소프트웨어융합기초1
useEffect/useRef/라우팅

김경민

Hook : useEffect

- 컴포넌트 내에서 랜더링이 수행된 이후 실행되는 메서드
- 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]) ;
```

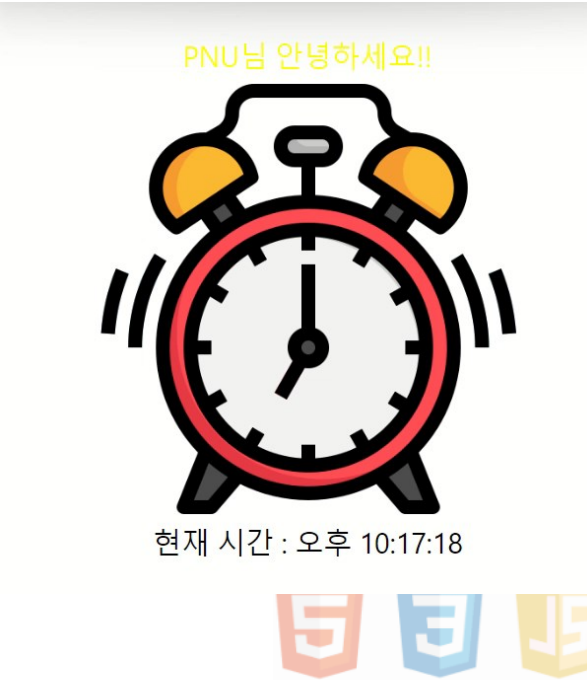
Hook : useEffect cleanup

- 컴포넌트가 unmount되어 DOM에서 제거될 경우 실행되는 메소드

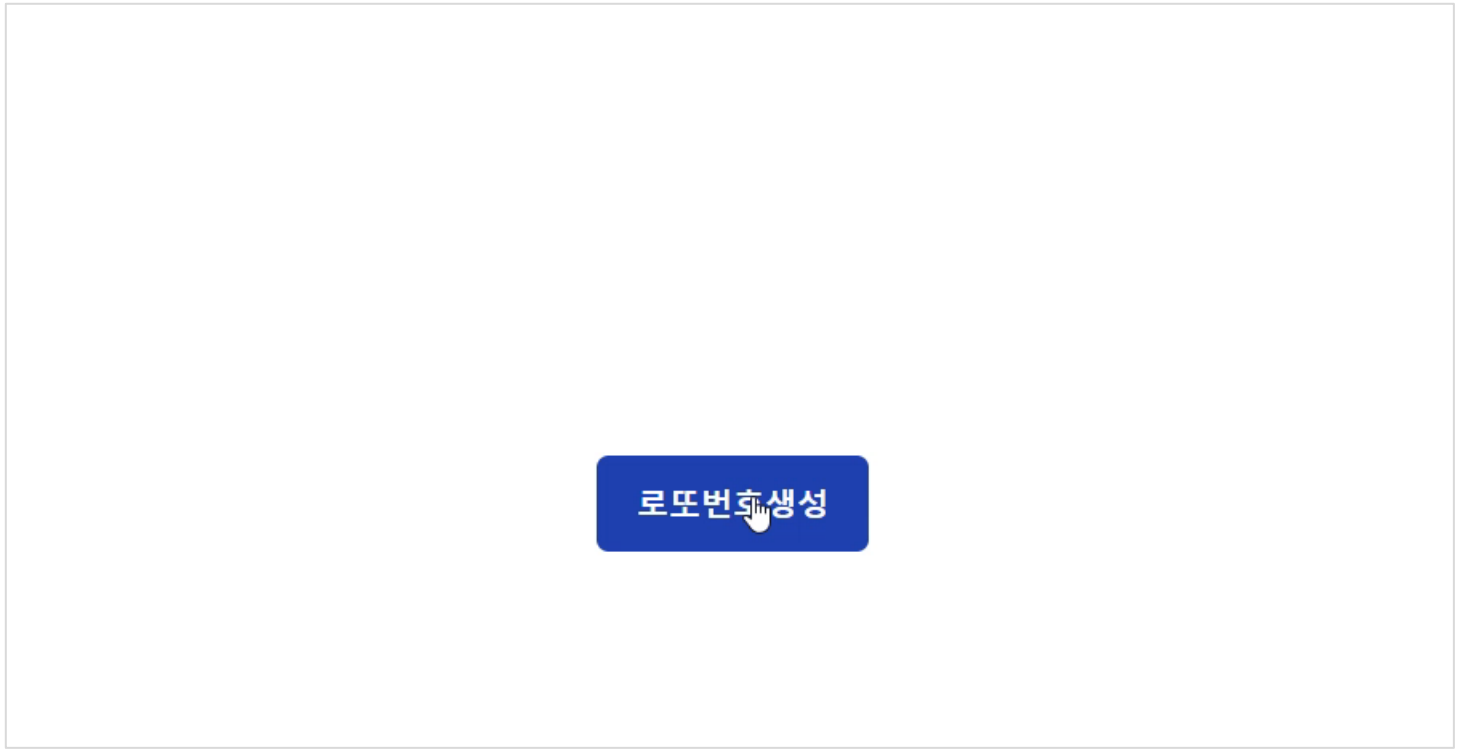
useEffect(() => { ... return ()=>{}}, dependency값)

```
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



해결문제 - 로또 번호 생성



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

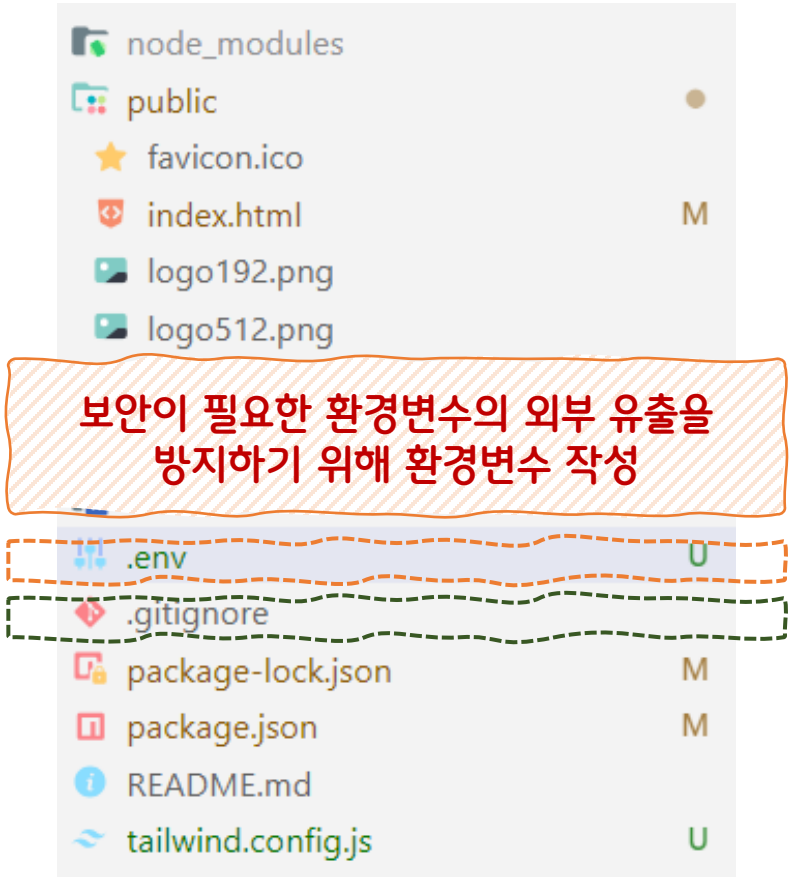
```
fetch(url)
  .then((resp) => resp.json())
  .then((data) => {
      const dailyBoxOfficeList = data.boxOfficeResult.dailyBoxOfficeList
      console.log(dailyBoxOfficeList)
  })
  .catch((err) => console.log(err));
```

성공

성공

실패

환경변수 설정



.env 파일은 최상위 루트에 작성
환경변수명은 반드시 REACT_APP_으로 시작

```
.env
1   REACT_APP_API_KEY = "8qw7g%2FC%2E
```

```
const apikey = process.env.REACT_APP_API_KEY ;
```

.env 파일이 올라가면 안 되기 때문에
gitIgnore에 .env를 꼭 추가

```
.gitignore
20  .env
```

HTML CSS JavaScript React JS

해결문제

- 영화진흥위원회 박스오피스

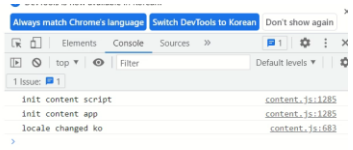
https://www.kobis.or.kr/kobisopenapi/homepg/apiservice/searchServiceInfo.do?serviceId=searchDailyBoxOffice

Hook : useRef()

- .current 프로퍼티로 전달된 인자(initialValue)로 초기화된 변경 가능한 ref 객체를 반환하고 반환된 객체는 컴포넌트의 전 생애주기를 통해 유지
- 변수 관리
  - useRef로 관리하는 값은 값이 변해도 화면이 렌더링되지 않음
- DOM 요소 선택
  - useRef() 를 사용하여 Ref객체를 만들고, 이 객체를 선택하고 싶은 DOM 에 ref값으로 설정
  - Ref객체의 .current값은 DOM 을 가리

Hook : useRef()

- 변수 관리




```
let cnt1 = 0;
const [cnt2, setCnt2] = useState(0) ;
const cnt3 = useRef(0) ;

const showCnt = () => {
    console.log(`cnt1 = ${cnt1}, state cnt2 = 
}
const upCnt1 = () => {
    cnt1 = cnt1 + 1;
    showCnt();
}
const upCnt2 = () => {
    setCnt2(cnt2 + 1);
    showCnt();
}
const upCnt3 = () => {
    cnt3.current = cnt3.current + 1;
    showCnt();
}
```

컴포넌트에서 선언된 변수는 랜더링에 영향을 주지 않음으로 화면에 반영되지 않고 핸더링이 되는 경우 초기화

State변수값이 변경되면 화면이 재 랜더링이 되고 State변수값 유지

current속성에 값저장

ref변수 값은 변경이되어도 재랜더링이 되지 않고 재랜더링이 된 경우에도 이전 값을 그대로 유지

Hook : useRef DOM 핸들링

```jsx
const [item, setItem] = useState([]) ;
const [viewTag, setViewTag] = useState() ;
const txtref = useRef() ;

useEffect(()=>{
    txtref.current.focus();
}, []);

useEffect(()=>{
    console.log(item);
    txtref.current.value = '' ;
    txtref.current.focus();

    setViewTag(item.map((i, idx) =>
        <div key={'i'+idx} className={style.d}>{i}</div
    ));
}, [item]);

const addItem = (e) => {
    e.preventDefault() ;
    let temp = [...item, txtref.current.value] ;
    temp = new Set(temp) ;
    temp = [...temp] ;

    setItem(temp) ;
}

const resetItem = (e) => {
    e.preventDefault() ;
    let temp = [] ;
    setItem(temp) ;
}
```

컴포넌트가 맨처음 랜더링되면 input요소에 포커스

기존 배열에 추가

```jsx
<main className="container">
    <article>
        <header>
        <form>
        <div classNam
            <div>
                <label
                <input ref={txtref} type="text" id="txt1" name="txt1" required />
            </div>
            <div>
                <button onClick={(e) => addItem(e)}>등록</button>
                <button onClick={(e) => resetItem(e)}>취소</button>
            </div>
        </div>
        </form>
        </header>
        <div>
            {viewTag}
        </div>
    </article>
</main>
```

input 요소에 ref속성을 지정하여 useRef 훅으로 지정한 변수 지정

해결문제

- 날짜를 선택하면 해당일의 박스오피스를 보여주기
  - 단, 처음 페이지가 로딩되면 어제 날짜의 박스오피스가 보여지기

순위	영화명	매출액	증감
1	슈퍼 마리오 브라더스	356,316,803	-
2	드림	319,364,559	-
3	존 윅 4	192,839,325	-
4	스즈메의 문단속	125,553,887	-
5	옥수역귀신	68,557,500	▲1
6	리바운드	37,857,757	▼1
7	더 퍼스트 슬램덩크	35,651,083	-
8	렌필드	12,476,199	▲11
9	킬링 로맨스	16,140,706	▼1
10	무명	11,138,275	▼1

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"
  },
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
```

라우터 적용

- **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

```jsx
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>
    );
}
```