Class-06: React Router

1. React Router Basics

React Router is a popular library for managing navigation in a React application. It enables single-page applications (SPAs) to have multiple views while maintaining a seamless user experience.

Installation

To use React Router, install it using npm or yarn:

```
npm install react-router-dom
```

Basic Setup

In a React application, React Router provides different components to define routes:

- **BrowserRouter**: Wraps the entire application to enable routing.
- **Routes**: A container for route definitions.
- **Route**: Defines a specific path and its corresponding component.
- **Link**: Used for navigation without reloading the page.

Example:

```
import { BrowserRouter as Router, Routes, Route, Link } from
'react-router-dom';
import Home from './Home';
import About from './About';

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
      </nav>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}
export default App;
```

2. Nested Routing

Nested routing allows rendering child routes inside parent routes, making the application structure more organized.

Example:

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

function Dashboard() {
  return (
    <div>
      <h2>Dashboard</h2>
      <nav>
        <Link to="profile">Profile</Link>
        <Link to="settings">Settings</Link>
      </nav>
      <Outlet />
    </div>
  );
}

function Profile() { return <h3>Profile Page</h3>; }
function Settings() { return <h3>Settings Page</h3>; }

function App() {
  return (
    <Router>
      <Routes>
        <Route path="dashboard" element={<Dashboard />}>
          <Route path="profile" element={<Profile />} />
          <Route path="settings" element={<Settings />} />
        </Route>
      </Routes>
    </Router>
  );
}

export default App;
```

Here, Outlet is used as a placeholder to render nested components dynamically.

3. Dynamic Routing

Dynamic routing enables pages to display content based on parameters in the URL.

Example:

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

function UserProfile() {
  let { userId } = useParams();
  return <h2>User Profile for ID: {userId}</h2>;
}

function App() {
  return (
    <Router>
      <Routes>
        <Route path="user/:userId" element={<UserProfile />} />
      </Routes>
    </Router>
  );
}

export default App;
```

Here, :userId is a dynamic segment that gets replaced with actual values, and useParams helps extract the parameter.

4. Redirects

Redirects help navigate users to different routes programmatically or automatically.

Using Navigate for Redirects

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

function Home() {
  const isLoggedIn = false; // Change this to true to allow access
  return isLoggedIn ? <h2>Welcome Home!</h2> : <Navigate to="/login" />;
}

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/home" element={<Home />} />
        <Route path="/login" element={<h2>Login Page</h2>} />
      </Routes>
    </Router>
  );
}

export default App;
```

If isLoggedIn is false, users will be redirected to the login page.

Conclusion

React Router enhances SPA navigation by providing structured routing, nested routes, dynamic parameters, and redirects. Understanding these concepts is crucial for building scalable React applications.