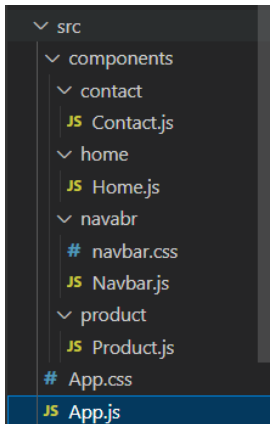


1. >npx create-react-app routing
2. Create components folder in src folder
  - a. Create 4 folders in src – navbar, home, product, contact
  - b. Create js file for each in respective folder



3. Code for navbar

```
JS Navbar.js X
src > components > navabr > JS Navbar.js > ...
1  import React from 'react'
2  import './navbar.css'
3
4  export default function Navbar() {
5    return (
6      <div className='navbar'>
7        <nav>
8          <h1>Reat Routing</h1>
9
10         <a href="">Home</a>
11         <a href="">Product</a>
12         <a href="">Contact</a>
13       </nav>
14     </div>
15   )
16 }
17

# navbar.css X
src > components > navabr > # navbar.css > .navbar nav h1
1  .navbar{
2    background-color: aqua;
3  }
4  .navbar nav
5  {
6    display: flex;
7    gap: 10px;
8    align-items: center;
9    margin: 0 20px;
10 }
11
12 .navbar nav h1
13 {
14   margin-right: auto;
15 }
```

4. Display navbar component in App component

```
JS App.js X
src > JS App.js > ...
1  import logo from './logo.svg';
2  import './App.css';
3  import Navbar from './components/navbar/Navbar';
4
5  function App() {
6    return (
7      <>
8        <Navbar/>
9      </>
10   );
11 }
12
13 export default App;
14
```

5. Npm start and check output



6. Implementing routing

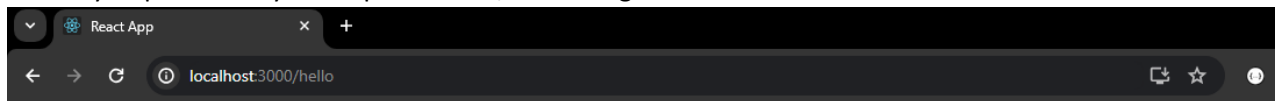
- >npm i react-router-dom
- Create router array and attach it to the RouterProvider as property in index.js

```
JS index.js X
src > JS index.js > ...
4  import App from './App';
5  import reportWebVitals from './reportWebVitals';
6  import { RouterProvider, createBrowserRouter } from 'react-router-dom';
7
8  const routes=createBrowserRouter([
9    {
10     path: "/",
11     element:<App/>
12   }
13 ])
14
15 const root = ReactDOM.createRoot(document.getElementById('root'));
16 root.render(
17   // <React.StrictMode>
18   // <App />
19   // </React.StrictMode>
20   <RouterProvider router={routes}/>
21 );
22
```

c. Output



- d. If we try to provide any other path than "/" we will get error



## Unexpected Application Error!

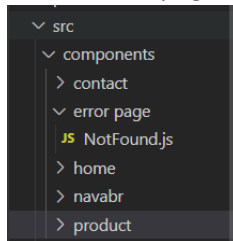
### 404 Not Found

Hey developer 💡

You can provide a way better UX than this when your app throws errors by providing your own `ErrorBoundary` or `errorElement` prop on your route.

## 7. Handling Error page

- a. Create error page folder inside components folder. In error page folder create NotFound.js



- b. Code for NotFound.js

```
import React from 'react'

export default function NotFound() {
  return (
    <center>
      
    </center>
  )
}
```

- c. Check output



404 error  
page not found



## 8. Handling routing from navbar

- a. In index.js provide children array

```
JS index.js X
src > JS index.js > ...
11
12 const routes=createBrowserRouter([
13   {
14     path:"/",
15     element:<App/>,
16     errorElement:<NotFound/>,
17     children:[
18       {
19         path:'/home',
20         element:<Home/>
21       },
22       {
23         path:'/contact',
24         element:<Contact/>
25       },
26       {
27         path:'/product',
28         element:<Product/>
29       }
30     ]
31   }
32 ])
```

- b. In Navbar.js provide links

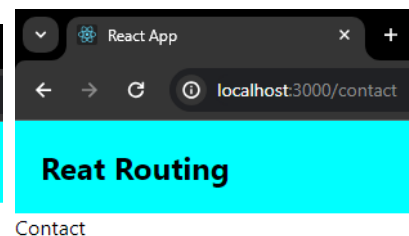
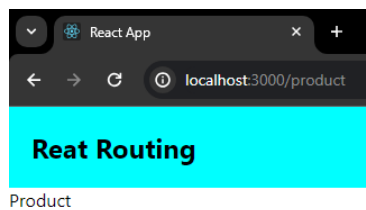
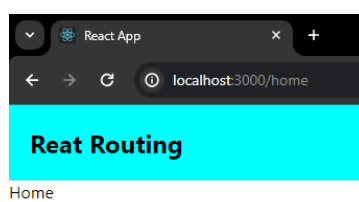
```
export default function Navbar() {
  return (
    <div className='navbar'>
      <nav>
        <h1>Reat Routing</h1>
        <a href="/home">Home</a>
        <a href="/product">Product</a>
        <a href="/contact">Contact</a>
      </nav>
    </div>
  )
}
```

- c. In App.js provide <Outlet/> for component switching

```
function App() {
  return (
    <>
      <Navbar/>
      <Outlet/>
    </>
  );
}

export default App;
```

- d. Output

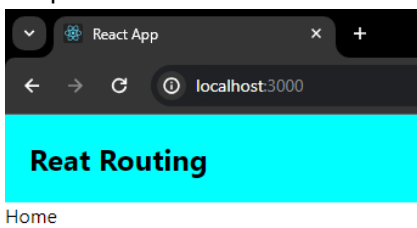


## 9. Handling index page (displaying home component as home/index page)

### a. In index.js provide index:true

```
11
12 const routes=createBrowserRouter([
13   {
14     path:"/",
15     element:<App/>,
16     errorElement:<NotFound/>,
17     children:[
18       {
19         index:true, ✓
20         element:<Home/> ✓
21       },
22       {
23         path:'/home',
24         element:<Home/>
25       },
26     ],
27   },
28 ], {})
```

### b. Output



### c.

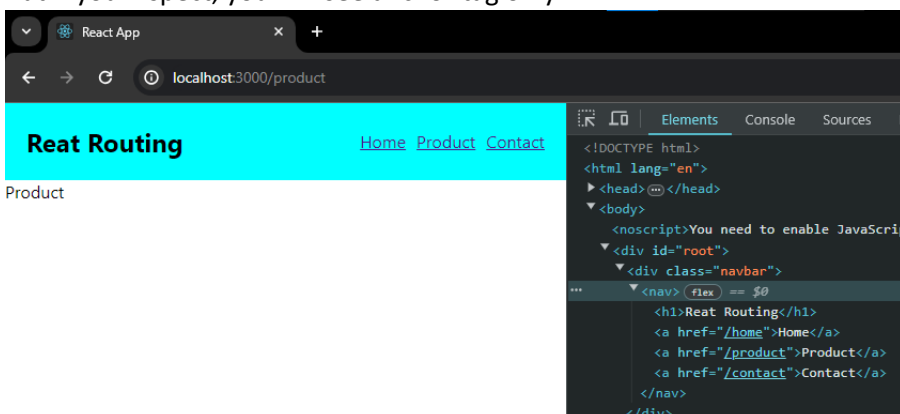
## 10. Preventing page from getting refersh

- a. If we click on home/contact/product from navbar then ypu will see, our page is getting refersh. As REACT is used to create SPA, we have to prevent our page from getting refresh. We can achieve this bye converting anchor tag to Link tag provided by react-router-dom

```
export default function Navbar() {
  return (
    <div className='navbar'>
      <nav>
        <h1>Reat Routing</h1>
        <a href="/home">Home</a>
        <a href="/product">Product</a>
        <a href="/contact">Contact</a>
      </nav>
    </div>
  )
}
```

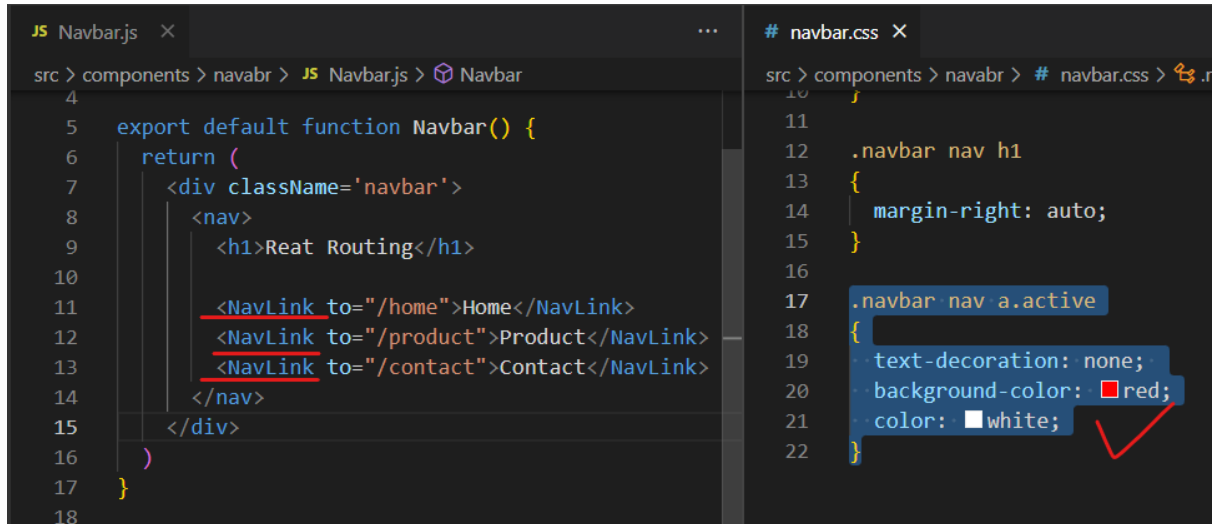
```
export default function Navbar() {
  return (
    <div className='navbar'>
      <nav>
        <h1>Reat Routing</h1>
        <Link to="/home">Home</Link>
        <Link to="/product">Product</Link>
        <Link to="/contact">Contact</Link>
      </nav>
    </div>
  )
}
```

But if you inspect, you will see anchor tag only



## 11. Providing CSS on navbar

- Convert Link to NavLink – this will automatically add active class on anchor tag (inspect and check)

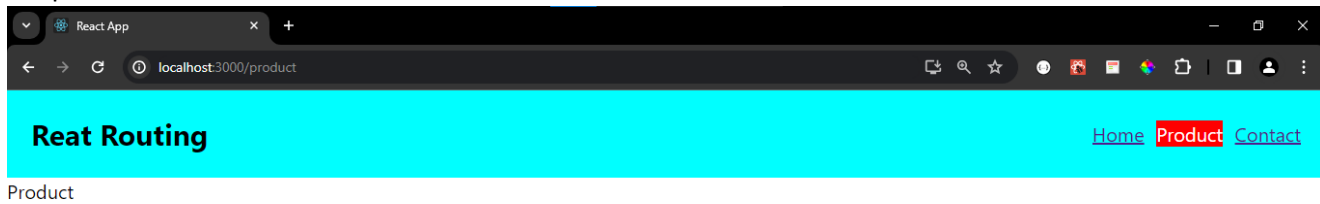


The screenshot shows two files in a code editor. The left file, `Navbar.js`, is a React component that exports a default function `Navbar()`. It returns a JSX element consisting of a `div` with the class `navbar`, which contains a `nav` element. Inside the `nav` element, there is an `h1` tag with the text "Reat Routing" and three `NavLink` components with `to` attributes of `"/home"`, `"/product"`, and `"/contact"`. The `NavLink` components are highlighted with red boxes. The right file, `navbar.css`, contains CSS rules. The first rule targets `.navbar nav h1` and sets `margin-right: auto;`. The second rule targets `.navbar nav a.active` and sets `text-decoration: none;`, `background-color: red;`, and `color: white;`. The second rule is highlighted with a blue box and a red checkmark.

```
JS Navbar.js
src > components > navbar > JS Navbar.js > Navbar
4
5 export default function Navbar() {
6   return (
7     <div className='navbar'>
8       <nav>
9         <h1>Reat Routing</h1>
10
11         <NavLink to="/home">Home</NavLink>
12         <NavLink to="/product">Product</NavLink>
13         <NavLink to="/contact">Contact</NavLink>
14       </nav>
15     </div>
16   )
17 }
18

# navbar.css
src > components > navbar > # navbar.css > navbar.css
10
11
12 .navbar nav h1
13 {
14   margin-right: auto;
15 }
16
17 .navbar nav a.active
18 {
19   text-decoration: none;
20   background-color: red;
21   color: white;
22 }
```

- Output



## 12. Implementing dynamic routing

- Create products.json inside src folder
- Code for products.json



The screenshot shows a code editor with the contents of a `products.json` file. The file contains a JSON array of three product objects. Each object has the following properties: `id`, `title`, `price`, and `image`. The first product is a "Backpack" with a price of 109.95. The second product is "Slim T-Shirts" with a price of 22.3. The third product is a "Cotton Jacket" with a price of 55.99. The `image` property for each product is a URL from a fake store API.

```
[
  {
    "id": 1,
    "title": "Backpack",
    "price": 109.95,
    "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg"
  },
  {
    "id": 2,
    "title": "Slim T-Shirts ",
    "price": 22.3,
    "image": "https://fakestoreapi.com/img/71-3HjGNDUL._AC_SY879._SX._UX._SY._UY_.jpg"
  },
  {
    "id": 3,
    "title": "Cotton Jacket",
    "price": 55.99,
    "image": "https://fakestoreapi.com/img/71li-ujt1UL._AC_UX679_.jpg"
  }
]
```

- c. Fetch data from products.json into Product.js and display into a card

```
import React from 'react'
import products from '../products.json'
import './card.css'
import { Link } from 'react-router-dom'

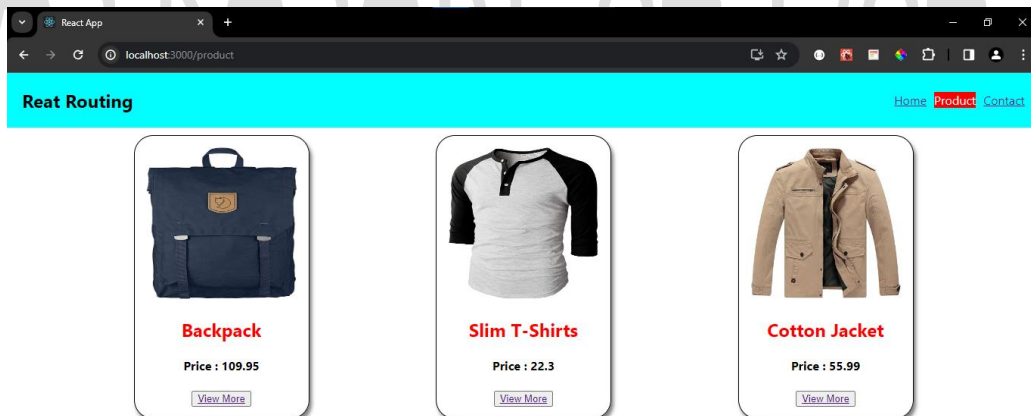
export default function Product() {
  return (
    <div className='card'>
      {
        products.map(product =>
          {
            return (
              <div className='card-items'>
                <img src={product.image} width="200px"
height="200px"/>
                <h2>{product.title}</h2>
                <h4>Price : {product.price}</h4>
                <button><Link
to={` /product/${product.id}`}>View More</Link></button>
              </div>
            )
          }
        )
      }
    </div>
  )
}
```

Create card.css in product folder. Code –

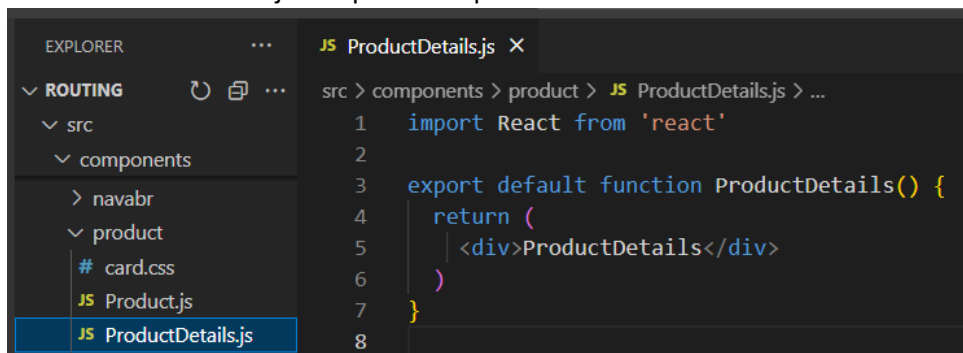
```
.card
{
  display: flex;
  justify-content: space-evenly;
  margin-top: 10px;
}

.card-items
{
  text-align: center;
  border: 1px solid black;
  padding: 15px;
  border-radius: 25px;
  box-shadow: 3px 5px 5px grey;
}

.card-items h2
{
  color: red;
}
```



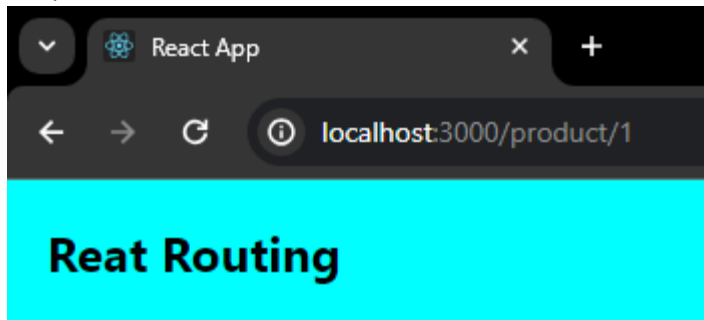
- d. Create ProductDetails.js component in product folder



- e. Provide path for ProductDetails from Index.js

```
JS index.js x
src > JS index.js > routes > children
30     },
31     {
32       path: '/product',
33       element: <Product/>
34     },
35     {
36       path: '/product/:id',
37       element: <ProductDetails/>
38     }
39   ]
40 }
41 )
```

Output :



ProductDetails

13. Displaying product in ProductDetails.js