This tutorial continues React Tutorial #1.

In React, every application is a 'single page' one. That is why React is so efficient, there is not much content in the application itself in terms of multiple .html files to be served to the client. However, single page web sites can only go so far, and we do have to provide a way of differentiating content based the traditional notion of using pages, albeit virtual ones in the case of React.

What is Page Routing? This is being able to navigate to different pages of an application, demonstrated by the use of a navigation bar containing menu options, like the options Home, About and Contact. Page routing would support navigation to a Home page, an About page and a Contact page. React App does not support page routing, so in order to implement routing a very popular solution (amongst others!) is to use the code base from React Router – this is what we do here. React Router supports the addition of multiple routes through the application, like a traditional set of pages to navigate to the home page, an about page and a contact page.
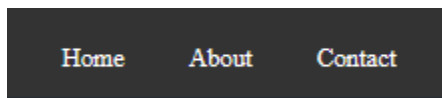
Before we start, **create a GitHub repository for your existing React example**. Open your existing code base, and let's add Routes for Home, About and Contact.

In this tutorial, you will

- Create a set of 3 routes using React, that enable 'navigation' to other 'panels'. The panels are virtual pages, because they are not stored as markup in traditional .html files
- Wire the routes to React code that handles each route and render the expected content

1) Installing the Base Router Code

The first step is to **pull in the code that supports routes in React**.

:s\meeting_alerts_routes> npm install react-router-dom

In the **src** folder, create a new file, **index.js**.

We will do a little manipulation, and add the **App component** here, and rearrange files and functions to represent Home, About and Contact, but store these in pages folder, as they are the pages of the web site.

In index.js, in the src folder,
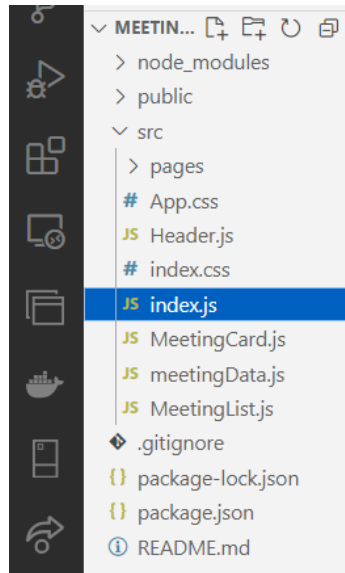
... you will need to import the base code necessary for adding routes, by importing components from the react-router-dom package.

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

What does the import accomplish?

This import gets specific components from the react-router-dom library, which are needed to handle routing in React applications. We are pulling in code from 3 defined react components – BrowserRouter, Routes and Route.

BrowserRouter is a router implementation that keeps the UI content in sync with the URL, so that the correct content appears for each route. Remember, React is effectively a single page implementation, with content added to the root element as opposed to loading a new .html file. React keeps a virtual copy of what you see on the DOM, and the visible DOM must be kept in sync with the virtual React DOM, as this is where the 'web page' content comes from.

The **Routes** component defines a set of routes for the application, making it possible to define different content in separate .js (or .jsx) files, and load the content from these files, into the UI. Think of the Routes component as a List element, ready to store List Items.

The **Route** component renders a UI component when the URL matches the path that has been defined for that route. Think of this as a List Item in a List element.

This is pretty sophisticated in terms of tooling to support the loading of new content, but the efficiency of React is the absence of multitudinous .html files, all of which take up storage space and additional time to transfer over the wires of the internet.

Next, let's add the 3 routes for Home, About and Contact. This is done within the App component, below.

2) Adding The Routes

We **add the following to the index.js file**, which is invoked when we start the application. For this step, all modifications are to the index.js file.

```
export default function App() {
  return ( ···
  );
}


const root = ReactDOM.createRoot(document.querySelector("#root"));
root.render(<App />);
```

You can give the root component's first child any title you want, however it is typical to see an App component.

Let's create those routes!

Adding routes does not make sense without also adding a navigation bar to navigate to the new content provided by the routes. This navigation bar should be accessible from all of the routes, it should be visible on the Home, About and Contact 'pages'. When you have all the components in place and wired together, you will see that the same menu bar, with the same content and links, appears on each of the virtual pages that represent Home, About and Contact.
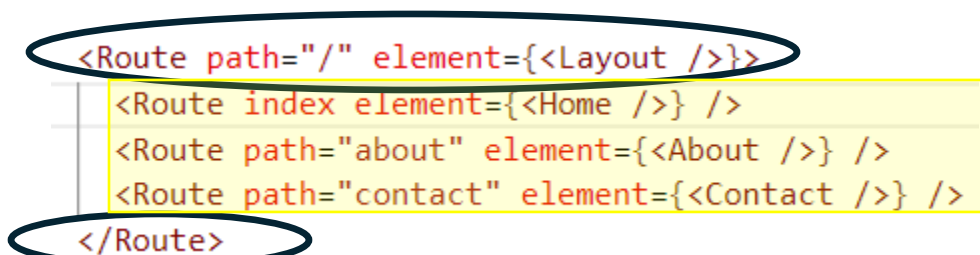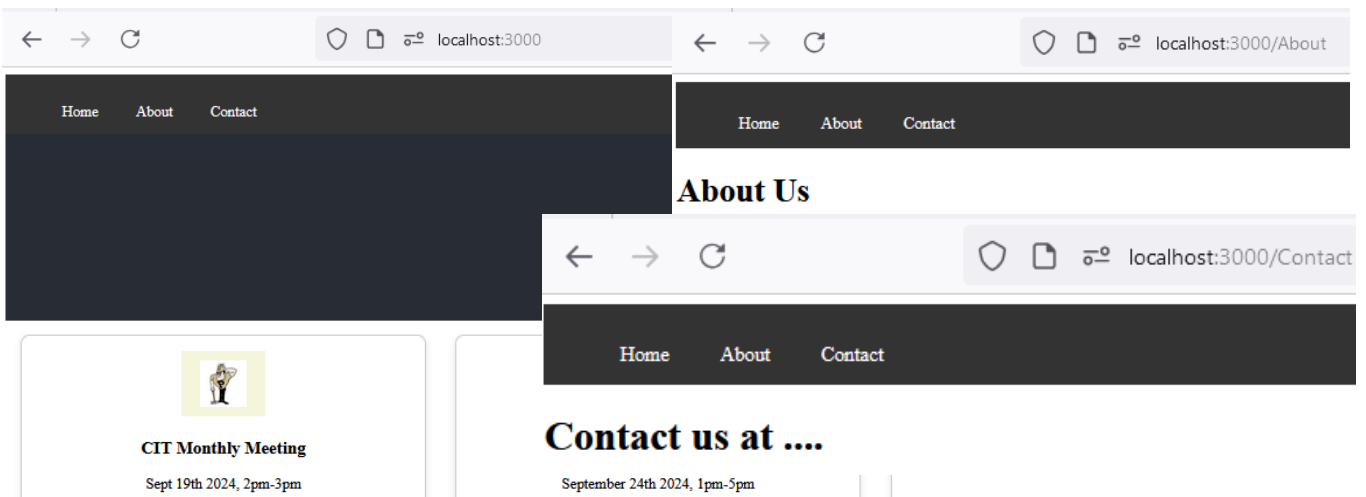
React Tutorial #2 – Routes

You will return the following from the App component

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Layout />}>
      <Route index element={<Home />} />
      <Route path="about" element={<About />} />
      <Route path="contact" element={<Contact />} />
    </Route>
  </Routes>
</BrowserRouter>
```

Now you can see the references to the imported content, **BrowserRouter**, **Routes** and **Route**. React uses the BrowserRouter component to contain and control the routes and keep the *visible* React DOM, which is the user interface, synchronized with the React *virtual* DOM.

The Layout component contains the navigation bar, and because this should appear on all of the routes when they are rendered, the layout component surrounds the 3 routes. You could say that the 3 routes, Home, About, Contact are *nested* within the Layout component.

```
<Route path="/" element={<Layout />}>
  <Route index element={<Home />} />
  <Route path="about" element={<About />} />
  <Route path="contact" element={<Contact />} />
</Route>
```

The navigation pathways are defined for the Home, About and Contact routes, with the Home route being identified as the effective home page because it has the index attribute, instead of a path specification.

```
export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="about" element={<About />} />
          <Route path="contact" element={<Contact />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

We must define content for the Home, About and Content, as well as the Layout, and import them into index.js in order for these components to be recognized when they are referenced. Let's add the imports for all of these (to the index.js file). We will be storing them in the src/pages folder, to be created in the next step. For now, these are placeholders.

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import About from "./pages/About";
import Contact from "./pages/Contact";
import "./App.css";
```

We will include App.css here too as it is now the top level source.

Let's see this index.js file in context.

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import About from "./pages/About";
import Contact from "./pages/Contact";
import "./App.css";


export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="about" element={<About />} />
          <Route path="contact" element={<Contact />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

const root = ReactDOM.createRoot(document.querySelector("#root"));
root.render(<App />);
```
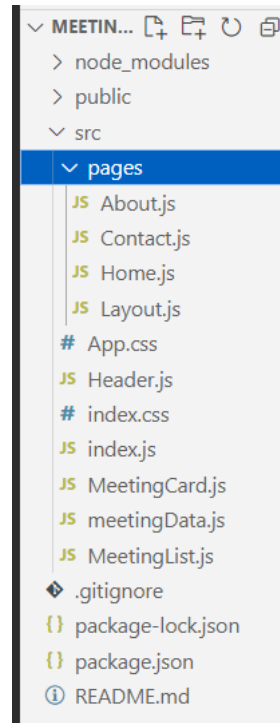
We will be adding the component content for Layout, Home, About and Contact, in the next steps.

3) The Layout Component, A Consistent Menu Bar

As mentioned, the Layout component is one that provides shared content – in this case, the navigation bar.

Let's create a folder, **pages**, inside the **src** folder. This **pages** folder will store content related to the **virtual pages** that represent the 3 pages in the web suite, Home, About and Contact.

At this point let's focus on the Layout component, in Layout.js.

```
∨ MEETIN...  ⌷₊ ⌷₊ ↻ ⊟
  > node_modules
  > public
  ∨ src
    ∨ pages
      JS About.js
      JS Contact.js
      JS Home.js
      JS Layout.js
    # App.css
    JS Header.js
    # index.css
    JS index.js
    JS MeetingCard.js
    JS meetingData.js
    JS MeetingList.js
  ◈ .gitignore
  {} package-lock.json
  {} package.json
  ⓘ README.md
```

The menu bar should link to other content, and be able to do the same thing that the anchor tag (<a href="....">) does. The support for this comes from 2 additional elements that translate to an anchor block, and the anchor block's ability to switch between pages or locations. These 2 elements are the Outlet element and the Link element. These should be imported from react-router-dom.

```
import { Outlet, Link } from "react-router-dom";
```

The Layout component is set it up to return JSX.

```
const Layout = () => {

    return (···
    );
};
```

```
6 export default Layout;
```

```
return (
  <>
    <nav>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/About">About</Link>
          </li>
          <li>
            <Link to="/Contact">Contact</Link>
          </li>
        </ul>
    </nav>
    <Outlet />
  </>
);
```

The JSX is composed of Link and Outlet elements, and uses core elements – the nav tag, a list and list items.

The <Outlet/> component renders the current route selected.

<Link> is used to set the current URL and keep track of browsing history, and effectively links to an internal path. <Link> replaces the anochor block, <a href="">.
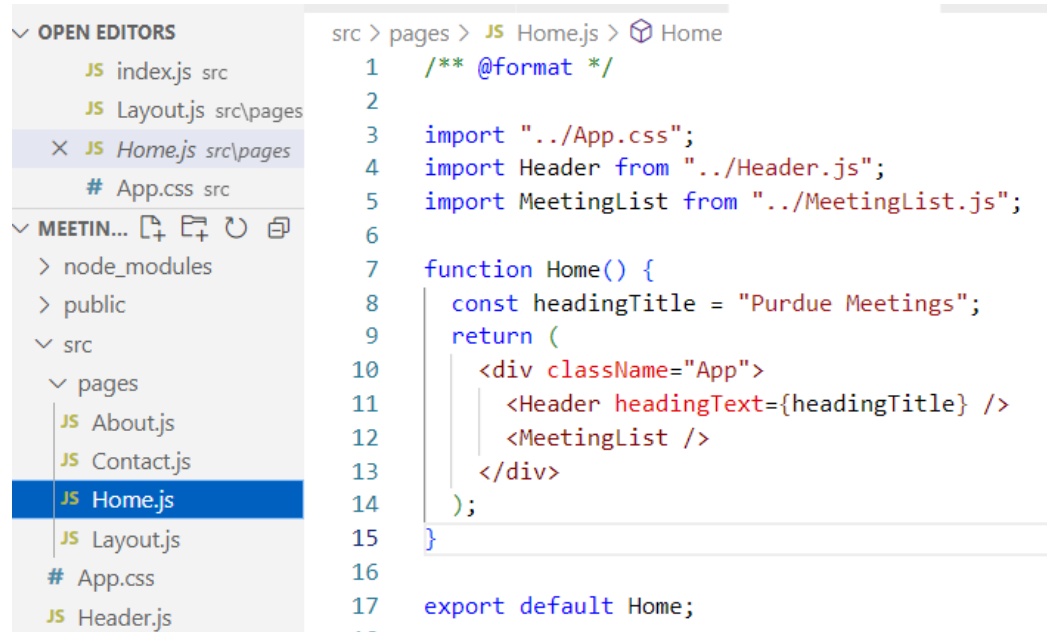
Here is the full Layout component in context.

```
import { Outlet, Link } from "react-router-dom";


const Layout = () => {

  return (
    <>
      <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/About">About</Link>
            </li>
            <li>
              <Link to="/Contact">Contact</Link>
            </li>
          </ul>
      </nav>
      <Outlet />
    </>
  );
};

export default Layout;
```

4) Home, About, Contact Components

For the Home component, let's copy and the existing content from the old App.js, and place it in the Home.js file.
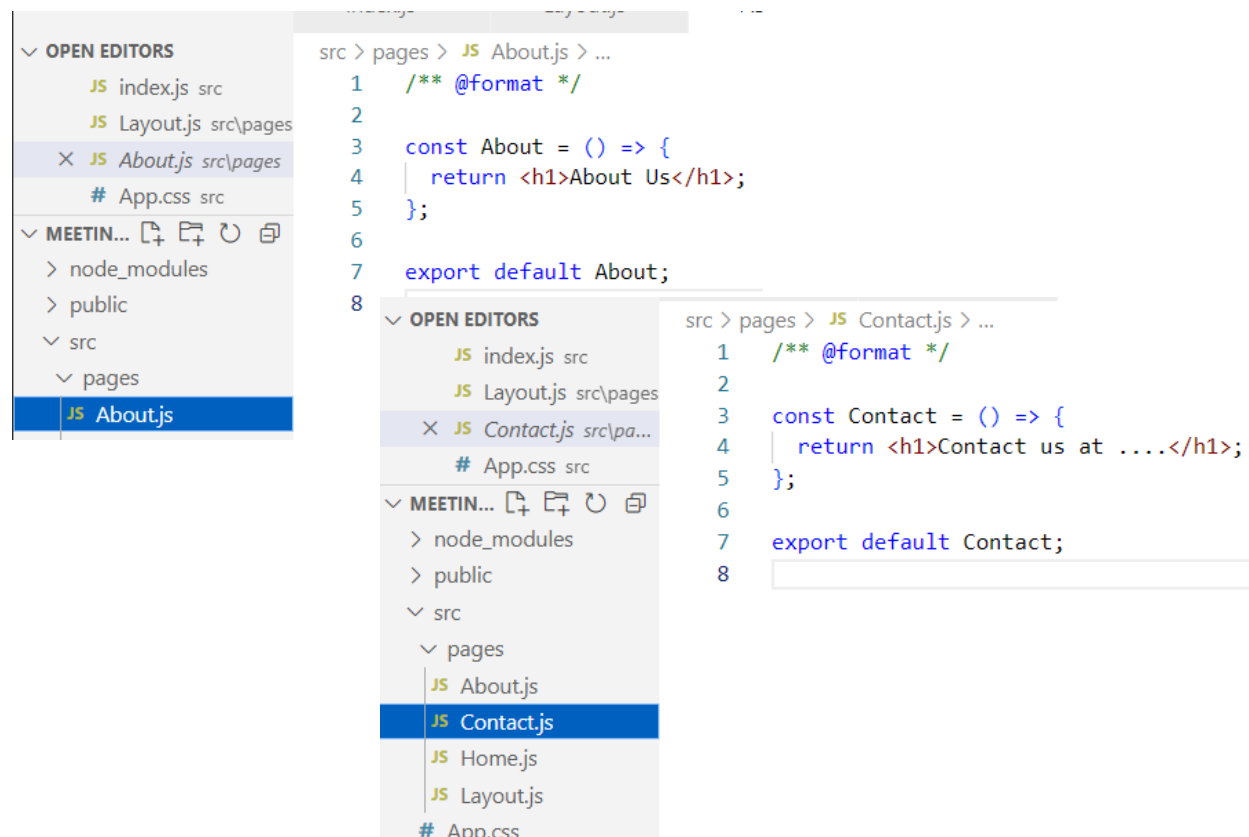
```
src > pages > JS Home.js > ⊙ Home
1   /** @format */
2
3   import "../App.css";
4   import Header from "../Header.js";
5   import MeetingList from "../MeetingList.js";
6
7   function Home() {
8     const headingTitle = "Purdue Meetings";
9     return (
10      <div className="App">
11        <Header headingText={headingTitle} />
12        <MeetingList />
13      </div>
14    );
15  }
16
17  export default Home;
```

We can add pseudo content for the other 2 pages.

```
src > pages > JS About.js > ...
1   /** @format */
2
3   const About = () => {
4     return <h1>About Us</h1>;
5   };
6
7   export default About;
8
```

```
src > pages > JS Contact.js > ...
1   /** @format */
2
3   const Contact = () => {
4     return <h1>Contact us at ....</h1>;
5   };
6
7   export default Contact;
8
```

Make sure to include the menu bar styling! Add this to App.css, in the src folder.

```
nav {
 background-color: #333;
 overflow: hidden;
}

nav a {
 float: left;
 display: block;
 color: white;
 text-align: center;
 padding: 14px 20px;
 text-decoration: none;
}

nav ul {
 list-style-type: none;
}
```

This is all you need for a base react application with Routes, navigable links to different content enabled in the traditional way, from links on a navigation bar.