08-simple-ajax

#01. 프로젝트 생성 및 초기화

1) 프로젝트 생성

> 프로젝트 이름은 영어 소문자만 사용 가능함.

```
yarn create react-app 07-hook-event
```

2) 프로젝트 생성 후 기초작업

프로젝트 초기화

대상	작업 내용
src폴더	App.css, App.test.js, index.css, logo.svg, setupTests.js, reportWebVitals.js 삭제
App.js	App.css와 logo.svg에 대한 참조(import) 구문 제거
index.js	index.css와 reportWebVitals.js에 대한 참조(import) 구문 제거 맨 마지막 행에 있는 `reportWebVitals()` 부분 삭제

3) 추가 패키지 설치

패키지 이름	설명
react-router-dom	SPA앱을 만들 때 사용. URL에 따라 실행할 Javascript를 분기한다.
react-helmet-async	`<head>`태그 내의 SEO관련 태그를 설정한다.
sass	sass와 scss 컴파일 기능 제공
styled-components	styled component 지원
styled-components-breakpoints	styled component에서 media query를 쉽게 사용할 수 있게 한다.
dayjs	날짜 처리 기능 제공 (리액트와 직접적인 연관은 없음)
axios	Ajax 요청 라이브러리
react-loader-spinner	로딩바(spinner) 컴포넌트

```
yarn add react-router-dom react-helmet-async sass styled-components
styled-components-breakpoints dayjs axios react-loader-spinner
```

4) Router 적용

index.js

import 구문 추가

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

index.js 파일에서 `<App />`을 `<BrowserRouter><App /></BrowserRouter>`로 변경

App.js

```
import { NavLink, Routes, Route } from "react-router-dom";
```

혹은

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

5) 프로젝트 실행

프로젝트를 VSCode로 열고, `Ctrl` + `~`를 눌러 터미널 실행

```
yarn start
```

#02. 백엔드와의 정보 교환

1) 새로운 내용을 화면에 반영하기 위한 전통적 방법

정적인 HTML 페이지는 새로운 내용을 화면에 표시하기 위해 다른 HTML 페이지로 리다이렉션 하는 것이 유일한 방법이었음.

사용자가 입력한 내용에 대한 조회, 입력, 수정, 삭제 등을 하기 위해서는 `<form>`태그에 action 속성에 명시된 웹 프로그램(ASP,PHP,JSP등)의 URL로 직접 submit 처리를 수행해야 했으며 이 과정에서도 페이지 이동이 발생하게 됨.

이러한 처리 과정때문에 전통적 웹 프로그램은 자신의 동작 결과를 HTML로 출력하여 사용자에게 페이지가 이어지도록 하는 경험을 제공해 왔음.

2) 모던 웹 사이트의 데이터 처리 방법

Ajax라는 방법이 고안되면서 웹 프로그램에게 페이지 이동 없이 백그라운드로 데이터를 전송하고 결과를 전달받을 수 있는 방법이 등장함.

Ajax를 활용하여 `<form>`의 입력 내용을 페이지 이동 없이 웹 프로그램에게 전송할 수 있었기 때문에 웹 프로그램은 처리 결과를 HTML이 아닌 내용만 표현하는 JSON으로 출력함.

HTML페이지는 Javascript를 통해 웹 프로그램이 전달한 JSON을 되돌려 받고 이를 화면에 반영하는 형태로 변모하였음.

이때부터 본격적으로 FrontEnd와 Backend의 개념이 생기기 시작함.

3) Ajax를 처리하는 방법

수많은 방법이 있지만 그 중에서 활발히 활용되는 방법은 axios 라이브러리의 활용임.

```
(async () => {
    let json = null;

    try {
        // ajax 처리 구현
        const response = await axios.요청방법함수('요청URL', 파라미터);
        json = response.data;
    } catch (e) {
        // 에러 발생시 처리
        alert("AJAX 요청에 실패했습니다.");
    }

    if (json != null) {
        // Ajax 처리 결과를 화면에 출력
    }
})();
```

4) RestfulAPI

하나의 URL이 어떤 개체(ex-상품,회원 등)를 의미하고 GET, POST, PUT, DELETE 전송방식에 따라 조회,입력,수정,삭제 기능을 구분하는 구현 형태

대부분 OpenAPI는 Restful API 방식을 따른다.

전송방식	수행할 동작	의미	SQL
POST	입력	Create	INSERT
GET	조회	Read	SELECT
PUT	수정	Update	UPDATE
DELETE	삭제	Delete	DELETE

위의 네가지 방식을 CRUD라 부른다.

즉, Restful API는 CRUD 방식을 따르는 표준.

전송방식에 따른 URL 예시

장바구니를 다루는 Restful API라고 가정할 경우

전송방식	URL	동작	파라미터

전송방식	URL	동작	파라미터
POST	http://localhost:3000/shopping/cart	장바구니에 담기	상품정보는 POST 방식으로 전달한다. 저장된 항목을 의미하는 고유한 일련번호값이 생성된다.
GET	http://localhost:3000/shopping/cart/:일련번호	해당 일련번호에 대한 장바구니 내역 조회	조회대상을 Path 파라미터로 전달한다.
PUT	http://localhost:3000/shopping/cart/:일련번호	해당 일련번호에 대한 장바구니 내역 수정(주문수량 등)	수정할 대상을 의미하는 일련번호는 Path 파라미터로 전달하고 수정할 내용은 PUT 파라미터로 전달한다.
DELETE	http://localhost:3000/shopping/cart/:일련번호	해당 일련번호에 대한 장바구니 항목 삭제	삭제대상을 Path 파라미터로 전달한다.

#03. 컴포넌트 성능 최적화 방법

1) `React.memo()` 사용.

컴포넌트 함수를 export 하기 전 `React.memo()`로 처리한다.

```
import React from 'react';

const News = () => {
    //... 생략 ...
};

// React.memo()를 사용하여 함수형 컴포넌트의 리렌더링 성능을 최적화
export default React.memo(NewsList);
```

2) `React.useCallback()` 사용

이벤트 핸들러 함수에 `React.useCallback()`을 적용한다.

```
const 함수이름 = React.useCallback(e => {
    //... 이벤트 처리 구현
}, []);
```

3) 함수형 상태값 업데이트

`useState()`로 상태값을 업데이트 할 때 상태값이 배열이나 JSON객체 타입일 경우 단순히 값을 전달하는 것이 아니고 함수형으로 처리한다.

적용전

```
const [target, setTarget] = React.useState([]);

// 상태값으로 적용할 배열 생성
const newArray = [...];
setTarget(newArray);
```

적용후

```
const [target, setTarget] = React.useState([]);

setTarget(function(target) {
    // target(원본)을 활용하여 상태값으로 활용할 새로운 배열 생성
    return 새로운배열;
});
```

적용후(축약형)

```
const [target, setTarget] = React.useState([]);
setTarget(target => 새로운배열);
```