tutorial Lab

Personal Website using React Router

In this tutorial lab, we will be building a personal website using React Router. Here is an example of what it should look like at the end:

https://codesandbox.io/s/3qp8q073q

Here are a list of features our application will have:

- Navigation Bar with links to About, Resume, Projects, and Contacts

Step 1. Setting up your project

In this step, we will set up the project structure so that you can keep your project organized. A completed example of this step can be found here : https://codesandbox.io/s/p9wzm674oj

1. Go to https://codesandbox.io/s/new to create a new React project.
2. Fork the project so that you can save it to your account.
3. Give the project a title such as “Personal Website using React Router”
4. Add `react-router-dom` as a dependency.
5. Create a new folder inside the `src` folder. Name it `components`.
6. Create a new file inside the `components` folder. Name it `App.jsx`

7. Copy the following inside `App.jsx`:

```
import React from "react";
import { BrowserRouter } from "react-router-dom";

const App = () => {
  return (
    <BrowserRouter>
      <div>
        <h1>Hello World</h1>
      </div>
    </BrowserRouter>
  );
};

export default App;
```

The App component contains our entire app. We must include the BrowserRouter component so that other React Router components can work in the future.

8. Inside `index.js`, delete the existing App component and instead import App from `./components/app.jsx`:

```
import React from "react";
import ReactDOM from "react-dom";
import "./styles.css";
import App from "./components/App.jsx";

const rootElement = document.getElementById("root");
ReactDOM.render(<App />, rootElement);
```

9. The screen should display Hello World. You are now ready for the next step.

Step 2. Creating the NavBar

In this step, we will be creating the NavBar component which will allow us to navigate to different sections of our application. A completed example of this step can be found here : https://codesandbox.io/s/0oy7ro513n

1. Create a new file inside `components`. Name it `NavTab.jsx`. This will be a component that represents an individual tab on the NavBar.
2. Copy the following inside `NavTab.jsx`:

```
import React from "react";
import { NavLink } from "react-router-dom";

const NavTab = props => {
  var activeStyle = {
    color: "green",
    fontWeight: "bold"
  };

  var navStyle = {
    margin: "10px"
  };
  return (
    <NavLink style={navStyle} activeStyle={activeStyle} to={props.to
      {props.label}
    </NavLink>
  );
};

export default NavTab;
```

The NavTab component is just a NavLink component with some style and an activeStyle attribute that makes it turn green and bold when it is active. Its `to` attribute is passed down as a props and will determine where it will link to. Its label is also passed down as a props.

4. Create a new file inside `components`. Name it `NavBar.jsx`:

5. Copy the following inside `NavBar.jsx`:

```
import React from "react";
import NavTab from "./NavTab.jsx";

const NavBar = () => {
  return (
    <div>
      <NavTab to="/about" label="About" />
      <NavTab to="/resume" label="Resume" />
      <NavTab to="/projects" label="Projects" />
      <NavTab to="/contact" label="Contact" />
    </div>
  );
};

export default NavBar
```

The NavBar component contains several NavTab components and provides them with a couple of properties that will define where they are linked to and how they are labeled.

6. Now import the NavBar component into `App.jsx` and render it to test it out:

```
import React from "react";
import { BrowserRouter } from "react-router-dom";
import NavBar from "./NavBar.jsx";

const App = () => {
  return (
    <BrowserRouter>
      <div>
        <NavBar />
      </div>
    </BrowserRouter>
  );
};

export default App;
```

7. The screen should now show a Navigation bar. Clinking on the links should take you to /about, /resume, /projects and /contact. Try it out!

Step 3. Implementing the About, Resume, and Error components

In this step, we will be creating the About and Resume components that will be rendered when we click on the different links. A completed example of this step can be found here : https://codesandbox.io/s/7wvy84wq0j.

1. Create a new file inside the `components` folder. Name it `About.jsx`.
2. Copy the following inside `About.jsx`:

```
import React from "react";

const About = () => {
  return <h2>My name is YOUR NAME and I am a React Developer.</h2>;
};

export default About;
```

The About component is just a simple component that just displays some text. Try importing and rendering it individually within `App.jsx` to test if it works.

3. Create a new file inside the `components` folder. Name it `Resume.jsx`.
4. Copy the following inside `Resume.jsx`:

```
import React from "react";

const Resume = () => {
  return (
    <div>
      <h1>YOUR NAME</h1>
      <h2>Education</h2>
      <h3>YOUR SCHOOL</h3>
      <ul>
        <li>Bachelors of Science in Computer Engineering</li>
        <li>Class of 20XX</li>
        <li>GPA: X</li>
      </ul>
      <h2>Work History</h2>
      <h3>YOUR JOB TITLE</h3>
      <ul>
        <li>JOB DESCRIPTION 2</li>
        <li>JOB DESCRIPTION 1</li>
      </ul>
      <h2>Skills</h2>
      <ul>
        <li>React.js</li>
        <li>React Router</li>
        <li>Node.js</li>
        <li>SKILL</li>
      </ul>
    </div>
  );
};

export default Resume;
```

The Resume component is just a bunch of basic HTML. Feel free to edit edit the structure to make it your own. Try importing rendering it individually within `App.jsx` to test if it works.

5. Create a new file inside the `components` folder. Name it `Error.jsx`.
6. Copy the following inside `Error.jsx`:

```
import React from "react";

const Error = () => {
  return <h2>Error: Invalid URL</h2>;
};

export default Error;
```

The Error component is used to display an error when the user navigates to an invalid page url.

6. Import the Resume, About, Resume and Error components into `App.jsx`. Next add a Switch component and a couple of Route and Redirects components to `App.jsx`.

```
import React from "react";
import { BrowserRouter, Switch, Redirect, Route } from "react-router
import NavBar from "./NavBar.jsx";
import About from "./About.jsx";
import Resume from "./Resume.jsx";
import Error from "./Error.jsx";

const App = () => {
  return (
    <BrowserRouter>
      <div>
        <NavBar />
        <Switch>
          <Redirect exact path="/" to="/about" />
          <Route path="/about" component={About} />
          <Route path="/resume" component={Resume} />
          <Route path="/projects" component={null} />
          <Route path="/contact" component={null} />
          <Route component={Error} />
        </Switch>
      </div>
    </BrowserRouter>
  );
};

export default App;
```

Make sure to import all the components you need from the `react-router-dom` library. The Switch component will render the first Route or Redirect that matches. The Route components will each render a specific component relating the to the URL. Insert the About and Resume components into the `component` attribute of the `./about` and `./resume` Route components. Leave the other components null for now. We will implement those in the next steps.

7. Test out the app by clicking the links. You should see the About and Resume components render when you go to the appropriate links. You should also see the Error component render when you navigate to an invalid URL.

Step 4. Implementing the Projects component

In this step, we will be creating the Projects component and the Contact component. A completed example of this step can be found here : https://codesandbox.io/s/3qp8q073q

1. Create a new file inside the `components` folder. Name it `Projects.jsx`.

2. Copy the following inside `Projects.jsx`:

```
import React from "react";
import { Link, Switch, Route } from "react-router-dom";

const Projects = ({ match }) => {
  return (
    <div>
      <ul>
        <li>
          <Link to={match.url + "/game_project"}>Game Project</Link>
        </li>
        <li>
          <Link to={match.url + "/react_project"}>React Project</Lin
        </li>
        <li>
          <Link to={match.url + "/database_project"}>Database Projec
        </li>
        <li>
          <Link to={match.url + "/machine_learning_project"}>
            Machine Learning Project
          </Link>
        </li>
      </ul>
      <Switch>
        <Route path={match.url + "/:projectName"} render = { ({match

        <Route
          exact
          path={match.url}
          render={() => <div>Pick a project to view!</div>}
        />
      </Switch>
    </div>
  );
};

export default Projects;
```

The Projects component will contain several Links and Routes to individual projects. `match.url` is used to append the additional URL segments to whatever base URL was used to render the Projects component. This way, if the base url changes, the Link and

Route paths will still work correctly. Currently, there are only two Routes. The first route will match when the user visits `projects/:projectName`. It will simply display the projectName URL parameter, which is the second segment of the URL path. For example, if the URL is projects/myCoolProject, it will display myCoolProject. The second route will render a simple text statement when the URL is exactly `/projects`.

3. Create a new file inside the `components` folder. Name it `Contact.jsx`.

4. Copy the following inside `Contact.jsx`:

```
import React from "react";
import { Prompt } from "react-router-dom";

class Contact extends React.Component {
  constructor(props) {
    super(props);
    this.state = { value: "" };
  }
  handleChange(event) {
    this.setState({ value: event.target.value });
  }
  handleSubmit() {
    this.setState({ value: "" });
  }
  render() {
    var style = {
      width: 300,
      height: 60,
      margin: 10
    };
    return (
      <div>
        <input
          value={this.state.value}
          onChange={this.handleChange.bind(this)}
          style={style}
        />
        <div>
          <button onClick={this.handleSubmit.bind(this)}>Send</butto
        </div>
        <Prompt
          when={this.state.value !== ""}
          message="are you sure you want to leave without sending a 
        />
      </div>
    );
  }
}

export default Contact;
```

The Contact component contains a input form and because of that it must be a controlled component and must handle its own state. This is done by first updating the component state whenever there is an onChange event. To finalize this, the input display value is tied to the component state. This will ensure that the component state is always being displayed in the input form and that any changes to the input form will be reflected in the state.

There is also a Prompt component that will prompt the user when they try to navigate away and the form is filled out and not submitted. Since we have a controlled component, we can easily access the state from the input form and check if its empty or not.

5. Import the Projects and Contact components into `App.jsx` and add them into the appropriate Routes:

```
import React from "react";
import { BrowserRouter, Switch, Redirect, Route } from "react-router
import NavBar from "./NavBar.jsx";
import About from "./About.jsx";
import Resume from "./Resume.jsx";
import Error from "./Error.jsx";
import Projects from "./Projects.jsx";
import Contact from "./Contact.jsx";

const App = () => {
  return (
    <BrowserRouter>
      <div>
        <NavBar />
        <Switch>
          <Redirect exact path="/" to="/about" />
          <Route path="/about" component={About} />
          <Route path="/resume" component={Resume} />
          <Route path="/projects" component={Projects} />
          <Route path="/contact" component={Contact} />
          <Route component={Error} />
        </Switch>
      </div>
    </BrowserRouter>
  );
};

export default App;
```

6. You are all finished! Test out the application by clicking all the links to see different components rendered at different routes. Try clicking the additional links that appear when you visit `./Projects`. Also try filling out the form and navigating away without pressing submit in `./Contact`.

Again, the finished example can be found here: https://codesandbox.io/s/3qp8q073q