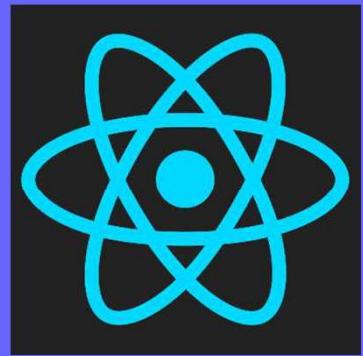


1

React Router

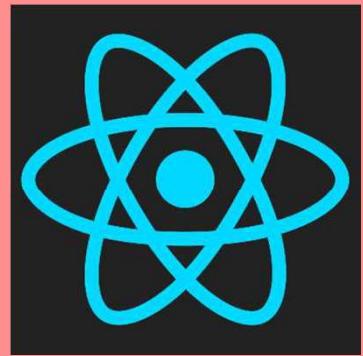


Introduction to React Routing

- It is a fully featured client and server side routing library for react, the library helps create and navigate between different urls that make up your web application.
- It also provides unique urls for different components in the app and makes the ui easily shareable with other users
- If you're building a medium too large scale react app react router is a must have package

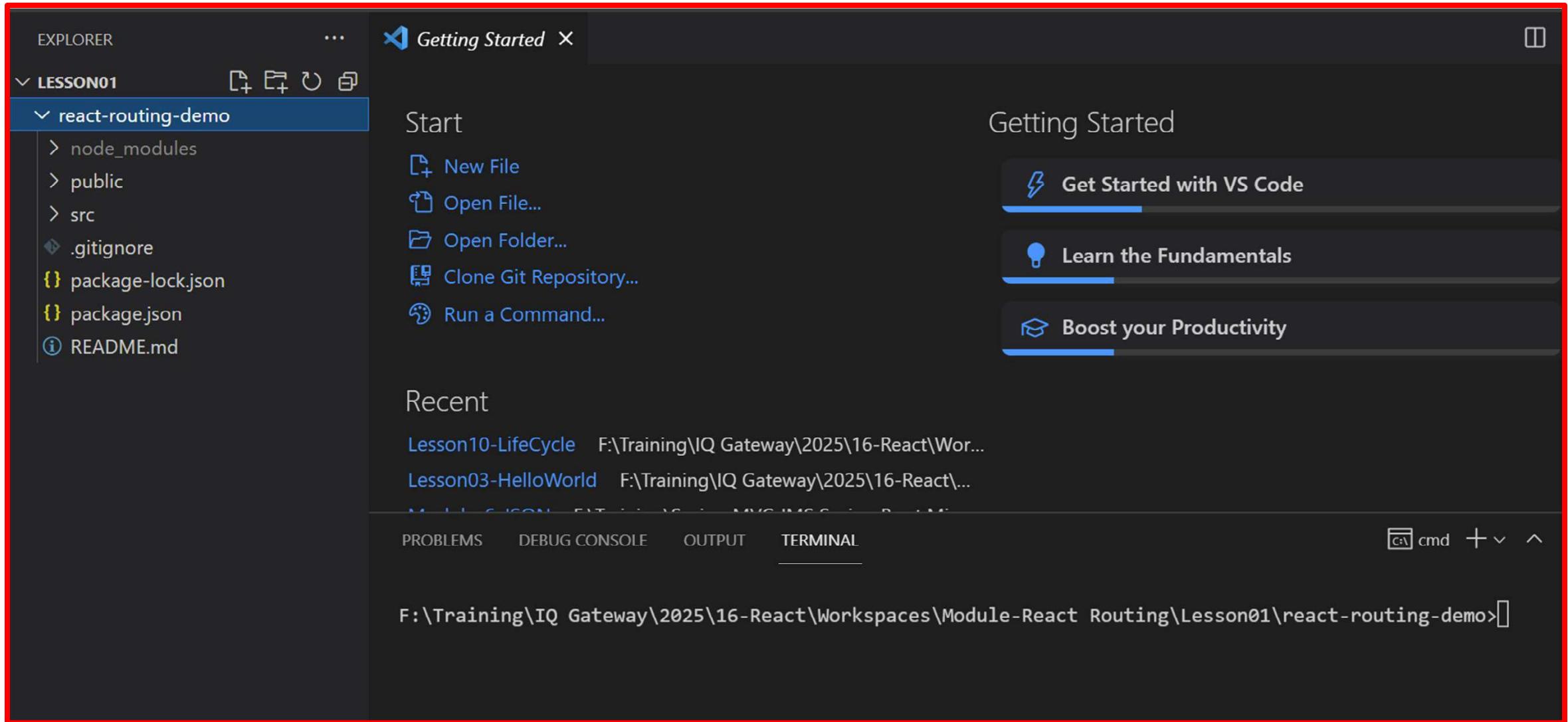
Focus on

1. Configuring Routes
2. Navigating on Button Click
3. Navigating Programmatically
4. Dynamic Routes
5. Nested Routes
6. Route Parameters



Installation of React Routing

Step1 : Create React App



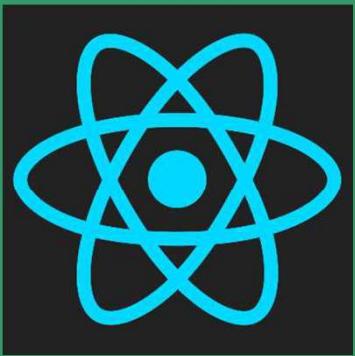
Step 2 : Install React Router Package

The screenshot shows the VS Code interface with the title bar "Getting Started X". The left sidebar displays a project structure for "react-routing-demo" under "LESSON01", including files like "node_modules", "public", "src", ".gitignore", "package-lock.json", "package.json", and "README.md". The main area has a "Start" menu with options: New File, Open File..., Open Folder..., Clone Git Repository..., and Run a Command... . Below it is a "Recent" section listing "Lesson10-LifeCycle" and "Lesson03-HelloWorld". The bottom navigation bar includes tabs for PROBLEMS, DEBUG CONSOLE, OUTPUT, and TERMINAL. A red box highlights the terminal tab, which contains the command: `F:\Training\IQ Gateway\2025\16-React\Workspaces\Module-React Routing\Lesson01\react-routing-demo >npm install react-router-dom@6`. The status bar at the bottom right shows "cmd + ^ x". On the right side, there's a "Getting Started" sidebar with three sections: "Get Started with VS Code" (underlined), "Learn the Fundamentals", and "Boost your Productivity".

{ } package.json M X

react-routing-demo > { } package.json > ...

```
1  {
2    "name": "react-routing-demo",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/dom": "^10.4.0",
7      "@testing-library/jest-dom": "^6.6.3",
8      "@testing-library/react": "^16.3.0",
9      "@testing-library/user-event": "^13.5.0",
10     "react": "^19.1.0",
11     "react-dom": "^19.1.0",
12     "react-router-dom": "^6.30.1", ^6.30.1
13     "react-scripts": "5.0.1",
14     "web-vitals": "^2.1.4"
15   },
```

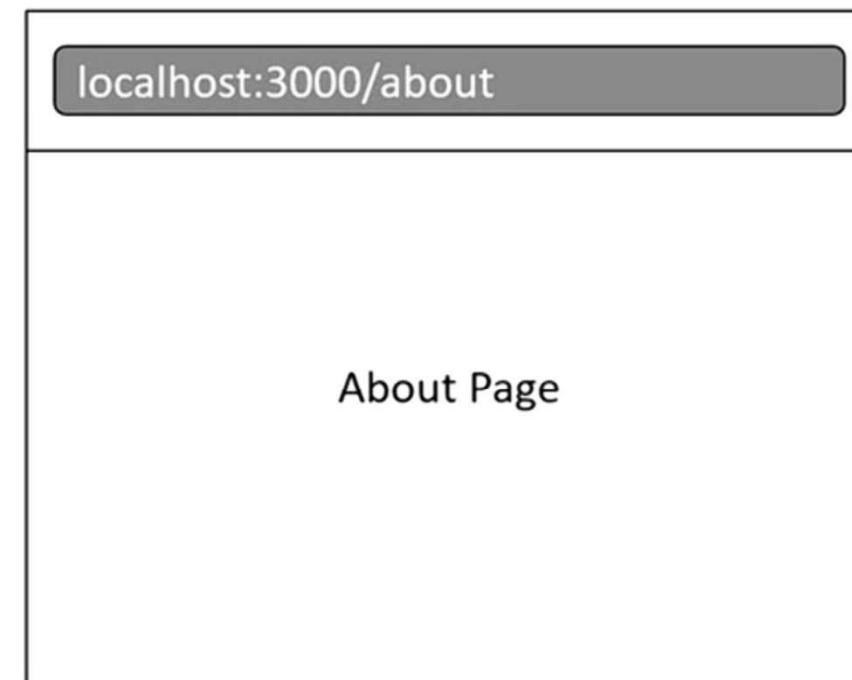
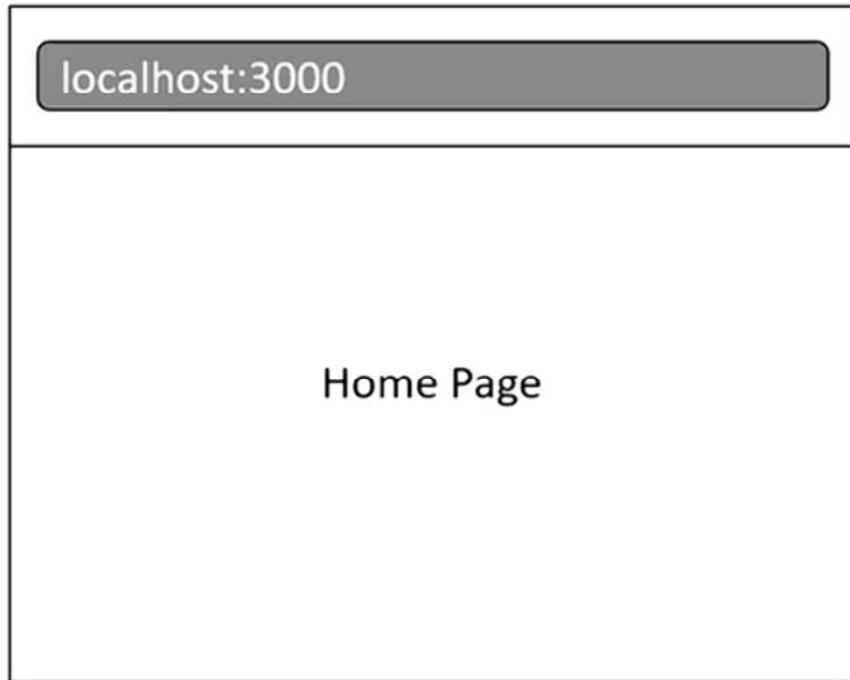


Configuring Routes

Configuring Routes

Scenario : We're going to set up two routes the first route is the home route

- If the user navigates to **localhost:3000** they should be able to see the **home page**
- If the visit **localhost:3000/about** they should be able to see the **about page**



We render the app component to the DOM

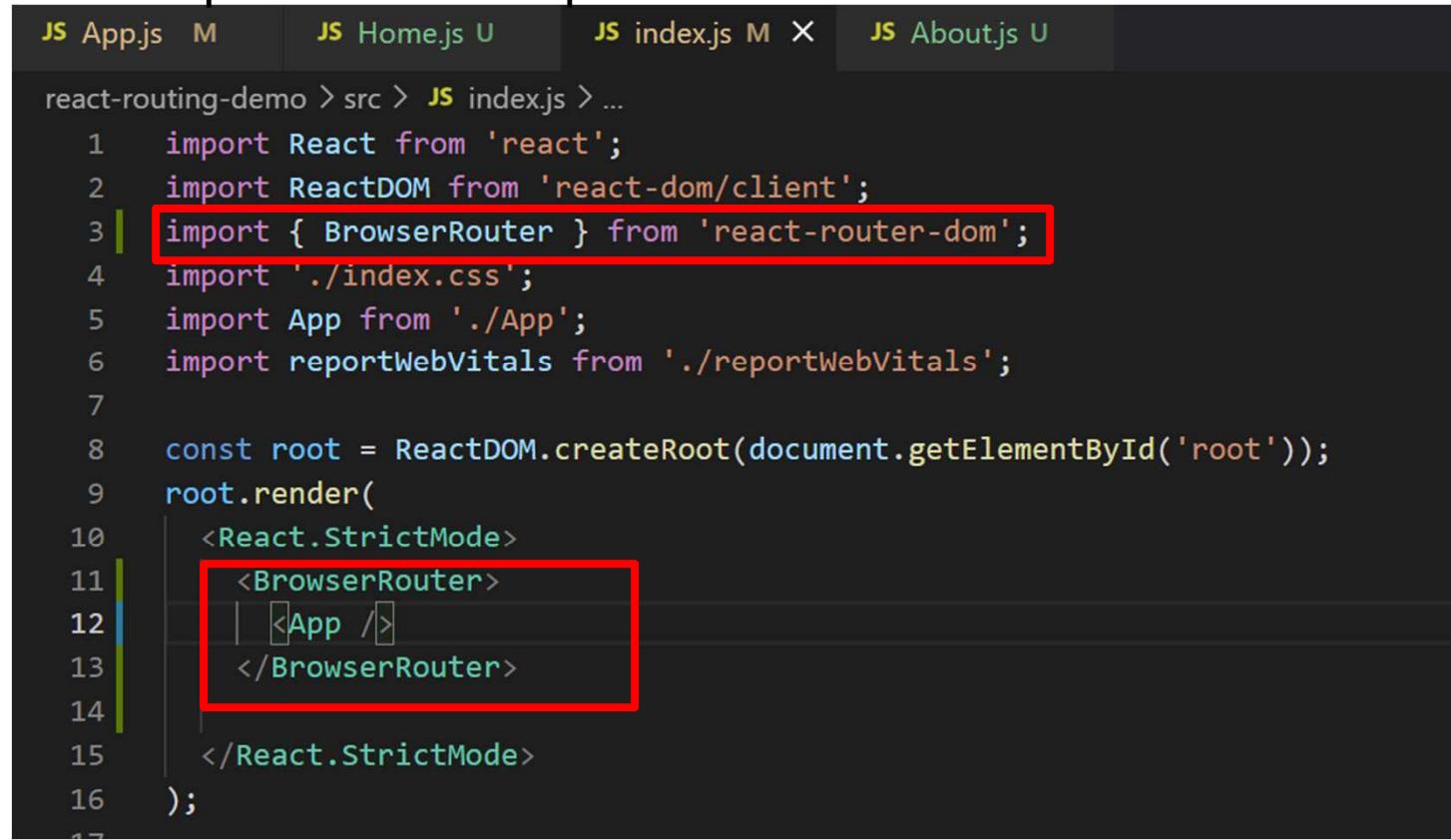
```
JS index.js X  
react-routing-demo > src > JS index.js > ...  
1 import React from 'react';  
2 import ReactDOM from 'react-dom/client';  
3 import './index.css';  
4 import App from './App';  
5 import reportWebVitals from './reportWebVitals';  
6  
7 const root = ReactDOM.createRoot(document.getElementById('root'));  
8 root.render(  
9   <React.StrictMode>  
10    <App />  
11   </React.StrictMode>  
12 );  
13  
14 // If you want to start measuring performance in your app, pass a function  
15 // to log results (for example: reportWebVitals(console.log))  
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
17 reportWebVitals();  
18
```

App Component Renders

```
JS App.js      X  
react-routing-demo > src > JS App.js > ...  
1 import logo from './logo.svg';  
2 import './App.css';  
3  
4 function App() {  
5   return (  
6     <div className="App">  
7       <header className="App-header">  
8         <img src={logo} className="App-logo" alt="logo" />  
9         <p>  
10          | Edit <code>src/App.js</code> and save to reload.  
11        </p>  
12        <a  
13          | className="App-link"  
14          | href="https://reactjs.org"  
15          | target="_blank"  
16          | rel="noopener noreferrer"  
17          |>  
18          | Learn React  
19        </a>  
20      </header>  
21    </div>  
22  );  
23}  
24  
25 export default App;
```

Configuring

- The first step to configuring routes with react router is to connect the url in the browser with our react application
- For that react router provides a component called **BrowserRouter**



```
JS App.js M JS Home.js U JS index.js M X JS About.js U

react-routing-demo > src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import { BrowserRouter } from 'react-router-dom';
4  import './index.css';
5  import App from './App';
6  import reportWebVitals from './reportWebVitals';
7
8  const root = ReactDOM.createRoot(document.getElementById('root'));
9  root.render(
10    <React.StrictMode>
11      <BrowserRouter>
12        <App />
13      </BrowserRouter>
14    </React.StrictMode>
15  );
16
```

A screenshot of a code editor showing the file index.js. The file contains code for a React application. A red box highlights the import statement for the BrowserRouter component and the component itself within the render function. The code is as follows:

```
JS App.js M JS Home.js U JS index.js M X JS About.js U

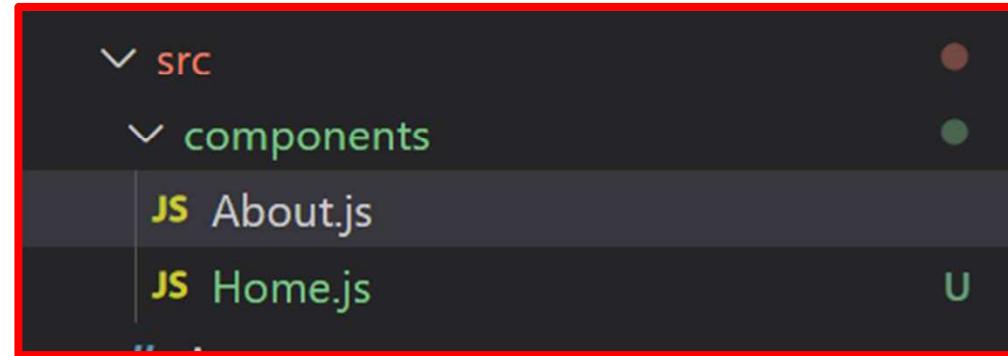
react-routing-demo > src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import { BrowserRouter } from 'react-router-dom';
4  import './index.css';
5  import App from './App';
6  import reportWebVitals from './reportWebVitals';
7
8  const root = ReactDOM.createRoot(document.getElementById('root'));
9  root.render(
10    <React.StrictMode>
11      <BrowserRouter>
12        <App />
13      </BrowserRouter>
14    </React.StrictMode>
15  );
16
```

- This allows us to do is use all the features from react router within the app component tree

Let's Configure the routes in App.js

JS App.js 1, M X

```
react-routing-demo > src > JS App.js > ...
1  function App() {
2    return (
3      ...
4    );
5  }
6
7
8  export default App;
9
```



JS App.js 1, M X JS Home.js U X JS About.js U

```
react-routing-demo > src > components > JS Home.js > ...
1  import React from 'react'
2
3  export const Home = () => {
4    return (
5      ...
6      Home Page
7    );
8  }
9
10
```

JS App.js 1, M JS Home.js U JS About.js U X

```
react-routing-demo > src > components > JS About.js > [?] About
1  import React from 'react'
2
3  export const About = () => {
4    return (
5      ...
6      About Page
7    );
8
9 }
```

- For the route configuration we need two components from react router

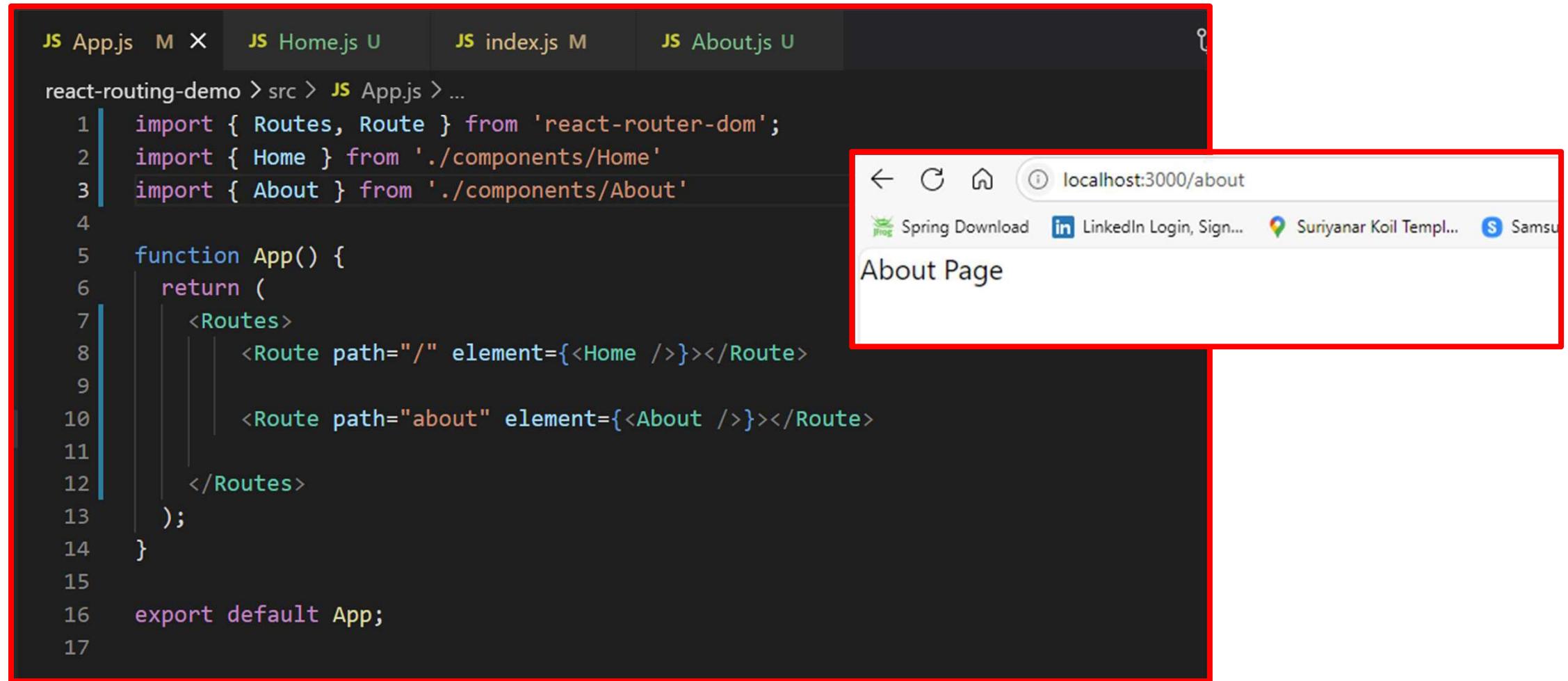
The image displays three windows illustrating the setup of a React Router application:

- Code Editor (Top):** Shows the `App.js` file with the following code:

```
JS App.js > ...
| import { Routes, Route } from 'react-router-dom'
```
- Code Editor (Bottom Left):** Shows the `App.js` file with the following code:

```
JS App.js M X JS Home.js U JS About.js U
react-routing-demo > src > JS App.js > App
1 import { Routes, Route } from 'react-router-dom';
2 import { Home } from './components/Home'
3
4 function App() {
5   return (
6     <Routes>
7       <Route path="/" element={<Home />}></Route>
8     </Routes>
9   );
10 }
11
12 export default App;
```
- Browser (Bottom Right):** Shows a browser window at `localhost:3000` displaying the "Home Page". The address bar shows `localhost:3000`. The page content is "Home Page".

Second Route



The image shows a code editor and a web browser side-by-side, both with red borders.

Code Editor (Left):

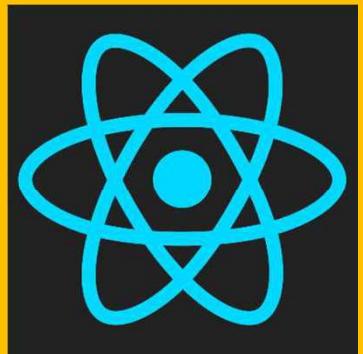
```
react-routing-demo > src > JS App.js > ...
1 import { Routes, Route } from 'react-router-dom';
2 import { Home } from './components/Home'
3 import { About } from './components/About'
4
5 function App() {
6   return (
7     <Routes>
8       <Route path="/" element={<Home />}></Route>
9
10      <Route path="about" element={<About />}></Route>
11
12    </Routes>
13  );
14}
15
16 export default App;
17
```

Browser (Right):

localhost:3000/about

About Page

- This is pretty much how you configure Routes with React Router

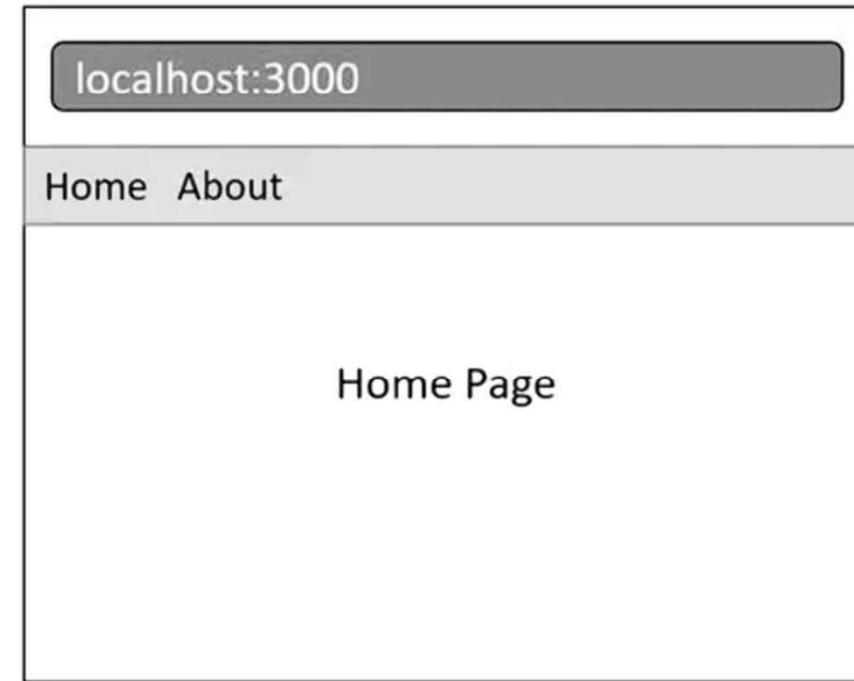
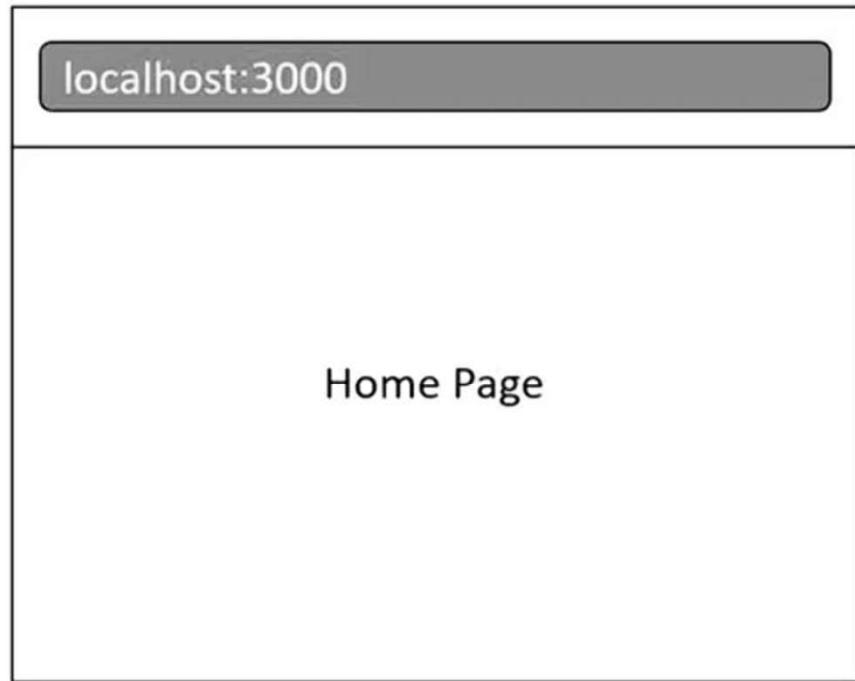


Links

Links

Scenario : Let's add a Navbar in the app component with two links

- One to the **Home page** and One to the **About page**
- Click of these links we should be able to navigate between the two route



```
JS App.js M JS NavBar.js U X JS Home.js U JS index.js M JS All  
react-routing-demo > src > components > JS NavBar.js > [?] NavBar  
1 import React from 'react'  
2 import { Link } from 'react-router-dom' ←  
3  
4 export const NavBar = () => {  
5   return (  
6     <nav>  
7       <Link to="/">Home</Link>  
8       <Link to="/about">About</Link>  
9     </nav>  
10    )  
11  }  
12
```

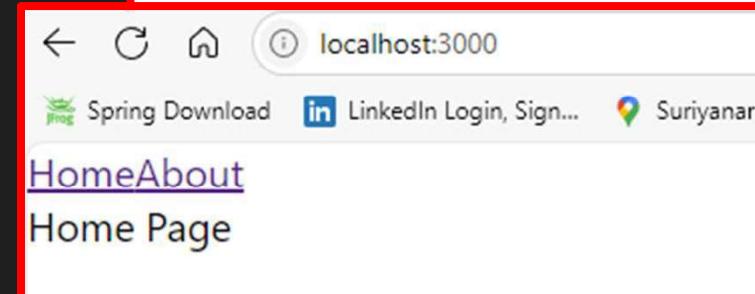
Clickable elements to navigate between the routes to navigate to another route

React Router provides us with the **Link component**

```
JS App.js M X JS NavBar.js U JS Home.js U JS index.js M JS About.js U  
react-routing-demo > src > JS App.js > ...  
1 import { Routes, Route } from 'react-router-dom';  
2 import { Home } from './components/Home'  
3 import { About } from './components/About'  
4 import { NavBar } from './components/NavBar'  
5 function App() {  
6   return (  
7     <>  
8       <NavBar />  
9       <Routes>  
10      <Route path="/" element={<Home />}></Route>  
11  
12      <Route path="about" element={<About />}></Route>  
13    </Routes>  
14  </>  
15);  
16}  
17  
18  
19 export default App;  
20
```

Fragments [Place Where Components are Rendered in SPA]

We Have Included NavBar Component

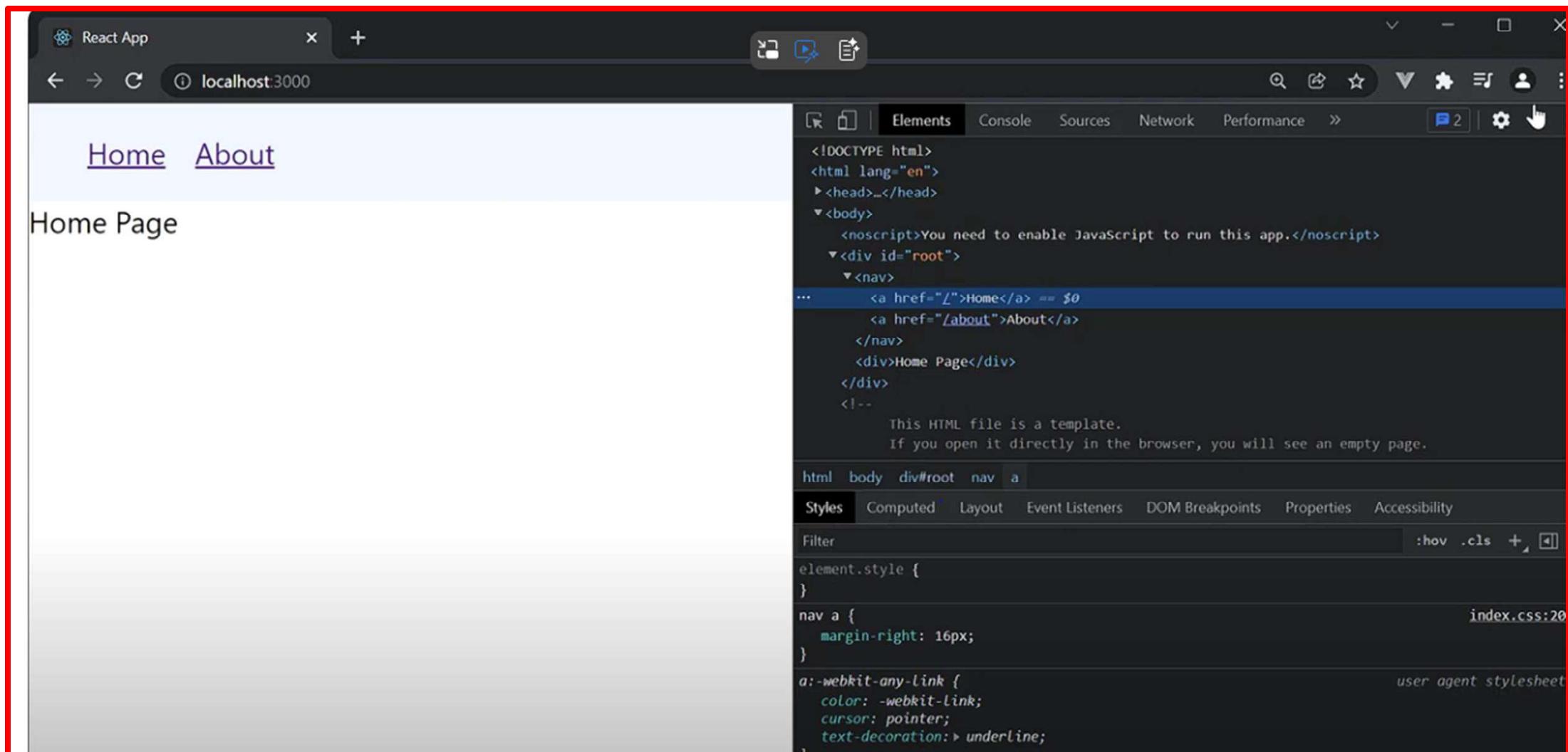


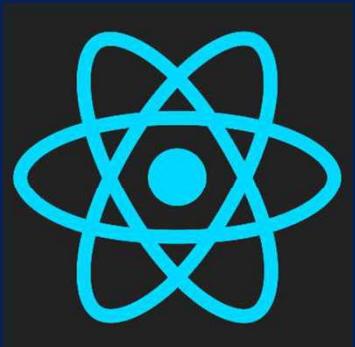
Let's Add Some Styling to Make it Presentable

The screenshot shows a code editor with several tabs open, including App.js, index.css, NavBar.js, Home.js, index.js, About, and a few others. The index.css tab is active, displaying CSS code for styling the application. A red box highlights the styling rules for the 'nav' element and its children 'a' elements.

```
react-routing-demo > src > # index.css > nav a
1 body {
2   margin: 0;
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
4   |   'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
5   |   sans-serif;
6   -webkit-font-smoothing: antialiased;
7   -moz-osx-font-smoothing: grayscale;
8 }
9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
12   |   monospace;
13 }
14
15 nav {
16   background-color: #aliceblue;
17   padding: 16px 32px;
18 }
19
20 nav a {
21   margin-right: 16px;
22 }
```

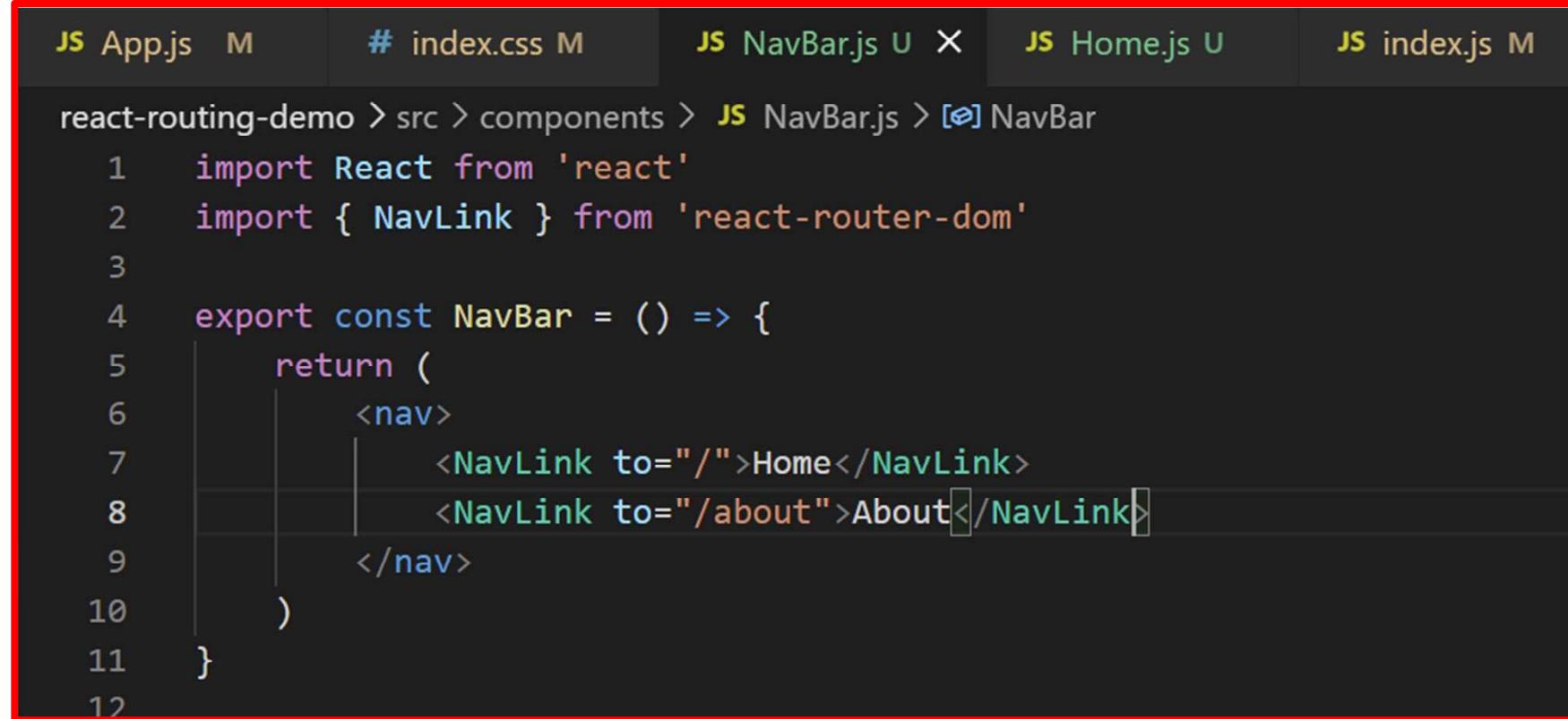
To the right of the code editor is a browser window showing the application running at localhost:3000. The browser displays a navigation bar with 'Home' and 'About' links, and the main content area displays the text 'Home Page'. The browser window is also highlighted with a red box.





Active Links

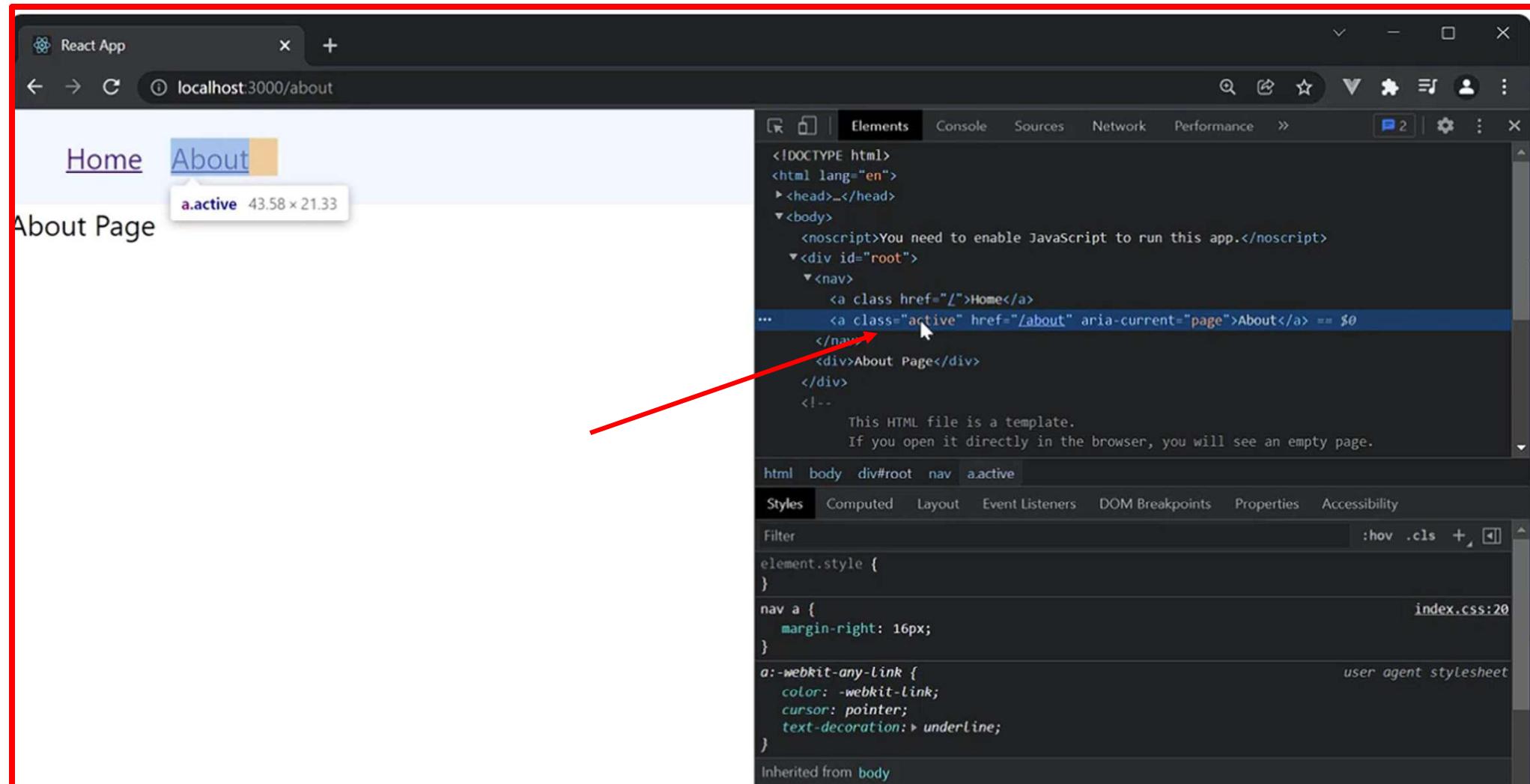
- In practical web applications it is common to style the link corresponding to the active route in a different way this also benefits the user from a user experience point of view
- **React Router** provides another component called **Nav Link** which knows whether or not it is the active link we can use that component to style the **Active Link in our NavBar**



```
JS App.js M      # index.css M      JS NavBar.js U X      JS Home.js U      JS index.js M

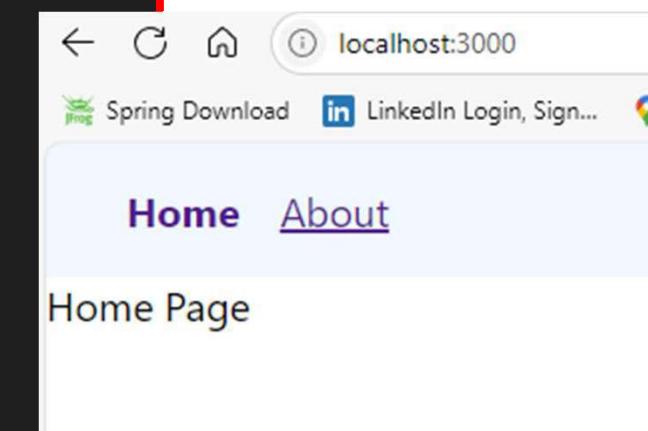
react-routing-demo > src > components > JS NavBar.js > [e] NavBar
1 import React from 'react'
2 import { NavLink } from 'react-router-dom'
3
4 export const NavBar = () => {
5   return (
6     <nav>
7       <NavLink to="/">Home</NavLink>
8       <NavLink to="/about">About</NavLink>
9     </nav>
10  )
11}
12
```

- What is special about this **NavLink Component** though is that by default it receives an active class when the link is the current route.



In index.css

```
14  
15  nav {  
16    background-color: #aliceblue;  
17    padding: 16px 32px;  
18  }  
19  
20  nav a {  
21    margin-right: 16px;  
22  }  
23  
24  nav a.active {  
25    text-decoration: none;  
26    font-weight: bold;  
27  }
```



Active Link [Css in JS] with Style prop

The image shows a code editor and a browser window side-by-side, both with red boxes highlighting specific sections.

Code Editor (NavBar.js):

```
react-routing-demo > src > components > NavBar.js > ...
1 import React from 'react'
2 import { NavLink } from 'react-router-dom'
3
4 export const NavBar = () => {
5
6     const navLinkStyles = ({ isActive }) => {
7         return {
8             fontWeight: isActive ? 'bold' : 'normal',
9             textDecoration: isActive ? 'none': 'underline'
10        }
11    }
12
13    return (
14        <nav>
15            <NavLink style={navLinkStyles} to="/">Home</NavLink>
16            <NavLink style={navLinkStyles} to="/about">About</NavLink>
17        </nav>
18    )
19}
20
```

Code Editor (index.css):

```
23
24 /* nav a.active {
25 |   text-decoration: none;
26 |   font-weight: bold;
27 | }
```

Browser Screenshot:

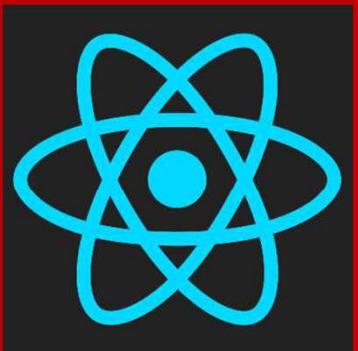
The browser shows a navigation bar with two links: "Home" and "About". The "Home" link is bold and underlined, indicating it is the active page. The "About" link is regular text.

localhost:3000

Spring Download LinkedIn Login, Sign... Google

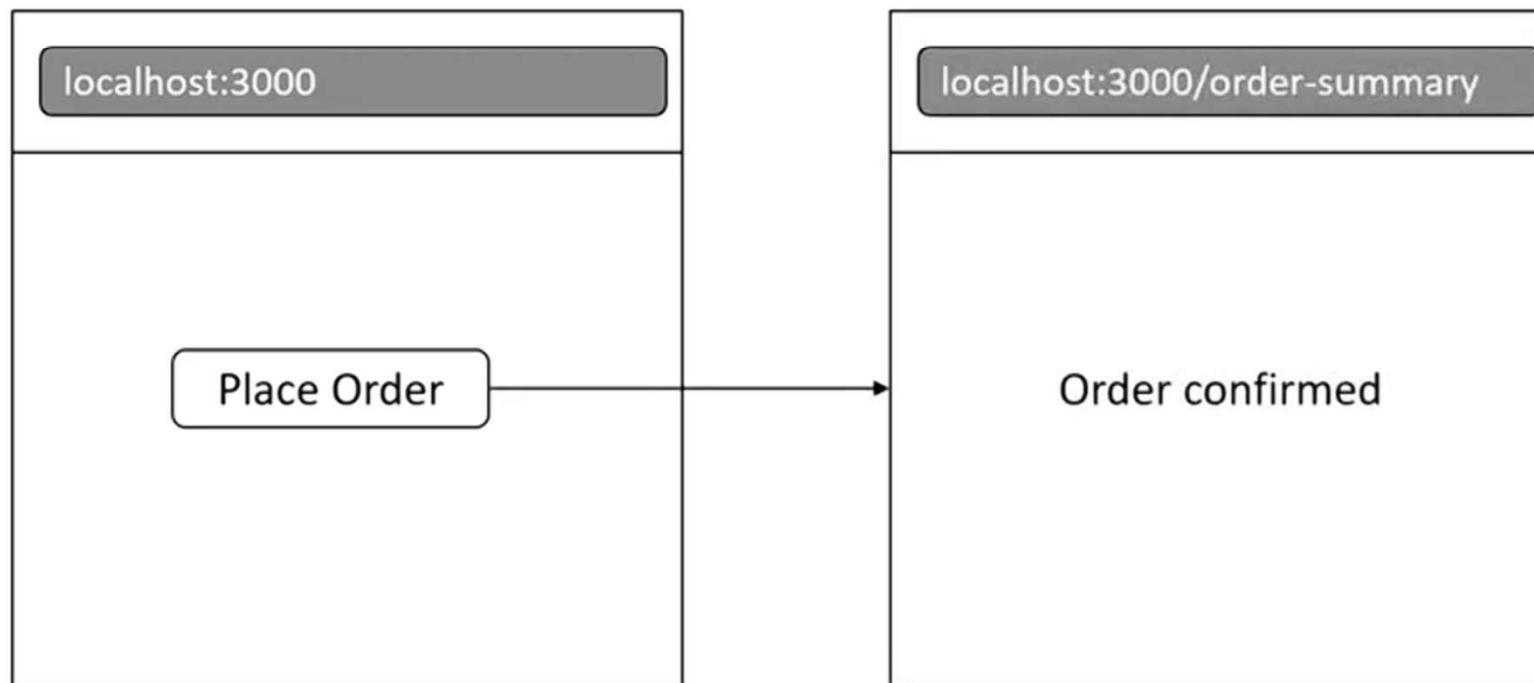
Home About

Home Page



Navigating Programmatically

- Till now we learned how to navigate to different routes using the link and NavLink components for the most part it is pretty much all you need however at times you might need to programmatically navigate to a particular route



- Let's Implement the scenario, by a button click handler where we programmatically navigate to a different route in our application.
- On click of Order button in the home page let's navigate to the order summary page

The screenshot shows a dark-themed interface of VS Code. On the left is the Explorer sidebar with a red border, displaying the project structure:

- LESSON01
- react-routing-demo
 - node_modules
 - public
- src
 - components
 - About.js
 - Home.js
 - NavBar.js
 - OrderSummary.js
- # App.css

On the right, the main editor area shows the content of OrderSummary.js:

```
JS OrderSummary.js U X JS App.js M
react-routing-demo > src > components > JS OrderSummary.js >
1 import React from 'react'
2
3 export const OrderSummary = () => {
4   return (
5     <div>
6       | Order Confirmed
7     </div>
8   )
9 }
10 }
```

The screenshot shows a development environment with the following details:

- EXPLORER:** Shows the project structure under "LESSON01".
 - react-routing-demo**: Contains "node_modules", "public", and "src".
 - src**: Contains "components" (with "About.js", "Home.js", "NavBar.js", "OrderSummary.js"), "# App.css", "App.js" (selected), "App.test.js", "# index.css", "index.js", "logo.svg", "reportWebVitals.js", "setupTests.js", ".gitignore", "package-lock.json", and "package.json".
- FILES:** Shows files with their status (U for untracked, M for modified). The current file is "App.js".
- CODE:** The content of "App.js" is as follows:

```
1 import { Routes, Route } from 'react-router-dom';
2 import { Home } from './components/Home'
3 import { About } from './components/About'
4 import { NavBar } from './components/NavBar'
5 import { OrderSummary } from './components/OrderSummary'
6 function App() {
7   return (
8     <>
9     <NavBar />
10    <Routes>
11      <Route path="/" element={<Home />}></Route>
12
13      <Route path="about" element={<About />}></Route>
14
15      <Route path="order-summary" element={<OrderSummary />}></Route>
16    </Routes>
17  );
18}
19
20
21
22 export default App;
```
- BROWSER PREVIEW:** A window titled "React App" shows the application at "localhost:3000/order-summary". It displays a "NavBar" with "Home" and "About" links, and a main content area with the text "Order confirmed!".

The screenshot shows a dark-themed interface of a code editor, likely Visual Studio Code. The left sidebar is titled 'EXPLORER' and displays a file tree under 'LESSON01'. The 'src' folder contains 'components' with files 'About.js', 'Home.js', and 'NavBar.js', and other files like 'OrderSummary.js', 'App.css', 'App.js', and 'App.test.js'. The right pane is titled 'JS Home.js U' and shows the source code for the 'Home' component:

```
react-routing-demo > src > components > JS Home.js > [U] Home
1 import React from 'react'
2
3 export const Home = () => {
4     return (
5         <>
6             <div>
7                 Home Page
8                 <button>Place Order</button>
9             </div>
10        </>
11    )
12 }
13
```

- To Navigate Programming React Router provides **useNavigate** Hook

The screenshot illustrates the use of the `useNavigate` hook in a React application. On the left, a code editor shows `Home.js` with the following code:

```
react-routing-demo > src > components > JS Home.js > ...
1 import React from 'react'
2 import { useNavigate } from 'react-router-dom'
3
4 export const Home = () => {
5   const navigate = useNavigate()
6   return (
7     <>
8       <div>
9         Home Page
10        </div>
11        <button onClick={() => navigate('order-summary')}>Place Order</button>
12      </>
13    )
14  }
15 }
16 }
```

The code uses the `useNavigate` hook to define a `navigate` function that can be called from within the component's body. Below the code editor are two browser windows. The left window shows the `Home` page at `localhost:3000`, which contains the text "Home Page" and a "Place Order" button. The right window shows the `OrderSummary` page at `localhost:3000/order-summary`, which displays the message "Order Confirmed". Both browser windows have red boxes around them, and a large red box surrounds the entire code editor area.

- Now implementing a back button programmatically is also done with the **useNavigate** hook

The screenshot illustrates the implementation of a back button using the `useNavigate` hook in a React application.

Code Editor:

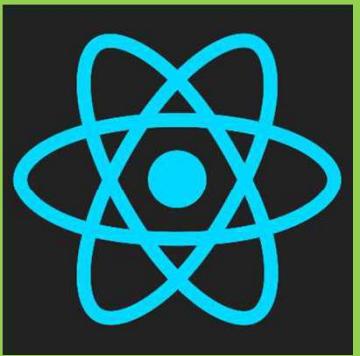
```

JS OrderSummary.js U X JS Home.js U JS App.js M
react-routing-demo > src > components > JS OrderSummary.js > ...
1 import React from 'react'
2 import { useNavigate } from 'react-router-dom'
3
4 export const OrderSummary = () => {
5   const navigate = useNavigate()
6   return (
7     <>
8       <div>
9         |   Order Confirmed
10        </div>
11        <button onClick={() => navigate(-1)}>Go Back</button>
12      </>
13    )
14  }
15

```

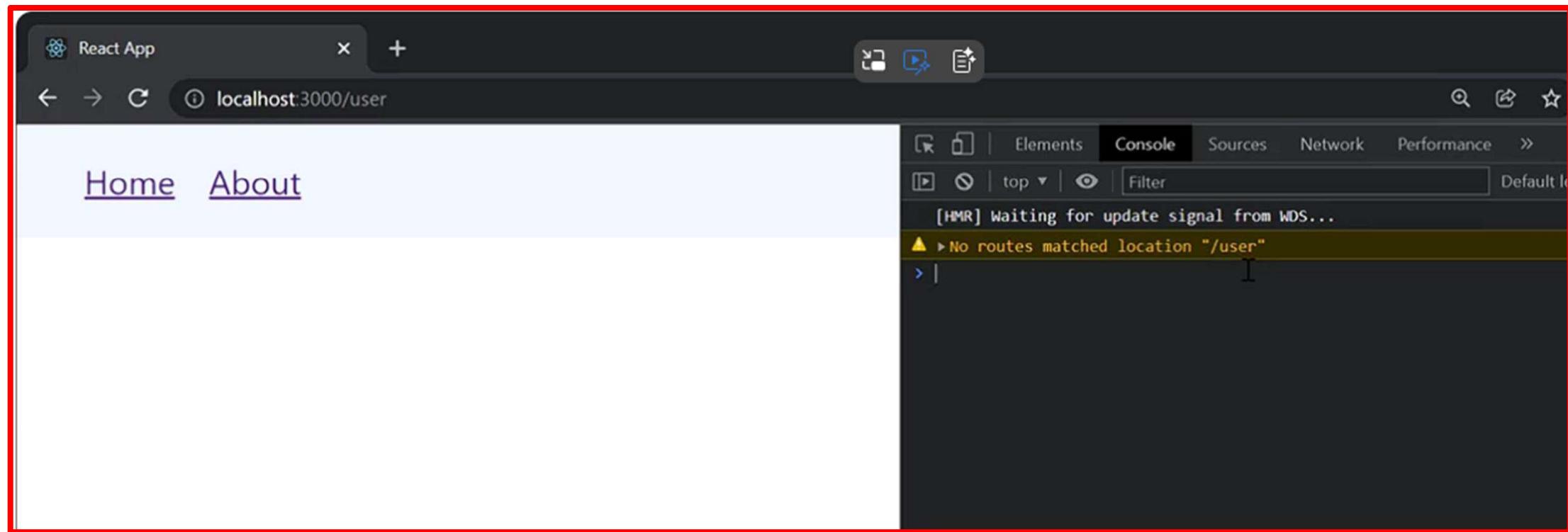
Browsers:

- Top Browser:** Shows the "Home" page with a "Place Order" button. The URL is `localhost:3000`.
- Bottom Browser:** Shows the "Order Confirmed" page with a "Go Back" button. The URL is `localhost:3000/order-summary`.

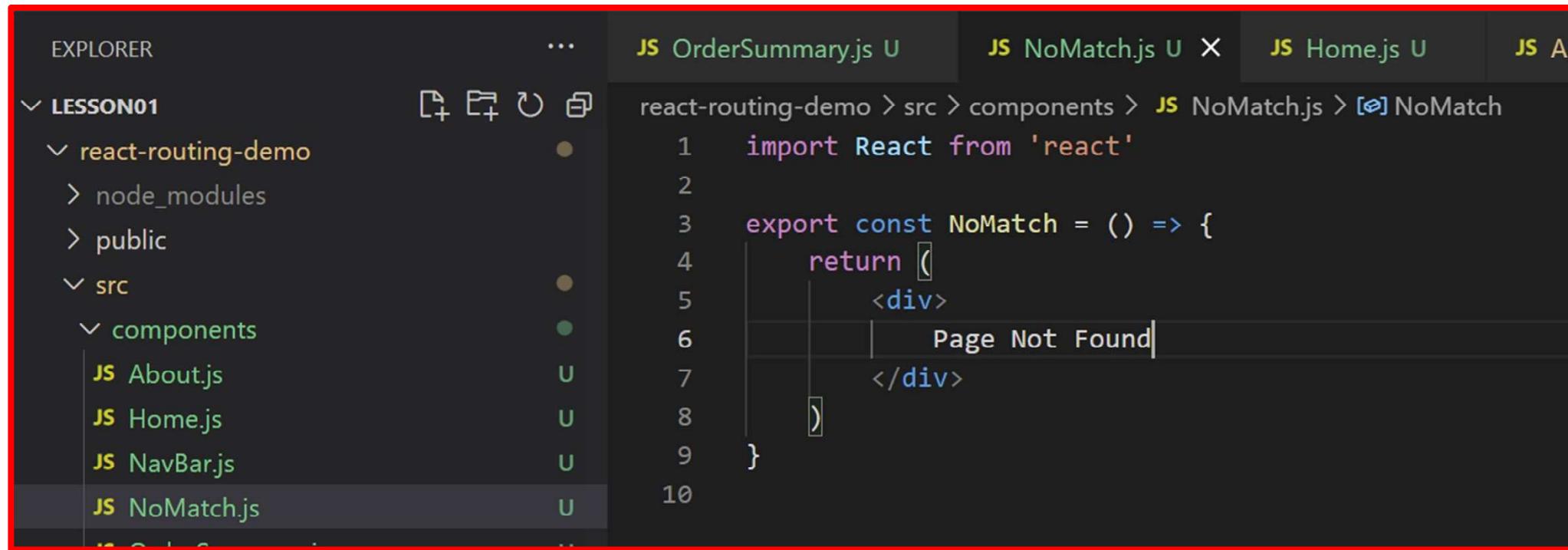


No Match Route

- This scenario is not favorable from a user point of view a user might assume that the app is still loading or there is an error in the **/user route**



- It would be better if we inform the user that the url does not match any route in our application



The screenshot shows a portion of the Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of a project structure under 'LESSON01'. The 'src' folder contains 'components' and 'NoMatch.js'. The 'components' folder has files 'About.js', 'Home.js', and 'NavBar.js'. 'NoMatch.js' is currently selected and highlighted with a red border. In the center, there's a status bar showing file paths: 'react-routing-demo > src > components > NoMatch.js > [NoMatch] NoMatch'. On the right is the main editor area displaying the code for 'NoMatch.js':

```
import React from 'react'
export const NoMatch = () => {
  return (
    <div>
      Page Not Found
    </div>
  )
}
```

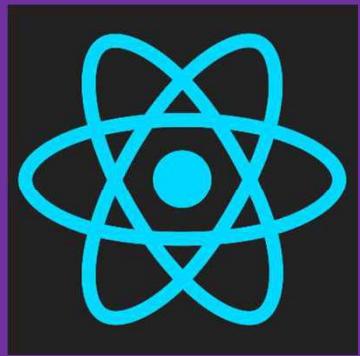
The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists files and folders under 'LESSON01' and 'react-routing-demo'. The 'App.js' file is currently selected and highlighted in the sidebar. The main area displays the contents of 'App.js'.

```
import { Routes, Route } from 'react-router-dom';
import { Home } from './components/Home'
import { About } from './components/About'
import { NavBar } from './components/NavBar'
import { OrderSummary } from './components/OrderSummary'
import { NoMatch } from './components/NoMatch'

function App() {
  return (
    <>
    <NavBar />
    <Routes>
      <Route path="/" element={<Home />}></Route>
      <Route path="about" element={<About />}></Route>
      <Route path="order-summary" element={<OrderSummary />}></Route>
      <Route path="*" element={<NoMatch />}></Route>
    </Routes>
  </>
)
}

export default App;
```

To the right of the code editor is a browser window showing the application's UI. The address bar indicates the URL is 'localhost:3000/order-summary1'. The page content includes a navigation bar with links to 'Home' and 'About', followed by a 'Page Not Found' message.

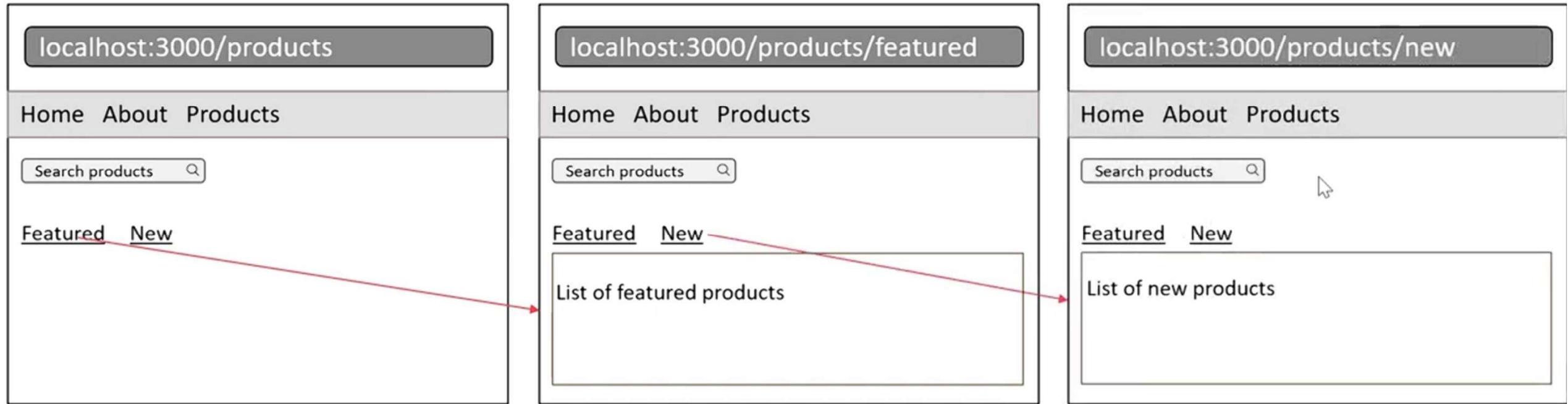


Nested Routes

- So far we have seen how react router helps us navigate across pages in our application but what you should know is that react router also helps to switch between a portion of the view inside the page

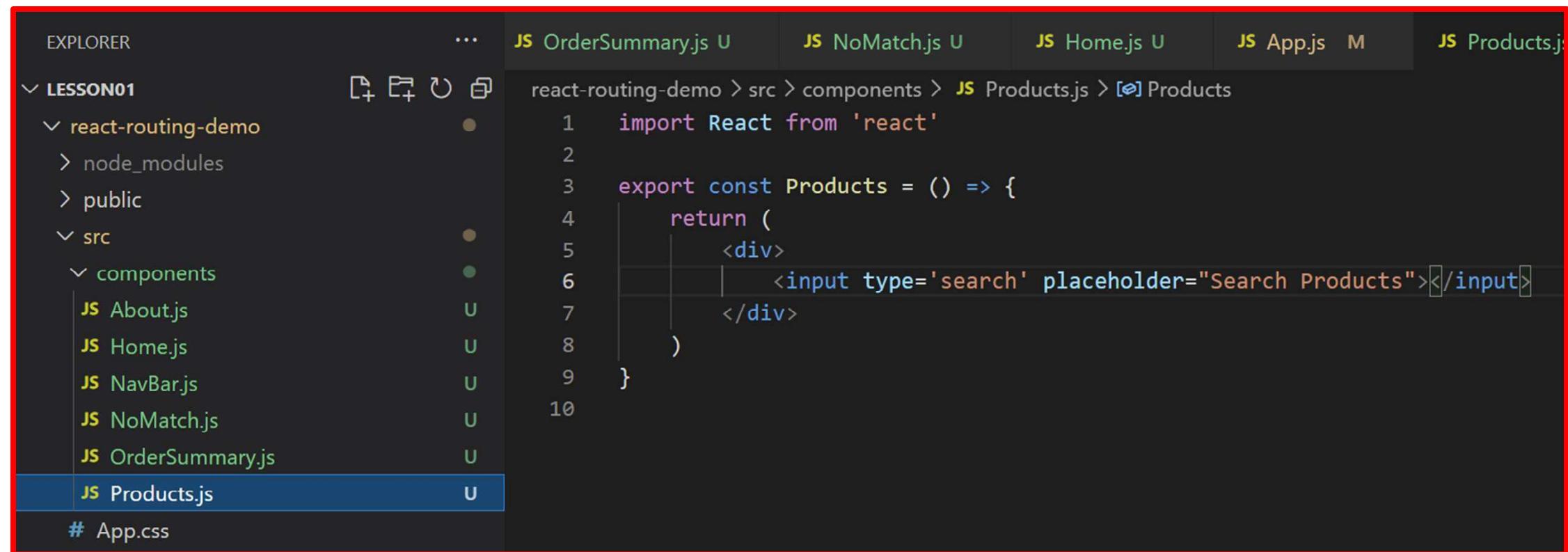


To Achieve this we make use of Nested Routes



Step 1 :

- We are going to configure a new route to the products page and add a link in the nav bar



The screenshot shows a code editor interface with a red border around the main content area. On the left is a file explorer sidebar with the following structure:

- LESSON01
 - react-routing-demo
 - node_modules
 - public
 - src
 - components
 - About.js
 - Home.js
 - NavBar.js
 - NoMatch.js
 - OrderSummary.js
 - Products.js
- # App.css

The main editor area displays the content of `Products.js`:

```
import React from 'react'
export const Products = () => {
  return (
    <div>
      <input type='search' placeholder="Search Products"></input>
    </div>
  )
}
```

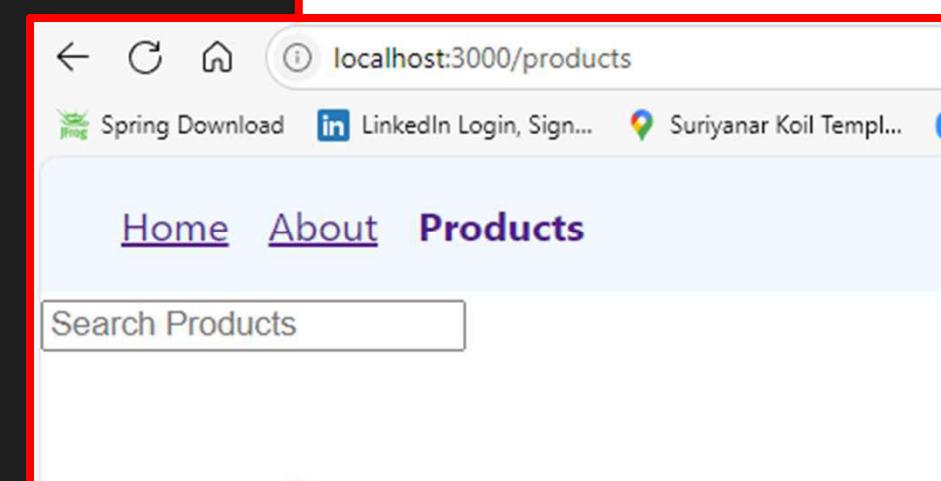
The screenshot shows the VS Code interface with a red border around the main workspace. On the left is the Explorer sidebar showing the project structure:

- LESSON01
- react-routing-demo
 - node_modules
 - public
- src
 - components
 - About.js
 - Home.js
 - NavBar.js
 - NoMatch.js
 - OrderSummary.js
 - Products.js
 - # App.css
 - App.js
 - App.test.js
 - # index.css
 - index.js
 - logo.svg
 - reportWebVitals.js
 - setupTests.js
- .gitignore

On the right is the editor tab bar with tabs for OrderSummary.js, NoMatch.js, Home.js, App.js (selected), and Products.js. The code in the editor is:

```
react-routing-demo > src > JS App.js > App
4 import { NavBar } from './components/NavBar'
5 import { OrderSummary } from './components/OrderSummary'
6 import { NoMatch } from './components/NoMatch'
7 import { Products } from './components/Products'
8 function App() {
9   return (
10   <>
11   <NavBar />
12   <Routes>
13   <Route path="/" element={<Home />}></Route>
14
15   <Route path="about" element={<About />}></Route>
16
17   <Route path="order-summary" element={<OrderSummary />}></Route>
18
19   <Route path="products" element={<Products />}></Route>
20
21   <Route path="*" element={<NoMatch />}></Route>
22   </Routes>
23   </>
24 }
25
26
27 export default App;
```

```
JS Products.js U      JS App.js M      JS NavBar.js U X
react-routing-demo > src > components > JS NavBar.js > ...
1  import React from 'react'
2  import { NavLink } from 'react-router-dom'
3
4  export const NavBar = () => {
5
6      const navLinkStyles = ({ isActive }) => {
7          return {
8              fontWeight: isActive ? 'bold' : 'normal',
9              textDecoration: isActive ? 'none': 'underline'
10         }
11     }
12
13     return (
14         <nav>
15             <NavLink style={navLinkStyles} to="/">Home</NavLink>
16             <NavLink style={navLinkStyles} to="/about">About</NavLink>
17             <NavLink style={navLinkStyles} to="/products">Products</NavLink>
18         </nav>
19     )
20 }
21
```



Step 2:

- Lets add another set of navigation links within the products component

The image shows a development environment with a code editor and a browser window. The code editor on the left displays the file `Products.js` with the following content:

```
JS Products.js U X JS App.js M JS NavBar.js U
react-routing-demo > src > components > JS Products.js > ...
1 import React from 'react'
2 import { Link } from 'react-router-dom'
3
4 export const Products = () => {
5   return (
6     <>
7       <div>
8         <input type='search' placeholder="Search Products"></input>
9       </div>
10      <nav>
11        <Link to='featured'>Featured</Link>
12        <Link to='new'>New</Link>
13      </nav>
14    </>
15  )
16}
17
18}
```

The browser window on the right shows the application running at `localhost:3000/products`. The page has a header with [Home](#), [About](#), and [Products](#). Below the header is a search bar labeled "Search Products". Underneath the search bar are two links: [Featured](#) and [New](#). A red box highlights the `to='featured'` and `to='new'` paths in the code editor, and a red callout box points to them with the text "Relative Path Without /".

Products.js

```
react-routing-demo > src > # index.css > ↗ primary-nav
  5   | sans-serif;
  6   | -webkit-font-smoothing: antialiased;
  7   | -moz-osx-font-smoothing: grayscale;
  8 }
  9
10  code {
11    font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
12    monospace;
13 }
14
15 .primary-nav {
16   background-color: #aliceblue;
17   padding: 16px 32px;
18 }
19
20 nav a {
21   margin-right: 16px;
22 }
23
24 /* nav a.active {
25   text-decoration: none;
26   font-weight: bold;
27 } */
```

NavBar.js

```
react-routing-demo > src > components > JS NavBar.js > ↗ NavBar
  1 import React from 'react'
  2 import { NavLink } from 'react-router-dom'
  3
  4 export const NavBar = () => {
  5
  6   const navLinkStyles = ({ isActive }) => {
  7     return {
  8       fontWeight: isActive ? 'bold' : 'normal',
  9       textDecoration: isActive ? 'none': 'underline'
 10     }
 11   }
 12
 13   return (
 14     <nav className='primary-nav'>
 15       <NavLink style={navLinkStyles} to="/">Home</NavLink>
 16       <NavLink style={navLinkStyles} to="/about">About</NavLink>
 17       <NavLink style={navLinkStyles} to="/products">Products</NavLink>
 18     </nav>
 19   )
 20
 21 }
```

Browser Screenshot

localhost:3000/products

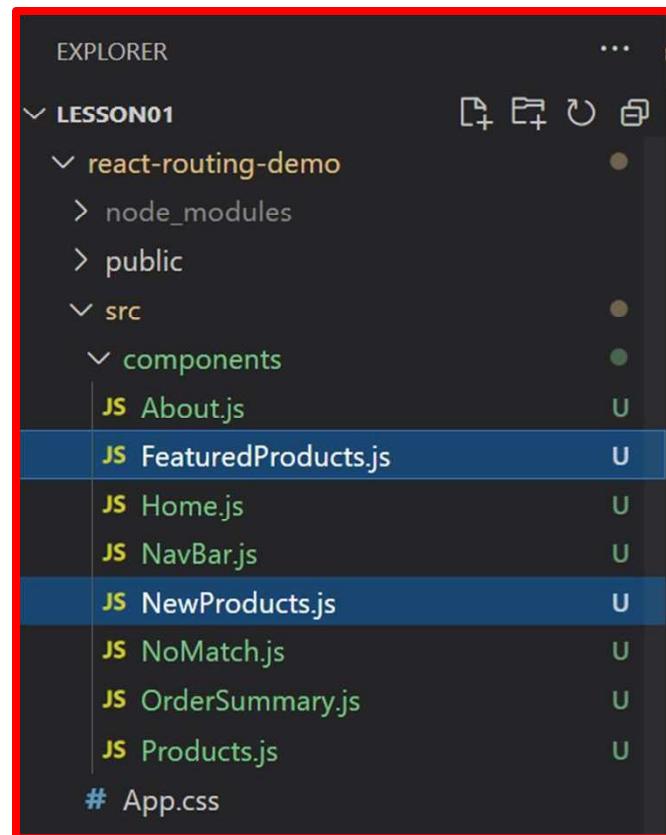
Home About **Products**

Search Products

Featured New

Step 3:

- We will create 2 New Components that need to be rendered for **Featured Products** and **New Products** and also configure the new routes



JS App.js M **JS** FeaturedProducts.js U X **JS** NewProducts.js U

react-routing-demo > src > components > **JS** FeaturedProducts.js > ...

```
1 import React from 'react'
2
3 export const FeaturedProducts = () => {
4     return (
5         <div>
6             | List of Featured Products
7         </div>
8     )
9 }
10 |
```

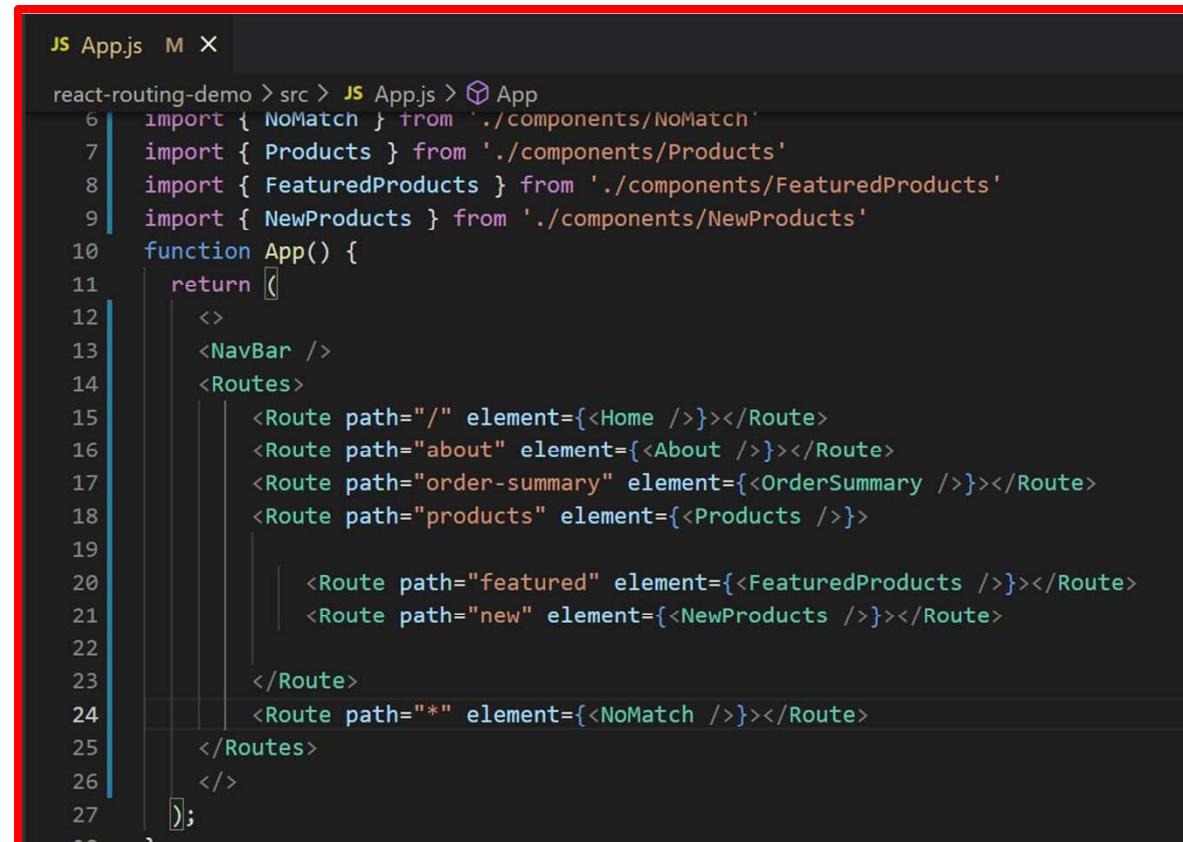
JS App.js M X **JS** FeaturedProducts.js U **JS** NewProducts.js U X

react-routing-demo > src > components > **JS** NewProducts.js > ...

```
1 import React from 'react'
2
3 export const NewProducts = () => {
4     return (
5         <div>
6             | List of New Products
7         </div>
8     )
9 }
```

Now Configuring Nested Routes

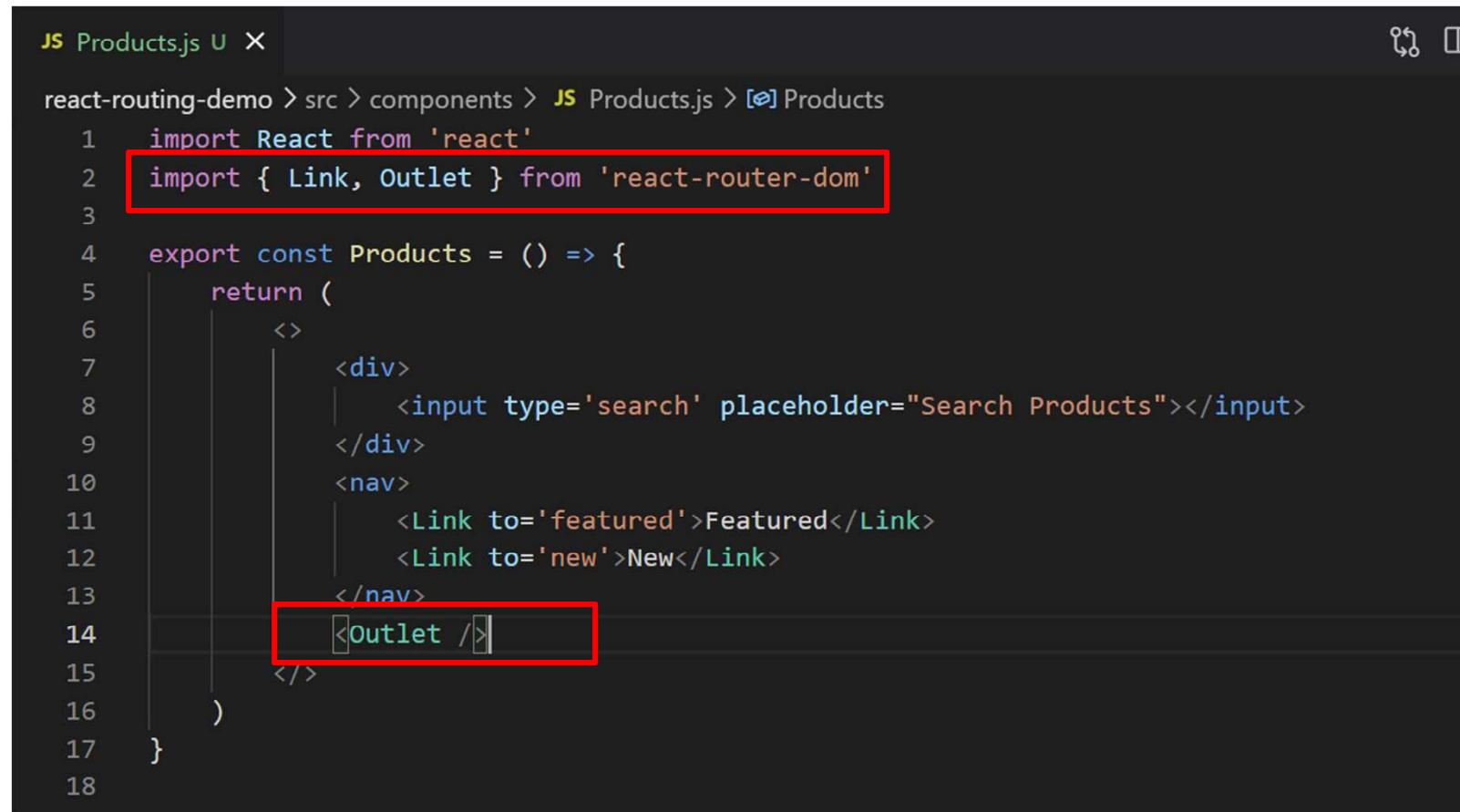
- Both the components have to be nested within the **products** route



```
JS App.js M X
react-routing-demo > src > JS App.js > App
  6 import { NoMatch } from './components/NoMatch'
  7 import { Products } from './components/Products'
  8 import { FeaturedProducts } from './components/FeaturedProducts'
  9 import { NewProducts } from './components/NewProducts'
 10 function App() {
 11   return [
 12     <>
 13     <NavBar />
 14     <Routes>
 15       <Route path="/" element={<Home />}></Route>
 16       <Route path="about" element={<About />}></Route>
 17       <Route path="order-summary" element={<OrderSummary />}></Route>
 18       <Route path="products" element={<Products />}>
 19         <Route path="featured" element={<FeaturedProducts />}></Route>
 20         <Route path="new" element={<NewProducts />}></Route>
 21       </Route>
 22       <Route path="*" element={<NoMatch />}></Route>
 23     </Routes>
 24   </>
 25   ];
 26 }
 27 
```

- What is special about nested routes is that react router automatically forms the full path to the children routes

- However Products Page Doesn't Know Where to Render Child Components
 - So React Router Provides Outlet Component



```
JS Products.js × react-routing-demo > src > components > JS Products.js > [?] Products
1 import React from 'react'
2 import { Link, Outlet } from 'react-router-dom'
3
4 export const Products = () => {
5   return (
6     <>
7       <div>
8         <input type='search' placeholder="Search Products"></input>
9       </div>
10      <nav>
11        <Link to='featured'>Featured</Link>
12        <Link to='new'>New</Link>
13      </nav>
14      <Outlet />
15    </>
16  )
17}
18
```

The image displays three separate browser windows, each showing a different page of a web application. All three windows have identical header and footer sections.

Header:

- Back, Forward, Home, Refresh buttons
- Address bar: localhost:3000/products
- Links: Spring Download, LinkedIn Login, Sign..., Suriyanar Koil Templ..., Samsung

Page 1 (Top Right): localhost:3000/products

- Navigation: Home, About, **Products**
- Search: Search Products
- Filter: Featured, New

Page 2 (Bottom Left): localhost:3000/products/featured

- Navigation: Home, About, **Products**
- Search: Search Products
- Filter: Featured, New
- Content: List of Featured Products

Page 3 (Bottom Right): localhost:3000/products/new

- Navigation: Home, About, **Products**
- Search: Search Products
- Filter: Featured, New
- Content: List of New Products

