React Router

Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application. When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route.

React Router is a standard library system built on top of the React and used to create routing in the React application using React Router Package. It provides the synchronous URL on the browser with data that will be displayed on the web page. It maintains the standard structure and behavior of the application and mainly used for developing single page web applications.

Need of React Router

React Router plays an important role to display multiple views in a single page application. Without React Router, it is not possible to display multiple views in React applications. Most of the social media websites like Facebook, Instagram uses React Router for rendering multiple views.

React Router Installation

React contains three different packages for routing. These are:

1. **react-router:** It provides the core routing components and functions for the React Router applications.
2. **react-router-native:** It is used for mobile applications.
3. **react-router-dom:** It is used for web applications design.

It is not possible to install react-router directly in your application. To use react routing, first, you need to install react-router-dom modules in your application. The below command is used to install react router dom.

```
$ npm install react-router-dom --save
```

Components in React Router

There are two types of router components:

- **<BrowserRouter>:** It is used for handling the dynamic URL.

- **<HashRouter>:** It is used for handling the static request.

Example

Step-1: In our project, we will create two more components along with **App.js**, which is already present.

About.js

```
import React from 'react'
class About extends React.Component {
  render() {
    return <h1>About</h1>
  }
}
export default About
```

Contact.js

```
import React from 'react'
class Contact extends React.Component {
  render() {
    return <h1>Contact</h1>
  }
}
export default Contact
```

App.js

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

```
  }
}
export default App
```

Step-2: For Routing, open the index.js file and import all the three component files in it. Here, you need to import line: **import { Route, Link, BrowserRouter as Router } from 'react-router-dom'** which helps us to implement the Routing. Now, our index.js file looks like below.

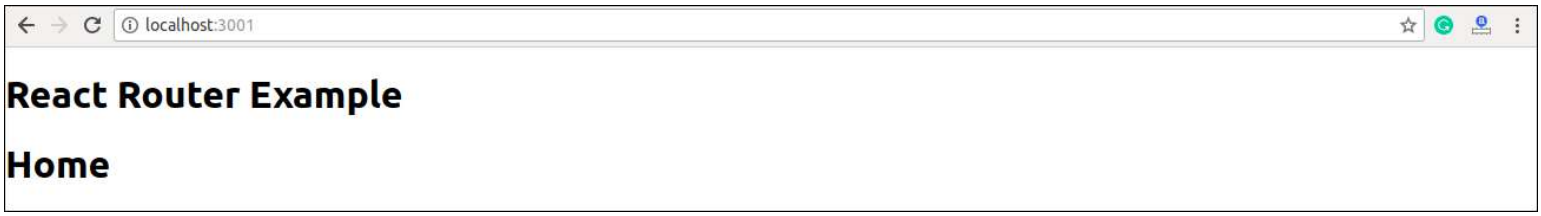
What is Route?

It is used to define and render component based on the specified path. It will accept components and render to define what should be rendered.

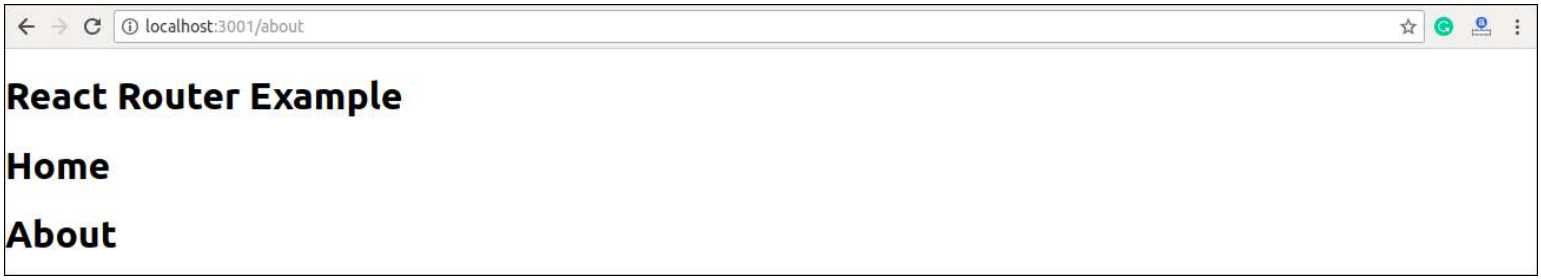
Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Route, Link, BrowserRouter as Router } from 'react-router-dom'
import './index.css';
import App from './App';
import About from './about'
import Contact from './contact'

const routing = (
  <Router>
    <div>
      <h1>React Router Example</h1>
      <Route path="/" component={App} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'));
```

Step-3: Open **command prompt**, go to your project location, and then type **npm start**. You will get the following screen.



Now, if you enter **manually** in the browser: **localhost:3000/about**, you will see **About** component is rendered on the screen.



Step-4: In the above screen, you can see that **Home** component is still rendered. It is because the home path is '**/**' and about path is '**/about**', so you can observe that **slash** is common in both paths which render both components. To stop this behavior, you need to use the **exact** prop. It can be seen in the below example.

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Route, Link, BrowserRouter as Router } from 'react-router-dom'
import './index.css';
import App from './App';
import About from './about'
import Contact from './contact'

const routing = (
  <Router>
    <div>
      <h1>React Router Example</h1>
      <Route exact path="/" component={App} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
```

```
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'));
```

Output

Adding Navigation using Link component

Sometimes, we want to need **multiple** links on a single page. When we click on any of that particular **Link**, it should load that page which is associated with that path without **reloading** the web page. To do this, we need to import **<Link>** component in the **index.js** file.

What is < Link> component?

This component is used to create links which allow to **navigate** on different **URLs** and render its content without reloading the webpage.

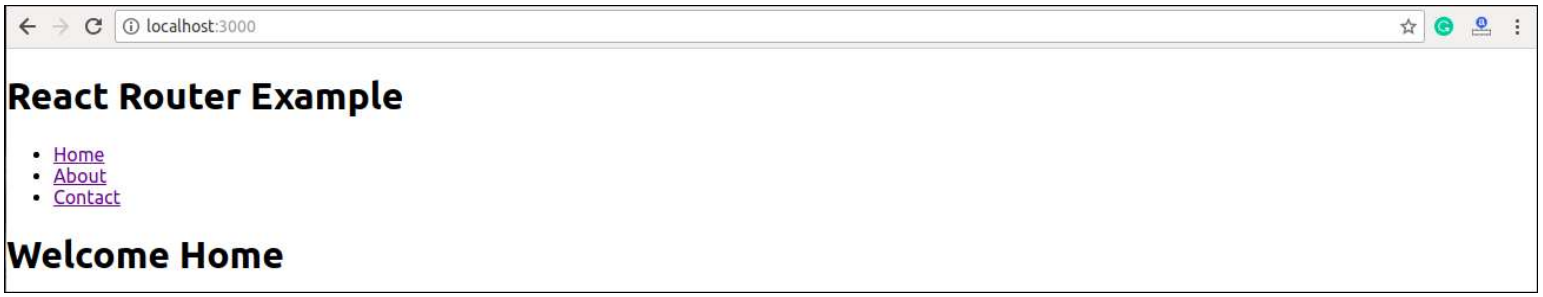
Example

Index.js

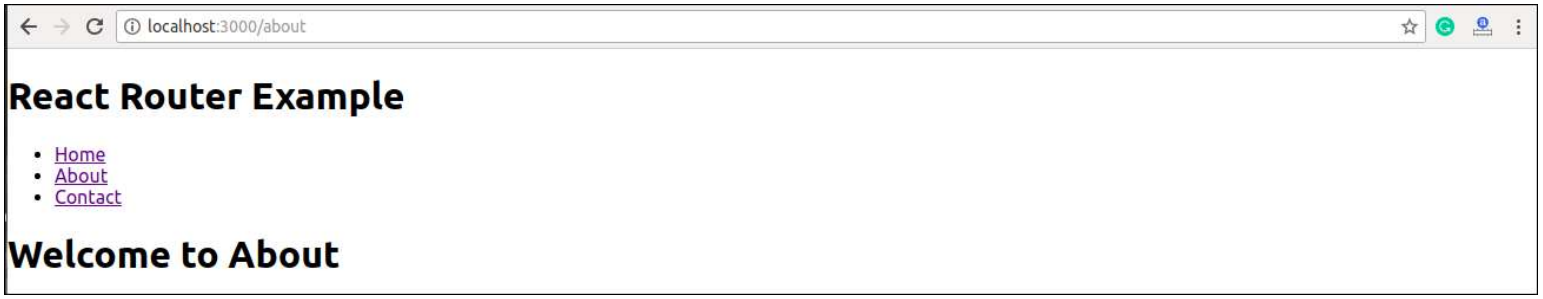
```
import React from 'react';
import ReactDOM from 'react-dom';
import { Route, Link, BrowserRouter as Router } from 'react-router-dom'
import './index.css';
import App from './App';
import About from './about'
import Contact from './contact'

const routing = (
  <Router>
    <div>
```

```
      <h1>React Router Example</h1>
      <ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/about">About</Link>
        </li>
        <li>
          <Link to="/contact">Contact</Link>
        </li>
      </ul>
      <Route exact path="/" component={App} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'));
```

Output



After adding Link, you can see that the routes are rendered on the screen. Now, if you click on the **About**, you will see URL is changing and About component is rendered.



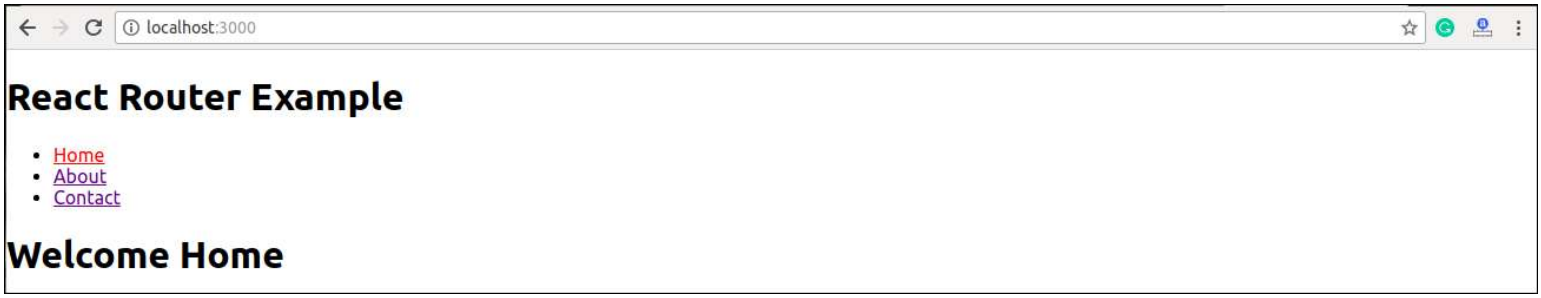
Now, we need to add some **styles** to the Link. So that when we click on any particular link, it can be easily **identified** which Link is **active**. To do this react router provides a new trick **NavLink** instead of **Link**. Now, in the index.js file, replace Link from Navlink and add properties **activeStyle**. The activeStyle properties mean when we click on the Link, it should have a specific style so that we can differentiate which one is currently active.

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Link, NavLink } from 'react-router-dom'
import './index.css';
import App from './App';
import About from './about'
import Contact from './contact'

const routing = (
  <Router>
    <div>
      <h1>React Router Example</h1>
      <ul>
        <li>
          <NavLink to="/" exact activeStyle={
            {color:'red'}
          }>Home</NavLink>
        </li>
        <li>
          <NavLink to="/about" exact activeStyle={
            {color:'green'}
          }>About</NavLink>
        </li>
        <li>
          <NavLink to="/contact" exact activeStyle={
            {color:'magenta'}
          }>Contact</NavLink>
        </li>
      </ul>
      <Route exact path="/" component={App} />
```

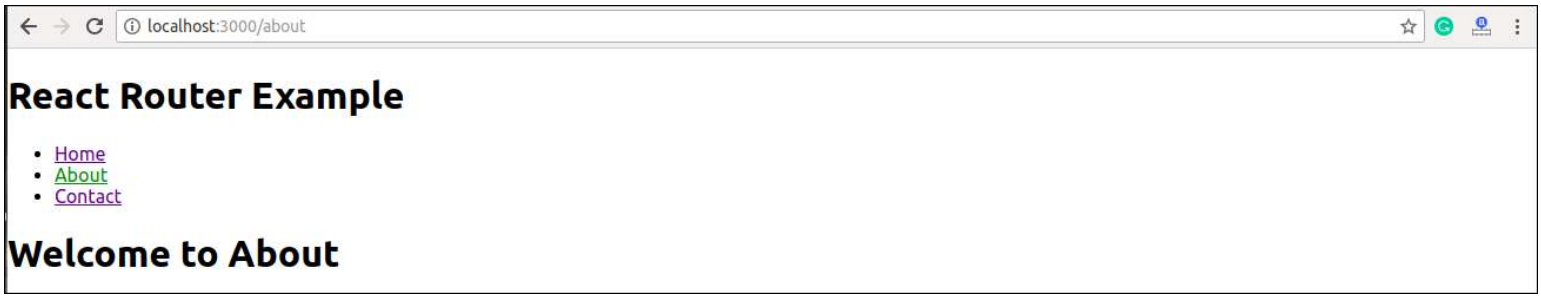
```
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'));
```

Output

When we execute the above program, we will get the following screen in which we can see that **Home** link is of color **Red** and is the only currently **active** link.



Now, when we click on **About** link, its color shown **green** that is the currently **active** link.



<Link> vs <NavLink>

The Link component allows navigating the different routes on the websites, whereas NavLink component is used to add styles to the active routes.

React Router Switch

The <**Switch**> component is used to render components only when the path will be **matched**. Otherwise, it returns to the **not found** component.

To understand this, first, we need to create a **notfound** component.

notfound.js

```
import React from 'react'
const Notfound = () => <h1>Not found</h1>
export default Notfound
```

Now, import component in the index.js file. It can be seen in the below code.

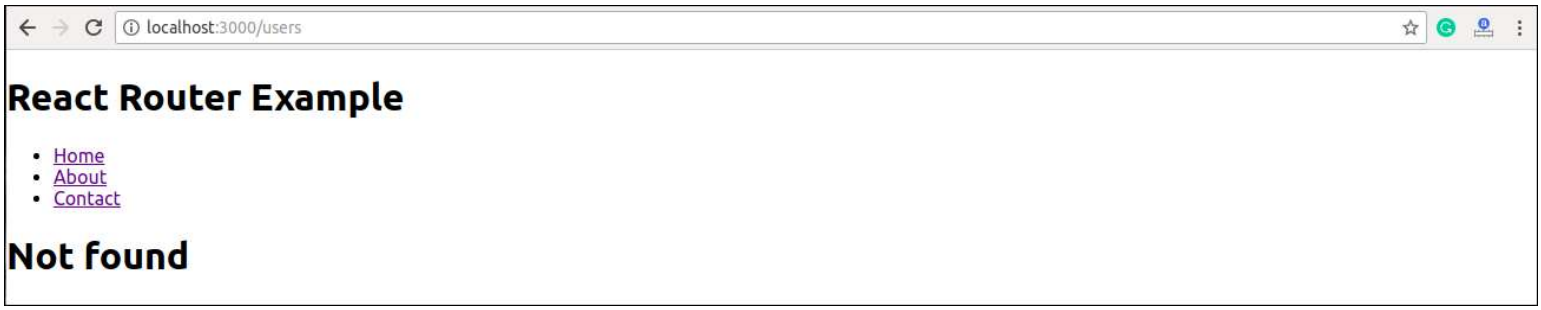
Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Link, NavLink, Switch } from 'react-router-dom'
import './index.css';
import App from './App';
import About from './about'
import Contact from './contact'
import Notfound from './notfound'

const routing = (
  <Router>
    <div>
      <h1>React Router Example</h1>
      <ul>
        <li>
          <NavLink to="/" exact activeStyle={
            {color:'red'}
          }>Home</NavLink>
        </li>
        <li>
          <NavLink to="/about" exact activeStyle={
            {color:'green'}
          }>About</NavLink>
        </li>
        <li>
          <NavLink to="/contact" exact activeStyle={
            {color:'magenta'}
```

```
        }>Contact</NavLink>
        </li>
      </ul>
      <Switch>
        <Route exact path="/" component={App} />
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
        <Route component={Notfound} />
      </Switch>
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'));
```

Output

If we manually enter the **wrong** path, it will give the not found error.



React Router <Redirect>

A <Redirect> component is used to redirect to another route in our application to maintain the old URLs. It can be placed anywhere in the route hierarchy.

Nested Routing in React

Nested routing allows you to render **sub-routes** in your application. It can be understood in the below example.

Example

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Link, NavLink, Switch } from 'react-router-dom'
import './index.css';
import App from './App';
import About from './about'
import Contact from './contact'
import Notfound from './notfound'

const routing = (
  <Router>
    <div>
      <h1>React Router Example</h1>
      <ul>
        <li>
          <NavLink to="/" exact activeStyle={
            {color:'red'}
          }>Home</NavLink>
        </li>
        <li>
          <NavLink to="/about" exact activeStyle={
            {color:'green'}
          }>About</NavLink>
        </li>
        <li>
          <NavLink to="/contact" exact activeStyle={
            {color:'magenta'}
          }>Contact</NavLink>
        </li>
      </ul>
      <Switch>
        <Route exact path="/" component={App} />
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
        <Route component={Notfound} />
      </Switch>
```

```
    </div>
  </Router>
)
ReactDOM.render(routing, document.getElementById('root'));
```

In the **contact.js** file, we need to import the **React Router** component to implement the **subroutes**.

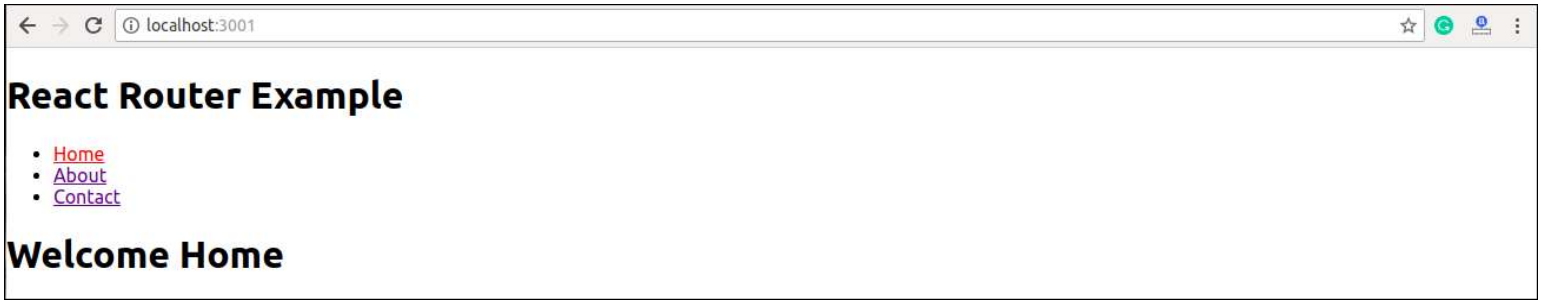
contact.js

```
import React from 'react'
import { Route, Link } from 'react-router-dom'

const Contacts = ({ match }) => <p>{match.params.id}</p>

class Contact extends React.Component {
  render() {
    const { url } = this.props.match
    return (
      <div>
        <h1>Welcome to Contact Page</h1>
        <strong>Select contact Id</strong>
        <ul>
          <li>
            <Link to="/contact/1">Contacts 1 </Link>
          </li>
          <li>
            <Link to="/contact/2">Contacts 2 </Link>
          </li>
          <li>
            <Link to="/contact/3">Contacts 3 </Link>
          </li>
          <li>
            <Link to="/contact/4">Contacts 4 </Link>
          </li>
        </ul>
        <Route path="/contact/:id" component={Contacts} />
```

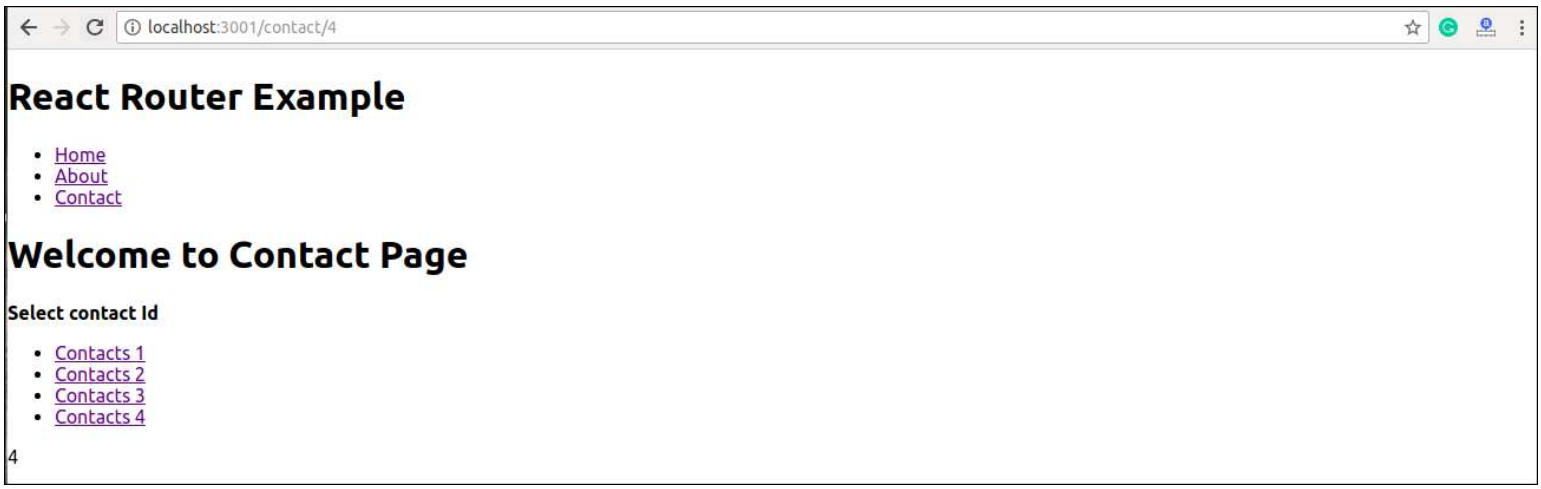
```
      </div>
    )
  }
}
export default Contact
```

Output

When we execute the above program, we will get the following output.



After clicking the **Contact** link, we will get the contact list. Now, selecting any contact, we will get the corresponding output. It can be shown in the below example.



Benefits Of React Router

The benefits of React Router is given below:

- In this, it is not necessary to set the browser history manually.
- Link uses to navigate the internal links in the application. It is similar to the anchor tag.
- It uses Switch feature for rendering.
- The Router needs only a Single Child element.
- In this, every component is specified in .