9. React Routing

Last updated by | Subramanya Dixit | May 5, 2025 at 7:59 PM GMT+5:30

📘 Why React Router Is Important

React Router allows you to add navigation to your single-page applications. It helps in managing URLs and rendering specific components based on those URLs without reloading the page, providing a smoother user experience.

🧠 Key Concepts

✅ React Router Basics

React Router provides a collection of components to handle routing in a React app. The core components include:

- **`BrowserRouter`** : The main container for managing routing.
- **`Route`** : Defines a mapping between a URL and a component.
- **`Link`** : Provides navigation between different routes in your app.
- **`Switch`** : Renders only the first `<Route>` or `<Redirect>` that matches the current location.

Example of basic routing setup:

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

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
      </nav>
      <Switch>
        <Route path="/" exact>
          <Home />
        </Route>
        <Route path="/about">
          <About />
        </Route>
      </Switch>
    </Router>
  );
}

function Home() {
  return <h2>Home Page</h2>;
}

function About() {
  return <h2>About Page</h2>;
}
```

✅ Dynamic Routing

React Router supports dynamic routes with URL parameters. These allow you to pass data through the URL.

```
<Route path="/user/:id" component={User} />

function User({ match }) {
  const { id } = match.params;
  return <h2>User ID: {id}</h2>;
}
```

✅ Programmatic Navigation

You can programmatically navigate using the `history` object, which is provided by `useHistory()` hook in React Router.

```
import { useHistory } from 'react-router-dom';

function NavigateButton() {
  const history = useHistory();

  const handleClick = () => {
    history.push("/about");
  };

  return <button onClick={handleClick}>Go to About</button>;
}
```

✅ Redirects

You can redirect users from one route to another using the `<Redirect>` component.

```
import { Redirect } from 'react-router-dom';

function Home() {
  const isAuthenticated = false;
  return (
    <>
      {isAuthenticated ? <h2>Welcome</h2> : <Redirect to="/login" />}
    </>
  );
}
```

✅ Nested Routing

React Router allows nested routes, which means you can render routes inside other routes. This is useful when you want to display a section of a page depending on the URL.

```
<Route path="/dashboard" component={Dashboard}>
  <Route path="/settings" component={Settings} />
</Route>
```

💡 Guidelines

- Use **`BrowserRouter`** as the top-level component for routing.
- **`Link`** is used for navigation, while **`Route`** defines which components to render based on the current path.
- Use **URL parameters** to pass dynamic data in the route.

- For programmatic navigation, use `useHistory` or `useNavigate` in React Router v6 and above.
- For conditional rendering, use `Redirect` to change routes dynamically.

📝 Practice Exercises

1. Set up basic routing with `Route` and `Link`.
2. Implement dynamic routing with URL parameters.
3. Create a component that redirects based on authentication status.
4. Implement nested routes for a more complex layout (e.g., dashboard with sub-routes).

❓ Quiz Questions

1. What is the role of the `<Route>` component in React Router?
a) To create links between different components
b) To define a URL path and associate it with a component
✅ **c) To define the component to render based on the current URL**
d) To manage state across different routes

---

2. How can you programmatically navigate to a different route in React Router?
a) By using `Link`
b) By using `Switch`
✅ **c) By using the `useHistory` or `useNavigate` hook**
d) By using `Redirect`

---

3. How do you handle dynamic routes with URL parameters in React Router?
a) Use a separate `Route` for each parameter
✅ **b) Use `:parameterName` in the `path` prop of `Route`**
c) Use `useParams` inside the parent component
d) Pass parameters directly into the `Link` component
