**useEffect를 사용하여 마운트/언마운트/업데이트 시 할 작업 설정하기**

**-**마운트 : 처음 나타났을 때

-언마운트 : 사라질 때

-업데이트 : 특정 props가 바뀔 때

**마운트/언마운트 시 주로 하는 작업**

-마운트

-props로 받은 값을 커폼넌트의 로컬 상태로 설정

-외부 API 요청(REST API 등)

-라이브러리 사용(DE, Video.js 등)

-setInterval 을 통한 반복작업 혹은 setTimeout을 통한 작업 예약

-언마운트

-setInterval, setTimeout 을 사용하여 등록한 작업들 clear하기(clearInterval, claearTimeout)

-라이브러리 인스턴스 제거

**마운트/언마운트 관리**

//UserList.js

import React, {useEffect } from ‘react;

function User({ user, onRemove, onToggle }) {

//User 컴포넌트가 실행될 때마다 밑 useEffect가 실행됨.

**//deps 을 비워둔 버전 : 컴포넌트가 처음 나타날 때만 실행됨.**

useEffect(() => {

console.log(‘컴포넌트가 화면에 나타남’);

return () => {

console.log(‘컴포넌트가 화면에서 사라짐’);

};

}, []);

**//deps를 채운 버전 : 컴포넌트가 처음 나타나거나 지정된 값이 변경될 때마다 실행됨**

**//그리고 언마운트 시에도 호출이 되고, 값이 바뀌기 직전에도 호출이 됨.**

useEffect(() => {

console.log(‘컴포넌트가 화면에 나타남’);

console.log(user);

return () => {

console.log(‘컴포넌트가 화면에서 사라짐’);

console.log(user);

};

}, [user]);

**//deps를 생략한 버전 : 컴포넌트가 리렌더링 될 때마다 호출됨.(업데이트때마다)**

useEffect(() => {

console.log(user);

});

…

-useEffect 함수 설명

-첫번째 파라미터는 함수를 넣음

-함수를 반환할 수 잇는데, 이를 cleanup 함수라고 불림.

-deps가 비어있는 경우에 컴포넌트가 사라질 때 cleanup 함수가 호출됨.

-두번째 파라미터엔 의존값이 들어있는 배열(deps)를 넣음

-만약 **deps 배열을 비우게 되면,** 컴포넌트가 처음 나타날때만 useEffect에 등록한

함수가 호출됨.

-**deps를 넣게 되면**, 컴퍼넌트가 처음 마운트 될때에도 호출이 되고 지정한 값이 바뀔

때에도 호출이 됨. 언마운트 시에도 호출되고 값이 바뀌기 직전에도 호출이 됨.

-즉, 처음 나타날 때 -> 값이 바뀌기 전(cleanup) -> 값이 바뀐 후

-> 사라질 때(cleanup)

-useEffect 안에서 사용하는 상태나, props가 있다면 deps에 넣어주는 게 규칙이다.

-**deps파라미터를 생략**하면 컴포넌트가 리렌더링 될때마다 호출됨.

-> 리액트 컴포넌트는 기본적으로 부모 컴포넌트가 리렌더링 되면 자식

컴포넌트 또한 리렌더링됨. 따라서 vitualDOM은 실제 DOM에 변화가 없어라도 자식컴포넌트를 전부 리렌더링함.

(위 코드처럼 하면, 1 2 3중 아무거나 바뀌어도 전체다 연계되서 호출됨.

아마 이 컴포넌트들의 부모컴포넌트가 이 컴포넌트 중 하나가 리렌더링 되면 같이 리렌더링돼서 전부다 같이 리렌더링 되는 것인듯)

나중에 virtual DOM 에서 렌더링하는 리소스를 아끼는 것도 배움