01

Route

0. React Router Dom

라이브러리 설치

- npm install react-router-dom

<Router>

- <BrowserRouter>: 웹 애플리케이션에서 가장 흔히 사용하는 라우터로, HTML5의 히스토리 API를 사용하여 URL을 관리

- <HashRouter>: 정적 파일만을 제공하는 서버나 백엔드 서버가 없는 정적 웹사이트 호스팅 환경에서 사용

01. <BrowserRouter>

- 애플리케이션을 감싸는 라우터의 기본 컴포넌트
- HTML5의 히스토리 API를 사용하여 URL을 관리

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

function App() {
    return (
        <BrowserRouter>
            <MainRoutes />
        </BrowserRouter>
    );
}
```

02. <Routes>와 <Route>

- <Routes>는 여러 개의 <Route>를 감싸서 각 URL 경로에 맞는 컴포넌트를 렌더링
- <Routes>는 하위 <Route> 중 일치하는 path를 가진 <Route> 를 실행하고 나머지 <Route>는 무시

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

function MainRoutes() {
    return (
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
        </Routes>
    );
}
```

03. Route의 index 속성

- 중첩된 라우트에서 기본 경로로 표시할 페이지를 설정

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

function App() {
    return (
        <Routes>
            <Route path="/" element={<Layout />}>
                <Route index element={<Home />} />
                 {/* 기본 페이지 */}
                <Route path="/about" element={<About />} />
            </Route>
        </Routes>
    );
}
```

04. <Link>

- **두 태그 모두 페이지 간 이동을 위한 링크**
  - <Link>: 기본 링크를 제공
  - <a>의 기본 기능인 페이지 리로딩을 하지 않는다.

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

function Navbar() {
 return (
   <nav>
    <Link to="/">Home</Link>
    <Link to="/about">About</Link>
   </nav>
    );
}
```

05. useNavigate

- 프로그래밍 방식으로 페이지 이동을 제어하는 훅.
- 이벤트 발생시 특정 페이지로 이동할 때 유용.
- 비동기 적으로 처리되면 현재 컴포넌트의 랜더링이 끝난 다음 경로를 변경.

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

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

    return (
        <button onClick={() => navigate('/about')}>
            Go to About
        </button>
    );
}
```

06. useParams

- URL 파라미터를 가져오는 훅.
- 특정 ID로 데이터를 불러올 때 유용.

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

function Post() {
    const { postId } = useParams();

    return <h2>Post ID: {postId}</h2>;
}
```

  - /post/:postId 경로에서 postId 파라미터를 가져올 수 있다.

07. Outlet 컴포넌트

- 중첩된 라우트를 렌더링할 때 자식 경로가 표시될 위치를 지정

```
import { Routes, Route, Outlet } from 'react-router-dom';
function Layout() {
    return (
        <div>
            <h1>Main Layout</h1>
            <Outlet /> {/* 자식 컴포넌트가 여기에 렌더링 */}
        </div>
    );}
function App() {
  return (
     <Routes>
       <Route path="/" element={<Layout />}>
           <Route path="about" element={<About />} />
           <Route path="contact" element={<Contact />} />
            </Route>
       </Routes>
    );}
```

08. useOutletContext()

- React Router에서 제공하는 특수 훅
- 중첩된 라우트 구조에서 부모 컴포넌트가 <Outlet />을 통해 전달한 데이터를 하위 라우트 컴포넌트들이 사용할 수 있도록 전달
- React의 Context API와 유사하지만, 특정 라우트 구조 내에서만 데이터를 공유하는 데 사용

08. useOutletContext()

```
function App() {
return (
 <BrowserRouter>
  <Routes>
    <Route path="parent" element={<ParentComponent />}>
     <Route path="child" element={<ChildComponent />}>
       <Route path="grandchild“
                    element={<GrandchildComponent/>} />
     </Route>
    </Route>
  </Routes>
 </BrowserRouter>
    );
}

export default App;
```

08. useOutletContext()

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

function ParentComponent() {
    const sharedData = "This data is shared with all
                        nested routes";

    return (
        <div>
            <h1>Parent Component</h1>
            <Outlet context={{ sharedData }} />
             {/* 데이터를 context로 전달 */}
        </div>
    );
}

export default ParentComponent;
```

08. useOutletContext()

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

function ChildComponent() {
    const { sharedData } = useOutletContext();
    // 상위 컴포넌트의 데이터를 받아옴

    return (
        <div>
            <h2>Child Component</h2>
            <p>Data from Parent: {sharedData}</p>
            <Outlet />
        {/* GrandchildComponent를 렌더링할 Outlet */}
        </div>
    );
}

export default ChildComponent;
```

08. useOutletContext()

```
import React from 'react';
import { useOutletContext } from 'react-router-dom';

function GrandchildComponent() {
    const { sharedData } = useOutletContext();
    // 상위 컴포넌트의 데이터를 받아옴

    return (
        <div>
            <h3>Grandchild Component</h3>
            <p>Data from Parent: {sharedData}</p>
        </div>
    );
}

export default GrandchildComponent;
```