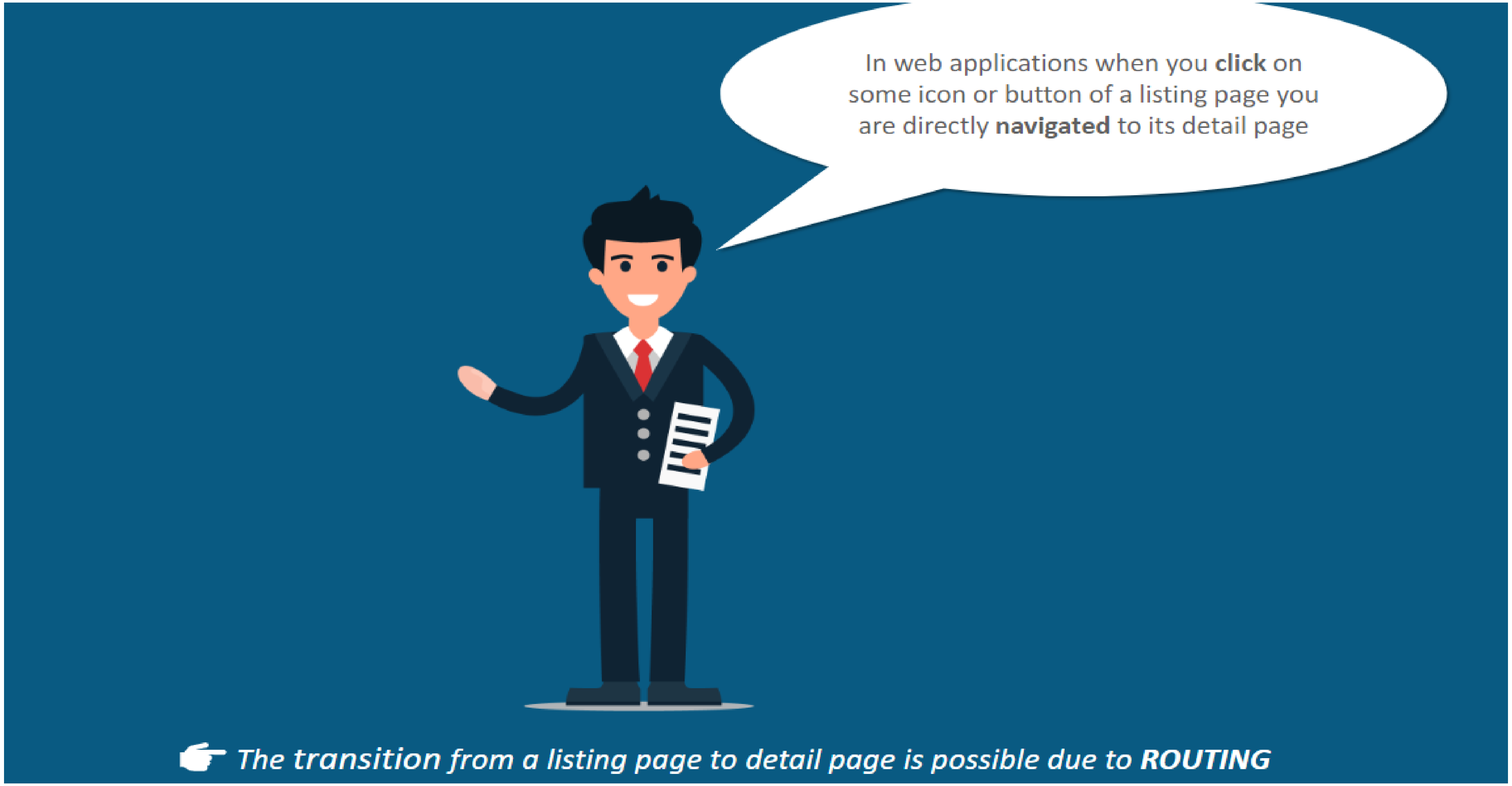
SPA


In web applications when you **click** on some icon or button of a listing page you are directly **navigated** to its detail page

The transition from a listing page to detail page is possible due to ***ROUTING***

SPA

❑Single-page applications are websites that have pages that load inline within the same page. Read on to find how you can create a single page application using React.

❑A single page application (SPA) is essentially a webpage that interacts with the web browser dynamically by rewriting the current web page with the data obtained from the webserver. Hence, in a single page application, the webpage does not reload the page during its runtime and instead works within a browser.

Single-page Application

Reloading: Single-page applications work inside a browser and do not require page reloading during webpage executing.

UI/UX: Offers outstanding user experience by imitating a natural environment with the browser by eliminating wait time and page reloads. It consists of a single web page that loads all content using JavaScript. It requests the markup and data

Why choose a single-page application?

The benefits of choosing single-page applications (SPA) are:

- ❑ SPA is quicker since all the webpage resources are loaded only once throughout the application, and data is the only resource that is transmitted.
- ❑ SPA caches local storage effectively as it sends one request, stores all the data, and uses it even when offline.
- ❑ SPA simplifies and streamlines development activities as it eliminates the need to write code to render pages on the server.
- ❑ SPA can be debugged with ease with Chrome as it is possible to investigate page elements and monitor network operations.

SPA

Routing

Routing **is a process of mapping from the URL to rendered content.**

It helps user to move to the different parts of an application, when he enters the URL or clicks an element (link, button, icon, image etc) within the application.

To define a route, we require the ***URL path*** to match with and the ***component*** to render.

SPA

react-router

react-router is a popular routing library used to integrate routes in React applications.

- This library is split into two parts: ***react-router-dom and react-router-native***

- *react-router-dom* is used to write an application that runs in ***browser*** and *react-router-native* used to write the ***mobile*** applications

- You can include react-router using: ***npm install react-router-dom***

SPA

Features Of react-router Library

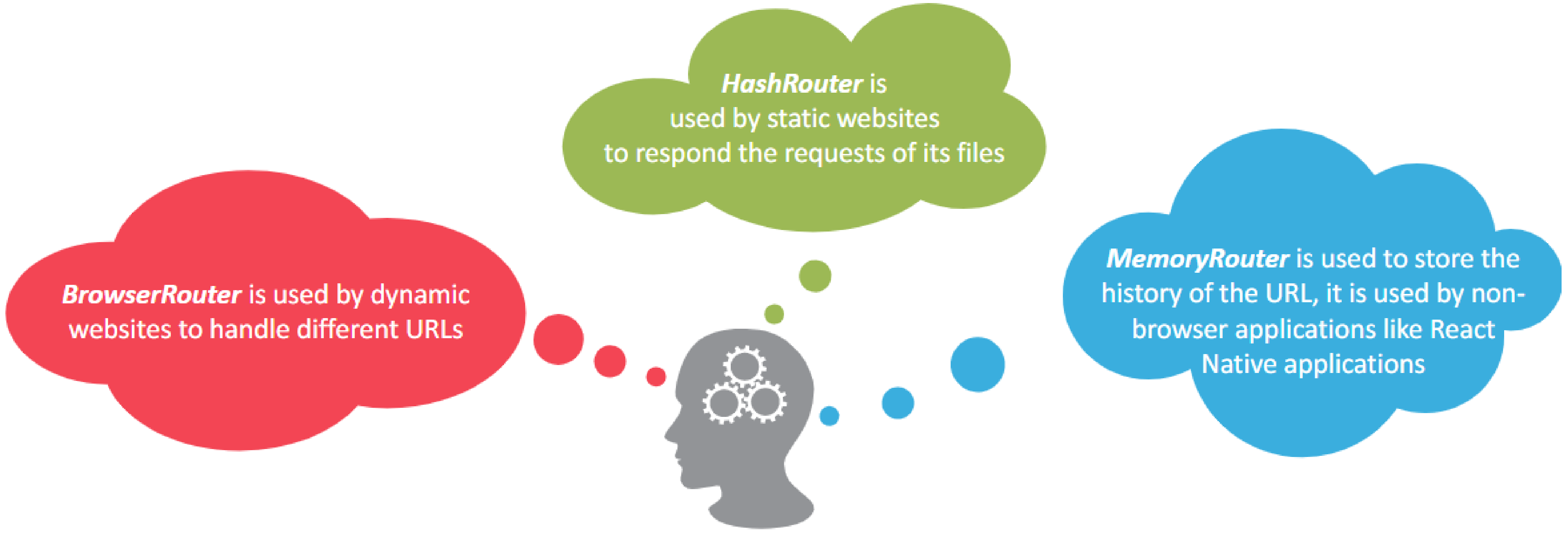
In order to support ***multiple views*** within the ***single page application***, react-router library includes following features:

- ***Navigating*** backward and forward through the application, maintaining the ***history***, and ***restoring*** the state of the application
- ***Rendering*** appropriate page components as per the ***URL***
- ***Redirecting*** the user from one route to the other
- 404 Page not foud — Rendering a ***404 page*** when ***none*** of the ***routes match*** the URL

SPA

React Router

The ***react-router*** package includes a number of routers that we can take advantage of depending on the platform we are targeting. These include BrowserRouter, HashRouter, and MemoryRouter.



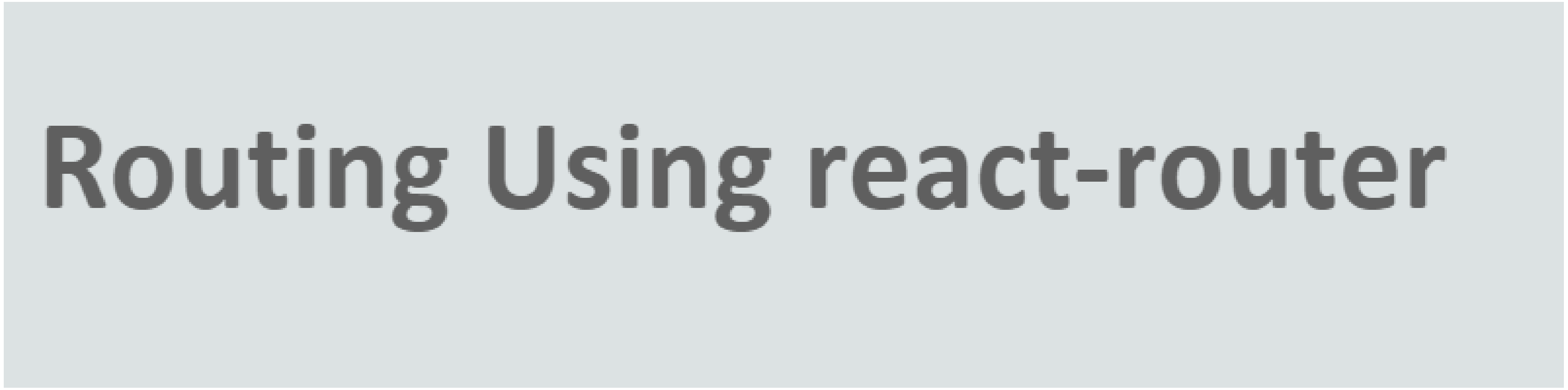
HashRouter is used by static websites to respond the requests of its files

BrowserRouter is used by dynamic websites to handle different URLs

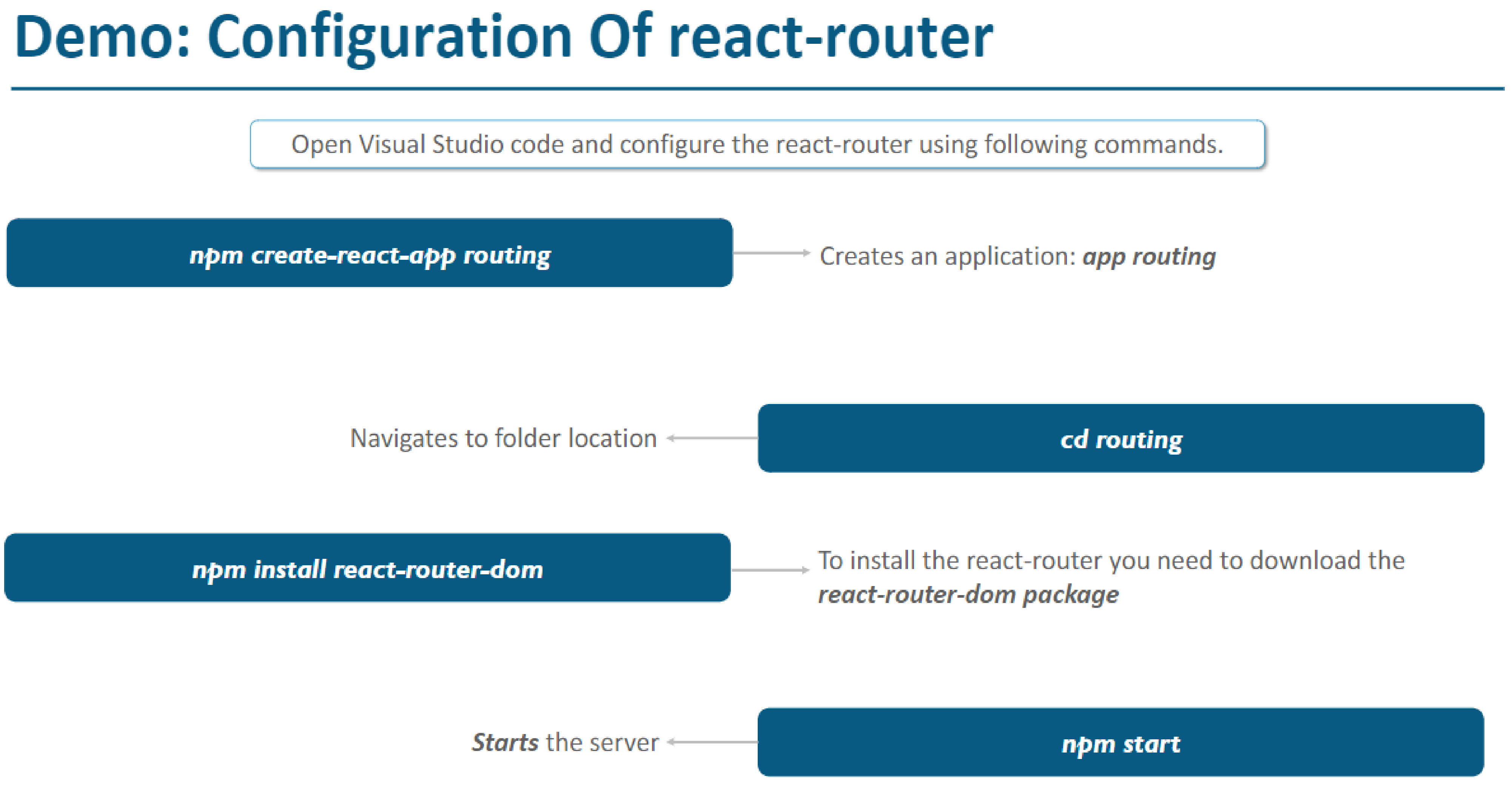
MemoryRouter is used to store the history of the URL, it is used by non-browser applications like React Native applications

For Browser based applications BrowserRouter and HashRouter are a good fit.

SPA


Routing Using react-router

SPA



Demo: Configuration Of react-router

Open Visual Studio code and configure the react-router using following commands.

```
npm create-react-app routing
```

Creates an application: ***app routing***

```
cd routing
```

Navigates to folder location

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

To install the react-router you need to download the ***react-router-dom package***

```
npm start
```

Starts the server

SPA

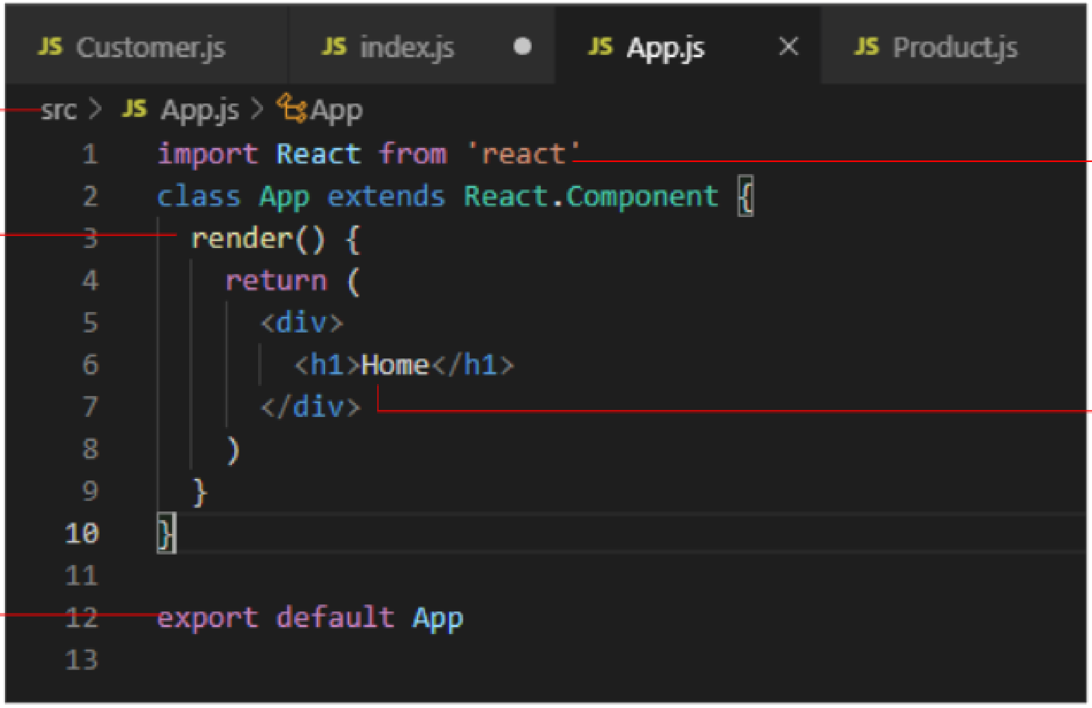
Demo: Configure The Application Components

Configure *three* components: ***App, Customer and Product***

Create a file called App.js within the ***Source folder***

Method that takes input data and returns JSX data

The entire class is exported, so that we can import this component into index.js file



```
import React from 'react'
class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Home</h1>
      </div>
    )
  }
}

export default App
```

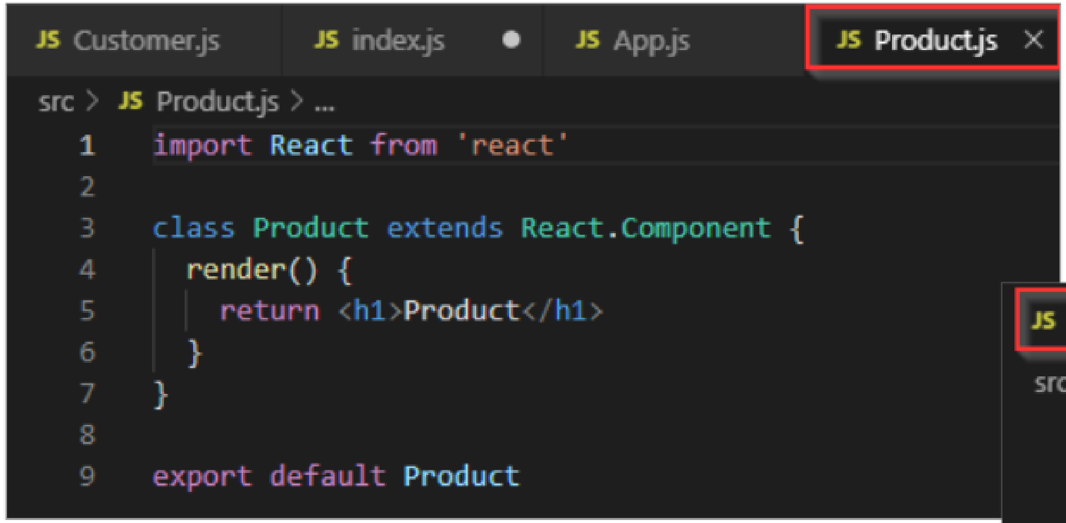
Imports react library

Data to be displayed when application is routed to App.js

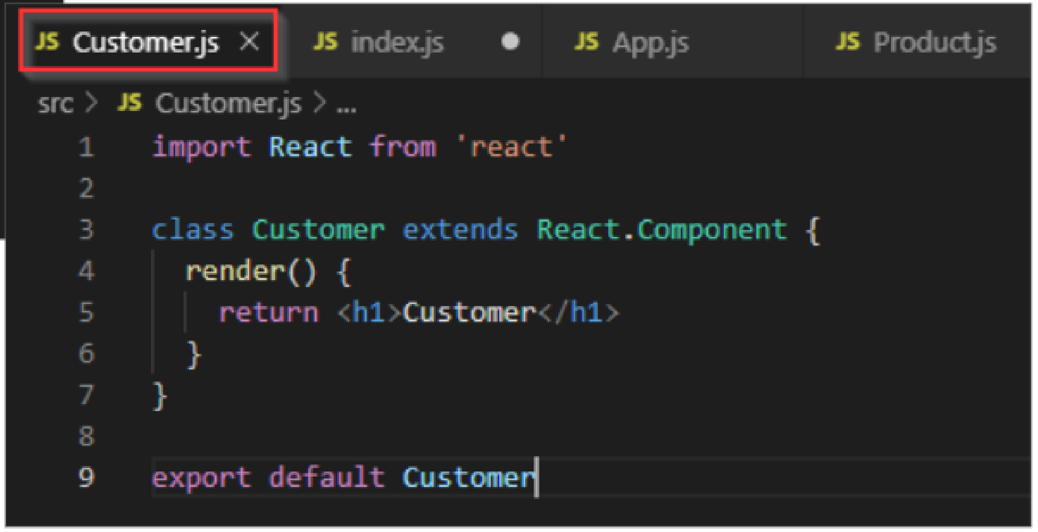
SPA

Demo: Configure The Application Components (contd.)

Similarly create ***Customer*** and ***Product*** components.



```
import React from 'react'

class Product extends React.Component {
  render() {
    return <h1>Product</h1>
  }
}

export default Product
```

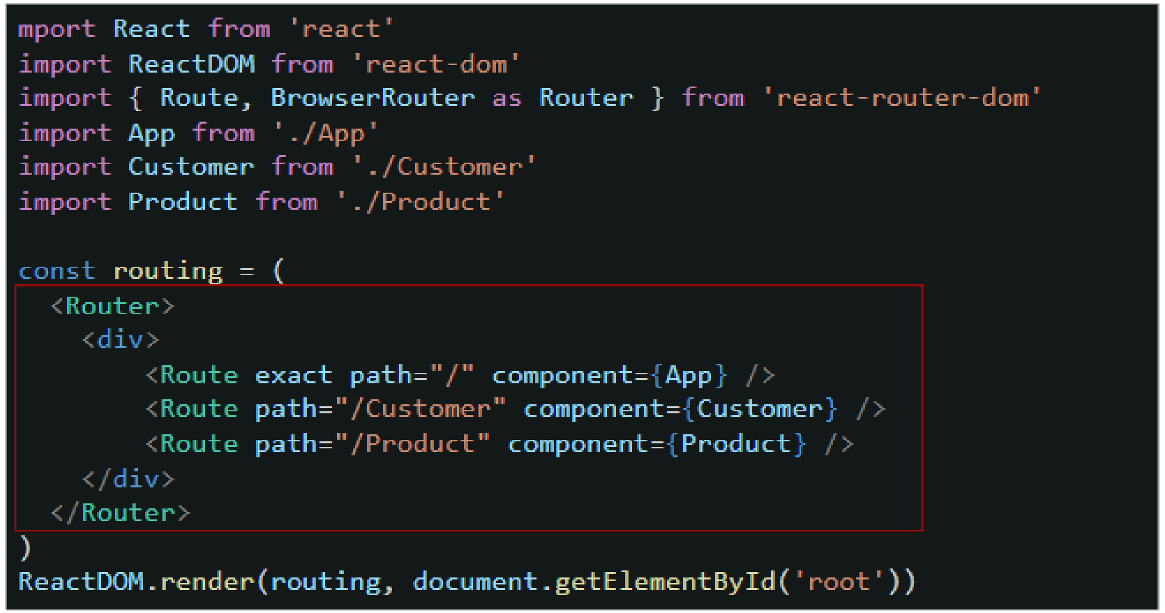


```
import React from 'react'

class Customer extends React.Component {
  render() {
    return <h1>Customer</h1>
  }
}

export default Customer
```

SPA

Demo: Import Components In Index.js File

Import the components and ***specify*** the ***path*** and ***component*** in routing section.

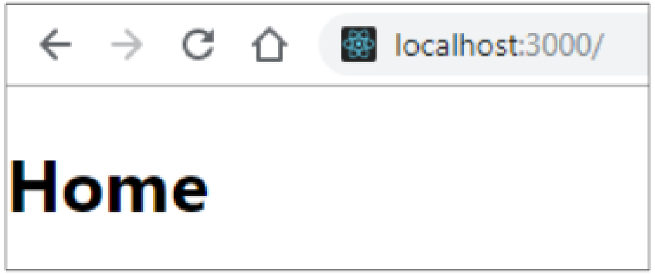


```
mport React from 'react'
import ReactDOM from 'react-dom'
import { Route, BrowserRouter as Router } from 'react-router-dom'
import App from './App'
import Customer from './Customer'
import Product from './Product'

const routing = (
  <Router>
    <div>
        <Route exact path="/" component={App} />
        <Route path="/Customer" component={Customer} />
        <Route path="/Product" component={Product} />
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'))
```

SPA

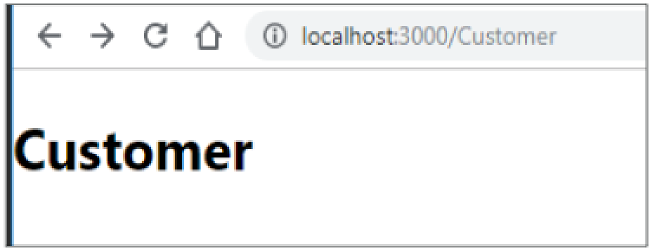
Check The Output

Open the browser, check the routing of application at ***localhost:3000*** using the paths.


localhost:3000/

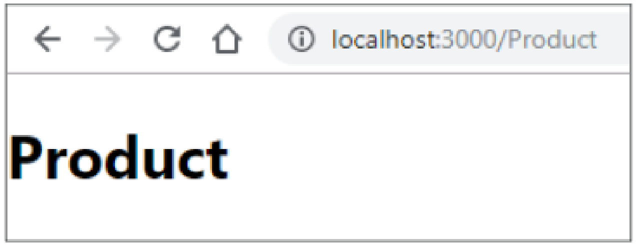
Home

Home component rendered at path /


localhost:3000/Customer

Customer

Customer component rendered at path ***/Customer***


localhost:3000/Product

Product

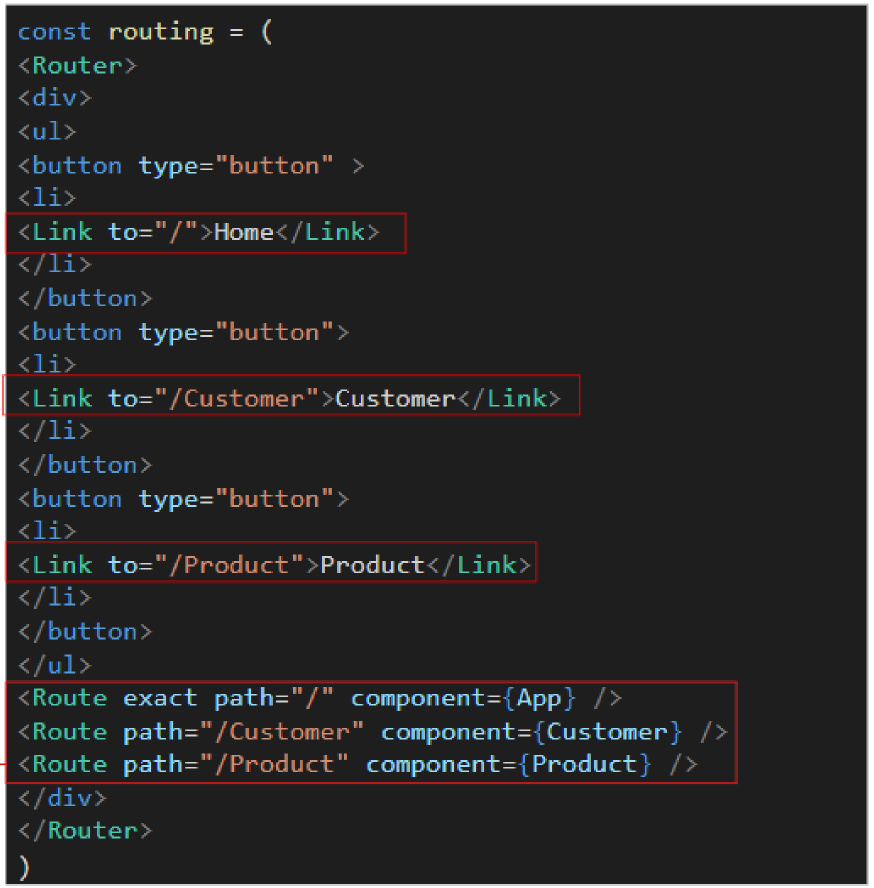
Product component rendered at path ***/Product***

SPA

Manage Navigations Using Links

Link component will let you see the routes rendered directly on the screen by clicking the component, instead of mentioning path in URL.

Declaration of path and its corresponding component



```
const routing = (
<Router>
<div>
<ul>
<button type="button" >
<li>
<Link to="/">Home</Link>
</li>
</button>
<button type="button">
<li>
<Link to="/Customer">Customer</Link>
</li>
</button>
<button type="button">
<li>
<Link to="/Product">Product</Link>
</li>
</button>
</ul>
<Route exact path="/" component={App} />
<Route path="/Customer" component={Customer} />
<Route path="/Product" component={Product} />
</div>
</Router>
)
```

SPA

404 Page

01

When user navigates to wrong path, to indicate wrong path, to the user ***404 page*** is used. It is also Known by the term ***Page Not Found***

02

In order to add this page to the website we need to import ***Switch*** component provided by react-router

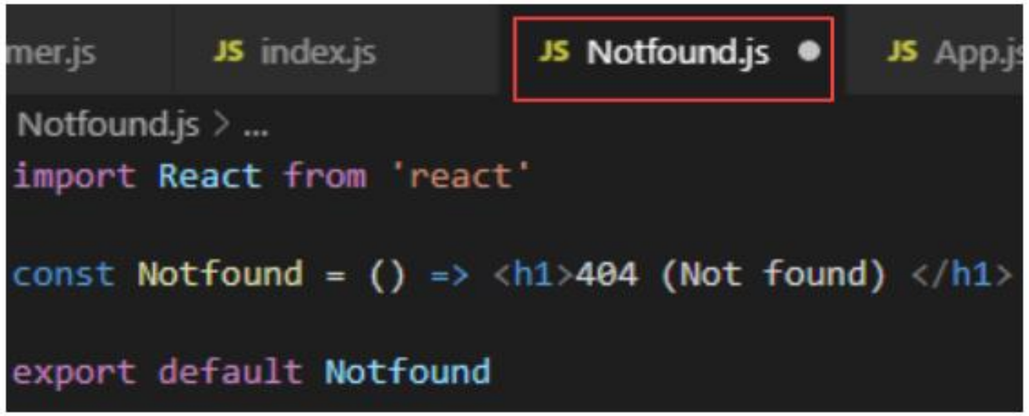
03

Switch component renders the respective component only if ***path matches*** otherwise it displays 404 page

SPA

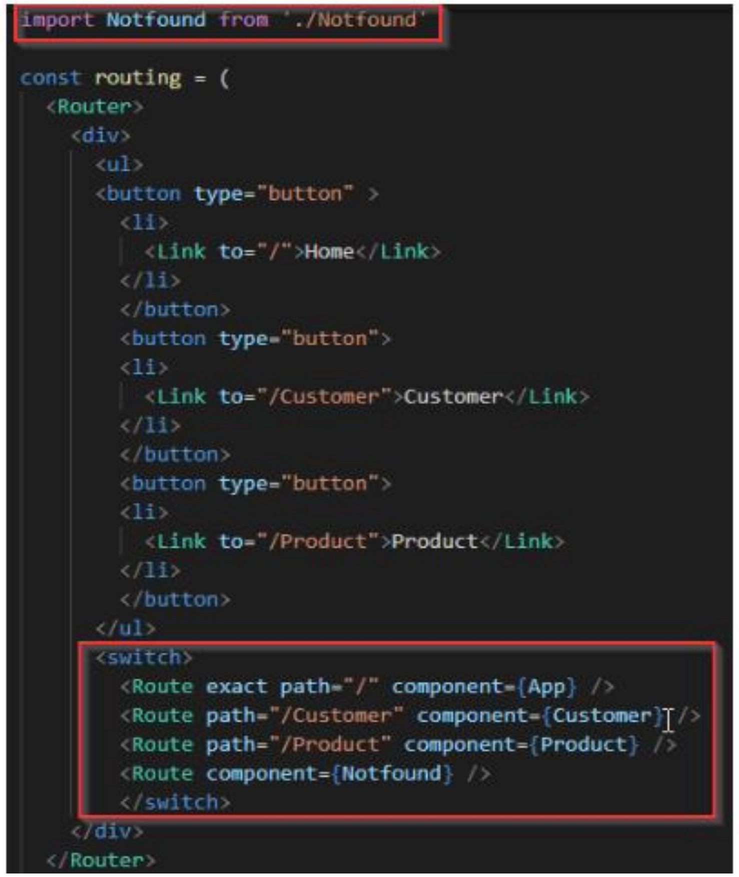
Configure 404 Page

1 Create a component called Notfound



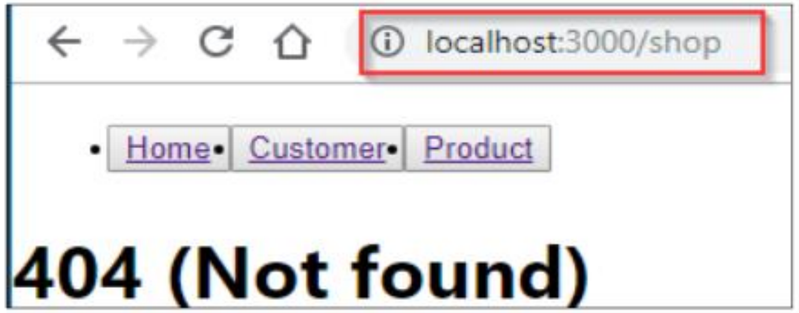
```
import React from 'react'

const Notfound = () => <h1>404 (Not found) </h1>

export default Notfound
```

2 Add Switch component to index.js



```
import Notfound from './Notfound'

const routing = (
  <Router>
    <div>
      <ul>
      <button type="button" >
        <li>
          <Link to="/">Home</Link>
        </li>
        </button>
        <button type="button">
        <li>
          <Link to="/Customer">Customer</Link>
        </li>
        </button>
        <button type="button">
        <li>
          <Link to="/Product">Product</Link>
        </li>
        </button>
      </ul>
      <switch>
        <Route exact path="/" component={App} />
        <Route path="/Customer" component={Customer} />
        <Route path="/Product" component={Product} />
        <Route component={Notfound} />
        </switch>
    </div>
  </Router>
```

3 Check the working by entering wrong path



404 (Not found)