Web Front-End Development
Week 15: React Router

React Router
https://reactrouter.com/
Although many React apps are single-page web apps there are many cases where you'll want to include the ability to navigate to different pages or sections.
React Router provides components that let you add navigational components to your React app. These components enable us to build a single-page web app with dynamic, client-side routing. This means that navigation is achieved without the page refreshing as the user clicks on links to navigate. React Router uses a component structure to call components, which display the appropriate information.

To use React Router you need to install react-router-dom
```
npm install react-router-dom
```

React Router has three main categories of components.
https://reactrouter.com/web/guides/primary-components

Routers
The router is the core component that does all the logic of displaying various components. React Router provides two routers for web projects – BrowserRouter and HashRouter
We'll be using the BrowserRouter component which uses regular URL paths.
To use a router it must be rendered as the root of your component hierarchy. In most React apps the <App> component is the highest parent component and is rendered by index.js. With React Router the < BrowserRouter> component needs to be the top-level component so you'll need to wrap <App /> in < BrowserRouter> in index.js.

```
ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById('root')
);
```

Don't forget you'll need to import BrowserRouter
```
import {BrowserRouter} from 'react-router-dom';
```

Route Matchers
There are two components to handle route matching.
The <Switch> component is similar to a switch statement in JavaScript. It searches through its children <Route> elements to match one whose path matches the current URL. When it finds one it will render that <Route> component and all further <Route> components will be ignored. So you should put <Route> components with more specific (typically longer) paths before less-specific ones. If no <Route> matches, the <Switch> renders nothing (null).

The <Route> component connects the link with the React component that will appear on the page.

```
<Route path="/about" component={About} />
```

The path in the <Route> component matches the beginning of the URL not the whole thing. A <Route path="/"> will always match the URL so if you use that for a home page it should be the last <Route> in the <Switch>. Or use <Route exact path="/"> which does match the entire URL.

Navigation

The <Link> component lets you define a link, similar to an <a> tag but it's actually a React component. (this JSX actually is compiled to an <a href> tag in JavaScript) It will navigate to the linked component, and the URL will change, but the page won't reload so a page refresh is avoided.

```
<Link to="/about">About Us </Link>
```

The value of the "to" attribute must match the value of the "path" attribute in the <Route> tag to connect the Link and the Route. This is how the router renders the matching Route when a Link is clicked.

Example

https://repl.it/@aileenjp/react-favorites-router

I've added some navigation to my Favorites example.

In repl.it I had to go under packages and search for react-router-dom and add it (this took a few minutes).

To use React Router in your create-react-app you'll need to install react-router-dom

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

index.js

I restructured index.js so <BrowserRouter> is the parent element.

```
import {BrowserRouter} from 'react-router-dom';

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById('root')
);
```

New components

Create components for your new pages:

About.js

```
import React from 'react';
function About() {
  return (
    <div className="header">
      <h1>About</h1>
    </div>
  )
}
export default About;
```

Tips.js

```
import React from 'react';
function Tips() {
  return (
    <div className="header">
      <h1>Tips</h1>
    </div>
  )
}
export default Tips;
```

Also create a Navbar component with links to the pages.

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

function Navbar() {
  return (
    <div className="nav">
      <Link className="navItem" to="/">Favorites </Link>
      <Link className="navItem" to="/tips">Healthy Eating
Tips</Link>
      <Link className="navItem" to="/about">About Us </Link>
    </div>
  )
}

export default Navbar;
```

Note the import and export statements in each component.
I also had to reorganize and add some CSS so the nav didn't look terrible.

App.js
Now we update App.js as our main app container.
Import all the new components as well as the Route and Switch components from react-router-dom.
We still need to return one element which remains our <div>.

```
import React from 'react';
import {Route, Switch } from 'react-router-dom';
import './App.css';
import FavoriteList from './FavoriteList'
import About from './About'
import Tips from './Tips'
import Navbar from './Navbar'

function App() {
  return (
    <div>
      <Navbar />
      <Switch>
        <Route path="/" component={FavoriteList} exact />
        <Route path="/about" component={About} />
        <Route path="/tips" component={Tips} />
      </Switch>
    </div>
  )
}

export default App;
```

Open the repl in a separate tab and when you navigate you'll notice there's no page refresh. React Router only handles client-side routing as React is a client-side framework. To handle server side reloads and routing you'd need something like next.js which handles server-side rendering and more.