**React**

**Day 66th**

**How to do Routing**

In react we can do routing in two ways. Here we will be discussing what’s the difference between these ways and which one should we use.

We have already discussed if we want to do routing then we need to import three main things from the react-router-DOM library and these are `**BrowserRouter, Routes, Route**`. Now we will be understanding what these things are and why do we need it.

**🤷🏼‍♀️ BrowserRouter**

BrowserRouter is a router implementation that uses the **HTML5 History API** (pushState, replaceState, and popstate) to keep your UI in sync with the URL. It's commonly used for **Single Page Applications (SPAs)** in React, where page navigation doesn't reload the page but still updates the URL.

**✅ Key Points:**

* **Acts as the parent router component**, wrapping all route components.
* It **uses clean URLs** (e.g., /dashboard instead of /#/dashboard) unlike HashRouter.

**🔄 History API Methods:**

1. **history.pushState()**
   * Adds a new entry to the browser's session history stack.
   * Used when navigating to a new route.
   * Allows the user to go "back" to the previous route.
2. **history.replaceState()**
   * Replaces the current entry in the history stack.
   * Does **not** create a new entry (no effect on the back button).
3. **popstate event**
   * Triggered when the user navigates using the browser’s back/forward buttons.
   * Useful for listening to manual user navigation.

**🧠 Example: Authentication & Token**

Let’s consider Instagram. Without logging in, you can't view content like posts or reels. After logging in, the app stores an **authentication token**, often in localStorage or sessionStorage, or uses cookies. The UI updates to reflect the authenticated state and redirects to the dashboard. This routing transition uses BrowserRouter and programmatic navigation (useNavigate() or Navigate) after login.

**🤷🏼‍♀️ Routes**

In react-router-dom v6+, the <Routes> component **replaces** the older <Switch> component. It is used to define all available route paths.

* It looks through its children <Route> elements and renders the **first one** that matches the current URL.
* It's responsible for route matching and rendering logic.
* Routes **must** be direct children of a Router (like BrowserRouter).

**🤷🏼‍♀️ Route**

The <Route> component is used inside <Routes> to define a path and what component should render for that path.

**Example:**

<Routes>

<Route path="/" element={<Home />} />

<Route path="/login" element={<Login />} />

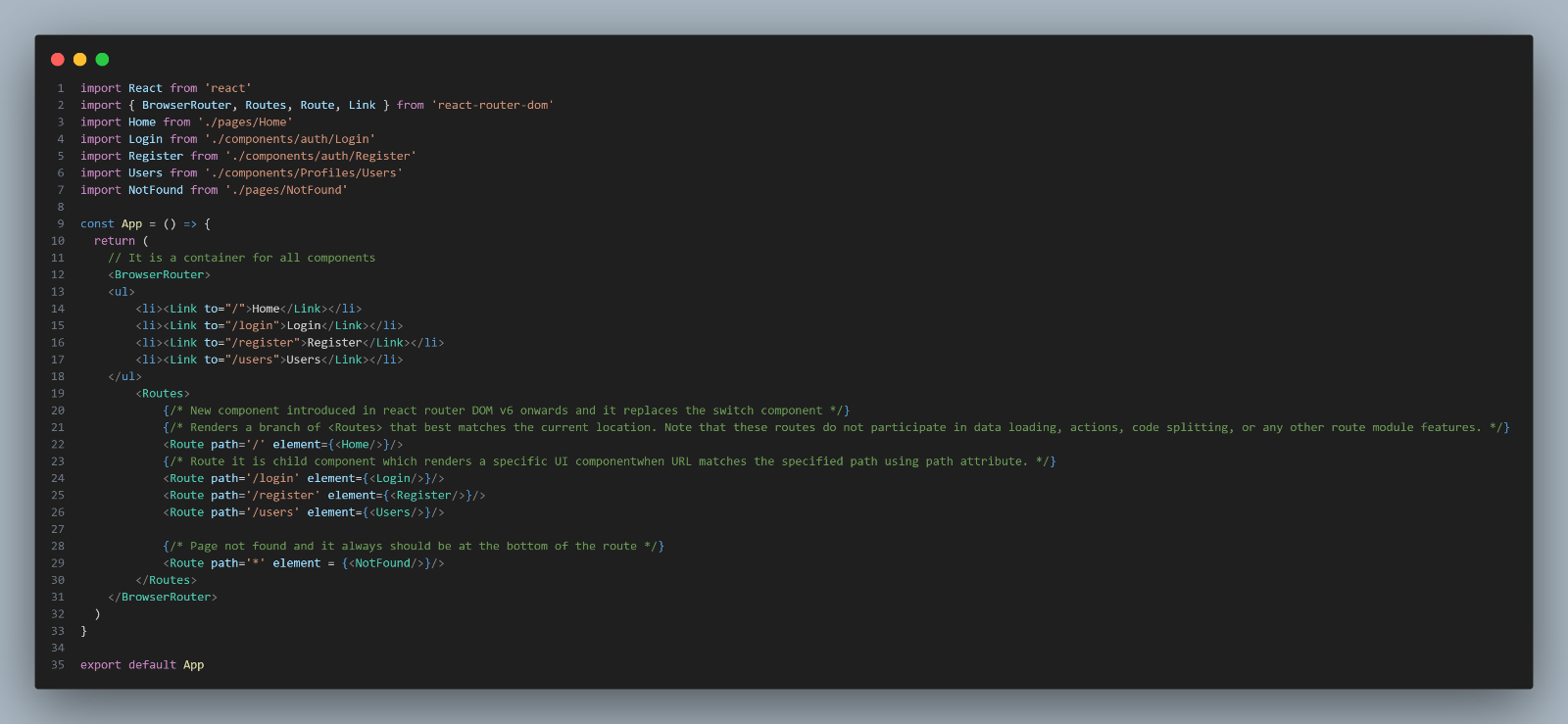
<Route path="\*" element={<PageNotFound />} /> {/\* Catch-all \*/}

</Routes>

* element prop takes the JSX component to render.
* The route with path="\*" should always be **at the bottom** to catch unmatched paths and display a "Page Not Found" UI.

**🔁 Summary:**

| **Concept** | **Description** |
| --- | --- |
| **BrowserRouter** | Syncs URL with the UI using HTML5 history API |
| **Routes** | Container for all route definitions |
| **Route** | Defines path and corresponding UI for rendering |

**Example code**

This is how we can setup routing by using first way. This is also known as traditional way of routing.

**Another way of Routing in react**

Another way we have is data API. It was introduced in React-Router v6.4, this is a more advanced routing approach that supports data loading, form submissions, deferred loading, and better code splitting through route modules. It uses `**createBrowserRouter(), RouterProvider, Route, Link**`. Here we have the example code.

Great question! Let me explain the two key React Router concepts clearly:

**What is createBrowserRouter?**

createBrowserRouter is a **function** introduced in **React Router v6.4+** to create a router instance that uses the **HTML5 History API** (like BrowserRouter does) but with enhanced features for data loading, mutations, and more.

**What it does:**

* It **defines your routes declaratively** with support for loaders, actions, error handling, etc.
* It creates a **router object** that manages route matching, navigation, and data lifecycle.
* This router object is then passed into a <RouterProvider> component to integrate with React.

**Why use it instead of <BrowserRouter>?**

<BrowserRouter> is a simple component wrapping the history API and rendering routes, but it doesn't have built-in support for:

* Data loading before rendering
* Form submissions and mutations
* Error boundaries specific to routes

createBrowserRouter supports all these features by defining routes as objects or JSX with extra options like loader, action, and errorElement.

**What is RouterProvider?**

RouterProvider is a **React component** that takes the router object created by createBrowserRouter (or other router creators) and **renders the app’s routes**.

**What it does:**

* Provides the router context to the entire React app
* Listens for navigation changes (URL changes, user actions)
* Manages route matching, calling loaders/actions, and rendering matched route components
* Handles error boundaries and loading states defined in the router configuration

**How do they work together?**

1. You **create** a router instance using createBrowserRouter by defining your routes and optionally adding data loaders or actions.
2. You **pass** that router to <RouterProvider> which renders the appropriate components based on the current URL.
3. React Router manages the rest — navigation, data fetching, error handling — automatically for you.

**Simple analogy:**

* createBrowserRouter = **Create a navigation map and rules** (with extra features)
* RouterProvider = **The navigator** who uses that map and controls where you go and what you see.

Here we have code example

Here we have another component as layout where we are having outlet. Now the terms comes here what is outlet and why do we need it.

**Outlet Comp**

Outlet is a special type of component provided by React Router that acts as a placeholder in your layout or parent route component where child route components will be rendered.

**Why Do we need it??**

React router uses a nested routing model, where routes can be nested inside other routes to create a hierarchy or layout structure.

* The parent route usually renders some common UI like navigation side bars, header, footer
* The child routes render specific page content based on the URL.

Outlet tells the react router where inside the parent component the matched child route’s UI should appear.