React Router

안 화 수

React Router

❖ React Router

1. 라우팅이란 사용자가 요청한 URL에 따라 해당 URL에 맞는 페이지를 보여주는 것을 의미한다.
2. 리액트에서는 라우팅 관련 라이브러리가 많이 있는데, 이중 가장 많이 사용되는 리액트 라우터(React Router)를 사용해보자.
3. React-Router는 각각의 url에 따라 선택된 데이터를 하나의 페이지에서 렌더링 해주는 라이브러리 이다.

SPA

- **SPA(Single Page Application) 방식**

  MPA(Multi Page Application) 방식처럼 여러 개의 페이지를 사용하여 새로운 페이지를 로드하는 방식이 아니라, 하나의 페이지 안에서 필요한 데이터만 가져와서 개발하는 방식이다.

React Router

- **프로젝트 생성**

```
npx    create-react-app    router01
```

React Router

- **React Router 설치**

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

React Router

React Router의 태그

- <BrowserRouter>

  React Router기능을 구현할 때 가장 바깥쪽에 사용하는 태그이다.

- <Routes>

  <Routes>태그는 여러 Route를 감싸고 있고 그 중 규칙에 일치하는 Route 하나만을 렌더링 시켜주는 역할을 한다.

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

React Router

React Router의 태그

- <Route>

  1. Route태그는 path속성에 경로, element속성에는 컴포넌트를 넣어 준다.

  2. 형식 : <Route path="경로" element={<컴포넌트 />} />

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

React Router

React Router의 태그

- <Link>

1. 웹 페이지에서는 원래 링크를 보여줄 때 a태그를 사용 하지만 a태그는 클릭시 페이지를 새로 불러오기 때문에 React Router에서는 사용하지 않는다.
2. Link 태그는 a태그와 비슷한 기능이지만, History API를 통해 브라우저 주소의 경로만 바꾸는 기능이 내장되어 있다.
3. 형식 : <Link to="경로">링크명</Link>

```
<Link to="/">홈</Link>
<Link to="/about">소개</Link></li>
```

4. Link 태그를 사용하기 위해서는 import 를 하고 사용해야 된다.

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

React Router

- **React Router의 태그**

```
<BrowserRouter>
    <h1>hi react</h1>
    <ul>
        <li><Link to="/">홈</Link></li>
        <li><Link to="/about">소개</Link></li>
    </ul>  <hr />
    <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
    </Routes>
</BrowserRouter>
```

React Router

예제1: src/App.js (1/2)

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

React Router

- 예제1: src/App.js (2/2)

```
function App() {
  return (
  <BrowserRouter>
          <h1>hi react</h1>
          <ul>
                  <li><Link to="/">홈</Link></li>
                  <li><Link to="/about">소개</Link></li>
          </ul> <hr />
          <Routes>
                  <Route path="/" element={<Home />} />
                  <Route path="/about" element={<About />} />
          </Routes>
  </BrowserRouter>
  );
}
export default App;
```

React Router

- **예제1: src/Home.js**

```
import React from "react";

function Home(){
    return(
        <div>
            <h1>홈</h1>
            <p>가장 먼저 보여지는 페이지</p>
        </div>
    );
};

export default Home;
```

React Router

- **예제1: src/About.js**

```
import React from "react";

function About(){
    return (
        <div>
                <h1>소개</h1>
                <p>라우터 프로젝트 소개 페이지</p>
        </div>
    );
};

export default About;
```

React Router

- **프로젝트 생성**

```
npx    create-react-app    router02
```

React Router

- **React Router 설치**

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

React Router

예제2: src/App.js (1/2)

```
import React from 'react';
import Header from './Header';
import Main from './Main';
import Product from './Product';
import NotFound from './NotFound';
import Footer from './Footer';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
```

React Router

- **예제2: src/App.js (2/2)**

```
const App = () => {
        return (
        <div className='App'>
            <BrowserRouter>
                <Header /><hr></hr>
                <Routes>
                        {/* 슬래시(/)로 요청하면 Main.js 컴포넌트를 실행 */}
                        <Route path="/" element={<Main />}></Route>
                        {/* /product/1로 요청하면 Product.js 컴포넌트를 실행 */}
                        <Route path="/product/*" element={<Product />}></Route>
                        {/* 일치하는 라우트가 없는경우 NotFound.js 컴포넌트를 실행 */}
                        <Route path="*" element={<NotFound />}></Route>
                </Routes>
                <Footer /><hr></hr>
            </BrowserRouter>
        </div>
        );
};
export default App;
```

React Router

- **예제2: src/Header.js**

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

function Header(props) {
    return (
        <div>
            <Link to="/">
                <h1>헤더입니다.(Header.js)</h1>
            </Link>
        </div>
    );
}

export default Header;
```

React Router

- **예제2: src/Main.js**

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

const Main = (props) => {
        return (
            <div>
                <h3>안녕하세요. 메인페이지 입니다.(Main.js)</h3>
                <ul>
                    <Link to="/product/1"><li>1번상품</li></Link>
                    <Link to="/product/2"><li>2번상품</li></Link>
                </ul>
            </div>
        );
};

export default Main;
```

React Router

- **예제2: src/Product.js**

```
import React from 'react';

const Product = (props) => {
    return (
        <div>
            <h3>상품 페이지입니다.(Product.js)</h3>
        </div>
    );
}

export default Product;
```

React Router

- **예제2: src/NotFound.js**

```
import React from 'react';

const NotFound = () => {
    return (
        <div>
            404 Error
        </div>
    );
};

export default NotFound;
```

React Router

- **예제2: src/Footer.js**

```
import React from "react";

const Footer = () =>{

    return (
        <div>
            <h1>Footer 파일 입니다.</h1>
        </div>
    )
}

export default Footer;
```

React Router

- **프로젝트 생성**

```
npx    create-react-app    router03
```

React Router

- **React Router 설치**

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

React Router

- **router02 파일을 router03에 복사**

App.js - 수정

Header.js

Main.js

Product.js - 수정

Footer.js

NotFound.js

React Router

URL 파라미터와 쿼리 스트링

URL 파라미터와 쿼리 스트링을 이용해서 유동적으로 값을 전달할 수 있다.

- URL 파라미터 : /product/1
- 쿼리 스트링 : /product?product=1&searchKeyword=productName

URL 파라미터

URL 파라미터

1. URL 파라미터는 페이지의 주소를 정의할때, 유동적인 값을 전달하기 위한 문법이다.
2. /product/:productId 와 같이 경로에 : 를 사용하여 설정 한다.
3. URL 파라미터가 여러개인 경우엔 /product/:productId/:productName 과 같은 형태로 설정할 수 있다.

   App.js

   ```
   <Route path="/product/:productId" element={<Product />}></Route>
   ```

4. productId를 받을 때는 다음과 같이 사용한다. – Product.js

```
import { useParams } from 'react-router-dom';
//    const { 파라미터명 } = useParams();
      const { productId } = useParams();
```

URL 파라미터

예제3: src/App.js (1/2) - 수정

```
import React from 'react';
import Header from './Header';
import Main from './Main';
import Product from './Product';
import NotFound from './NotFound';
import Footer from './Footer';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
```

URL 파라미터

- **예제3: src/App.js (2/2)** - 수정

```
const App = () => {
        return (
        <div className='App'>
            <BrowserRouter>
                <Header /><hr></hr>
                <Routes>
                        {/* 슬래시(/)로 요청하면 Main.js 컴포넌트를 실행 */}
                        <Route path="/" element={<Main />}></Route>
                        {/* /product/1로 요청하면 Product.js 컴포넌트를 실행 */}
                    <Route path="/product/:productId" element={<Product />}></Route>
                        {/* 일치하는 라우트가 없는경우 NotFound.js 컴포넌트를 실행 */}
                        <Route path="*" element={<NotFound />}></Route>
                </Routes>
                <Footer /><hr></hr>
            </BrowserRouter>
        </div>
        );
};
export default App;
```

URL 파라미터

- **예제3: src/Header.js**

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

function Header(props) {
    return (
        <div>
            <Link to="/">
                <h1>헤더입니다.(Header.js)</h1>
            </Link>
        </div>
    );
}

export default Header;
```

URL 파라미터

- **예제3: src/Main.js**

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

const Main = (props) => {
        return (
            <div>
                <h3>안녕하세요. 메인페이지 입니다.(Main.js)</h3>
                <ul>
                    <Link to="/product/1"><li>1번상품</li></Link>
                    <Link to="/product/2"><li>2번상품</li></Link>
                </ul>
            </div>
        );
};

export default Main;
```

URL 파라미터

- **예제3: src/Product.js - 수정**

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

const Product = (props) => {
//1. const {파라미터명} = useParams();
     const {productId} = useParams();
//2. const 변수명 = useParams().파라미터명;
     const param = useParams().productId;
    return (
        <div>
            <h3>{productId}번 상품 페이지입니다.(Product.js)</h3>
            <h3>{param}번 상품 페이지입니다.(Product.js)</h3>
        </div>
    );
}
export default Product;
```

URL 파라미터

- **예제3: src/NotFound.js**

```
import React from 'react';

const NotFound = () => {
    return (
        <div>
            404 Error
        </div>
    );
};

export default NotFound;
```

URL 파라미터

- **예제3: src/Footer.js**

```
import React from "react";

const Footer = () =>{

    return (
        <div>
            <h1>Footer 파일 입니다.</h1>
        </div>
    )
}

export default Footer;
```

쿼리 스트링

쿼리 스트링

1. 쿼리 스트링을 이용해서 유동적으로 값을 전달할 수 있다.
2. URL 주소가 아래와 같을때

   http://localhost:3000/product/1?search=productName&q=demo

   물음표(?) 뒤에 search=productName&q=demo가 쿼리스트링이 되는것이다.
3. 쿼리스트링을 나누는 방법은 &를 사용해서 구분한다.
4. 쿼리 스트링 값을 받을때 자주 사용하는 훅(hooks)

   useLocation

   useSearchParams

   useNavigate

쿼리 스트링 - useLocation

- **useLocation**

1. useLocation()은 useParams()와 동일하게 현재 페이지의 쿼리스트링이 반환된다.
2. useLocation 작성은 const location = useLocation(); 으로 작성한다.
3. useLocation에서 지원하는 속성

   pathname : 현재 주소의 경로(쿼리스트링 제외 ?앞의값)

   search : 맨앞에 ? 문자를 포함한 쿼리스트링 값

   state : 페이지로 이동할때 임의로 넣을수 있는 상태값

   key : location의 고유값, 초기에는 default이며 페이지가 변경될 때 마다 고유의값이 생성된다.

쿼리 스트링 - useLocation

- **예제3: src/Product.js(1/2) – 수정**

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

const Product = () => {
    const {productId} = useParams();
    const location = useLocation();
```

쿼리 스트링 - useLocation

❖ **예제3: src/Product.js(2/2) – 수정**

```
return (
        <div>
            <h3>{productId}번 상품 페이지 입니다.</h3>
            <h3>useLocation</h3>
            <ul>
                <li>hash : {location.hash}</li>
                <li>pathname : {location.pathname}</li>
                <li>search : {location.search}</li>
                <li>state : {location.state}</li>
                <li>key : {location.key}</li>
            </ul>
        </div>
    );
}
export default Product;
```

useLocation

- hash :
- pathname : /product/1
- search : ?search=productName&q=demo
- state :
- key : default

쿼리 스트링 - useSearchParams

- **useSearchParams**

1. useSearchParams는 다음과 같이 사용한다.

```
const [searchParams, setSearchParams]=useSearchParams();
```

2. useSearchParams는 get()과 set()을 사용할 수 있다.

http://localhost:3000/product/1?search=productName&q=demo

```
const search = searchParams.get('search');
const q = searchParams.get('q');
```

쿼리 스트링 - useSearchParams

- **예제3: src/Product.js(1/2) – 수정**

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

const Product = () => {
    const {productId} = useParams();
    const [searchParams, setSearchParams] = useSearchParams();
    const search = searchParams.get('search');
    const q = searchParams.get('q');
```

쿼리 스트링 - useSearchParams

❖ **예제3: src/Product.js(2/2) – 수정**

```
return (
        <div>
            <h3>{productId}번 상품 페이지 입니다.</h3>
            <h3>useSearchParams</h3>
            <ul>
                <li>search : {search}</li>
                <li>q : {q}</li>
            </ul>
        </div>
    );
}
export default Product;
```

useSearchParams

- search : productName
- q : demo

쿼리 스트링 - useNavigate

- **useNavigate**

  1. Link태그를 사용하지 않고 뒤로가기 등에 사용하는 훅(Hooks) 이다.

쿼리 스트링 - useNavigate

- **예제3: src/Product.js(1/2) – 수정**

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

const Product = () => {
    const {productId} = useParams();
    const navigate = useNavigate();
```

쿼리 스트링 - useNavigate

❖ **예제3: src/Product.js(2/2) – 수정**

```
return (
        <div>
            <h3>{productId}번 상품 페이지 입니다.</h3>
            <h3>navigate</h3>
            <ul>
                <li><button onClick={() => navigate(-2)}>Go 2 pages back</button></li>
                <li><button onClick={() => navigate(-1)}>Go back</button></li>
                <li><button onClick={() => navigate(1)}>Go forward</button></li>
                <li><button onClick={() => navigate(2)}>Go 2 pages forward</button></li>
                <li><button onClick={() => navigate('/')}>Go Root</button></li>
            </ul>
        </div>
    );
}
export default Product;
```