5차시 스크립트

실습 - themealDB를 활용한 레시피 앱 만들기

- React 프로젝트 구조를 익히기
- useState, useEffect사용방법 익히기
- API 활용방법 익히기

1. React 프로젝트 생성

```
npx create-react-app recipe-app
cd recipe-app
npm install axios
```

2. 디렉토리 구조 잡기

```
src/
├── components/
│   └── RecipeList.js
│   └── RecipeCard.js
├── pages/
│   └── Home.js
├── App.js
├── api.js
```

3. API연동 및 데이터 가져오기

```
import axios from 'axios';

const BASE_URL = 'https://www.themealdb.com/api/json/v1/1';

export const getRandomRecipe = async () => {
  const res = await axios.get(`${BASE_URL}/random.php`);
```

```
  return res.data.meals[0];
};
```

Axios는 **Promise 기반의 HTTP 클라이언트 라이브러리**로, 브라우저와 Node.js 환경 모두에서 사용 가능하다. 주로 React 등 프론트엔드 프레임워크에서 **API 요청**을 할 때 사용된다.

AJAX → Fetch → Axios 순으로 발전

✅ 왜 Axios를 사용하는가?

1. **간결한 문법**
   - `fetch`보다 코드를 짧고 명확하게 작성 가능
2. **자동 JSON 변환**
   - 응답 데이터를 자동으로 JSON으로 변환
3. **요청/응답 인터셉터 지원**
   - 요청 전/후 가로채기 가능 → 예: 토큰 자동 추가
4. **타임아웃, 응답 상태 처리**
   - 요청 제한 시간, 에러 처리 등을 쉽게 설정
5. **브라우저 간 호환성 우수**

✅ 주요 기능 정리

```
axios.get(url)
axios.post(url, data)
axios.put(url, data)
axios.delete(url)

axios.defaults.baseURL = 'https://api.example.com';
axios.defaults.headers.common['Authorization'] = 'Bearer TOKEN';

axios.interceptors.request.use(config => {
  // 요청 전에 작업 가능 (예: 토큰 삽입)
  return config;
```

```
});

axios.interceptors.response.use(response => {
  // 응답 가공 가능
  return response;
});

// src/api/axiosInstance.js
import axios from 'axios';

const instance = axios.create({
  baseURL: 'https://www.themealdb.com/api/json/v1/1',
  timeout: 3000,
});

export default instance;
```

✅ 1. AJAX란?

- *AJAX (Asynchronous JavaScript and XML)**는 웹 페이지를 **전체 새로고침 없이** 서버와 비동기적으로 데이터를 주고받는 기술이다.

> 즉, 페이지를 다시 로드하지 않고 일부 데이터만 서버에서 받아와 UI를 갱신할 수 있게 하는 기술 방식.

📌 AJAX 예시 (기존 방식: XHR)

```
const xhr = new XMLHttpRequest();
xhr.open("GET", "https://api.example.com/data");
xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    console.log(JSON.parse(xhr.responseText));
  }
};
xhr.send();
```

✅ 2. Fetch API

📌 정의

`fetch()`는 **XHR의 현대적 대체제**로, **Promise 기반**으로 작성되어 더 깔끔하고 유지보수하기 쉬움.

📌 기본 문법

✅ GET 요청

```
fetch('https://api.example.com/data')
  .then(response ⇒ {
    if (!response.ok) throw new Error("Network error");
    return response.json(); // 수동으로 JSON 파싱 필요
  })
  .then(data ⇒ console.log(data))
  .catch(error ⇒ console.error(error));
```

- **AJAX**는 개념이고 기술들의 조합임. 핵심은 **비동기 데이터 전송**.
- **XHR**은 오래된 방식, 복잡하고 불편함.
- **Fetch API**는 깔끔하지만 에러 처리나 JSON 파싱이 수동임.
- **Axios**는 Fetch보다 사용성 좋고 기능이 풍부함 (인터셉터, 취소 토큰 등).

4. useEffect로 데이터 가져오기

```
import React, { useEffect, useState } from 'react';
import { getRandomRecipe } from '../api';

const Home = () ⇒ {
  const [recipe, setRecipe] = useState(null);

  useEffect(() ⇒ {
    getRandomRecipe().then(data ⇒ setRecipe(data));
  }, []);

  return (
    <div>
```

```
      {recipe && (
        <div>
          <h1>{recipe.strMeal}</h1>
          <img src={recipe.strMealThumb} alt={recipe.strMeal} width="300" />
          <p>{recipe.strInstructions}</p>
        </div>
      )}
    </div>
  );
};

export default Home;
```

5. 컴포넌트 분리

```
const RecipeCard = ({ recipe }) => (
  <div>
    <h2>{recipe.strMeal}</h2>
    <img src={recipe.strMealThumb} alt={recipe.strMeal} width="250" />
  </div>
);
```

6. 향후 진행 방향

1. React router를 통해서 상세페이지로 분리
2. zustand와 같은 state management library연결, 혹은 context API사용