## Main Differences Between Absolute Path and Relative Path

* Absolute paths describe the location of a file from its root directory. Whereas, a relative path describes the location of a file related to its current working directory.
* Absolute URLs use absolute paths and relative URls use relative paths.
* Absolute URLs are used to link to pages which are not on the same domain. On the other hand, relative URls are generally used to link to pages that are on the same domain.
* Absolute URls generally begin with delimiting characters such as “/”, “\”, “,”, “:” etc.  
  Relative URLs do not begin with delimiting characters.
* Absolute Paths are also known as full paths or file paths. Relative paths are also known as non-absolute paths.
* / is the root folder of the filesystem.
* ./ usually denotes the current folder that your program or script is in, usually the same one with the file you run.
* ../ denotes the folder above the current one.
* A relative URL on a Linux system looks like ./public\_html/cgi-bin
* An absolute path on a Linux system looks like /home/users/c/computerhope/public\_html/cgi-bin

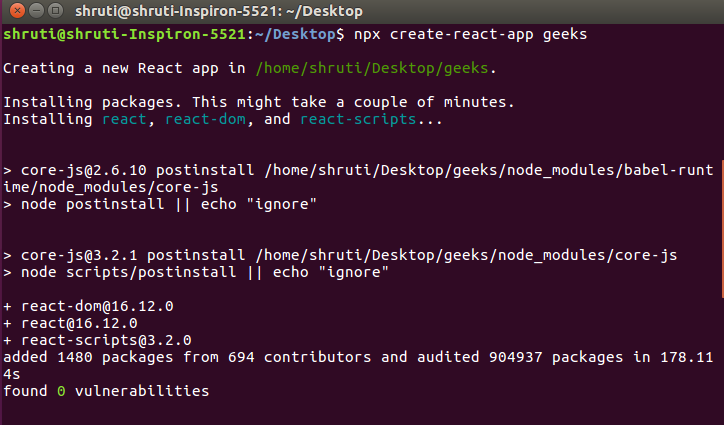
ROUTING in React JS

**React Router is a standard library for routing in React.** It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.  
Let us create a simple application to React to understand how the React Router works. The application will contain three components: home component, about a component, and contact component. We will use React Router to navigate between these components.

**Setting up the React Application:** Create a React application using [create-react-app](https://www.geeksforgeeks.org/reactjs-setting-development-environment/) and lets call it **geeks**.

**Note:** If you’ve previously installed create-react-app globally via npm, directly use the command below:

***npx create-react-app geeks***

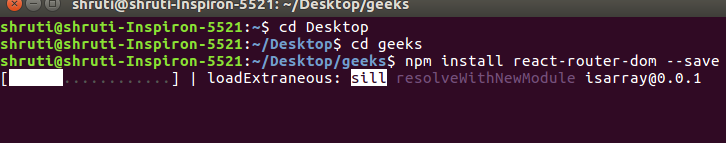
  
Your development environment is ready. Let us now install React Router in our Application.

**Installing React Router:** React Router can be installed via [npm](https://www.geeksforgeeks.org/node-js-npm-node-package-manager/) in your React application. Follow the steps given below to install Router in your React application:

**Step 1:** [cd](https://www.geeksforgeeks.org/cd-command-in-linux-with-examples/) into your project directory i.e **geeks**.

**Step 2:** To install the React Router use the following command:

***npm install – -save react-router-dom or npm i react-router-dom  
(its a updated command)***

  
After installing react-router-dom, add its components to your React application.

**Adding React Router Components:** The main Components of React Router are:

**BrowserRouter:** It is the parent component that is used to store all of the other components.

**Routes:** It’s a new component introduced in the v6 and a upgrade of the component.

Routes are chosen based on the best match instead of being traversed in order.

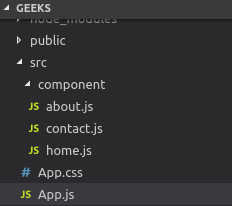
**Route:** Route is the conditionally shown component that **renders some UI when its path matches the current URL.**

**Link:** Link component is used to create links to different routes and implement navigation around the application. It works like HTML [anchor tag](https://www.geeksforgeeks.org/html-a-tag/).

To add React Router components in your application, open your project directory in the editor you use and go to **app.js** file. Now, add the below given code in app.js.

|  |
| --- |
| import {      BrowserRouter as Router,      Routes,      Route,      Link  } from 'react-router-dom'; |

**Note:** BrowserRouter is aliased as Router.

**Using React Router:** To use React Router, let us first create few components in the react application. In your project directory, create a folder named **component** inside the src folder and now add 3 files named **home.js**, **about.js** and **contact.js** to the component folder.  
  
Let us add some code to our 3 components:

**Home.js:**

|  |
| --- |
| import React from 'react';    **function** Home (){  **return** <h1>Welcome to the world of Geeks!</h1>  }    export **default** Home; |

**About.js:**

|  |
| --- |
| import React from 'react';    **function** About () {  **return** <div>          <h2>GeeksforGeeks is a computer science portal **for** geeks!</h2>            Read more about us at :          <a href="<https://www.geeksforgeeks.org/about/>">              https://www.geeksforgeeks.org/about/          </a>      </div>  }  export **default** About; |

**Contact.js:**

|  |
| --- |
| import React from 'react';    **function** Contact (){  **return** <address>              You can find us here:<br />              GeeksforGeeks<br />              5th & 6th Floor, Royal Kapsons, A- 118, <br />              Sector- 136, Noida, Uttar Pradesh (201305)          </address>  }    export **default** Contact; |

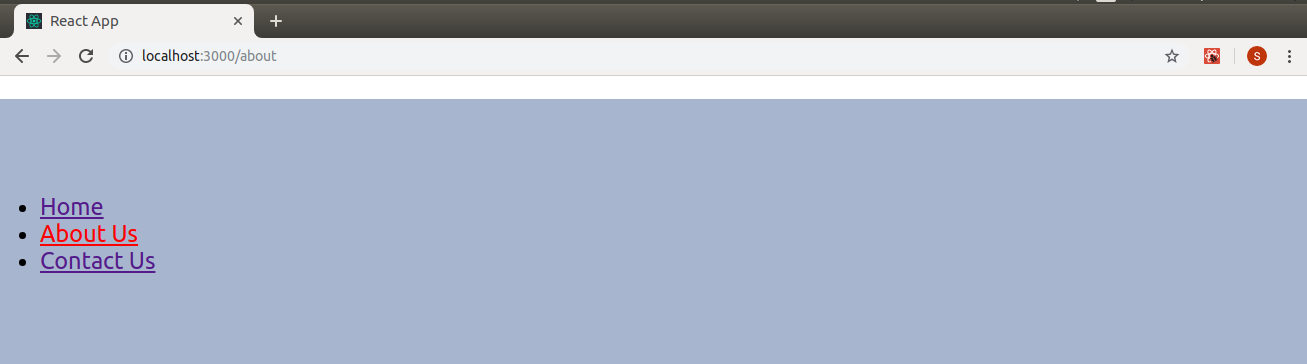
Now, let us include React Router components to the application:

**BrowserRouter:** Add BrowserRouter aliased as Router to your app.js file in order to wrap all the other components. BrowserRouter is a parent component and can have only single child.

|  |
| --- |
| class App extends Component {    render() {  **return** (         <Router>             <div className="App">             </div>         </Router>     );    }  } |

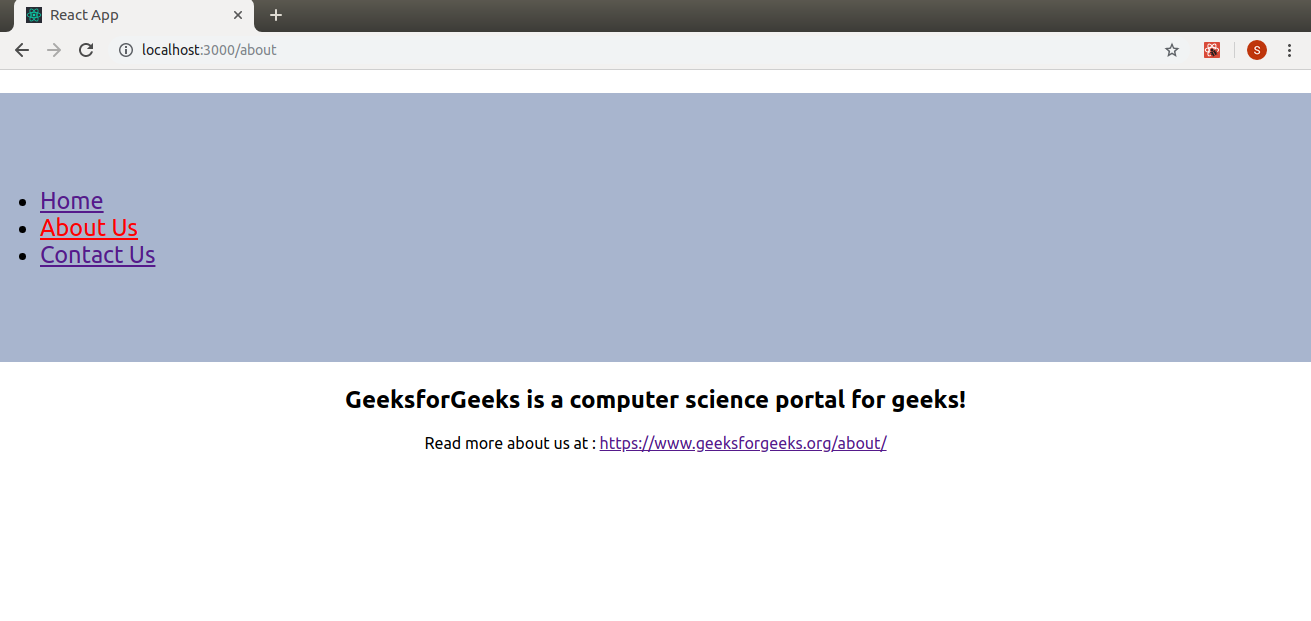
**Link:** Let us now create links to our components. Link component uses the **to** prop to describe the location where the links should navigate to.

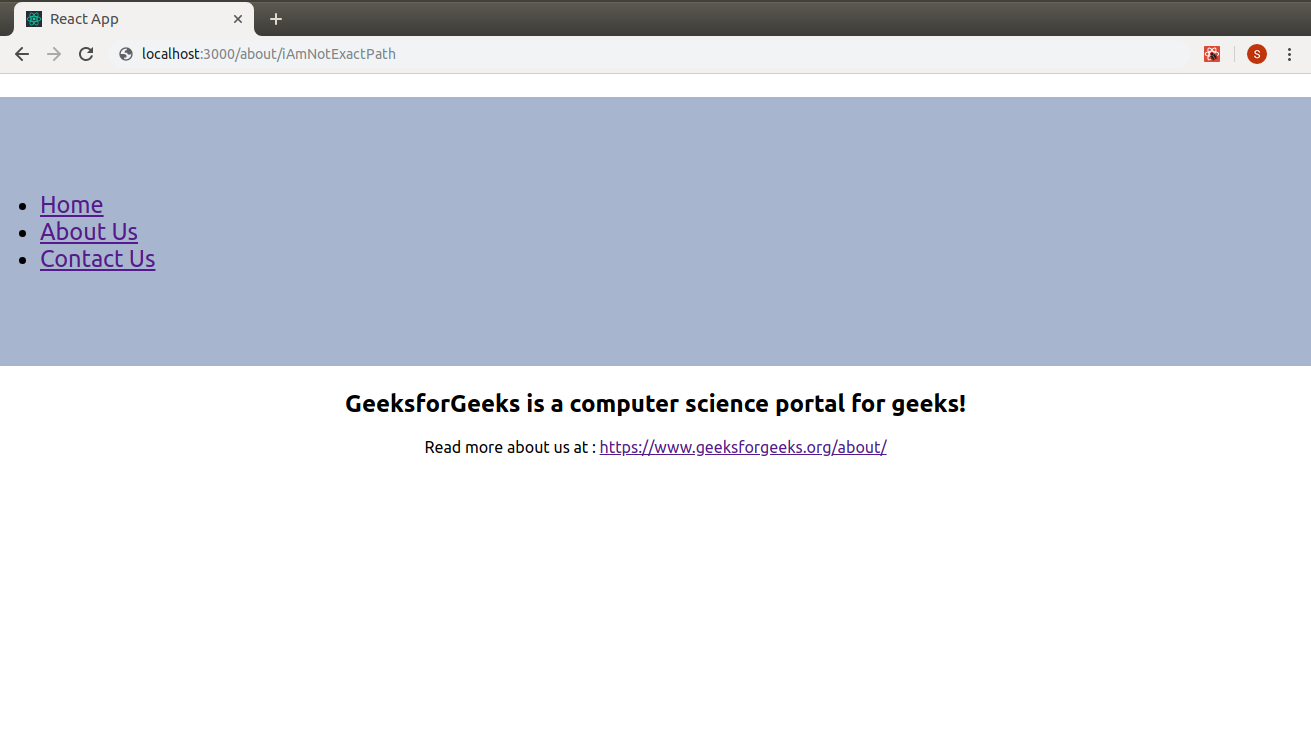
|  |
| --- |
| <div className="App">      <ul>        <li>          <Link to="/">Home</Link>        </li>        <li>          <Link to="/about">About Us</Link>        </li>        <li>          <Link to="/contact">Contact Us</Link>        </li>      </ul>  </div> |

Now, run your application on the local host and click on the links you created. You will notice the url changing according the value in **to** props of the Link component.  


**Route:** Route component will now help us to establish the link between component’s UI and the URL. To include routes to the application, add the code give below to your app.js.

|  |
| --- |
| <Route exact path='/' element={< Home />}></Route>  <Route exact path='/about' element={< About />}></Route>  <Route exact path='/contact' element={< Contact />}></Route> |

Components are linked now and clicking on any link will render the component associated with it.  
  
Let us now try to understand the props associated with the Route component.

**1.exact:** It is used to match the exact value with the URL. For Eg., exact path=’/about’ will only render the component if it exactly matches the path but if we remove exact from the syntax, then UI will still be rendered even if the structure is like /about/10.  


**2. path:** Path specifies a pathname we assign to our component.

**3. element:** It refers to the component which will render on matching the path.

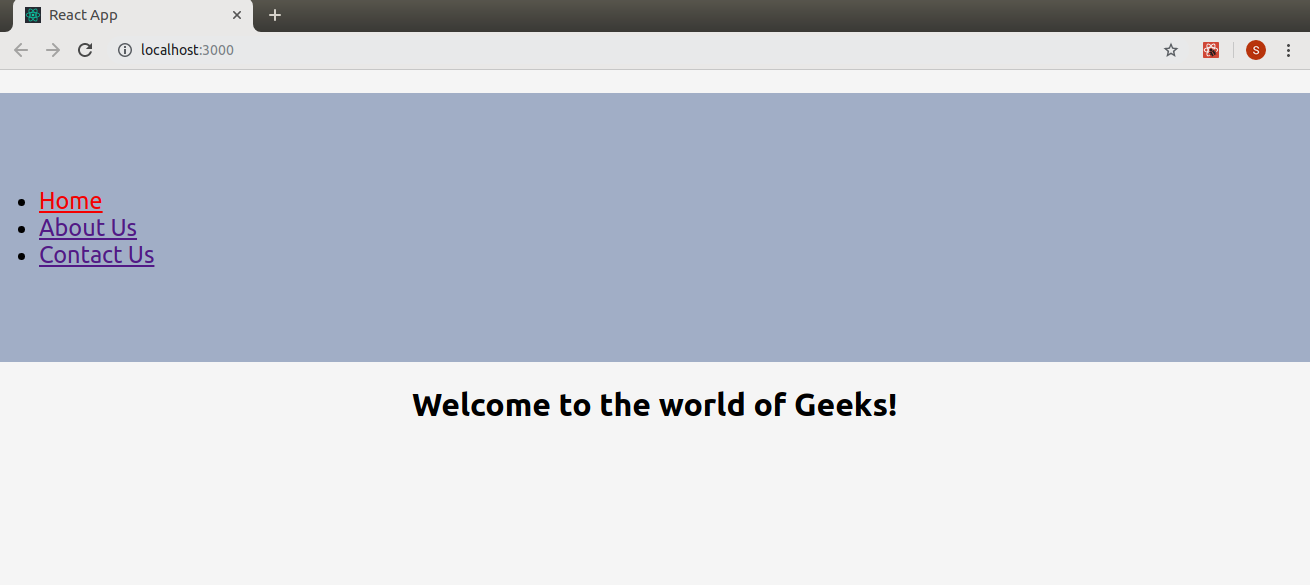
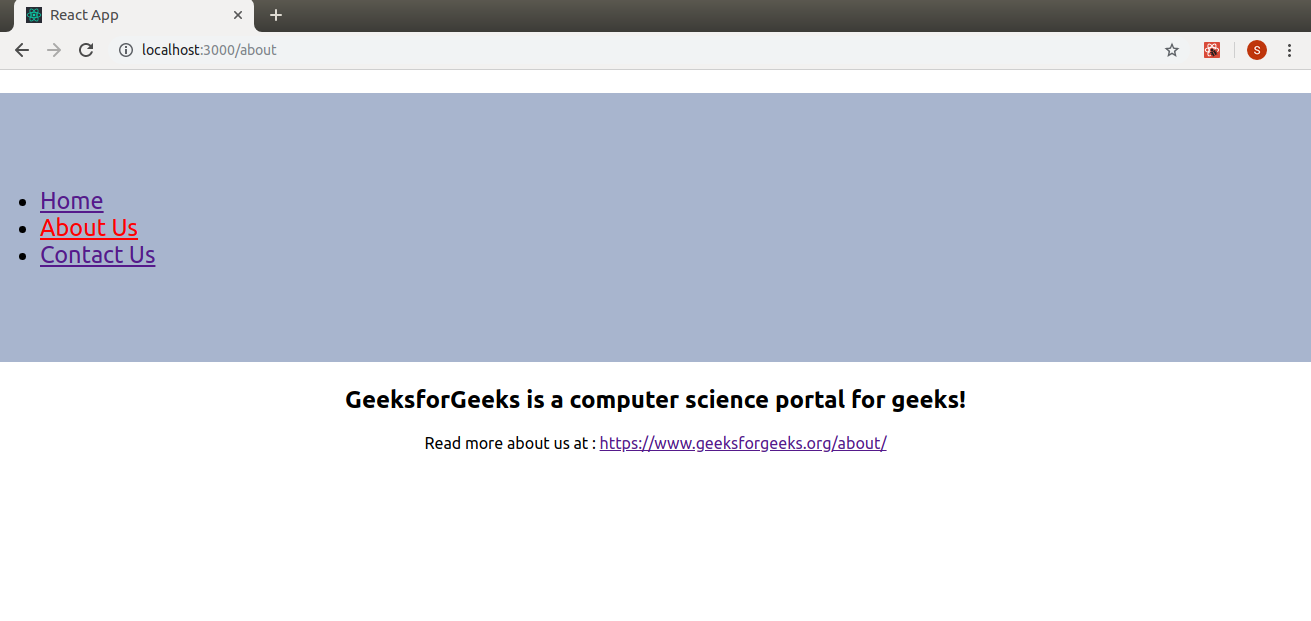
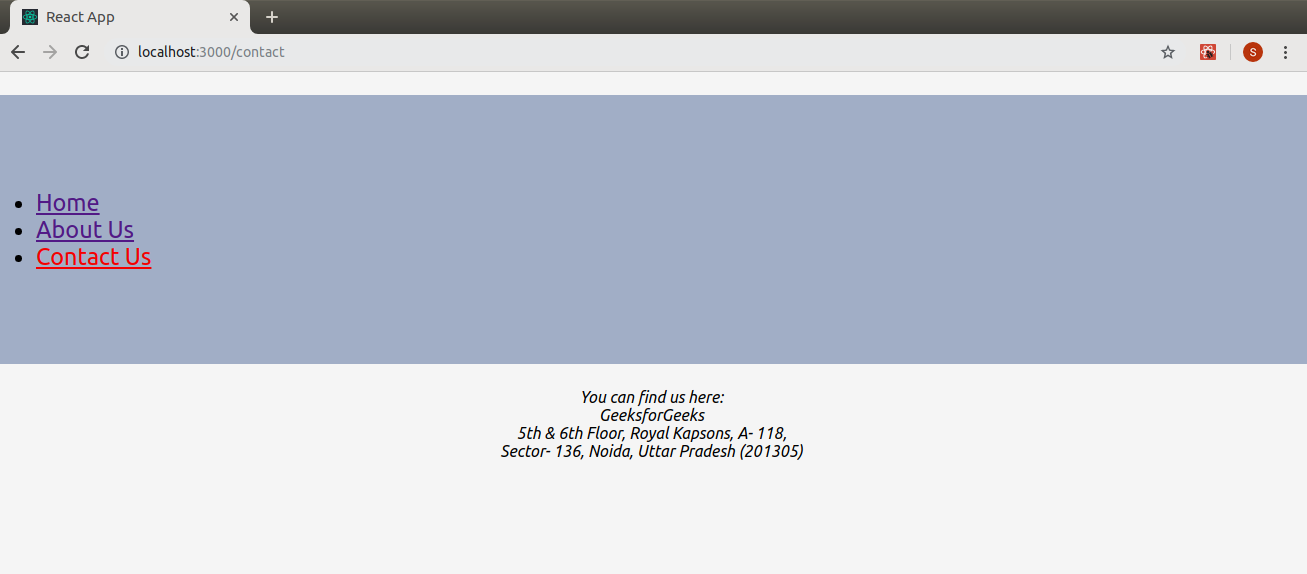
**Note:** By default, routes are inclusive which means more than one Route component can match the URL path and render at the same time. If we want to render a single component, we need to use **routes**.

**Routes:** To render a single component, wrap all the routes inside the Routes Component.

|  |
| --- |
| <Routes>      <Route exact path='/' element={< Home />}></Route>      <Route exact path='/about' element={< About />}></Route>      <Route exact path='/contact' element={< Contact />}></Route>  </Routes> |

In previous versions Switch groups together several routes, iterates over them and finds the first one that matches the path. Thereby, the corresponding component to the path is rendered.Now we use Routes.

|  |  |
| --- | --- |
| import React, { Component } from 'react';  import { BrowserRouter as Router,Routes, Route, Link } from 'react-router-dom';  import Home from './component/home';  import About from './component/about';  import Contact from './component/contact';  import './App.css';    class App extends Component {    render() {  **return** (         <Router>             <div className="App">              <ul className="App-header">                <li>                  <Link to="/">Home</Link>                </li>                <li>                  <Link to="/about">About Us</Link>                </li>                <li>                  <Link to="/contact">Contact Us</Link>                </li>              </ul>             <Routes>                   <Route exact path='/' element={< Home />}></Route>                   <Route exact path='/about' element={< About />}></Route>                   <Route exact path='/contact' element={< Contact />}></Route>            </Routes>            </div>         </Router>     );    }  }    export **default** App; |  |

Now, we you can click on the links and navigate to different components. React Router keeps your application UI in sync with the URL.  
  
  
  
Finally, we have successfully implemented navigation in our React application using React Router.

# ReactJS | Types of Routers

On the basis of the part of URL that the router will use to track the content that the user is trying to view, React Router provides three different kinds of routers:

1. [Memory Router](https://www.geeksforgeeks.org/reactjs-types-of-routers/#memory)
2. [Browser Router](https://www.geeksforgeeks.org/reactjs-types-of-routers/#browser)
3. [Hash Router](https://www.geeksforgeeks.org/reactjs-types-of-routers/#hash)

**Pre-requisite:** Before start this article you need to have basic knowledge of [React Router](https://www.geeksforgeeks.org/reactjs-router/).

**Memory Router:** Memory router keeps the URL changes in memory not in the user browsers. It keeps the history of the URL in memory and it does not read or write to the address bar so the user can not use the browser’s back button as well as the forward button. It doesn’t change the URL in your browser. It is very useful for testing and non-browser environments like React Native.

**Syntax:**

import { MemoryRouter as Router } from 'react-router-dom';

**Program:**

|  |
| --- |
| import React, { Component } from 'react';  import { MemoryRouter as Router, Route, Link, Switch }                      from 'react-router-dom';  import Home from './component/home';  import About from './component/about';  import Contact from './component/contact';  import './App.css';    class App extends Component {    render() {  **return** (          <Router>              <div >                  <ul >                    <li>                        <Link to="/">Home</Link>                    </li>                    <li>                        <Link to="/about">                          About Us                        </Link>                    </li>                    <li>                        <Link to="/contact">                          Contact Us                        </Link>                    </li>                  </ul>                  <Switch>                    <Route exact path='/'                        component={Home}>                    </Route>                    <Route exact path='/about'                        component={About}>                    </Route>                    <Route exact path='/contact'                        component={Contact}>                    </Route>                  </Switch>              </div>          </Router>      );    }  }    export **default** App; |

**Output:**



**Browser Router:** It uses HTML 5 history API (i.e. pushState, replaceState and popState API) to keep your UI in sync with the URL. It routes as a normal URL in the browser and assumes that the server is handling all the request URL (eg., /, /about) and points to root index.html. It accepts forceRefresh props to support legacy browsers which doesn’t support HTML 5 pushState API

**Syntax:**

**import { BrowserRouter as Router } from 'react-router-dom';**

**Program:**

|  |
| --- |
| import React, { Component } from 'react';  import { BrowserRouter as Router, Route, Link, Switch }                    from 'react-router-dom';    import Home from './component/home';  import About from './component/about';  import Contact from './component/contact';  import './App.css';    class App extends Component {    render() {  **return** (          <Router>              <div>                  <ul >                    <li>                        <Link to="/">Home</Link>                    </li>                    <li>                        <Link to="/about">About Us</Link>                    </li>                    <li>                        <Link to="/contact">                          Contact Us                        </Link>                    </li>                  </ul>                  <Switch>                    <Route exact path='/'                        component={Home}>                    </Route>                    <Route exact path='/about'                        component={About}>                    </Route>                    <Route exact path='/contact'                        component={Contact}>                    </Route>                  </Switch>              </div>          </Router>      );    }  }    export **default** App; |

**Output:**



**Hash Router:** Hash router uses client-side hash routing. It uses the hash portion of the URL (i.e. window.location.hash) to keep your UI in sync with the URL. Hash portion of the URL won’t be handled by the server, the server will always send the index.html for every request and ignore the hash value. It doesn’t need any configuration in the server to handle routes. It is used to support legacy browsers which usually don’t support HTML pushState API. It is very useful for the legacy browsers or you don’t have a server logic to handle the client-side. This route isn’t recommended to be used by the react-router-dom team.

**Syntax:**

import { HashRouter as Router } from 'react-router-dom';

**Program:**

|  |
| --- |
| import React, { Component } from 'react';  import { HashRouter as Router, Route, Link, Switch }                from 'react-router-dom';  import Home from './component/home';  import About from './component/about';  import Contact from './component/contact';  import './App.css';    class App extends Component {    render() {  **return** (          <Router>              <div >                  <ul >                    <li>                        <Link to="/">Home</Link>                    </li>                    <li>                        <Link to="/about">About Us</Link>                    </li>                    <li>                        <Link to="/contact">                            Contact Us                        </Link>                    </li>                  </ul>                  <Switch>                    <Route exact path='/'                        component={Home}>                    </Route>                    <Route exact path='/about'                        component={About}>                    </Route>                    <Route exact path='/contact'                        component={Contact}>                    </Route>                  </Switch>              </div>          </Router>      );    }  }    export **default** App; |

**Syntax:**

