리액트(React) 기초

useEffect 훅을 이용한 사이드 이팩트 관리

김경민

사이드 이펙트(Side Effect)

- 사이드 이펙트(Side Effect)
  - 컴포넌트가 렌더링될 때나 특정 상태가 변경될 때 수행해야 하는 외부 작업들
- 사이드 이펙트(Side Effect) 예
  - 데이터 페칭(Data Fetching)
    - 컴포넌트가 마운트될 때 API에서 사용자 데이터를 불러오는 작업
  - DOM 조작(DOM Manipulation)
    - 특정 컴포넌트가 렌더링된 후 포커스를 특정 입력 필드로 이동시키는 작업
  - 타이머 설정 및 비동기 작업
    - setTimeout 또는 setInterval을 사용해 타이머를 설정하는 작업
    - 사용자가 버튼을 클릭한 후 5초 후에 경고 메시지를 표시하는 작업
  - 로컬 스토리지와 같은 브라우저 API 사용
    - 사용자가 입력한 데이터를 로컬 스토리지에 저장해, 페이지를 새로고침 해도 데이터를 유지하는 작업

Hook : useEffect

- 컴포넌트 내에서 랜더링이 수행된 이후 실행되는 메서드
  - 컴포넌트가 렌더링될 때나, 특정 값이 변경될 때 실행되어야 하는 사이드 이펙트를 처리
- useEffect(() => {}, dependency값)
  - useEffect의 두 번째 인자는 의존성 배열로, 이 배열에 포함된 값이 변경될 때마다 첫 번째 인자로 전달된 함수가 다시 실행
  - 의존성 배열이 빈 배열 []이라면, 이 효과는 컴포넌트가 처음 마운트될 때만 실행되고, 그 후에는 실행되지 않음
  - 정리 함수 (Clean-up Function)
    - useEffect 내에서 반환되는 함수는 컴포넌트가 언마운트되거나, 이펙트가 재실행되기 전에 호출되어 리소스를 정리하는 데 사용

Hook : useEffect

- useEffect(() => {}, dependency값)

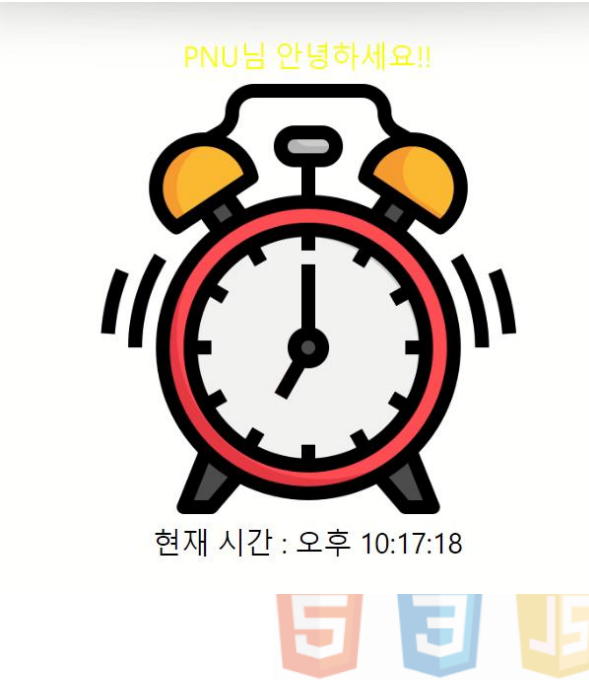
```
//useEffect : defendency가 없을 경우
useEffect(()=>{
    console.log('defendency가 없을 경우', pn) ;
}) ;

//useEffect : defendency가 빈배열
useEffect(()=>{
    console.log('defendency가 빈배열 경우', pn) ;
}, []) ;

//useEffect : defendency배열에 값이 있을 경우
useEffect(()=>{
    console.log('defendency배열에 값이 있을 경우', pn) ;
}, [pn]) ;
```

Hook : useEffect cleanup

- 컴포넌트가 unmount되어 DOM에서 제거될 경우 실행되는 메소드
- useEffect(() => { ... return ()=>{}}, dependency값)

```jsx
import { useState, useEffect } from "react";

function MyClockTime() {
  const [currentTime, setCurrentTime] = useState(new Date()) ;

  useEffect(() => {
    const tm = setInterval(() => {
      setCurrentTime(new Date()) ;
    } , 1000) ;

    return () => {
      clearInterval(tm);
    }
  },[]);

  return (
    <h1>
      현재 시각 : {currentTime.toLocaleTimeString()}
    </h1>
  )
}

export default MyClockTime ;
```

컴포넌트 제거 시
Cleanup 처리



전역 CSS와 모듈 CSS

전역 CSS

src > 03> My.css

```css
.pmy{
  background-color: white;
  color: black;
}
```

모듈 css : 컴포넌트의 스타일이 전역 범위로 확산되지 않고 컴포넌트 내에서만 적용

src > 03> My1.module.css

```css
.pmy{
  color: yellow;
}
```

src > 03> My2.module.css

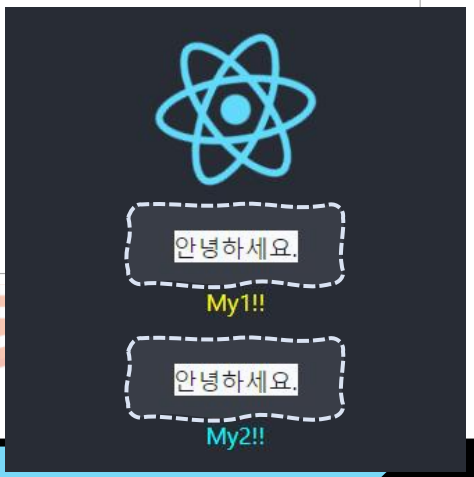
```css
.pmy{
  color: cyan;
}
```

src > 03 > My1.js

```jsx
import './My.css';
import styles from './My1.module.css';
export default function My1() {
  return(
    <div>
      <p className="pmy">안녕하세요. </p>
      <p className={styles.pmy}>My1!!</p>
    </div>
)
}
```

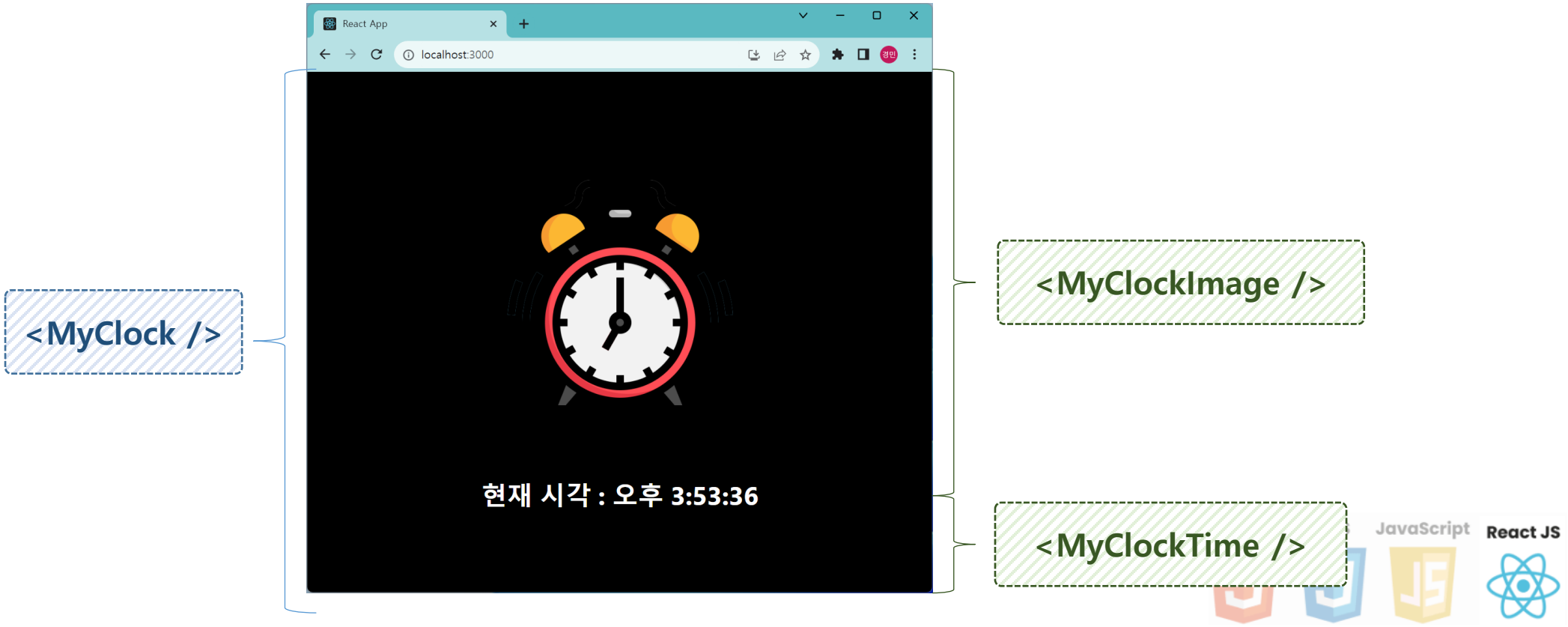
src > 03 > My2.js

```jsx
import styles from './My2.module.css';
export default function My2() {
  return(
    <div>
      <p className="pmy">안녕하세요. </p>
      <p className={styles.pmy}>My2!!</p>
    </div>
)
}
```



해결문제

- 시계가 1초에 한번씩 동작하도록 작성



Fetch API

- JavaScript에서 HTTP 요청을 보내고 응답을 받는 기능을 제공하는 API
  - Promise 기반으로 동작
  - fetch 함수를 사용하여 HTTP 요청을 보내고, 응답이 완료되면 Promise 객체를 반환
  - 응답은 Response 객체로 반환되며, 이 객체를 통해 응답의 상태와 데이터를 처리

```
fetch(url)
  .then((resp) => resp.json())
  .then((data) => {
      const dailyBoxOfficeList = data.boxOfficeResult.dailyBoxOfficeList
      console.log(dailyBoxOfficeList)
  })
  .catch((err) => console.log(err));
```

성공

성공

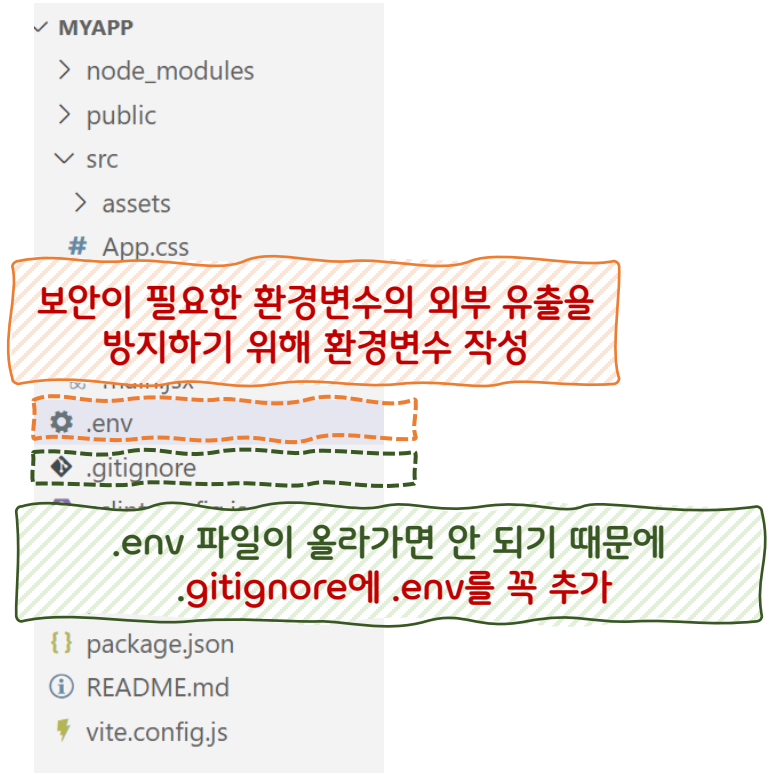
실패

비동기를 동기처럼 다루기

- async 함수
  - 함수 안의 비동기 코드를 Promise 기반으로 간결하게 작성
  - 항상 Promise를 반환
- await 키워드
  - Promise가 완료될 때까지 함수 실행을 잠시 중단
  - 마치 동기 코드처럼 결과 값을 직접 받아옴
- 에러 처리
  - try...catch 블록으로 동기 코드처럼 예외 처리 가능

```
const fetchData = async () => {
  try {
    const response = await fetch(url)
    const data = await response.json();
    console.log("데이터:", data);
  } catch (error) {
    console.error("에러 발생:", error);
  }
};
```

환경변수 파일 추가 및 설정



.env 파일은 최상위 루트에 작성
환경변수명은 반드시 VITE_으로 시작

.env

```
1  VITE_APP_API_KEY = "8qw7
2  VITE_APP_MV_KEY = "2a350
```

리액트 코드에서 환경변수 참조

```
const apiKey = import.meta.env.VITE_APP_MV_KEY;
```

해결문제

- 영화진흥위원회 박스오피스 fetch 활용

https://www.kobis.or.kr/kobisopenapi/homepg/apiservice/searchServiceInfo.do?serviceId=searchDailyBoxOffice

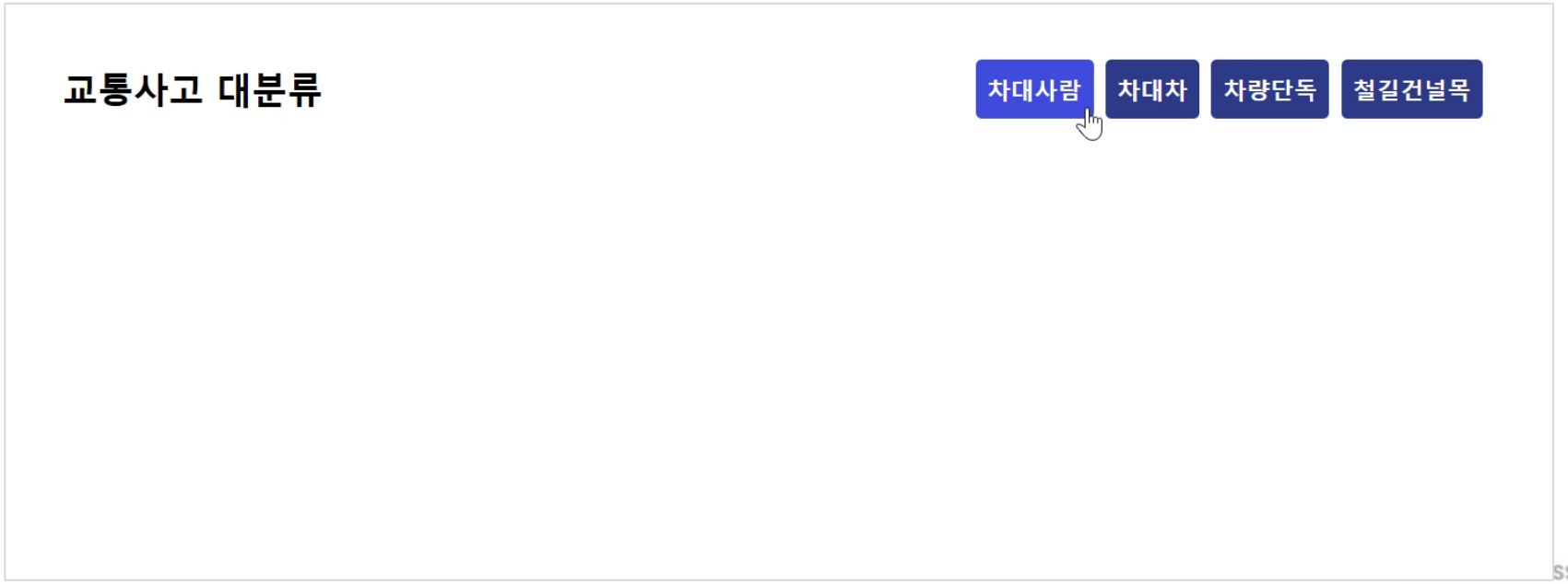
날짜기준 :

순위	영화명	매출액	관객수	누적 매출액	누적 관객수	증감율
1	미키 17	3,573,032,670	356,303	6,120,586,590	610,706	-
2	캡틴 아메리카: 브레이브 뉴 월드	419,486,060	43,733	14,973,919,566	1,508,014	-
3	퇴마록	331,801,340	33,725	2,258,914,060	236,764	-
4	괜찮아 괜찮아 괜찮아!	100,799,100	11,027	386,487,180	44,870	-
5	패딩턴: 페루에 가다!	95,930,900	10,543	742,174,300	87,881	↑1
6	그 시절, 우리가 좋아했던 소녀	88,663,900	9,850	1,124,671,500	127,272	↓1
7	백수아파트	54,650,640	5,921	179,396,220	20,975	↑4
8	컴플리트 언노운	56,717,120	5,579	266,399,020	28,533	↑2
9	첫 번째 키스	43,430,940	5,427	180,550,440	23,071	↓2
10	힘내라 대한민국	47,238,620	5,047	129,880,680	13,822	↓2

[OLD : 2025-02-26]백수아파트 , 상영한 스크린수 : 399 , 상영횟수 : 524

해결문제

- 공공데이터 포털 도로교통공단_사고유형별 교통사고 통계 데이터를 이용하여 다음과 같이 해당 자료를 출력



JavaScript React JS