**React Router**

**ReactJS Router is a library for handling navigation and routing in ReactJS applications**. It allows you to create a seamless user experience by mapping various URLs to components, enabling single-page application (SPA) navigation without refreshing the entire page.

This article will walk you through the basics of ReactJS Router, its features, the installation process, and how to implement routing in a React application.

**Table of Content**

* [What is React Router?](https://www.geeksforgeeks.org/reactjs-router/#what-is-a-react-router)
* [Features of React Router](https://www.geeksforgeeks.org/reactjs-router/#features-of-react-router)
* [Components of React Router](https://www.geeksforgeeks.org/reactjs-router/#components-of-react-router)
* [Uses of React Router](https://www.geeksforgeeks.org/reactjs-router/#react-router-components)
* [React Router – FAQ’s](https://www.geeksforgeeks.org/reactjs-router/#faqs-react-router)

**What is React Router?**

**ReactJS Router is a standard library for routing and navigation in**[**React applications**](https://www.geeksforgeeks.org/react-tutorial/)**.** It allows you to manage navigation in your app by defining routes that connect the URL paths to specific components.

With React Router, you can implement different views for different parts of your application without the need for a full-page refresh. This is a key feature of [single-page applications (SPAs)](https://www.geeksforgeeks.org/what-is-single-page-application/), where only the necessary content is updated as the user navigates.

The current latest verstion is [**React router dom v6**](https://www.geeksforgeeks.org/what-is-react-router-dom/)**.**

**Features of React Router**

* **Declarative Routing:** React Router v6 uses the Routes and Route components to define routes declaratively, making the routing configuration simple and easy to read.
* **Nested Routes:** It supports nested routes, allowing for complex and hierarchical routing structures, which helps in organizing the application better.
* **Programmatic Navigation:** The useNavigate hook enables programmatic navigation, allowing developers to navigate between routes based on certain conditions or user actions.
* **Route Parameters:** It provides dynamic routing with route parameters, enabling the creation of routes that can match multiple URL patterns.
* **Improved TypeScript Support:** Enhanced TypeScript support ensures that developers can build type-safe applications, improving development efficiency and reducing errors.

For a detailed exploration of routing in React, check out the guide on [ReactJS Course](https://gfgcdn.com/tu/St9/). This resource offers valuable insights into setting up React Router, defining routes, and utilizing components like <Route>, <Switch>, and <Link>. These components make it easy to implement navigation within your application, ensuring that users can move effortlessly from one view to another.

**Components of React Router**

React Router mainly comprises of the below components

**1. BrowserRouter and HashRouter**

* **BrowserRouter**: Uses the HTML5 history API to keep your UI in sync with the URL.
* **HashRouter**: Uses the hash portion of the URL (i.e., window.location.hash) to keep your UI in sync with the URL.

<BrowserRouter>  
 (/\* Your routes go here \*/}  
</BrowserRouter>

**2. Routes and Route**

* **Routes**: A container for all your route definitions.
* **Route**: Defines a single route with a path and the component to render.

<Routes>  
 <Route path="/" element={<Home />} />  
 <Route path="/about" element={<About />} />  
</Routes>

**3.**[Link and NavLink](https://www.geeksforgeeks.org/link-and-navlink-components-in-react-router-dom/)

* **Link**: Creates navigational links in your application.
* **NavLink**: Similar to Link but provides additional styling attributes when the link is active.

<nav>  
 <NavLink to="/" activeClassName="active">Home</NavLink>  
 <Link to="/about">About</Link>  
</nav>

**Steps to Implement React Router**

**Step 1: Initialize React Project**

Create React application using the following command.

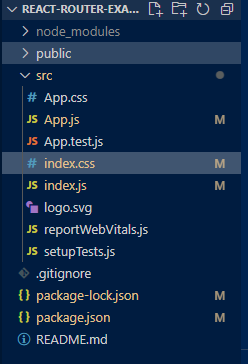
npx create-react-app react-router-example  
cd react-router-example

**Step 2: Install React Router**

Install react-router in your application write the following command in your terminal

npm install react-router-dom@6

**Project Structure:**



*Folder Structure*

**Dependencies**

"dependencies": {  
 "@testing-library/jest-dom": "^5.17.0",  
 "@testing-library/react": "^13.4.0",  
 "@testing-library/user-event": "^13.5.0",  
 "react": "^18.3.1",  
 "react-dom": "^18.3.1",  
 **"react-router-dom": "^6.24.1",**  
 "react-scripts": "5.0.1",  
 "web-vitals": "^2.1.4"  
}

**Example:**This example demonstrates implemeting basic routes in a React App.

CSSJavaScriptJavaScript

*// src/index.js*

**import** React **from** "react";

**import** ReactDOM **from** "react-dom/client";

**import** "./index.css";

**import** App **from** "./App";

**const** root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

<React.StrictMode>

<App />

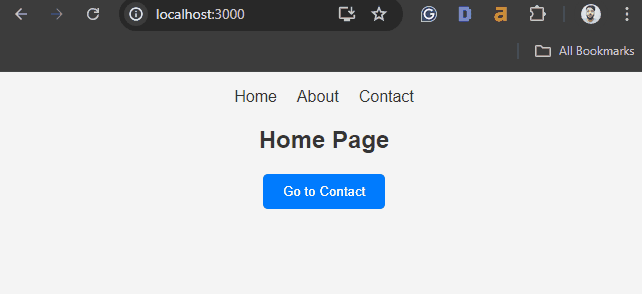
</React.StrictMode>

);

**Step 4:**Run the application using the following command.

npm start

**Output:**



*React Router*

**Uses of React Router**

1. **Navigation and Routing:** React Router provides a declarative way to navigate between different views or pages in a React application. It allows users to switch between views without refreshing the entire page.
2. **Dynamic Routing:** React Router supports dynamic routing, which means routes can change based on the application’s state or data, making it possible to handle complex navigation scenarios.
3. **URL Management:** React Router helps manage the URLs in your application, allowing for deep linking, bookmarkable URLs, and maintaining the browser’s history stack.
4. **Component-Based Approach:** Routing is handled through components, making it easy to compose routes and navigation in a modular and reusable way.