9강

Router-DOM

&

LINK

Router-DOM

Router - DOM

➢ Router – DOM이란 ?

- 리액트에서 페이지 이동(경로 전환) 을 담당하는 라이브러리이다.
- 전통적인 웹은 A.html, B.html처럼 페이지마다 별도 파일을 불러왔다.
- 하지만 리액트는 하나의 HTML 안에서 화면만 바꿔주는 방식(SPA) 으로 동작한다.
- 즉, 실제로는 페이지가 바뀌지 않지만 화면이 바뀌는 것처럼 보이게 만든다.

➢ SPA (Single Page Application)

- "하나의 페이지 안에서 여러 화면을 표현하는 웹앱 구조"
- 리액트, 뷰(Vue), 앵귤러(Angular) 등이 SPA 방식이다.
- 새로고침 없이 빠르게 화면이 바뀌는 것이 특징.
- HTML5의 History API를 이용해 브라우저 주소(URL)만 변경한다.

Router – DOM 설치

➢ Router – DOM 설치 방법

- VSC -> 새터미널 열기 -> Git Bash 선택
- 현재 React에서 작업하는 Project 폴더인 my_app 폴더로 이동한다.
- npm install react-rotuer-dom
- 단, react-router-dom 안에 react-router이 포함되어 있으므로 따로 설치할 필요가 없다.

➢ Router-DOM 설치 확인

package.json 파일을 열어 아래처럼 되어 있으면 설치가 완료된 상태임

```
"dependencies": {
  "react": "^19.1.1",
  "react-dom": "^19.1.1",
  "react-router-dom": "^7.9.5"
},
```

Router – DOM 사용 방법

- React Router 기본 구조
  - React Router 사용 할 때는 3개의 기본 컴포넌트를 반드시 알아야 한다.

컴포넌트	역할
<BrowserRouter>	라우터 기능 전체를 감싸는 컨테이너
<Routes>	여러 개의 경로(Route)를 묶는 영역
<Route>	URL 경로(path)와 보여줄 컴포넌트(element)를 연결

- Router DOM은 웹사이트의 길 안내표와 같은 역할을 한다.
  - <BrowserRouter> : "지도 전체를 감싸는 틀"
  - <Routes> : "길들을 하나씩 나열하는 구역"
  - <Route> : "실제로 가는 길(주소) 하나하나"

➢ *여기서 잠깐*

① Src -> pages 폴더 생성한다.

② Pages -> Home.jsx 파일을 생성한 후 아래처럼 작성한다.

③ Pages -> About.jsx 파일을 생성한 후 아래처럼 작성한다.

④ App.jsx 파일로 이동 한다.

⑤ import { BrowserRouter, Route, Routes } from 'react-router-dom' 반드시 import 해야 한다.

⑥ 페이지로 연결할 Home함수와 About함수를 반드시 import 해야 한다.

< Home.jsx >

```
export default function Home(){
  return <h2>여기는 Home Page</h2>
}
```

< About.jsx >

```
export default function About() {
  return <h2>여기는 About page</h2>;
}
```

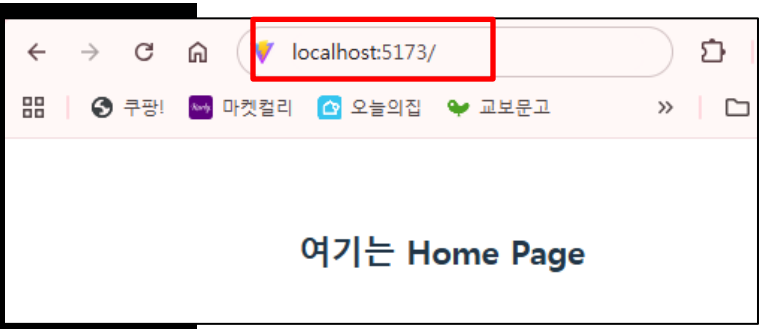
➤ 여기서 잠깐

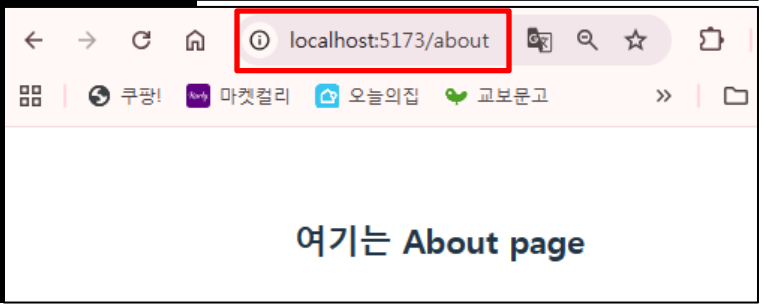
① **<Route>**
- "주소 하나 = 페이지 하나"
- 'URL 경로'와 '보여줄 컴포넌트'를 연결하는 역할을 한다.

② **path 속성**
- path="/" → 홈페이지 주소 , /는 웹사이트의 기본 주소 (루트) **예) http://localhost:5173/ 또는 https://내사이트.com/**
- path="/about" → 소개 페이지 주소, **예: http://localhost:5173/about**
- path는 주소줄에 어떤 URL을 쳤을 때를 의미한다.

③ **element 속성**
- 그 주소에서 보여줄 화면(컴포넌트)
- element={<Home />} : / 주소에 오면 Home 컴포넌트를 화면에 보여줘라
- element={<About />} : /about 주소에 오면 About 컴포넌트를 보여줘라

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


function App() {


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





➢ *왜 App.jsx Router DOM을 연결하는 가*

- 반드시 App.jsx 안에서만 Router를 써야 하는 건 아니다. 하지만 왜 대부분 App.jsx에서 사용해야 하냐면 Router는 "앱 전체를 감싸는 구조"여야 한다. React-router-dom의 핵심 컴포넌트인 <BrowserRouter>는 앱 전체의 라우팅 기능을 담당한다. 그래서 "앱의 최상위" 부분, 즉 모든 페이지를 감싸는 위치에 두는 게 일반적이다.
- 다만, 프로젝트 규모가 커지면 App.jsx가 너무 복잡해지기 때문에 Router를 "다른 파일"에 분리해서 Router.jsx 파일을 따로 만드는 경우도 있다.
- 또 다른 패턴은 main.jsx에서 <BrowserRouter>를 입력해 App.jsx 바깥에서 Router를 감싸기도 한다

➢ 왜 App.jsx Router DOM을 연결하는 가

< main.jsx >

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

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

BrowserRouter (전체 지도)
- Routes (길 목록)
  - Route path="/" → Home 화면
  - Route path="/about" → About 화면

< App.jsx >

```jsx
import { BrowserRouter, 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>
    );
}
```

Link

Link

➢ Link란 ?

- <Link>는 React Router(DOM)에서 제공하는 컴포넌트로, 사용자가 클릭했을 때 브라우저를 새로고침하지 않고 URL을 바꾸고 그에 맞는 컴포넌트를 렌더링하도록 해준다.
- 기본적으로는 <a>처럼 동작하지만 내부적으로 클릭 이벤트를 가로채서 HTML5 History API (pushState / replaceState)를 사용해 SPA 방식으로 이동한다.

➢ 왜 <Link>를 써야 하나? (<a>와의 차이)

- <a href="/about"> → 브라우저가 서버에 새 페이지 요청(전체 페이지 리로드).
- Link는 react-router-dom 패키지에서 제공된다.
- to 속성으로 이동할 경로를 지정한다.
- <Link to="/about">About</Link> → 자바스크립트가 주소만 바꾸고 현재 SPA 내에서 컴포넌트만 교체, 새로고침 없음 → 빠르고 상태 유지됨.
- Link도 반드시 import {Link} from 'react-router-dom';으로 import하여 사용한다.

➢ 여기서 잠깐

① Pages -> Home.jsx 파일을 아래처럼 수정하여 작성한다.

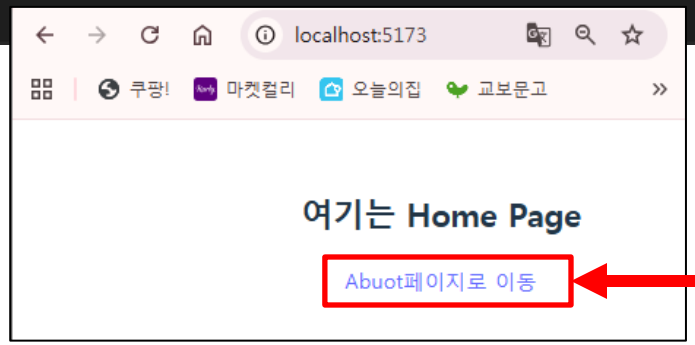
② Pages -> About.jsx 파일을 아래처럼 수정하여 작성한다.

③ App.jsx 파일로 이동 한다.

④ import {Link} from 'react-router-dom' 반드시 import 해야 한다.

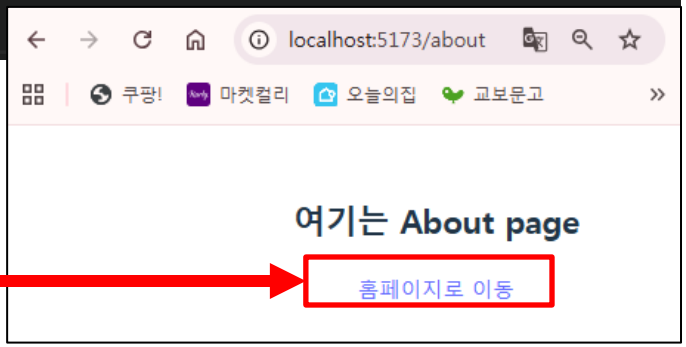
< Home.jsx >

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

export default function Home() {
  return (
    <>
      <h2>여기는 Home Page</h2>
      <Link to="/about">Abuot페이지로 이동</Link>
    </>
  );
}
```



< About.jsx >

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

export default function About() {
  return (
    <>
      <h2>여기는 About page</h2>;
      <Link to="/">홈페이지로 이동</Link>
    </>
  );
}
```



➢ *여기서 잠깐 - useParams() 함수*

- useParams()함수는 URL에서 동적으로 전달되는 값을 가져오기 위해 사용한다.
- React Router의 컴포넌트 내부에 존재한다.
- import { useParams } from "react-router-dom"; 반드시 import 해서 사용해야 한다.
- useParams() 함수를 이용해 URL의 파라미터 값을 받아 올 수 있다.
  <형식>
  const { **받아올 파라미터 값** } = useParams()
  📣반드시 함수처럼 호출 useParams()해야 하고, 호출하지 않고 useParams만 쓰면 undefined가 된다.

➢ *여기서 잠깐*

① Pages -> ProdApp.jsx 파일을 생성하고 아래 prodItem 오브젝트 배열을 이용해 작성한다.

```
const prodItem = [
 { id: 1, name: '노트북', price: 1200000 },
 { id: 2, name: '키보드', price: 50000 },
 { id: 3, name: '마우스', price: 35000 },
];
```

② Pages -> ProDetail.jsx 파일을 생성하고 아래처럼 작성한다.

③ ProdApp.jsx 에 <Link to={`/product/${item.id}`}>와 useParams()함수를 이용해 각 상품의 id를 Prodetail.jsx 파일에서 받아오도록 작성한다.

④ App.jsx 파일로 이동 한다.

➢ 여기서 잠깐

< ProdApp.jsx >

```jsx
export default function ProdApp() {
  const prodItem = [
    { id: 1, name: '노트북', price: 1200000 },
    { id: 2, name: '키보드', price: 50000 },
    { id: 3, name: '마우스', price: 35000 },
  ];


  return (
    <div>
      <h1>상품 목록</h1>
      <ul>
        {prodItem.map((Pitem) => (
          <li key={Pitem.id} >
            <Link to={`/product/${Pitem.id}`}>
              <span>
                {Pitem.name}-{Pitem.price}
              </span>
            </Link>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

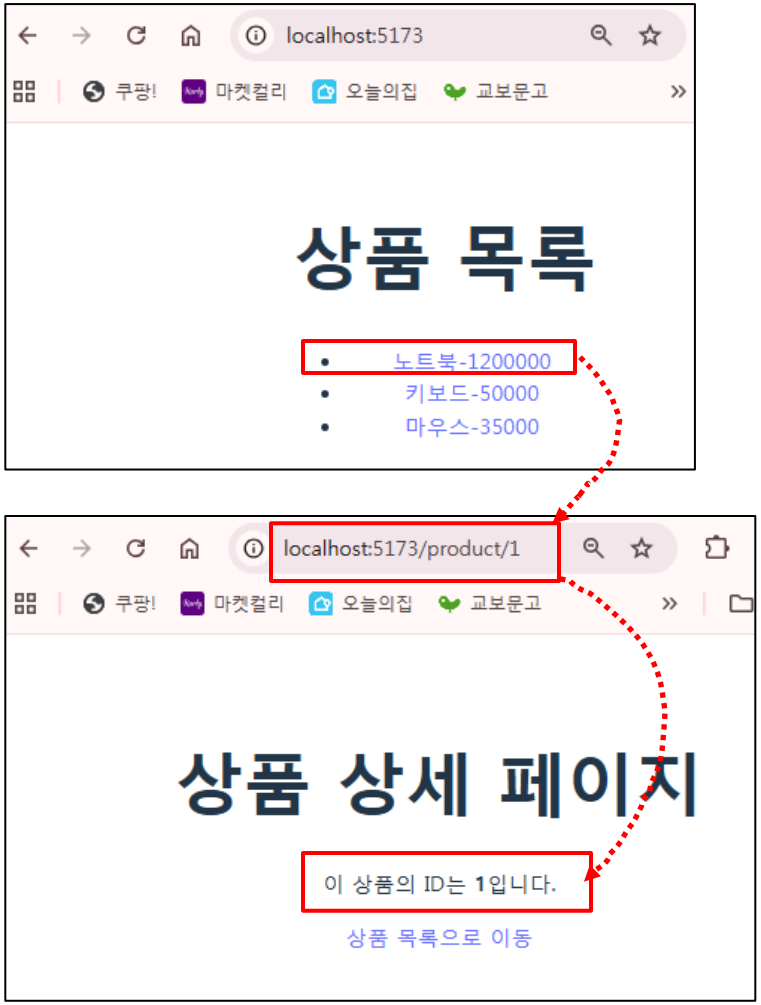
< ProDetail.jsx >

```jsx
export default function ProDetail() {
  // useParams()이용하기
  // URL의 http://localhost:5173/product/1 의 맨 뒤의 1을 파라미터 값으로 받아오는 함수 이다.
  const { id } = useParams();


  return (
    <div>
      <h1>상품 상세 페이지</h1>
      <p>
        이 상품의 ID는 <strong>{id}</strong>입니다.
      </p>
      <Link to="/">상품 목록으로 이동</Link>
    </div>
  );
}
```

➤ 여기서 잠깐

< App.jsx >

```
import ProdApp from './pages/ProdApp';
import ProDetail from './pages/ProDetail';

function App() {

  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<ProdApp />} />
        <Route path="/product/:id" element={<ProDetail />} />
      </Routes>
    </BrowserRouter>
  );
}
```



문제] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://fakestoreapi.com/products JSON 데이터를 fetch ~ then ~ catch ~ finally이용해 비동기 방식으로 가져와 작성한다.

② FakeStore 상품 목록 페이지와 상세 페이지를 React Router로 연결하는 프로젝트를 작성하시오.

③ Fakestore.jsx 에서 상품 목록을 Link 로 만들어 클릭 시 상세페이지로 이동하도록 작성하시오.

④ FakestoreDetail.jsx 파일을 새로 만들어, URL의 id 값을 이용해 해당 상품 상세 정보 출력하시오.

⑤ FakestoreDetail.jsx 파일에 [상품목록으로 이동]버튼 클릭하면 Fakestore.jsx 상품 목록으로 이동되도록 작성하시오.

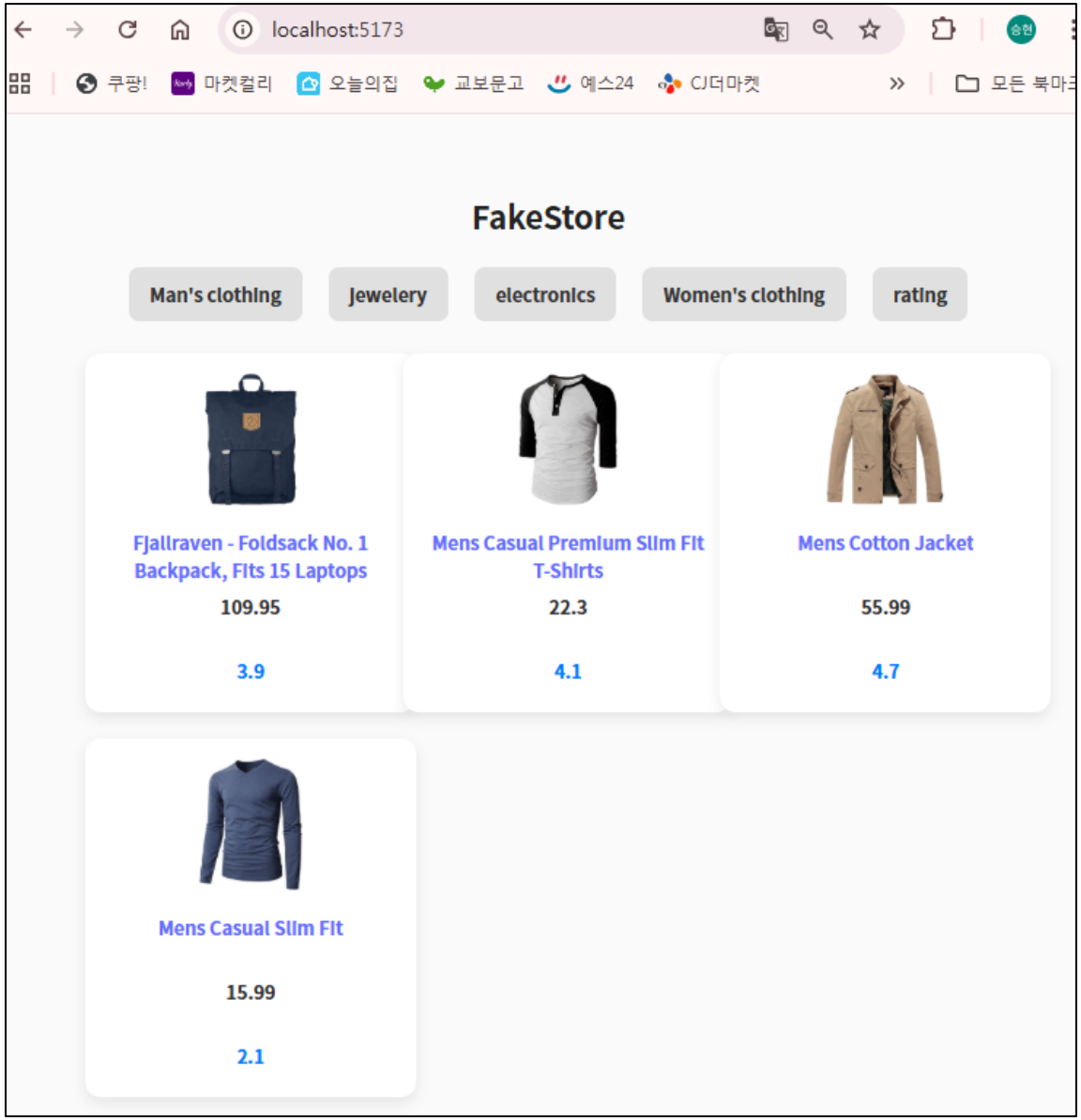
⑥ fakestore.css 파일을 새로 만들어 CSS를 작성하시오.

⑦ useState(), useEffect(), useParams(), Link를 이용해 작성하시오.

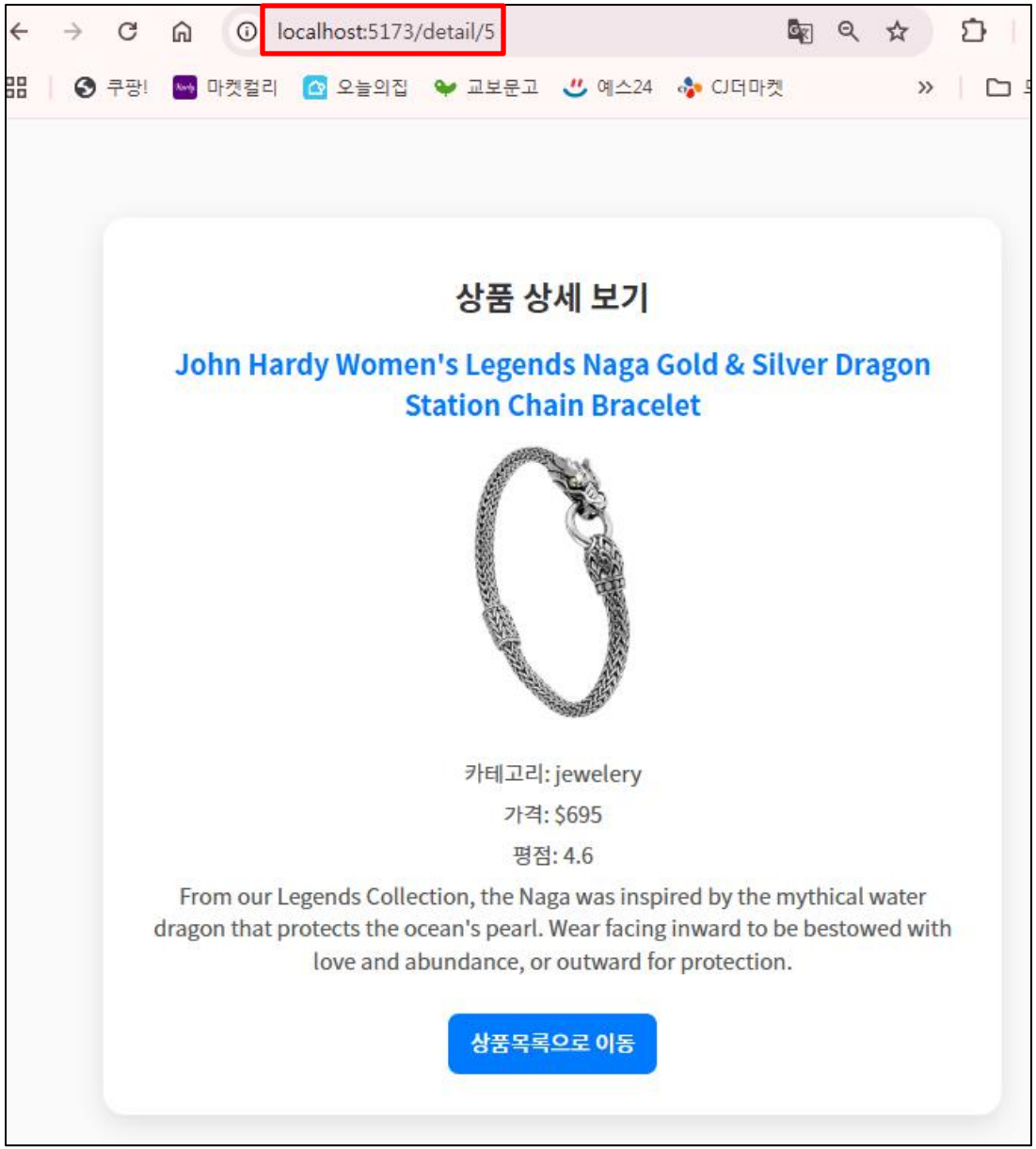
⑧ src 폴더안에 -> Pages 폴더 생성 -> Fakestore.jsx, FakestoreDetail.jsx, fakestore.css 파일을 생성하여 작성하시오.

⑨ Src 폴더안에 -> App.jsx 에서 BrowserRouter, Routes, Route 설정하여 적성하시오.

<상품 목록 출력 화면>


localhost:5173
쿠팡!
마켓컬리
오늘의집
교보문고
옥션24
CJ더마켓
모든 북마크
FakeStore
Man's clothing
Jewelery
electronics
Women's clothing
rating
FJallraven - Foldsack No. 1 Backpack, Fits 15 Laptops
109.95
3.9
Mens Casual Premium Slim Fit T-Shirts
22.3
4.1
Mens Cotton Jacket
55.99
4.7
Mens Casual Slim Fit
15.99
2.1

<상품 상세 보기 출력>


localhost:5173/detail/5
쿠팡!
마켓컬리
오늘의집
교보문고
예스24
CJ더마켓
상품 상세 보기
John Hardy Women's Legends Naga Gold & Silver Dragon Station Chain Bracelet
카테고리: jewelery
가격: $695
평점: 4.6
From our Legends Collection, the Naga was inspired by the mythical water dragon that protects the ocean's pearl. Wear facing inward to be bestowed with love and abundance, or outward for protection.
상품목록으로 이동

연습 문제

문제] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://dummyjson.com/recipes JSON 데이터를 fetch ~ then ~ catch ~ finally이용해 비동기 방식으로 가져와 작성한다.
② 탭 메뉴로 cuisine(예: "Italian", "Asian", "Mexican" 등) 별 필터링 기능 이용해 작성한다.
③ 탭 버튼 예시: "All", "Italian", "Asian", "Mexican" 등 (데이터에 있는 cuisine 값 기준)
④ 정렬 버튼 혹은 토글 기능: rating 순으로 내림차순 정렬.로 터된 상태에서 정렬 되도록 작성한다.
⑤ 목록 출력 시 각 레시피 항목에는 적어도: 이미지, 이름(name), cuisine, rating 보여주도록 작성한다.
⑥ 각 항목을 클릭하면 상세페이지로 이동할 수 있게 react-router-dom의 Link 사용 (예: /detail/:id). 상세페이지 컴포넌트 (예: RecipeDetail)로 작성한다.
⑦ Recip.css 파일을 새로 만들어 CSS를 작성하시오.
⑧ useState(), useEffect(), useParams(), Link를 이용해 작성하시오.
⑨ src 폴더안에 -> Pages 폴더 생성 -> Recipes 폴더 생성 -> RecipData.jsx, RecipeDetail.jsx, Recip.css RecipeList.jsx 파일을 생성하여 작성하시오.
⑩ src 폴더안에 -> App.jsx 에서 BrowserRouter, Routes, Route 설정하여 적성하시오.

전체 페이지

레시피 목록

All | Italian | Asian | American | Mexican | Mediterranean | Pakistani | Japanese | Moroccan | Korean | Greek | Thai | Indian | Turkish | Smoothie | Russian | Lebanese | Brazilian | Rating 순

Classic Margherita Pizza

요리 유형: Italian

평점: 4.6

Vegetarian Stir-Fry

요리 유형: Asian

평점: 4.7

Chocolate Chip Cookies

요리 유형: American

평점: 4.9

Chicken Alfredo Pasta

요리 유형: Italian

평점: 4.9

Mango Salsa Chicken

요리 유형: Mexican

평점: 4.9

Quinoa Salad with Avocado

요리 유형: Mediterranean

평점: 4.4

Tomato Basil Bruschetta

요리 유형: Italian

평점: 4.7

Beef and Broccoli Stir-Fry

요리 유형: Asian

평점: 4.7

Caprese Salad

요리 유형: Italian

평점: 4.6

Shrimp Scampi Pasta

요리 유형: Italian

평점: 4.3

레시피 상세

Classic Margherita Pizza

요리 유형: Italian

난이도: Easy

서빙 수: 4

칼로리: 300

평점: 4.6 (후기 98건)

조리 방법

- Preheat the oven to 475°F (245°C).
- Roll out the pizza dough and spread tomato sauce evenly.
- Top with slices of fresh mozzarella and fresh basil leaves.
- Drizzle with olive oil and season with salt and pepper.
- Bake in the preheated oven for 12-15 minutes or until the crust is golden brown.
- Slice and serve hot.

목록으로 돌아가기

좋아요 기능 추가

레시피 목록

All | Italian | Asian | American | Mexican | Mediterranean | Pakistani | Japanese | Moroccan | Korean | Greek | Thai | Indian | Turkish | Smoothie | Russian | Lebanese | Brazilian | Rating 순

Classic Margherita Pizza

요리 유형: Italian

평점: 4.6

❤ 좋아요 0

Vegetarian Stir-Fry

요리 유형: Asian

평점: 4.7

❤ 좋아요 0

Chocolate Chip Cookies

요리 유형: American

평점: 4.9

❤ 좋아요 0

Chicken Alfredo Pasta

요리 유형: Italian

평점: 4.9

❤ 좋아요 0

Mango Salsa Chicken

요리 유형: Mexican

평점: 4.9

❤ 좋아요 0

Quinoa Salad with Avocado

요리 유형: Mediterranean

평점: 4.4

❤ 좋아요 0

Tomato Basil Bruschetta

요리 유형: Italian

평점: 4.7

❤ 좋아요 0

Beef and Broccoli Stir-Fry

요리 유형: Asian

평점: 4.7

❤ 좋아요 0

Caprese Salad

요리 유형: Italian

평점: 4.6

❤ 좋아요 0

Shrimp Scampi Pasta

요리 유형: Italian

평점: 4.3

❤ 좋아요 0