13. React Router로 SPA 개발하기

13.1 Rounting?

- 리액트에서 라우트 시스템을 구축하기 위한 방법은 두 가지가 있다.
  1. reacu-router : 이 라이브러리는 가장 오래됐고 가장 많이 사용
  2. Next.js : react의 framework으로서 프로젝설정기능, 라우팅시스템, 최적화, 다국어지원, SSR등 다양한 기능제공
- 리액트 라우터를 사용하면 손쉽게 리액트 라우터로 SPA를 만들 수 있따.

13.2 리액트 라우터 적용 및 사용

13.2.1 프로젝트생성 및 라이브러리 설치

> - 설치
>   - `yarn create react-app .`
>   - `yarn add react-router-dom`

13.2.2 router 적용

- react-router를 적용할 때는 `index.js에서 react-router-dom에 내장되어 있는 BrouwserRouter을 사용`

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter} from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

13.2.3 Page Component 만들기

src/pages/Home.js

```
const Home = () => {
  return (
    <div>
      <h1>Home</h1>
      <p>Home Page 입니다!!</p>
    </div>
  );
}
```

```
export default Home;
```

src/pages/About.js

```
const About = () => {
  return (
    <div>
      <h1>About</h1>
      <p>About Page 입니다!!</p>
    </div>
  );
}

export default About;
```

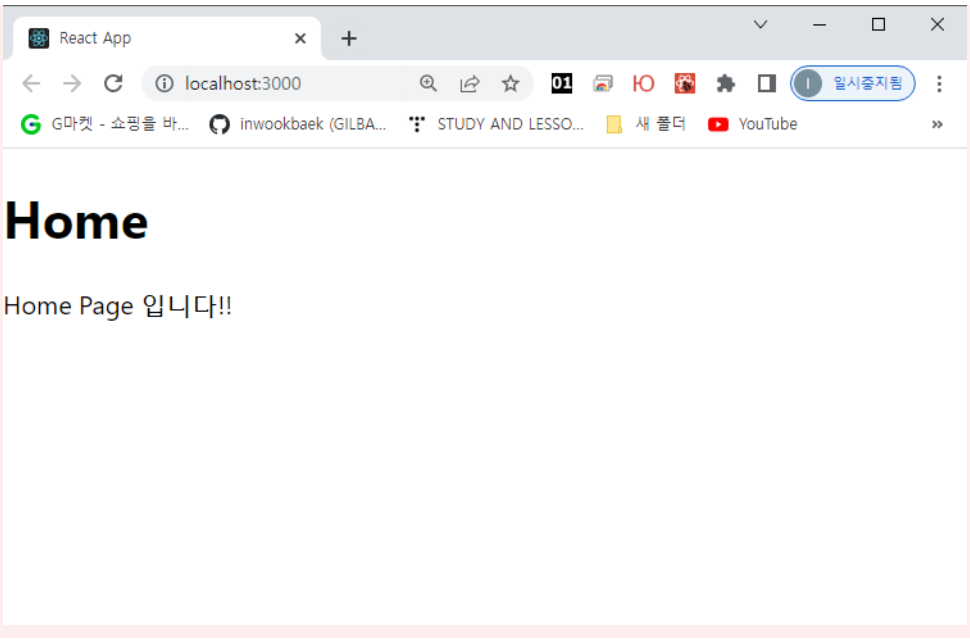
- Router Component는 다음과 같이 사용

  `<Route path='주소' element={ 컴퍼넌트 }>`

- Route컴퍼넌트는 `Routes` 컴퍼넌트 안에서 사용해야 한다.

src/App.js

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

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

export default App;
```

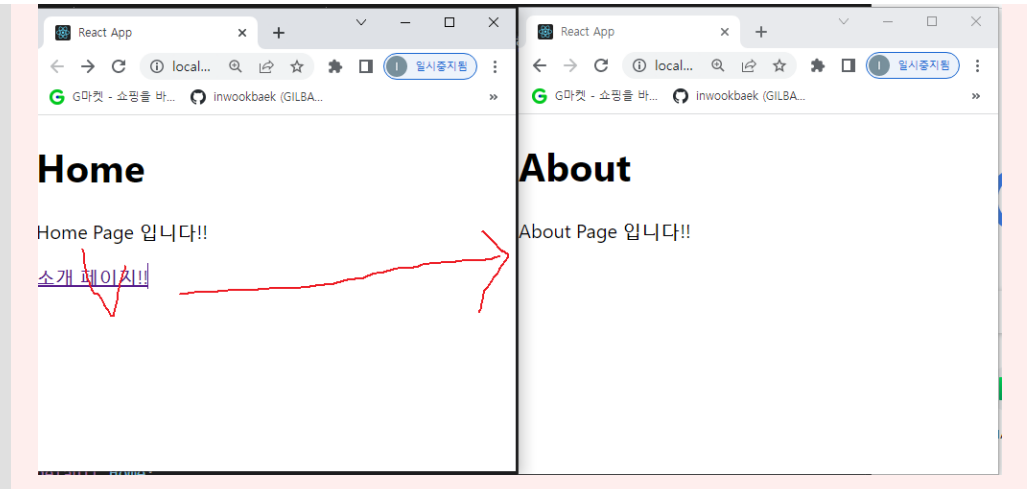


13.2.4 Link 컴퍼넌트

- 리액트라우터를 사용하는 프로젝트에서는 `a태그를 사용하면 않된다.`
- 그 이유는 a태그는 페이지를 이동할 때 페이지를 새로 불러오기 때문이다.
- Link컴퍼넌트로 a태그를 사용하지만 페이지를 `새로 불러오는 것을 막고`
- `History API를 통해 브러우저 주소의 경로만 변경하는 기능이 내장`되어 있다.
- Link컴퍼넌트는 `<Link to='경로'>링크이름</Link>`처럼 사용한다.

src/pages/About.js

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

const Home = () => {
  return (
    <div>
      <h1>Home</h1>
      <p>Home Page 입니다!!</p>
      <Link to='/about'>소개 페이지!!</Link>
    </div>
  );
}

export default Home;
```



13.4 URL 파라미터와 Query String

페이지주소 정의할 때

- URL파라미터 : `profile/posts`
  - URL파라미터는 주소의 경로에 유동적인 값을 넣는 형태이고
  - URL파라미터는 주로 id 또는 이름을 사용하여 특정 데이터를 조회할 때 사용
- 쿼리스트링 :: `comments?pave=1&search=react`
  - 쿼리스트링은 주소뒷부분에 `?문자열이후에 key=value형태로 정의하고 &로 구분하는 형태`이다.
  - 쿼리스트링은 키워드검색, 페이징, 정렬방식등의 데이터 조회헤 필요한 옵션을 전달할 때 사용

13.4.1 URL 파라미터

src/pages/Profile.js

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

const data = {
  sonny: {
    name: '손흥민',
    description: "EPL 토트넘 선수"
  },
  kangin: {
    name: '이강인',
    description: "스페인 마요르카 선수"
  }
};

const Profile = () => {

  const params = useParams();

  const profile = data[params.username];

  return (
    <div>
      <h1>축구선수 프로필</h1>
      {profile ? (
        <div>
          <h2>{profile.name}</h2>
          <p>{profile.description}</p>
        </div>
      ) : (
        <p>존재하지 않는 프로필입니다!!</p>
      )}
    </div>
  );
};

export default Profile;
```

- `URL 파라미터는 useParams Hook을 사용하여 객체형태로 조회 할 수 있다.`
- `URL 파라미터의 이름은 라우터를 설정할 때 Route 컴퍼넌트의 path props를 통해 설정 한다.`

src/App.js 수정

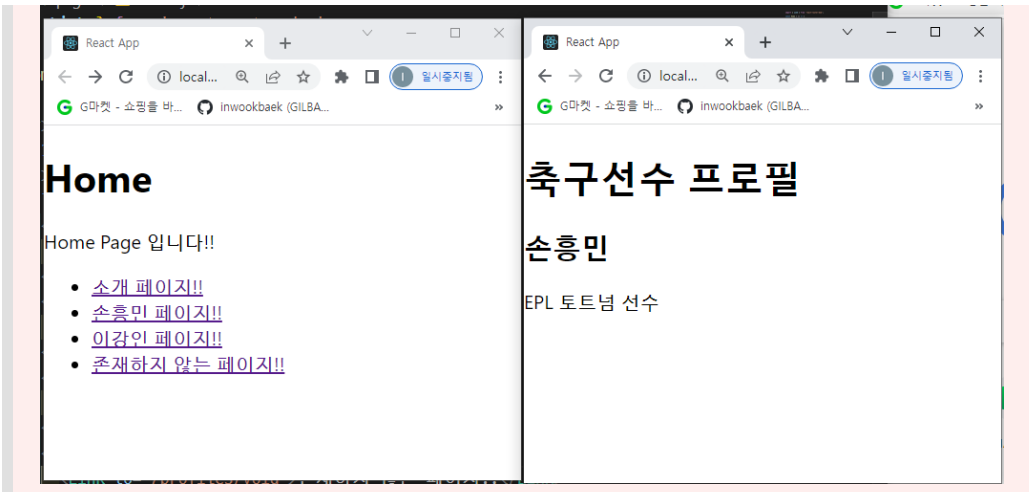
```
import { Route, Routes } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';
import Profile from './pages/Profile';

function App() {
  return (
    <Routes>
      <Route path="/" element={ <Home />} />
      <Route path="/about" element={ <About />} />
      <Route path="/profiles/:username" element={ <Profile />} />
    </Routes>
  );
}
```

```
export default App;
```

- URL파라미터는 `:username과 같이 경로에 :을 사용하여 설정`
- 여러개 인 경우 `/profiles/:username/:something와 같은 형태`로 설정한ㄷ.

src/pages/Home.js

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

const Home = () => {
  return (
    <div>
      <h1>Home</h1>
      <p>Home Page 입니다!!</p>
      <ul>
        <li>
          <Link to='/about'>소개 페이지!!</Link>
        </li>
        <li>
          <Link to='/profiles/sonny'>손흥민 페이지!!</Link>
        </li>
        <li>
          <Link to='/profiles/kangin'>이강인 페이지!!</Link>
        </li>
        <li>
          <Link to='/profiles/void'>존재하지 않는 페이지!!</Link>
        </li>
      </ul>
    </div>
  );
}

export default Home;
```



13.4.2 쿼리스트링

- 쿼리스트링은 URL파라미터와 달리 별도로 설정해야 하는 것이 없다.

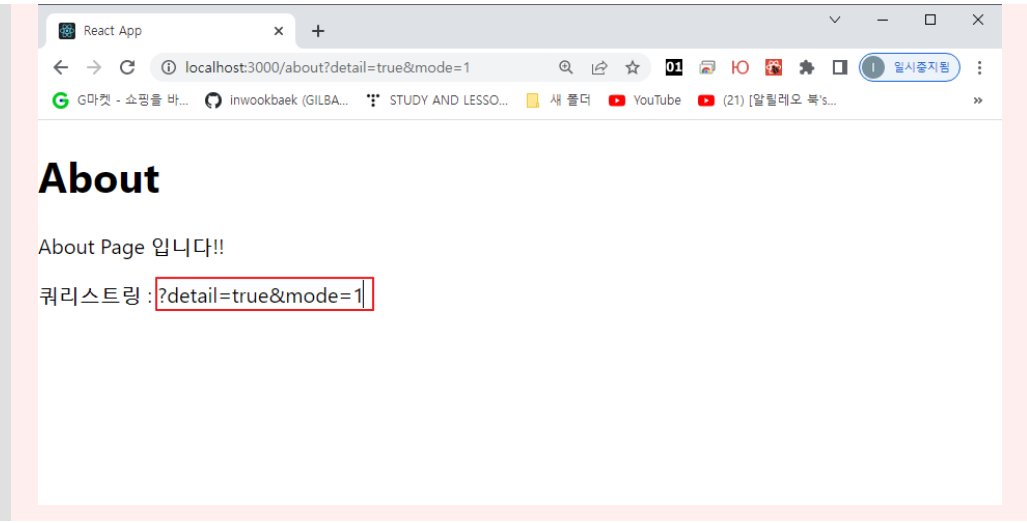
src/pages/About.js

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

```
const About = () => {

  const location = useLocation();
  return (
    <div>
      <h1>About</h1>
      <p>About Page 입니다!!</p>
      <p>쿼리스트링 : {location.search}</p>
    </div>
  );
}

export default About;
```

- `useLocation hook을 사용, 이 hook은 location객체를 반환한다.`
- 이 객체에는 페이지의 정보를 가지고 있다.

  1. pathname : 현재 주소의 경로(쿼리스트링제외)
  2. search : `?문자를 포함한 쿼리스트링 값`
  3. hash : 주소의 # 문자열 뒤의 값
     - 주로 History API가 지원되지 않는 구형 브라우저에서 클라우드라이팅을 하용할 때 쓰는 해시라우터에서 사용
  4. state : 페이지로 이동할 때 임의로 넣을 수 있는 상태 값
  5. key : location객체의 고유값, 초기에는 default이며 페이지변경될 때마다 고유값생성
- 주소창에 `http:localhost:3000/about?detail=true&mode=1` 로 확인



- key와 value를 파싱할 경우 `npm install qs 또는 npm install querysting` 설치

src/pages/About.js - qs 사용

- 사용하는 방법만 정리하고 실습을 하지 말 것

```
import React from 'react';
import qs from 'qs';

const About = ({ location }) => {
  const query = qs.parse(location.search, {
    ignoreQueryPrefix: true // 이 설정을 통하여 문자열 맨 앞의 ? 를 생략합니다.
```

```
  });
  const showDetail = query.detail === 'true'; // 쿼리의 파싱 결과값은 문자열입
니다.
  return (
    <div>
      <h1>소개</h1>
      <p>이 프로젝트는 리액트 라우터 기초를 실습해보는 예제 프로젝트입니다.
</p>
      {showDetail && <p>detail 값을 true 로 설정하셨군요!</p>}
    </div>
  );
};

export default About;
```

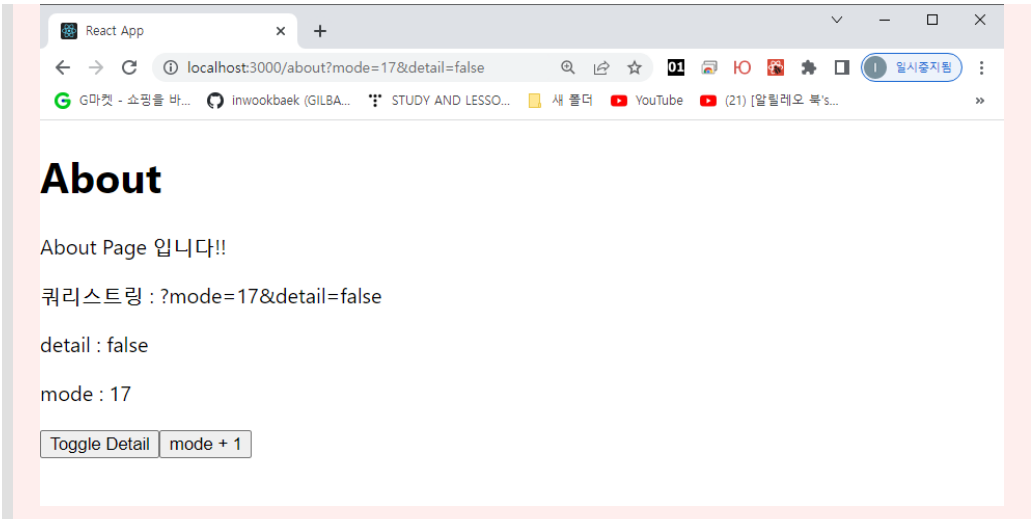
- react v6부터 `useSearchParams`라는 Hook을 통해서 쿼리스트링을 쉽게 조작할 수 있다

src/pages/About.js - useSearchParams 사용

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

const About = () => {

  const location = useLocation();

  const [searchParams, setSearchParams] = useSearchParams();
  const detail = searchParams.get('detail');
  const mode = searchParams.get('mode');

  const onToggleDetail = ()=> {
    setSearchParams({mode, detail: detail === 'true' ? false : true });
  }
  const onIncreaseMode = ()=> {
    const nextMode = mode === null ? 1 : parseInt(mode) + 1;
    setSearchParams({mode: nextMode, detail })
  }

  return (
    <div>
      <h1>About</h1>
      <p>About Page 입니다!!</p>
      <p>쿼리스트링 : {location.search}</p>
      <p>detail : {detail}</p>
      <p>mode : {mode}</p>
      <button onClick={onToggleDetail}>Toggle Detail</button>
      <button onClick={onIncreaseMode}>mode + 1</button>
    </div>
  );
}

export default About;
```

- `useSearchParams`는 배열타입의 값을 리턴
- 첫 번째는 쿼리파라미터를 조회하거나 수정하는 메서드들이 담긴 객체를 반환
- `get()`을 통해 특정 쿼리파라미터를 조회할 수 있고 `set()`을 통해 파라미터를 수정 할 수 있다.

- 만약, 파라미터가 없다면 null을 리턴
- 두 번째 원소는 파라미터를 객체형태로 수정하는 함수를 반환
- 주의할 점은 파라미터를 조회할 때 `값은 무조건 문자열 타입`
- 따라서, 비교할 때 문자열로 비교해야 하고 숫자형문자열을 parseInt로 숫자타입으로 리턴해 야한다.



13.5 중첩된 라우트

src/pages/Artices.js

```
import { Link } from "react-router-dom"

const Articles = () => {

  return (
    <ul>
      <li>
        <Link to="/articles/1">게시글 1</Link>
      </li>
      <li>
        <Link to="/articles/2">게시글 2</Link>
      </li>
      <li>
        <Link to="/articles/3">게시글 3</Link>
      </li>
    </ul>
  )
}

export default Articles
```

src/pages/Artice.js

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

const Article = () => {

  const { id } = useParams();

  return (
    <div>
```

```
        <h2>게시글 {id}</h2>
      </div>
    )
  }

  export default Article
```

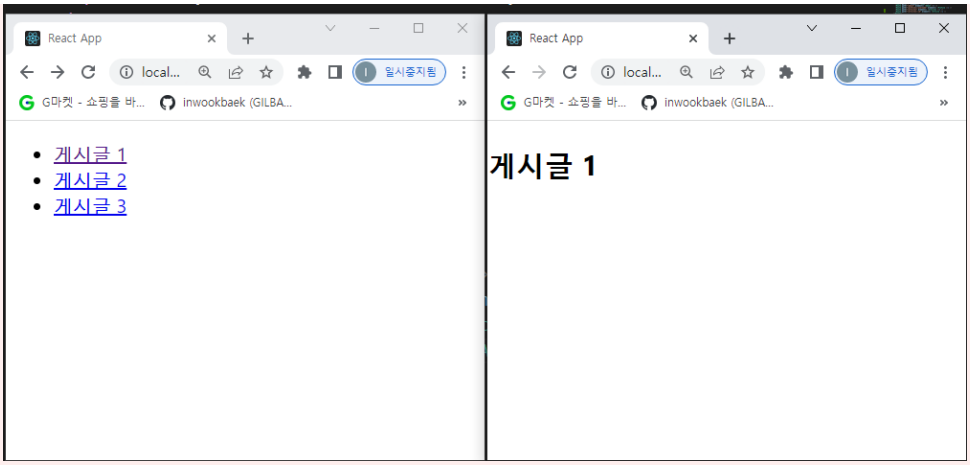
src/App.js

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

  function App() {
    return (
      <Routes>
        <Route path="/" element={ <Home />} />
        <Route path="/about" element={ <About />} />
        <Route path="/profiles/:username" element={ <Profile />} />
        <Route path="/articles" element={ <Articles />} />
        <Route path="/articles/:id" element={ <Article />} />
      </Routes>
    );
  }

  export default App;
```

src/pages/Home.js

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

  const Home = () => {
    return (
      <div>
        <h1>Home</h1>
        <p>Home Page 입니다!!</p>
        <ul>
          <li>
            <Link to='/about'>소개 페이지!!</Link>
          </li>
          <li>
            <Link to='/profiles/sonny'>손흥민 페이지!!</Link>
          </li>
          <li>
            <Link to='/profiles/kangin'>이강인 페이지!!</Link>
          </li>
          <li>
            <Link to='/profiles/void'>존재하지 않는 페이지!!</Link>
          </li>
          <li>
            <Link to='/articles'>게시글 목록</Link>
          </li>
        </ul>
      </div>
    );
```

```
}

export default Home;
```



nested route

- 중첩된 라우터를 구현하기 위해서는 `react-router의 Outlet 컴퍼넌트를 사용` 해야 한다.

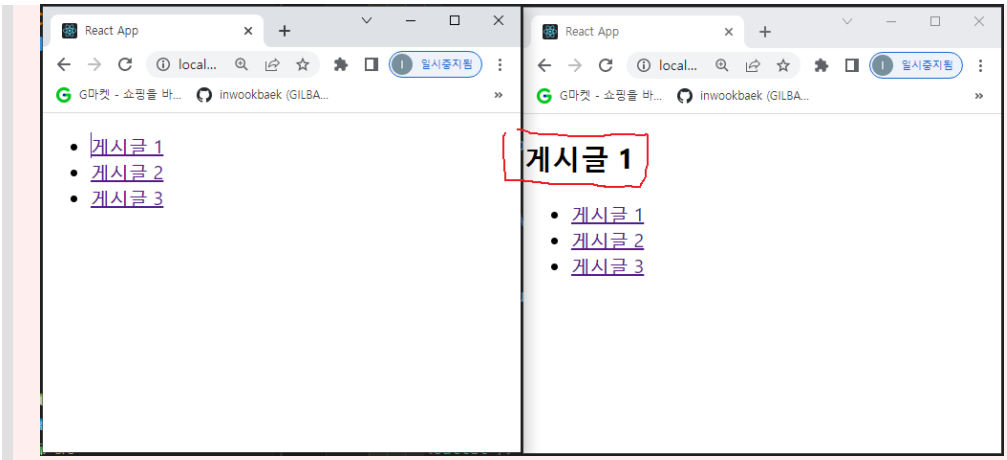
src/App.js

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

function App() {
  return (
    <Routes>
      <Route path="/" element={ <Home />} />
      <Route path="/about" element={ <About />} />
      <Route path="/profiles/:username" element={ <Profile />} />
      <Route path="/articles" element={ <Articles />}>
        <Route path="/articles/:id" element={ <Article />} />
      </Route>
    </Routes>
  );
}

export default App;
```

src/pages/Articles.js

```
import { Link, Outlet } from "react-router-dom"

const Articles = () => {

  return (
    <div>
      <Outlet />
      <ul>
        <li>
```

```
            <Link to="/articles/1">게시글 1</Link>
          </li>
          <li>
            <Link to="/articles/2">게시글 2</Link>
          </li>
          <li>
            <Link to="/articles/3">게시글 3</Link>
          </li>
        </ul>
      </div>
    )
}

export default Articles
```



13.5.1 공통 에이아웃 컴퍼넌트

- 중첨라우터와 Outlet은 페이지끼리 공통적으로 사용하는 레이아웃이 있을 때 유용하게 사용

공통레이아웃

src/Layout.js

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

const Layout = () => {
  return (
    <div>
      <header style={{background: 'mediumseagreen', padding: 16,
        color: 'white', fontSize: 24}}>공통레이아웃</header>
      <main>
        <Outlet />
      </main>
    </div>
  )
}

export default Layout;
```

src/App.js

```
import { Route, Routes } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';
import Profile from './pages/Profile';
import Articles from './pages/Articles';
import Article from './pages/Article';
import Layout from './Layout';

function App() {
  return (
    <Routes>
      <Route element={ <Layout/> }>
        <Route path="/" element={ <Home />} />
        <Route path="/about" element={ <About />} />
        <Route path="/profiles/:username" element={ <Profile />} />
      </Route>
      <Route path="/articles" element={ <Articles />}>
        <Route path="/articles/:id" element={ <Article />} />
      </Route>
    </Routes>
  );
}

export default App;
```

13.5.2 index props

- Rount 컴퍼넌트에는 `index라는 props`가 있다. 이 props는 `path=/`와 동일한 의미
- index props인 `path='/'`는 좀 더 명시저으로 표현하는 방법

13.6 리액트라우터 부가기능

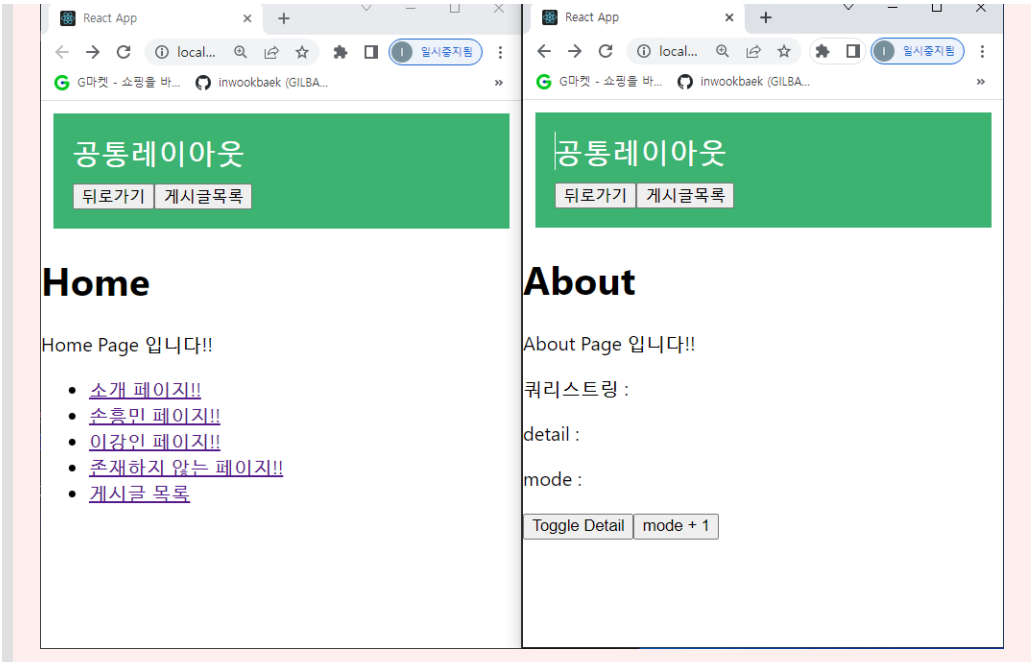
13.6.1 `useNavigate`

- useNavigatesms `Link컴퍼넌트를 사용하지 않고 다른 페이지로 이동`해야 할 때 사용하는 HOOK
- `navegate(n)함수를 사용할 때 양수면 앞으로 음수면 뒤로 지정된 숫자만큼 이동`한다.
- `navigate('/articles', {relace: true});`처럼 replace옵션을 true로 설정하면
- 직전페이지가 아니라 articles 전의 페이지가 나타나개 된다.

src/Layout.js

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

const Layout = () => {

  const navigate = useNavigate();

  const goBack = () => {
```

```
    navigate(-1); // 이전 페이지
  }

  const goArtices = () => {
    navigate('/articles');
    // navigate('/articles', {replace: true});
  }
  return (
    <div>
      <header style={{background: 'mediumseagreen', padding: 16, margin:10,
            color: 'white', fontSize: 24}}>공통레이아웃
        <div>
          <button onClick={goBack}>뒤로가기</button>
          <button onClick={goArtices}>게시글목록</button>
        </div>
      </header>
      <main>
        <Outlet />
      </main>
    </div>
  )
}

export default Layout;
```



13.6.2 NavLink

- 경로가 현재 라우트의 경로와 일치할 경우 스타일 또는 CSS클래스를 적용하는 컴퍼넌트 이다.
- 이 컴퍼넌트의 style과 className은 {isActive: boolean}을 전달 받는 함수타입의 값을 전달

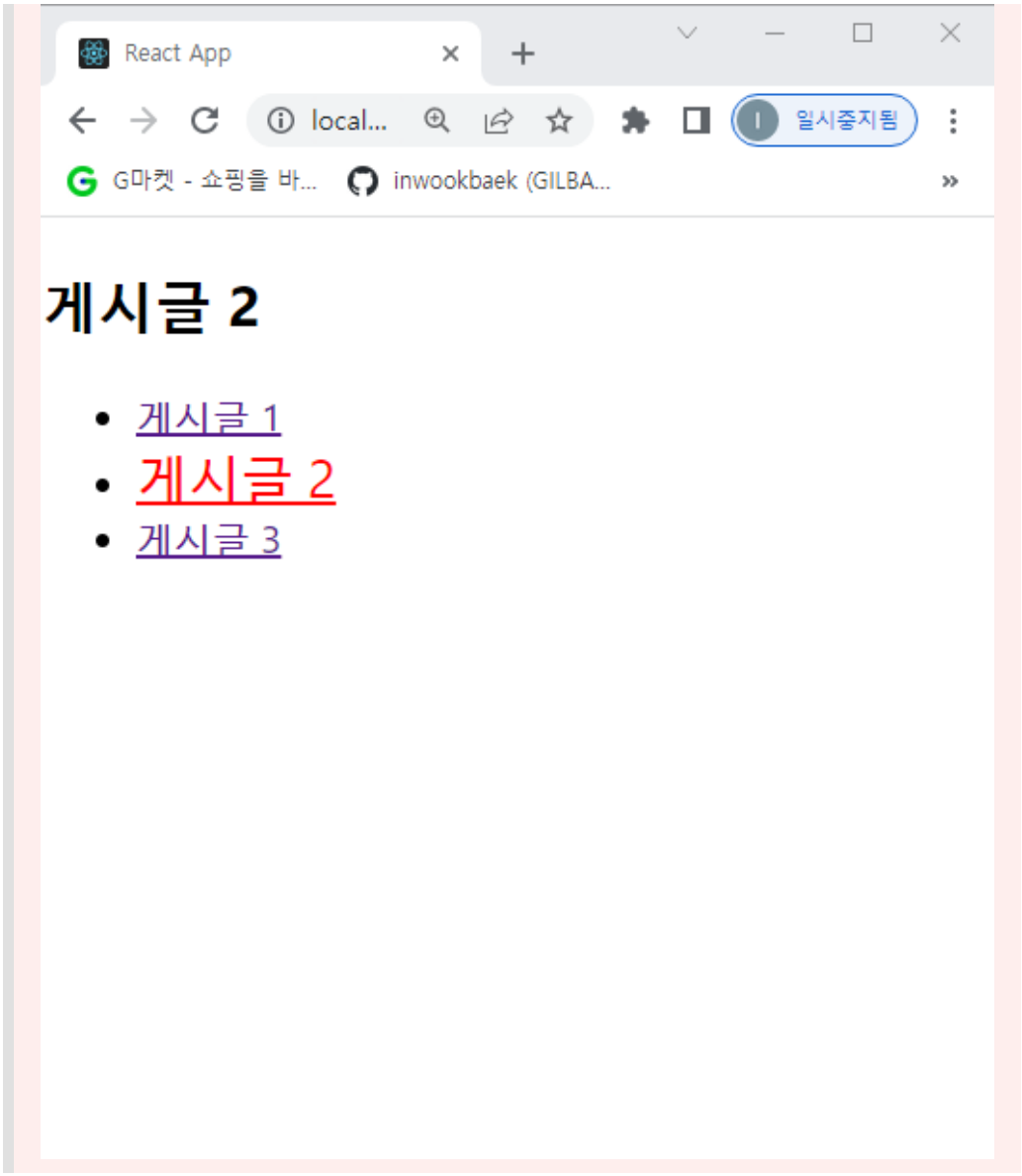
예시코드

```
<NavLink style={({isActive ? activeStyle : undefined})}>
  또는
```

```
  <NavLink style={({isActive ? 'active' : undefined})}>
```

src/pages/Articles.js

```
import { Link, NavLink, Outlet } from "react-router-dom"

const Articles = () => {

  const activeStyle = {
    color: 'red',
    fontSize: 22,
  }

  return (
    <div>
      <Outlet />
      <ul>
        <li>
          <NavLink to="/articles/1" style={({isActive}) => isActive ?
activeStyle: undefined}>게시글 1</NavLink>
        </li>
        <li>
          <NavLink to="/articles/2" style={({isActive}) => isActive ?
activeStyle: undefined}>게시글 2</NavLink>
        </li>
        <li>
          <NavLink to="/articles/3" style={({isActive}) => isActive ?
activeStyle: undefined}>게시글 3</NavLink>
        </li>
      </ul>
    </div>
  )
}

export default Articles
```



src/pages/Articles.js - refactoring 예시코드

- ArticleItem 컴퍼넌트가 없어 실행은 않됨
- 새로운 ArticleItem을 만들어 리팩토링해서 사용하는 방법을 권장

```
import { Link, NavLink, Outlet } from "react-router-dom"

const Articles = () => {

  return (
    <div>
      <Outlet />
      <ul>
        <li>
          <ArticleItem id={1} />
          <ArticleItem id={2} />
          <ArticleItem id={3} />
        </li>
      </ul>
    </div>
  )
  };

const ArricleItem = ({id}) => {
```

```
  const activeStyle = {
    color: 'red',
    fontSize: 22,
  }
  return (
    <li>
      <NavLink to={`/articles/${id}`} style={({isActive}) => isActive ?
activeStyle: undefined}>
        게시글 {id}
      </NavLink>
    </li>
  )
}

export default Articles
```

13.6.3 NotFound 페이지 만들기

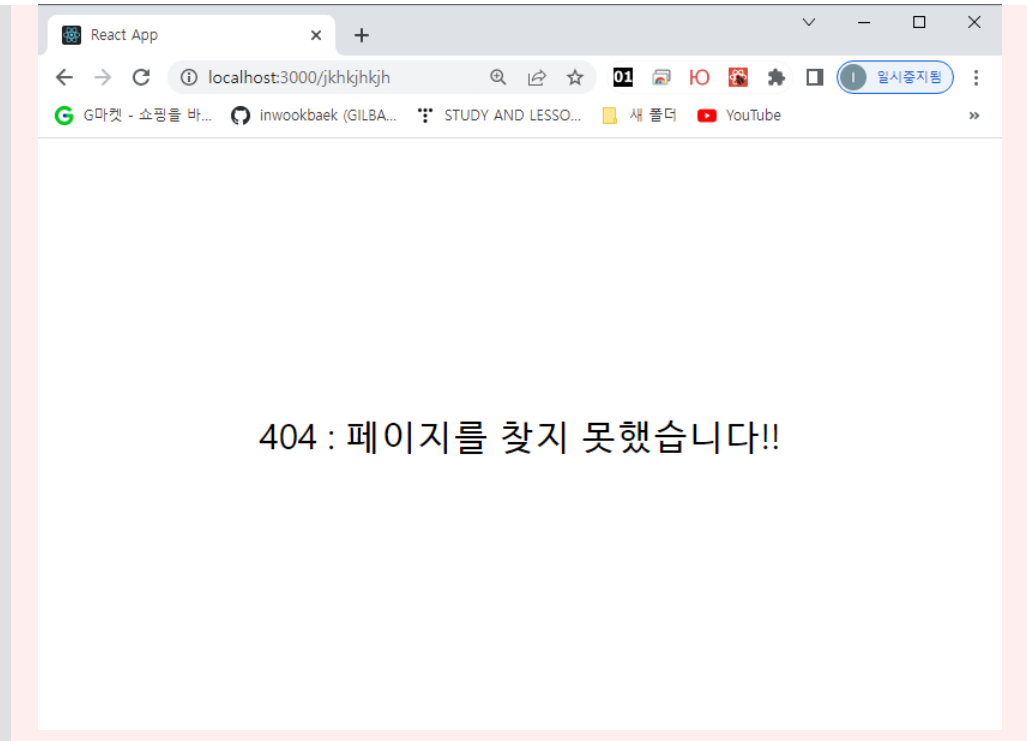
srccpages/NotFound.js

```
import React from 'react'

const NotFound = () => {
  return (
    <div
      style={{
        display: 'flex',
        alignItems: 'center',
        justifyContent: 'center',
        fontSize: 24,
        position: 'absolute',
        width: '100%',
        height: '100%'
      }}
    >
      404 : 페이지를 찾지 못했습니다!!
    </div>
  )
}

export default NotFound;
```

src/pages/Articles.js

```
import { Route, Routes } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';
import Profile from './pages/Profile';
import Articles from './pages/Articles';
import Article from './pages/Article';
import Layout from './Layout';
import NotFound from './pages/NotFound';

function App() {
  return (
    <Routes>
      <Route path='/' element={ <Layout/> }>
```

```
          <Route path="/" element={ <Home />} />
          <Route path="/about" element={ <About />} />
          <Route path="/profiles/:username" element={ <Profile />} />
        </Route>
        <Route path="/articles" element={ <Articles />}>
          <Route path="/articles/:id" element={ <Article />} />
        </Route>
        <Route path="*" element={ <NotFound />}/>
      </Routes>
    );
}

export default App;
```



13.6.4 Navigate 컴퍼넌트

- 이 컴퍼넌트를 보여주는 순간 다른 페이지로 이동할 때 사용 즉, `페이지를 리다이렉트` 할 경우 사용

src/pages/Login.js

```
import React from 'react'

const Login = () => {
  return (
    <div><h3>Login Page!!!</h3></div>
  )
}

export default Login
```

src/pages/MyPage.js

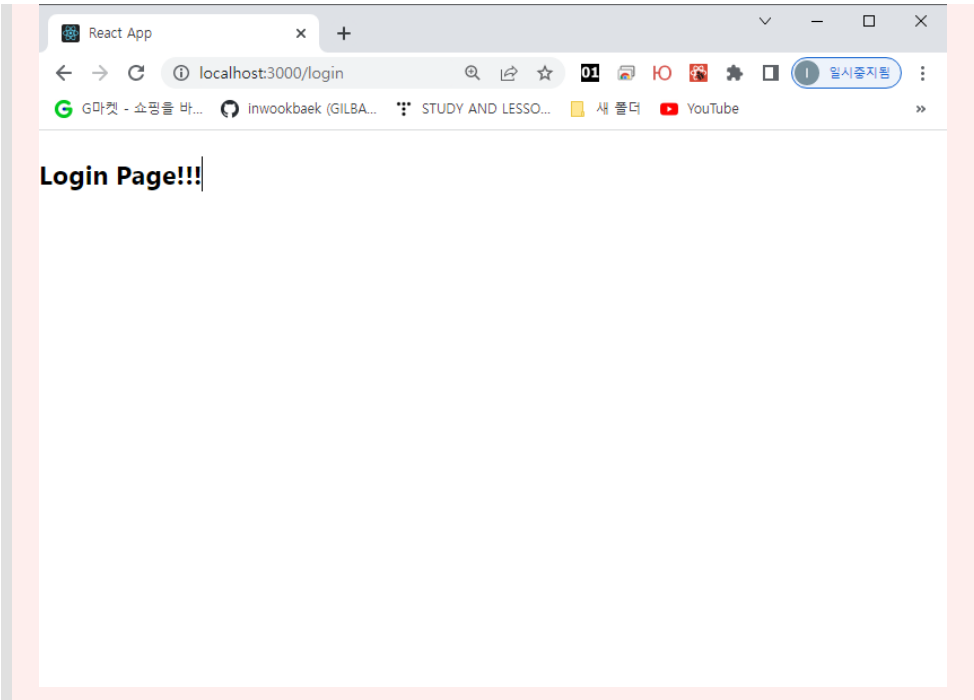
```
import { Navigate } from "react-router-dom";

const MyPage = () => {
```

```
  const isLoggedIn = false;

  if(!isLoggedIn) {
    return <Navigate to='/login' replace={true} />;
  }

  return <div>MyPage!!!!</div>
}

export default MyPage
```

src/App.js

```
import { Route, Routes } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';
import Profile from './pages/Profile';
import Articles from './pages/Articles';
import Article from './pages/Article';
import Layout from './Layout';
import NotFound from './pages/NotFound';
import MyPage from './pages/MyPage';
import Login from './pages/Login';

function App() {
  return (
    <Routes>
      <Route path='/' element={ <Layout/> }>
        <Route path="/" element={ <Home />} />
        <Route path="/about" element={ <About />} />
        <Route path="/profiles/:username" element={ <Profile />} />
      </Route>
      <Route path="/articles" element={ <Articles />}>
        <Route path="/articles/:id" element={ <Article />} />
      </Route>
      <Route path="/login" element={ <Login />}/>
      <Route path="/mypage" element={ <MyPage />}/>
      <Route path="*" element={ <NotFound />}/>
    </Routes>
  );
}

export default App;
```

- `localhost:3000/mypage`로 입력하는 순간 `login`페이지로 이동 확인


React App
localhost:3000/login
일시중지됨
G마켓 - 쇼핑을 바...
inwookbaek (GILBA...
STUDY AND LESSO...
새 폴더
YouTube
Login Page!!!