Router

React Router is a library that lets you handle routing (pages/navigation) in React apps — **without reloading the page**.

Install React Router :

npm install react-router-dom

<BrowserRouter>

```
<BrowserRouter>
    <App />
</BrowserRouter>
```

<BrowserRouter> is a **wrapper component** provided by react-router-dom that enables **client-side routing** in your React app.

It tells React:
"This app will use URL-based navigation handled entirely by React, not by reloading the browser."

Why Do You Need <BrowserRouter>?

React doesn't understand URLs like */about* or */contact* by itself.
When you use <Route> and <Link>, **React Router needs to manage the URL and the page view** — that's only possible **inside a <BrowserRouter>**.

Without it, React throws an error like:
-> You cannot use <Route> outside a <Router>

Routes, Route, Link & Outlet

<Routes> is a container that holds one or more <Route> components.

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

<Link> lets you navigate **without reloading the page**.
It replaces the traditional <a href="..."> and is React-friendly.

```
<Link to="/about">About Us</Link>
```

<Outlet /> is like a placeholder where **child routes** (nested pages) will be displayed.

Routes, Route, Link & Outlet

Element	Purpose
<Routes>	Wraps all route definitions
<Route>	Maps a path to a component
<Link>	Navigates between pages without reload
<Outlet>	Dynamically loads page content inside the main layout

If you want to **highlight active links**, use <NavLink> instead of <Link>:

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

<NavLink to="/about" className={({ isActive }) => isActive ? 'active' : ''}>
About
</NavLink>
```

React Context API

Normally in React:

- Data (like theme, language, auth) is passed from **parent → child → grandchild** using **props**
- This is called **prop drilling** and it becomes painful in large apps

Context API solves this by:

- Allowing **global access** to shared data — like theme, user, settings — without prop drilling.

Real Project: Theme Switcher App

- Uses createContext to store the theme state (light or dark)
- Uses Provider to wrap the app
- Any component can useContext() to read/update the theme