**Routing in React (React Router)**

**Question 1: What is React Router? How does it handle routing in single-page applications?**

**Ans:-**

What is React Router?

React Router is a popular library for React applications that enables client-side routing. It allows you to create single-page applications (SPAs) with navigation between different components or views without causing a full page reload.

How does React Router handle routing in single-page applications?

* Single-page applications (SPAs) load a single HTML page and dynamically update the content without refreshing the entire page.
* React Router manages the URL changes in the browser without making a new request to the server. It uses the History API (like pushState and popState) to manipulate the URL.
* When a user clicks on a link or navigates programmatically, React Router intercepts the event and renders the appropriate React component that matches the URL path, instead of loading a new page from the server.
* This routing happens inside the browser, making transitions fast and smooth, and preserving the app state.

How React Router works internally (simplified):

1. You define routes in your React app using components like <BrowserRouter>, <Routes>, and <Route>, specifying which component should render for which URL path.
2. When the URL changes, React Router matches the new URL to the route definitions.
3. It then renders the matching React component(s) inside the current page, updating the view.
4. The browser’s URL bar updates accordingly, reflecting the new route without a full reload.

Summary:

* React Router provides a way to map URLs to React components.
* It handles routing on the client side, enabling smooth, fast navigation inside SPAs.
* Uses the History API to manipulate the browser’s URL.
* Prevents full page reloads, keeping the app performant and stateful.

**Question 2: Explain the difference between BrowserRouter, Route, Link, and Switch components in React Router**

**ANS:-**

. BrowserRouter

* It’s the main router component that uses the HTML5 History API to keep your UI in sync with the URL.
* Wraps your entire React app or at least the part that needs routing.
* Manages the browser’s URL and listens for changes.
* Example usage:

jsx

CopyEdit

import { BrowserRouter } from 'react-router-dom';

<BrowserRouter>

{/\* Your routes/components go here \*/}

</BrowserRouter>

2. Route

* Defines a mapping between a URL path and a React component.
* When the URL matches the path prop, React Router renders the component specified by element (in React Router v6+) or component (in v5).
* Example:

jsx

CopyEdit

import { Route } from 'react-router-dom';

<Route path="/about" element={<AboutPage />} />

* You can have multiple routes for different paths.

3. Link

* Used to create navigation links that change the URL without triggering a full page reload.
* Works like an <a> tag but uses React Router internally to update the URL and render components client-side.
* Example:

jsx

CopyEdit

import { Link } from 'react-router-dom';

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

4. Switch (React Router v5) / Routes (React Router v6)

* Switch (v5): Renders only the first child <Route> or <Redirect> that matches the current URL. Prevents multiple routes from rendering at the same time.

Example:

jsx

CopyEdit

import { Switch, Route } from 'react-router-dom';

<Switch>

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

<Route path="/" component={HomePage} />

</Switch>

* Routes (v6): Replaces Switch and behaves similarly, rendering the first matching route.

Example:

jsx

CopyEdit

import { Routes, Route } from 'react-router-dom';

<Routes>

<Route path="/about" element={<AboutPage />} />

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

</Routes>

**Task 1: • Set up a basic React Router with two routes: one for a Home page and one for an About page. Display the appropriate content based on the URL.**

Step-by-step React Router Setup

1. Make sure you have React Router installed:

bash

CopyEdit

npm install react-router-dom

1. Create your basic components for Home and About:

jsx

CopyEdit

// Home.js

export function Home() {

return <h2>Welcome to the Home Page</h2>;

}

// About.js

export function About() {

return <h2>This is the About Page</h2>;

}

1. Setup the router in your main app file (App.js):

jsx

CopyEdit

import React from 'react';

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

import { Home } from './Home';

import { About } from './About';

function App() {

return (

<BrowserRouter>

<nav>

{/\* Navigation Links \*/}

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

</nav>

<Routes>

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

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

</Routes>

</BrowserRouter>

);

}

export default App;

Explanation:

* <BrowserRouter> wraps the whole app to enable routing.
* <Link> creates clickable links to navigate without page reload.
* <Routes> holds the route definitions.
* Each <Route> maps a URL path to a component.
* When you visit /, it renders the Home component.
* When you visit /about, it renders the About component.

Result:

* Navigating to http://localhost:3000/ shows:  
  Welcome to the Home Page
* Navigating to http://localhost:3000/about shows:  
  This is the About Page

**Task 2: • Create a navigation bar using React Router’s Link component that allows users to switch between the Home, About, and Contact pages.**

**ANS:-**

import React from 'react';

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

import { Home } from './Home';

import { About } from './About';

import { Contact } from './Contact';

function App() {

return (

<BrowserRouter>

{/\* Navigation Bar \*/}

<nav style={{ padding: '10px', backgroundColor: '#eee' }}>

<Link to="/" style={{ margin: '0 10px' }}>Home</Link>

<Link to="/about" style={{ margin: '0 10px' }}>About</Link>

<Link to="/contact" style={{ margin: '0 10px' }}>Contact</Link>

</nav>

{/\* Routes \*/}

<Routes>

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

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

<Route path="/contact" element={<Contact />} />

</Routes>

</BrowserRouter>

);

}

export default App;