리액트 심화 (5) useEffect

1. 유효성 검사

: 입력을 받는 input 값이 우리가 의도한 규칙에 맞게 들어오는지 검사

> `id` : 영문소문자와 숫자로 이루어진 10자 이내
>
> `password` : 영문대소문자, 숫자, 특수문자가 모두 들어간 8~12잘
>
> `email` : xxx@xxx.xx 등 유효한 이메일 양식에 맞게

```
useEffect(() => {
    setFormIsValid(
        enteredEmail.includes('@')
        && enteredPassword.trim().lenght > 6
    );
}, [enteredEmail, enteredPassword]);
```

> `emailChangeHandler` 와 `passwordChangeHandler` 의 공통점이 있다면, state가 바뀔때 실행이 되는 점이 공통점이다.

2. debounce

: 동일한 이벤트가 반복해서 실행되는 경우 일정 시간 지연 시킨 후 처리한다. 일정 시간 안에 동일 이벤트가 다시 발생하면 마지막 발생 시점부터 일정 시간을 지연한다.

> 네이버 검색창 - 알파벳 하나하나 입력하는 것들이 이벤트 발생이 된다
>
> 지도검색 서비스 - 스크롤 할때마다 위도와 경도 값이 바뀐다 - 스크롤 이벤트가 발생이 된다
>
> (동일한 이벤트) 매번 요청을 보내는 과정을 최적화 하기 위해 - debounce
>
> > 검색창, 지도검색 서비스

3. useEffect의 cleanUp 함수

```
useEfffect() => {
    ...
    return () => {
    ...
    }
}, [...]);
```

4. 실습 코드

```
useEffect(() => {
    setTimeout(() => {
        console.log("check validity");
        setFormIsValid(
            enteredEmail.includes('@')
```

```
            && enteredPassword.trim().lenght > 6
    );
    }, 500);

    //cleanUp 함수
    return () => {
        console.log("cleanup");
    };
}), [enteredEmail, enteredPassword]);
```

```
useEffect(() => {
    const identifier = setTimeout(() => {
        console.log("check validity");
        setFormIsValid(
            enteredEmail.includes('@')
            && enteredPassword.trim().lenght > 6
    );
    }, 500);

    //cleanUp 함수
    return () => {
        console.log("cleanup");
        clearTimeout(identifier);
    };
}), [enteredEmail, enteredPassword]);
```

- 유효성 검사는 input 값이 우리가 의도한 규치에 맞게 들어오는지 검사
- debounce는 동일한 이벤트가 반복해서 실행되는 경우 일정 시간 지연 시킨 후 처리