

AI 활용 빅데이터분석 풀스택웹서비스 SW 개발자 양성과정

React



부산대학교 소프트웨어교육센터
PUSAN NATIONAL UNIVERSITY SOFTWARE EDUCATION CENTER



리액트 라우터

- 브라우저의 주소를 다루기 위한 라이브러리
 - 브라우저의 URL을 감지하고 이에 따라 적절한 컴포넌트를 렌더링
- 프로젝트 생성 후 라우터 설치
 - `npm install react-router-dom`

Package.json

```
{
  "name": "rt",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.1",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  }
}
```

라우터 적용

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

- BrowserRouter

- React Router의 기본 구성 요소이며, URL을 관리하는 데 사용
- 애플리케이션의 최상위 레벨에 위치하며, React Router를 초기화

- Routes

- 라우팅 정보를 정의하는 컴포넌트
- 하위에 Route 컴포넌트를 렌더링하며, URL에 따라 매칭되는 Route 컴포넌트를 렌더링

- Route

- URL과 컴포넌트를 매핑하는 컴포넌트
- path 속성과 component 속성으로, path에 맞는 URL이 요청되면 component 속성에 지정된 컴포넌트를 렌더링

라우터 적용

```
const RouteMain = () => {
```

```
  return (
```

```
    <BrowserRouter>
```

```
      <main className='container'>
```

```
        <RouteNav />
```

```
        <Routes>
```

```
          <Route path='/' element={<RouteHome />} />
```

```
          <Route path='/page1' element={<RoutePage1 />} />
```

```
          <Route path='/page2' element={<RoutePage2 />} />
```

```
        </Routes>
```

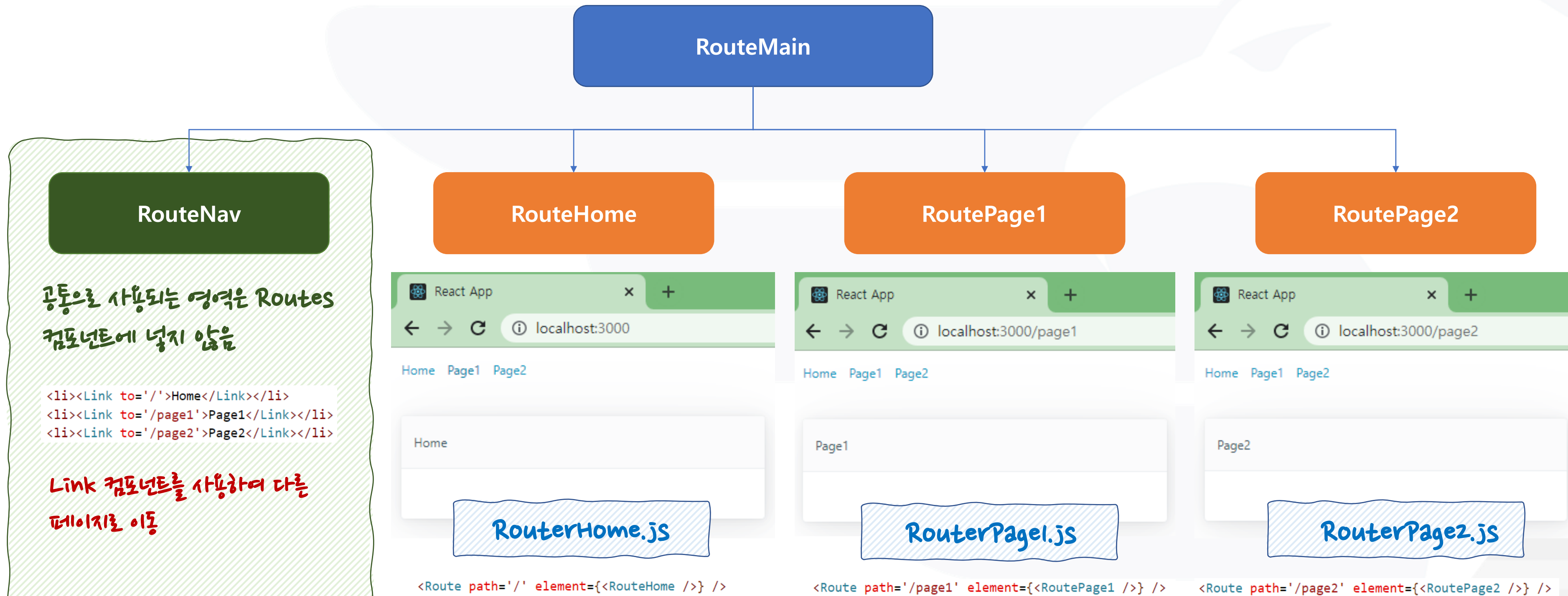
```
      </main>
```

```
    </BrowserRouter>
```

```
  );
```

```
}
```

라우터 적용



URL 파라미터

• 주소의 경로에 값을 넣어 전송

URL 파라미터로 사용할 파라미터를 :뒤에 작성

- 여러 개의 파라미터를 사용할 경우 /:i1/:i2 형식으로 사용

```
<Routes>
  <Route path="/page1/:item1" element={<RoutePage1 />} />
  <Route path="/page2" element={<RoutePage2 />} />
</Routes>
```

```
<h2>[URL 파라미터]</h2>
```

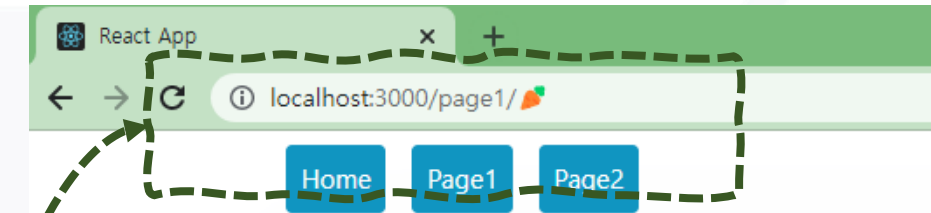
```
<ul>
```

```
  <li><Link to="/page1/🍎">사과🍎</Link></li>
```

```
  <li><Link to="/page1/🍌">바나나🍌</Link></li>
```

```
  <li><Link to="/page1/🥕">당근🥕</Link></li>
```

```
</ul>
```



Page1

🥕 과일이 아닙니다.

useParams 훅을 사용

```
import { useParams } from "react-router-dom";
const RoutePage1 = () => {
  const item = useParams().item1;
  const fruits = ['🍎', '🍌'];
  return (
    <article>
      <header><h1>Page1</h1></header>
      <h2>{item}{ fruits.includes(item)
        ? '과일입니다.'
        : '과일이 아닙니다.' }</h2>
    </article>
  );
}
```

useNavigate

- event가 발생할 때, url을 조작할 수 있는 interface

[useNavigate]

사과 🍏

바나나 🍌

당근

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

```
const RouteHome = () => {  
  const navigator = useNavigate();  
  return (
```

```
<article>
  <header><h1>Home</h1></header>
  <h2>[URL 파라미터]</h2>
  <ul>...
</ul>
  <h2>[useNavigate]</h2>
  <div className='grid'>
    <button onClick={() => nav}>
    <button onClick={() => nav}>
    <button onClick={() => nav}>
  </div>
</article>
```

클릭

클릭이벤트가 발생할 때 url 조작

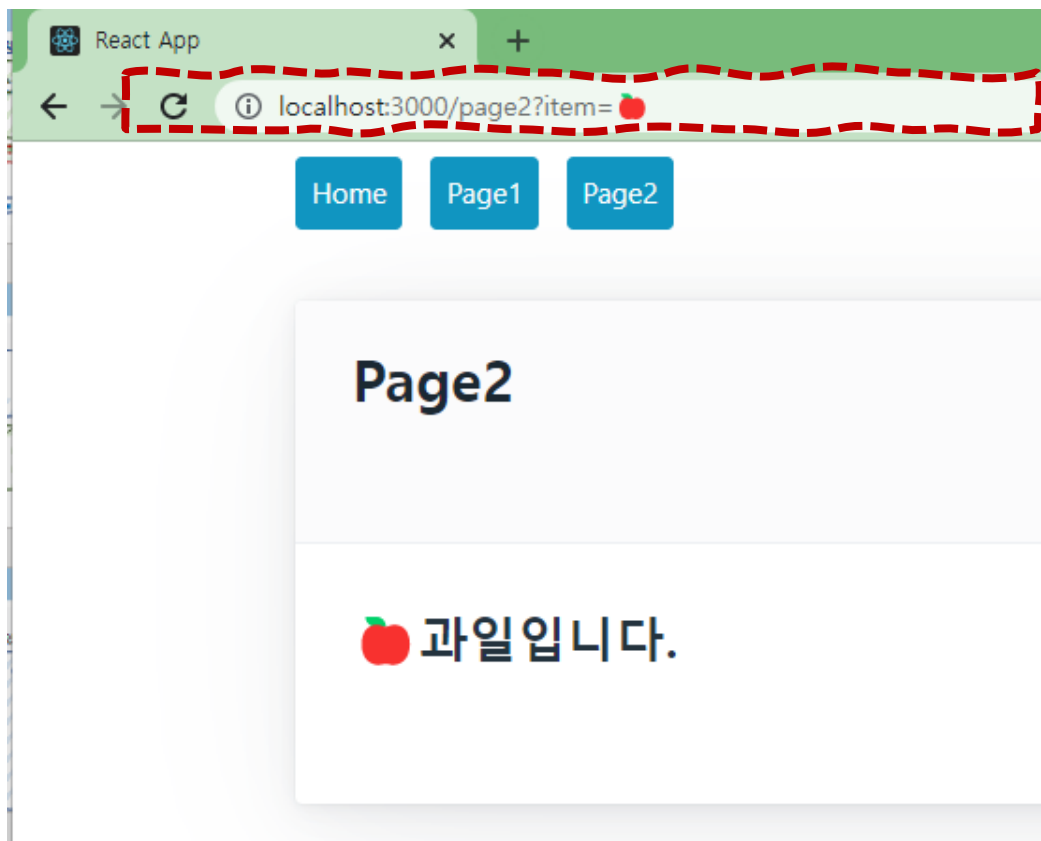
```

navigator('/page2?item=🍎')}> 사과🍎 </button>
navigator('/page2?item=🍌')}> 바나나🍌 </button>
navigator('/page2?item=🥕')}> 당근🥕 </button>

```

쿼리스트링

- Route 컴포넌트에 별도 설정할 필요없이 `useLocation` 훅을 사용하면 전달된 쿼리스트링을 확인할 수 있음
- 쿼리스트링을 오브젝트로 변환하기 위해서는 `query-string` 패키지 설치
 - `npm install query-string`



```
import { useLocation } from "react-router-dom";
import qs from 'query-string';

const RoutePage2 = () => {
  const location = useLocation().search ;
  const item = qs.parse(location).item ;

  const fruits = ['🍎', '🍌'] ;
  return (
    <article>
      <header><h1>Page2</h1></header>
      <h2>{item}{ fruits.includes(item)
        ? '과일입니다.'
        : '과일이 아닙니다.' }</h2>
    </article>
  );
}
```

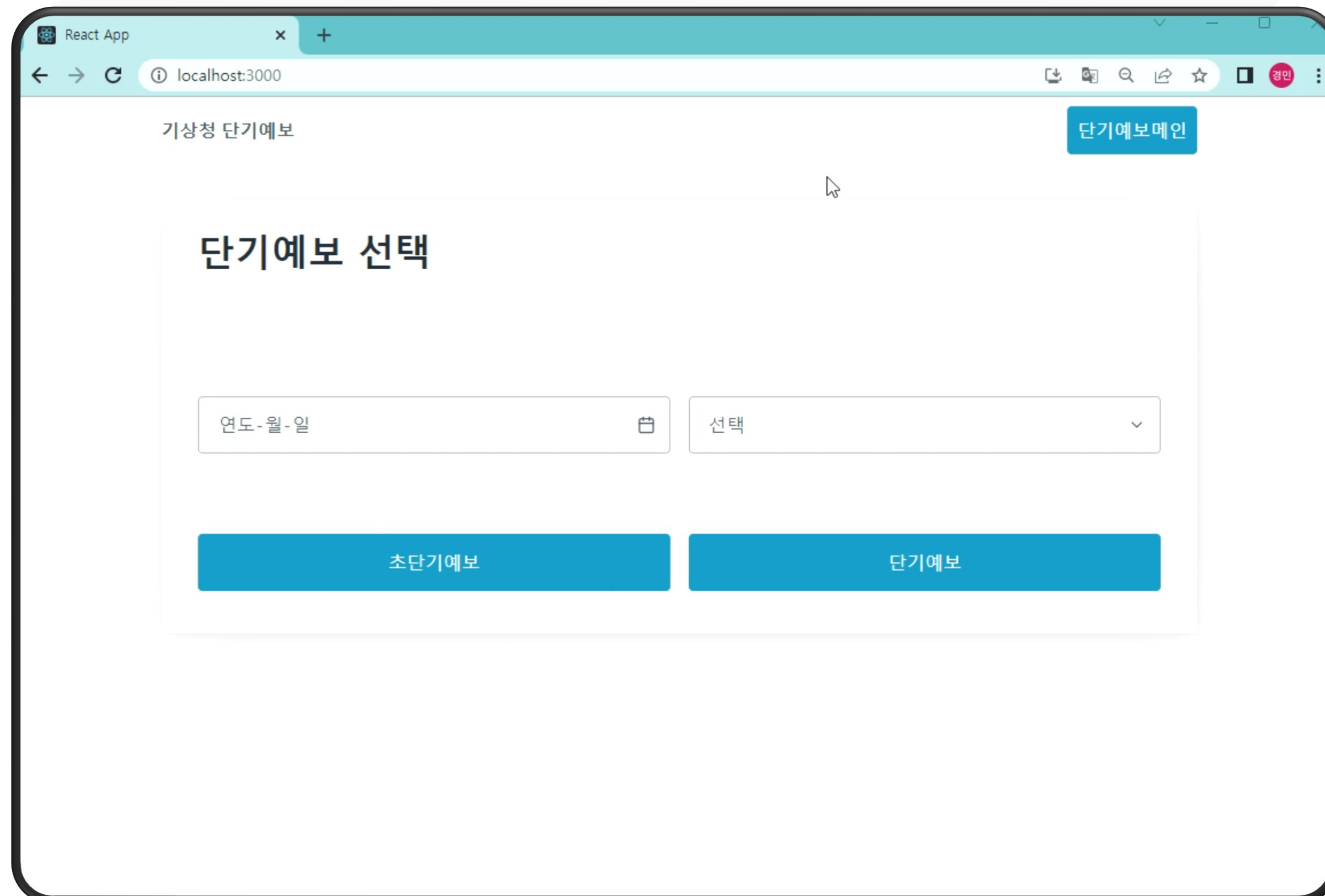
URL 주소로 전달되는 값은 `useLocation` 훅을 이용하여 전달된 쿼리스트링을 받음

useSearchParams

- v6 부터 쿼리스트링 제어

```
import { useSearchParams, useLocation } from "react-router-dom" ;  
export default function Rpage2() {  
  const loc = useLocation().search ;  
  console.log(loc)  
  
  const [sParams] = useSearchParams() ;  
  const item1 = sParams.get('item1') ;  
  const item2 = sParams.get('item2') ;  
  
  return (  
    <div className="flex flex-wrap justify-between items-center mx-auto max-w-screen-xl">  
      page2 : {item1} {item2}  
    </div>  
  )  
}
```

실습과제



React App x +

localhost:3000

기상청 단기예보

단기예보메인

단기예보 선택

연도-월-일

선택

초단기예보

단기예보

• 기상청_단기예보 ((구)_동네예보) 조회서비스

기본정보

데이터명	기상청_단기예보 ((구)_동네예보) 조회서비스	상세설명	
서비스유형	REST	심의여부	자동승인
신청유형	개발계정 활용신청	처리상태	
활용기간	2022-09-21 ~ 2024-09-21		

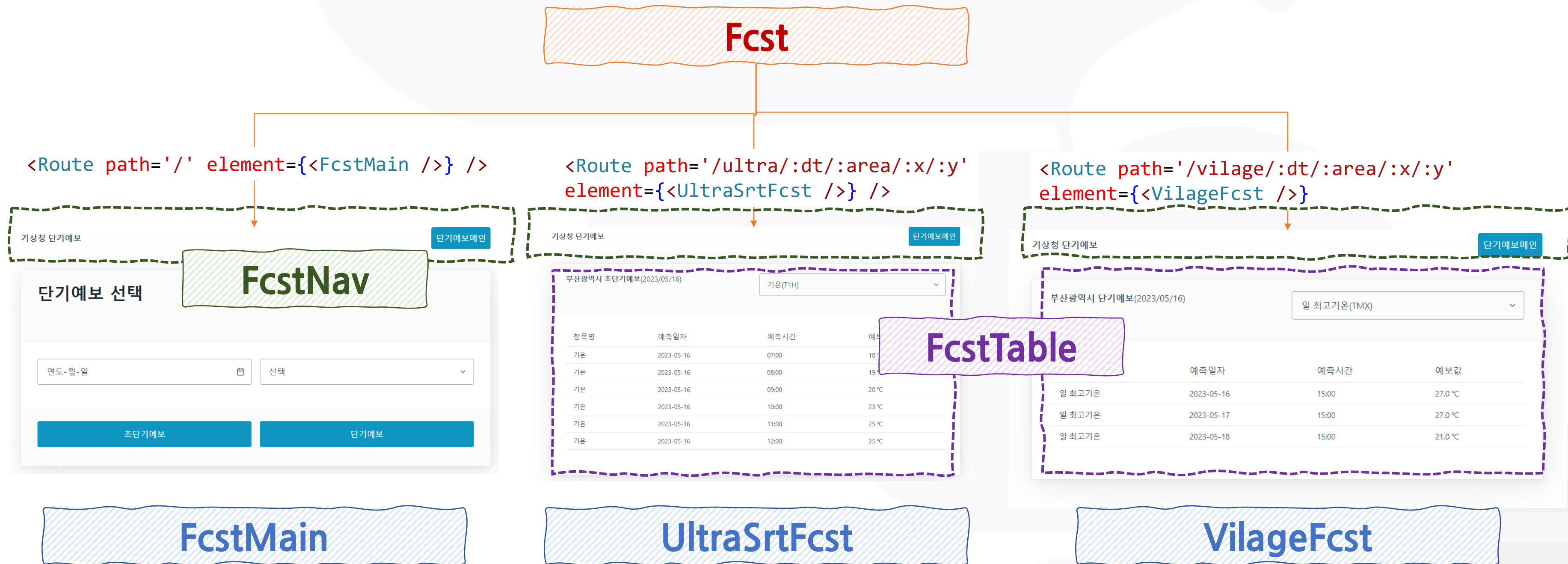
서비스정보

참고문서	기상청41_단기예보 조회서비스_오픈API활용가이드_최종.zip
데이터포맷	JSON+XML
End Point	http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0

활용신청 상세기능정보

NO	상세기능	설명	일일 트래픽	미리보기
1	초단기실황조회	실황정보를 조회하기 위해 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 조회 조건으로 자료구분코드, 실황값, 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 정보를 조회하는 기능	10000	<div>확인</div>
2	초단기예보조회	초단기예보정보를 조회하기 위해 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 조회 조건으로 자료구분코드, 예보값, 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 정보를 조회하는 기능	10000	<div>확인</div>
3	단기예보조회	단기예보 정보를 조회하기 위해 발표일자, 발표시각, 예보지점 X좌표, 예보지점 Y 좌표의 조회 조건으로 발표일자, 발표시각, 자료구분문자, 예보 값, 예보일자, 예보시각, 예보지점 X 좌표, 예보지점 Y 좌표의 정보를 조회하는 기능	10000	<div>확인</div>
4	예보버전조회	단기예보정보조회서비스 각각의 오퍼레이션(초단기실황, 초단기예보, 단기예보)들의 수정된 예보버전을 파악하기 위해 예보버전을 조회하는 기능	10000	<div>확인</div>

화면구성



화면구성2

