리액트와 상호작용

state와 on{Event}를 통해서...
이벤트핸들러는 순수함수일 필요가 없다

EventHandler의 이름은 어떻게?

이벤트 핸들러는 onEvent={ `eventHandler` }에 위치하게 되는 함수
event 객체를 자동으로 argument로 받게 된다.

이름을 어떻게 지어야 할까?

“ 일반 함수와 이벤트 핸들러의 차이가 뭘까?
이벤트 객체를 받아서 실행해야만 이벤트핸들러라고 봐야할까? ”

on- vs handle-

개인적인 생각으로, on- 함수는 무언가가 실행되고 나서

바깥으로 무언가를 알리거나, 전달해야 하는 경우

에 적합한 이름이지 않을까 라고 요즘은 생각한다

결국 prop으로 전달해서 하위 컴포넌트 등에서 특정 동작 시 추가로 실행되어야 하는 경우에 on- 형태로 전달하는 것이 적합할 것 같다

> 관습적으로 이벤트 핸들러 prop의 이름은 on으로 시작하여 대문자 영문으로 이어집니다. - 리액트 문서

반면 handle- 함수는

특정한 이벤트가 발생했을 때 실행되는 함수

라고 생각하며, `동작` 이 아닌 `대상+동작` 이 보다 구체적이어서 좋다고 생각한다

`handleClick`, `handleModalToggleButtonClick`

```
export const Profile = () => {
  return (
    <div>
      <ProfileHeader onMyPageClick={onMyPageClick} />
      <Button onClick={handleButtonClick}>버튼 클릭</Button>
    </div>
  );
};
```

이벤트 전파

컴포넌트 내에 동일 이벤트에 대해 여러 이벤트들이 등록되어야 한다면,

`event.stopPropagation()` 을 사용하자

이벤트 전파를 통한 캡처

`on{Event}Capture` 를 통해 로그 분석 등을 할 수 있다.

다만, 프로덕션에서는 제거되는 것이 좋을 수 있음

useState의 사용

왜 useState는 if, for, 함수 내에서 사용될 수 없을까?

React hooks:not magic, just arrays

위 글의 번역본

useState 뿐 아니라 모든 훅(React 19에 추가될 예정인 `use` 제외)이
그런 조건을 갖는 이유는 크게 `훅 호출 순서`와 관련이 깊다.

- 리렌더링마다 호출 순서를 동일하게 유지해야 React에 의한 상태 유지/업데이트가 올바르게 일어날 수 있음
- React 내부에 훅의 순서가 '배열'로 저장되기에 다음 인덱스에 영향을 줄 수 있음

setter에 function을 사용하기

예시들에 나와있는 대로 `setState(prevState + 1)`을 중첩해서 여러번 실행하게 되면

```
export default function Counter() {
  const [number, setNumber] = useState(0);

  return (
    <>
      <h1>{number}</h1>
      <button
        onClick={() => {
          setNumber(number + 1);
          setNumber(number + 1);
          setNumber(number + 1);
        }}
      >
        +3
      </button>
    </>
  );
```

```
expect(number).toBe(3);
```

실행 테스트를 해보면 아래와 같은 결과를 얻게 될 것입니다

```
Expected: 3
Received: 1
```

setter가 실행되면 `다음 렌더링`의 state를 변경하게 되어
세번의 setState 모두
`setState(0 + 1)`의 반복이 됩니다.

“실행 시점의 스냅샷”

이 중요 키워드가 될 수 있겠네요

우리가 원하는 대로, 0 -> 1 -> 2 -> 3이 되게 하려면 어떻게 해야될까요?

동일 state를 여러번 업데이트 하기 위해서는 이전 큐의 state를 기반으로 다음 state를 계산하게 해야 합니다

`setState(prevState => prevState + 1)` ~~like that~~

> 이는 단순히 state 값을 대체하는 것이 아니라 React에 "state 값으로 무언가를 하라"고 지시하는 방법입니다.

state를 이걸로 바꿔라 vssetter에 기존 state를 이용해 무언가를 하라의 차이가 됩니다.

그런데, state가 세번 변하는데 렌더링이 세번 일어날까요?

리액트는 state 업데이트가 일어나기 이전, 이벤트 핸들러 내의 코드가 모두 실행되는 것을 기다립니다.

그러기 위해서 앞서 나온 `큐` 가 사용되죠

객체, 배열 state

중요한 것은 state는 직접 변경되어서는 안된다는 점입니다.

객체와 배열 상태를 변경하려면 복사본을 만들고 그걸 새로운 state로 사용해야 합니다!

공식 문서는 중첩 객체의 깊은 복사를 위해 `Immer` 라이브러리를 소개하고 있습니다.

저는 이 API를 알게 된 후로는 1-depth를 넘는 객체에 대해서는 이것을 사용합니다.

structuredClone

ES2022에 추가된 신상 API입니다!

현재 대부분의 브라우저들은 모두 지원하고 있고 (그 브라우저 제외)
Node.js의 경우는 17에 도입되어 18부터는 안정적으로 지원하고 있습니다.

객체의 깊은 복사를 지원하는 API입니다.

옵셔널 프로퍼티 transfer가 있기는 하지만, 이는 파일 바이너리를 다룰때 이용되므로 지금은 넘어가죠

배열을 다룰 때

배열의 경우에도 Array.prototype에 추가된 메소드를 활용하면 좋을 것 같네요

toSorted, toSpliced

이 둘은 ES2023에 추가된 Array 메소드로, 원본 배열을 수정하지 않습니다!