useEffectEvent

useEffectEvent 는 **Effect** 내부의 비반응형 로직을 추출해 **Effect 이벤트**라고 불리는 재사용 가능한 함수로 만들 수 있게 해주는 **React Hook**입니다.

```
const onSomething = useEffectEvent(callback)
```


레퍼런스

useEffectEvent(callback)

Effect 이벤트를 선언하기 위해 컴포넌트의 최상위 레벨에서 useEffectEvent 를 호출하세요. Effect 이벤트는 useEffect 와 같이 Effect 내부에서 호출 가능한 함수입니다.

```
import { useEffectEvent, useEffect } from 'react';

function ChatRoom({ roomId, theme }) {
  const onConnected = useEffectEvent(() => {
    showNotification('Connected!', theme);
  });

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.on('connected', () => {
      onConnected();
```

```
    });
    connection.connect();
    return () => connection.disconnect();
  }, [roomId]);

  // ...
}
```

아래에서 더 많은 예시를 확인하세요.

매개변수

- `callback` : Effect 이벤트를 위한 로직을 포함하는 함수입니다. `useEffectEvent` 로 Effect 이벤트를 정의했을 때, `callback` 은 실행할 때마다 항상 최신의 props와 state 값을 참조합니다. 이를 통해 오래된 클로저 문제를 피할 수 있습니다.

반환값

Effect 이벤트 함수를 반환합니다. `useEffect` , `useLayoutEffect` 또는 `useInsertionEffect` 내부에서 이 함수를 호출할 수 있습니다.

주의 사항

- **Effect 내부에서만 호출하세요:** Effect 이벤트는 오로지 Effect 내부에서만 호출해야 합니다. 그것을 사용하는 Effect 이전에 그것을 정의하세요. 다른 컴포넌트나 훅으로 그것을 전달하지 마세요. `eslint-plugin-react-hooks` 린터(버전 6.1.1 또는 최신)는 Effect 이벤트를 잘못된 맥락에서 호출하는 것을 방지하기 위해 이 제한을 강제할 것입니다.
- **의존성 지름길이 아닙니다:** Effect의 의존성 배열에 의존성을 적는 것을 피하기 위해 `useEffectEvent` 를 사용하지 마세요. 이것은 버그를 숨기고 코드를 이해하는 것을 어렵게 합니다. 명시적으로 의존성을 작성하거나 필요한 경우 이전 값을 비교하기 위해 ref를 사용하세요.
- **비반응형 로직을 위해 사용하세요:** 변하는 값에 의존하지 않는 로직을 추출하기 위해서만 `useEffectEvent` 를 사용하세요.

사용법

최신 props와 state를 읽기

전형적으로, Effect 내부에서 반응형 값을 읽을 때, 의존성 배열에 그것을 포함해야 합니다. 이것은 값이 바뀔 때 마다 Effect가 다시 동작하도록 하고, 이것은 보통 바람직한 동작입니다.

그러나 몇몇의 사례에서, 이 값들이 변할 때 Effect가 다시 동작하지 않고 Effect 내부에서 가장 최신의 props 또는 state를 읽고 싶어할 수 있습니다.

Effect 내부에서 이 값들을 반응형으로 만드는 것 없이 최신 props와 state를 읽기 위해 Effect 이벤트 내부에 그것들을 포함하세요.

```
import { useEffect, useContext, useEffectEvent } from 'react';

function Page({ url }) {
  const { items } = useContext(ShoppingCartContext);
  const numberOfItems = items.length;

  const onNavigate = useEffectEvent((visitedUrl) => {
    logVisit(visitedUrl, numberOfItems);
  });

  useEffect(() => {
    onNavigate(url);
  }, [url]);

  // ...
}
```

이 예시에서, `url` 이 바뀔 때(새로운 페이지 방문을 로깅하기 위해) Effect는 랜더 이후에 다시 동작해야 합니다. 그러나 이것은 `numberOfItems` 가 바뀔 때 다시 동작하면 **안됩니다**. 로깅 로직을 Effect 이벤트로 감싸 `numberOfItems` 는 비반응형이 됩니다. 이것은 항상 Effect를 트리거 하는 것 없이 최신의 값을 읽습니다.

`url` 과 같은 반응형 값을 Effect 이벤트의 인수로 전달할 수 있습니다. 이로써, 이벤트 내부의 비반응형 값에 접근할 때 이것들을 반응형으로 유지할 수 있습니다.

이전
useEffect

다음
useId

React 학습하기

빠르게 시작하기

설치하기

UI 표현하기

상호작용성 더하기

State 관리하기

탈출구

API 참고서

React APIs

React DOM APIs

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관