1. BrowserRouter

- BrowserRouter wraps your entire React app to enable routing using the **browser's history API**.
- This allows React to **control the URL bar** without refreshing the browser, creating a **Single Page Application (SPA)** experience.

Key Points

- Without BrowserRouter, **none of the routing features (Route, Link, etc.) will work**.
- It keeps the **UI in sync with the URL**.

Where to use it?

Place it in **Main.jsx** or the **top-most component**:

```
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import { ReactDOM } from 'react-router-dom'
import App from './App.jsx'

ReactDOM.createRoot(document.getElementById('root')).r
ender(
  < BrowserRouter>
    <App />
  </ BrowserRouter>,
)
```

2. Routes & Route

- Think of **Routes** **as a container** that holds all the path-based navigation instructions.
- Each **Route** **maps a URL path to a React component** that should render when that path is active.

Example:

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

How it works

- When the URL is /, React renders the Home component.
- When the URL is /about, React renders the About component.

> React Router matches the **path to the URL**, and renders the matching component, replacing the view **without page reload**.

Difference between React Router v5 vs v6:

```
// React Router v5
<Route path="/" component={Home} />

// React Router v6 (NEW)
<Route path="/" element={<Home />} />
```

3. Link

- Link is used to **navigate between routes** without refreshing the page.

Why not use <a href="">?

- Traditional **<a href="">** **tags reload** the entire page — that's not ideal in React SPAs.
- The **Link component from React Router** changes the URL and renders the corresponding component **without reloading**.

Usage

```
<Link to="/">Home</Link>

<Link to="/contact">Contact Us</Link>
```

> Link ensures a **smooth, fast, and efficient SPA experience** by preventing full page refresh.

4. useNavigate

- **useNavigate** is a **hook** that allows you to **navigate programmatically** — i.e., through functions, not UI clicks.

Common Use Cases:

- Navigation depends on **logic (e.g. after form submission)**.
- After login/logout
- Inside event handlers

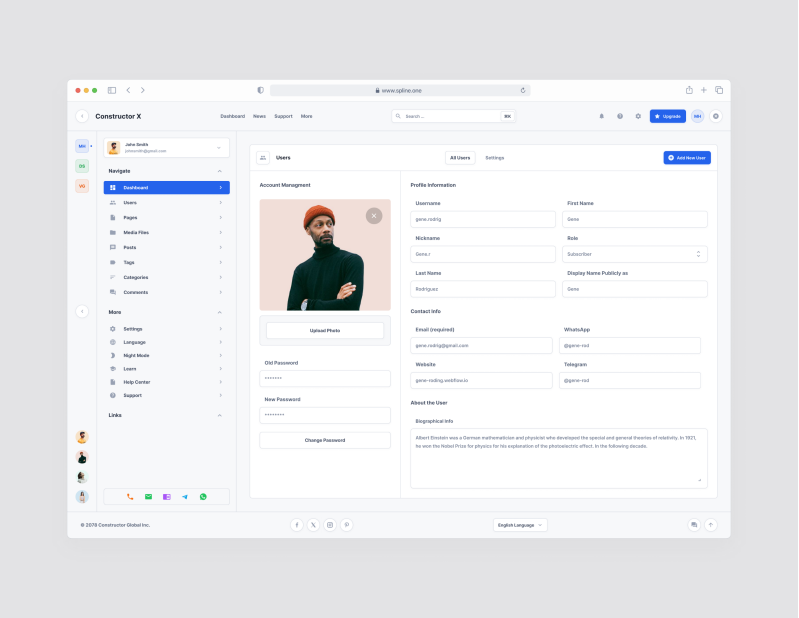
Example:

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

function LoginForm() {

   const navigate = useNavigate();

   function handleLoginSuccess() {

      navigate('/dashboard');

   }

   return <button  onClick={handleLoginSuccess}
>Login</button>;}
```

5. useParams

I’ve created a **class management website** where I store the details of **120 students** on my website. Each student has a **profile card**.

When **User1 logs in** and goes to their **profile section**, the website needs to show **only User1’s profile** — not all 120.



Instead of creating 120 different routes like:

- /profile/user1
- /profile/user2
- ……
- /profile/user120

I use **one smart dynamic route**:

```
<Route path="/profile/:id" element={<Profile />} />
```

Now here’s where useParams comes in…

useParams is a React Router hook that **reads the dynamic value** from the URL — in this case, the **:username**.

So when User1 goes to /profile/user1, inside my Profile component, I write:

```
const { id} = useParams();
```

And then I use this id to **fetch and display the correct profile** from my JavaScript object.

This way, every student sees **only their own profile**, and we don't have to write code for each student manually.

6. Nested Routing

- Nested routing is used when:
    - You have **shared layouts** like a sidebar or navbar.
    - You want to render **sub-routes within a parent route**.

Scenario Example

In a **Dashboard**, you have:

- /dashboard/profile
- /dashboard/settings

Instead of re-rendering the whole dashboard every time, use **nested routes** inside Dashboard:

Example:

App.jsx

```jsx
<Routes>
  <Route path="/dashboard" element={<Dashboard />}>
    <Route path="profile" element={<Profile />} />
    <Route path="settings" element={<Settings />} />
  </Route>
</Routes>
```

Dashboard.jsx

```jsx
import { Outlet, Link } from 'react-router-dom';

function Dashboard() {
  return (
    <div>
      <h2>Dashboard</h2>
      <Link to="profile">Profile</Link> |
      <Link to="settings">Settings</Link>
      <Outlet /> {/* Nested route renders here */}
    </div>
  );
}
```

Navigating to /dashboard/profile renders the Profile component **inside Dashboard**.

Table of Concepts

Feature	Use When...
BrowserRouter	You want to enable client-side routing
Routes	You need to map paths to views/components
Route	You want to show specific component by path
Link	You want navigation without page reload
useNavigate	You need redirection after events (login etc)
useParams	You want dynamic data in URLs (**/product/:id**)
Nested Routes	You need shared layout + sub-views