**🌐 React Router**

**Handling Navigation**

\* `react-router-dom` is the \*\*de facto standard\*\* library for client-side routing in React web apps.

\* It lets you build \*\*single-page applications (SPAs)\*\* where different URLs render different components without reloading the page.

\* Routes are declared using `<Route>` components mapping paths to UI components.

\* `<Link>` components replace anchor tags to enable navigation while preventing full page reloads.

**Example:**

```jsx

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

function App() {

return (

<Router>

<nav>

<Link to="/">Home</Link> | <Link to="/about">About</Link>

</nav>

<Route exact path="/" component={Home} />

<Route path="/about" component={About} />

</Router>

);

}

```

\* Smooth transitions between views.

\* Enables deep linking and bookmarking.

\* Supports nested routes and dynamic parameters.

**🛠️ JavaScript Destructuring**

**Array Destructuring**

\* Simplifies extracting multiple values from arrays into variables.

\* Reduces code verbosity and improves readability.

```js

const [first, second, third] = ['apple', 'banana', 'cherry'];

```

**9. Object Destructuring**

\* Extract specific properties from objects.

\* Can rename variables or provide default values.

\* Essential for working with complex nested data.

```js

const user = { name: 'Bob', age: 30, location: 'NYC' };

const { name, location: city, email = 'N/A' } = user;

```

---

**🔄 React Hooks Overview**

**10. Hooks Benefits**

Hooks like `useState` and `useEffect` allow functional components to have \*\*state and lifecycle features\*\* without classes.`useEffect` replaces lifecycle methods (`componentDidMount`, `componentDidUpdate`, `componentWillUnmount`) and handles side effects like data fetching or subscriptions. Hooks encourage better code reuse through custom hooks Reduce boilerplate and make component logic more intuitive and declarative.

Example of side effects with `useEffect`:

```jsx

function Timer() {

const [seconds, setSeconds] = React.useState(0);

React.useEffect(() => {

const interval = setInterval(() => setSeconds(s => s + 1), 1000);

return () => clearInterval(interval); // cleanup when unmounted

}, []); // empty dependency array = runs once after mount

return <div>Seconds: {seconds}</div>;

}