

COMP 3133 – Full Stack Development – Lab 10

- React Router

Developer Note:

- Create a new **create-react-app** application for this lab.

References:

<https://www.npmjs.com/package/react-router>

Exercise 1 – Basic Routes

1. Create a new React application using **create-react-app**

2. Install **React Router** using **npm install** command.

```
npm install --save react-router-dom
```

3. Navigate to the **App.js** file and remove the default JSX in the App component render method.

```
class App extends Component {  
  render() {  
    return (  
      );  
    }  
}
```

4. In the **App.js**, add an import **BrowserRouter** and **Route** from **react-router-dom**

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

5. In the **App.js**, above the App component add the following function component.

```
const NewRoute = () => {
  return (
    <div>
      <p>New Route</p>
    </div>
  );
};
```

6. Create a home component in the same **App.js**

```
const Home = () => {
  return (
    <div>
      <p>Home</p>
    </div>
  );
};
```

7. In the **App.js** component start setting up the Routes by adding the **BrowserRouter** and **Route** to the render method.

```
class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Route>
        </Route>
      </BrowserRouter>
    );
  }
}
```

8. To setup our routes we need to add **Link** and **Switch** to the import statement from **react-router-dom**

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

9. Add the first link and default path to load the **Home** component.

```

<BrowserRouter>
  <div>
    <ul>
      <li>
        <Link to="/">Home</Link>
      </li>
    </ul>
    <Route>
      <Route path="/" component={Home} />
    </Route>
  </div>
</BrowserRouter>

```

The browser should output as follows



- [Home](#)

Home

10. We need to add more Routes and Link to the Browser router. Use the **Switch** statement in this way to allow for multiple route paths. Test out the new routing links in the browser.

```

<BrowserRouter>
  <div>
    <ul>
      <li>
        <Link to="/">Home</Link>
      </li>
      <li>
        <Link to="/newroute">New Route</Link>
      </li>
    </ul>
    <Switch>
      <Route path="/" component={Home} />
      <Route path="/newroute" component={NewRoute} />
    </Switch>
  </div>
</BrowserRouter>

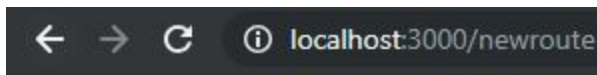
```

11. Testing the application the New Route will not working, since the default route "/" is being matched first when "/newroute" is requested. To solve this add the keyword **exact** to the default Home route.

```

<Route path="/" component={Home} exact />

```



- [Home](#)
- [New Route](#)

New Route

Exercise 2 – Multiple Routes

- 1.** Create a component folder and then three new functional components: About, Contact, Error.
Move the code for Home component in a file and place in the components folder.
- 2.** In each of the component files add similar code to the following. In render the name of the component in the JSX will be displayed

```
const Contact = () => {  
  return (  
    <div>  
      <p>Contact</p>  
    </div>  
  );  
};
```

- 3.** Import all of the components into the **App.js** file.

```
import Home from './components/Home';  
import About from './components/About';  
import Contact from './components/Contact';  
import Error from './components/Error';
```

- 4.** Add the new routes to the **BrowserRouter** as follows:

```

<Switch>
  <Route path="/" component={Home} exact />
  <Route path="/about" component={About} />
  <Route path="/contact" component={Contact} />
  <Route component={Error} />
</Switch>

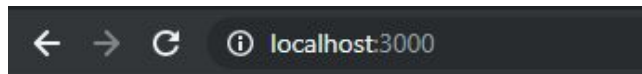
```

5. Add the new navigation links to the **JSX** unordered list to trigger the routes.

```

<ul>
  <li><Link to="/">Home</Link></li>
  <li><Link to="/contact">Contact</Link></li>
  <li><Link to="/about">About</Link></li>
</ul>

```



- [Home](#)
- [Contact](#)
- [About](#)

Home

6. Create a new **Navigation** component and move all the navigation links into the file. Import it into the **App.js** and include the component in the render in place of the `` links.

Exercise 3 – Router Parameters

1. Create a new component named **Student** and a new Link in the **Navigation** component.
2. Add a colon for a route parameter for studentname, followed by another optional route parameter for studentno

```

<Route
  path="/student/:studentname/:studentno?"
  component={Student}
/>

```

3. Update the navigation links to display and pass the following route parameters.

- [Home](#)
- [Contact](#)
- [About](#)
- [Student: Jim Smith](#)
- [Student: Jane Smith, Student No: 50001](#)

4. Add the following code in the **Student** component to output the **mandatory route parameter** for **Student name**.

```
const Student = ({ match }) => {  
  
  const { studentname } = match.params;  
  return (  
    <div>  
      <p>Student</p>  
      <div>  
        <div>`The student name is "${studentname}"!`</div>  
      </div>  
    </div>  
  );  
};
```

5. Update the **Student** component to display the **optional route parameter** for **Student no**. The output when navigating to this route should be as follows.

- [Home](#)
- [Contact](#)
- [About](#)
- [Student: Jim Smith](#)
- [Student: Jane Smith, Student No: 50001](#)

Student

The student name is "jane smith"!

The student no is 50001

Exercise 4 – Redirecting with History API

1. We need to create a **History** module and export it, so that we can use the History API in the **BrowserRouter**. To do this, we need to create a **History.js** file with the following code.

```
import createBrowserHistory from "history/createBrowserHistory";

const customHistory = createBrowserHistory();
export default customHistory;
```

2. Next, import the history module in the **App.js** and include the history object in the **BrowserRouter** to make it available to the entire application

```
<BrowserRouter history={history}>
```

3. Create a new navigation link, route and functional component named **Redirect**.
4. The **history object** will be available in the component props parameter. Add the following event handler in the **Redirect** component.

```
const handleRedirectClick = () => {
  const { history } = props;
  if (history) history.push("/");
  else console.log(`history not found in props`);
};
```

5. Add a button with a click handler to call the **handleRedirectClick** method and redirect the route to the home page.

