📌 React Router 완벽 가이드

🚀 1. React Router란?

📖 개념

- `React Router`는 **React 애플리케이션에서 클라이언트 사이드 라우팅을 구현하는 대표적인 라이브러리**입니다.
- 페이지를 새로고침하지 않고도 URL을 기반으로 **컴포넌트를 전환**할 수 있게 도와줍니다.
- SPA(Single Page Application)의 **내비게이션 구성의 핵심 도구**입니다.

설치 명령어 : npm install react-router-dom

💡 2. 핵심 개념 및 구성 요소

✅ `BrowserRouter`

- HTML5의 history API를 사용하여 URL을 관리하는 라우터
- 루트 컴포넌트를 감싸야 함

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

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

✅ `Routes` & `Route`

- `Routes`는 여러 `Route`를 포함하는 컨테이너
- `Route`는 특정 경로(`path`)에 컴포넌트를 매핑

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

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

✅ `Link` / `NavLink`

- 페이지 이동을 위한 컴포넌트 (a 태그 대체)
- `NavLink`는 활성화된 링크에 스타일 적용 가능

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

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

🛠️ 3. 동적 라우팅 (Dynamic Routing)

📌 URL 파라미터 사용

```
<Route path="/users/:id" element={<UserDetail />} />
import { useParams } from 'react-router-dom';

function UserDetail() {
  const { id } = useParams();
  return <p>유저 ID: {id}</p>;
}
```

✅ URL 경로의 값을 `useParams()` 훅으로 가져올 수 있음

📌 쿼리스트링 사용

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

function SearchPage() {
  const [searchParams] = useSearchParams();
  const keyword = searchParams.get('keyword');
  return <p>검색어: {keyword}</p>;
}
```

🔁 4. 중첩 라우트 (Nested Routing)

📌 라우트 안에 자식 라우트 정의

```
<Route path="/dashboard" element={<Dashboard />}>
  <Route path="stats" element={<Stats />} />
  <Route path="settings" element={<Settings />} />
</Route>
```

📌 Outlet 컴포넌트로 자식 컴포넌트 렌더링 위치 지정

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

function Dashboard() {
  return (
    <div>
      <h2>Dashboard</h2>
      <Outlet />
    </div>
  );
}
```

✅ 중첩 구조를 통해 레이아웃 유지가 용이함

🧩 5. Navigate (리디렉션)

📌 컴포넌트 내에서 경로 변경

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

function Login() {
  const navigate = useNavigate();

  const handleLogin = () => {
    // 로그인 로직...
    navigate('/dashboard');
  };

  return <button onClick={handleLogin}>로그인</button>;
}
```

📦 6. 404 페이지 처리

📌 없는 경로 처리하기

```
<Route path="*" element={<NotFound />} />
```

✅ 모든 라우트 뒤에 정의하여 일치하지 않는 경로를 처리할 수 있음

🧠 7. 실무 팁

- URL 파라미터와 쿼리스트링을 함께 쓰는 경우 분리해서 관리
- `useEffect`로 파라미터 변화에 따른 API 요청 처리
- `React.lazy`와 `Suspense`를 조합하면 라우트 코드 분할 가능 (Code Splitting)

```
const About = React.lazy(() => import('./pages/About'));
...
<Suspense fallback={<div>로딩중...</div>}>
  <Routes>
    <Route path="/about" element={<About />} />
  </Routes>
</Suspense>
```

🏁 8. 마무리

✅ 핵심 요약

- `react-router-dom`은 SPA의 페이지 이동을 구성하는 핵심 라이브러리
- `Routes` / `Route`, `Link`, `useNavigate`, `useParams` 등 핵심 컴포넌트와 훅을 익히면 강력한 내비게이션 구현 가능
- 실무에서는 동적 라우팅, 중첩 라우트, 404 처리, 코드 스플리팅까지 고려해야 함