

React

With Next.js 

3. Event

연사 & 멘토 소개

연사 : 박승범 (@killerwhale)

멘토 : 박승범 (@killerwhale)

멘토 : 박승범 (@killerwhale)

Event

OTL PLUS

!든 사람들 | 라이선스 | 개인정보취급방침

otlplus@sparcs.org

© 2016, SPARCS OTL Team

색

전체 ☒ 기필 ☐ 기선 ☐ 전필 ☐ 전선 ☐

교필 ☐ 인선 ☐ 공통 ☐ 석박 ☐ 자선 ☐

기타 ☐

전체 ☒ 인문 ☐ 건환 ☐ 기경 ☐ 기계 ☐

뇌인지 ☐ 물리 ☐ 바공 ☐ 반시공 ☐ 산공 ☐

산디 ☐ 생명 ☐ 생화공 ☐ 수리 ☐ 신소재 ☐

원양 ☐ 융인 ☐ 전산 ☐ 전자 ☐ 항공 ☐

화학 ☐ 기타 ☐

전체 ☒ 100번대 ☐ 200번대 ☐ 300번대 ☐

400번대 ☐

화요일 10:00 ~ 15:00

검색 취소

9

10

11

12


1

2

3

4

5



기필 0

전필 0

인선 0

0

학점

?

성적

?

날

시험시

월

화

수

목

금

드래그 처리하기 -> 마우스 Event Handling 필요

Event

React에서 이벤트는 특정 동작이 발생했을 때 실행할 함수를 props로 전달한다.

```
<button onClick={activateLasers}>  
  Activate Lasers  
</button>
```

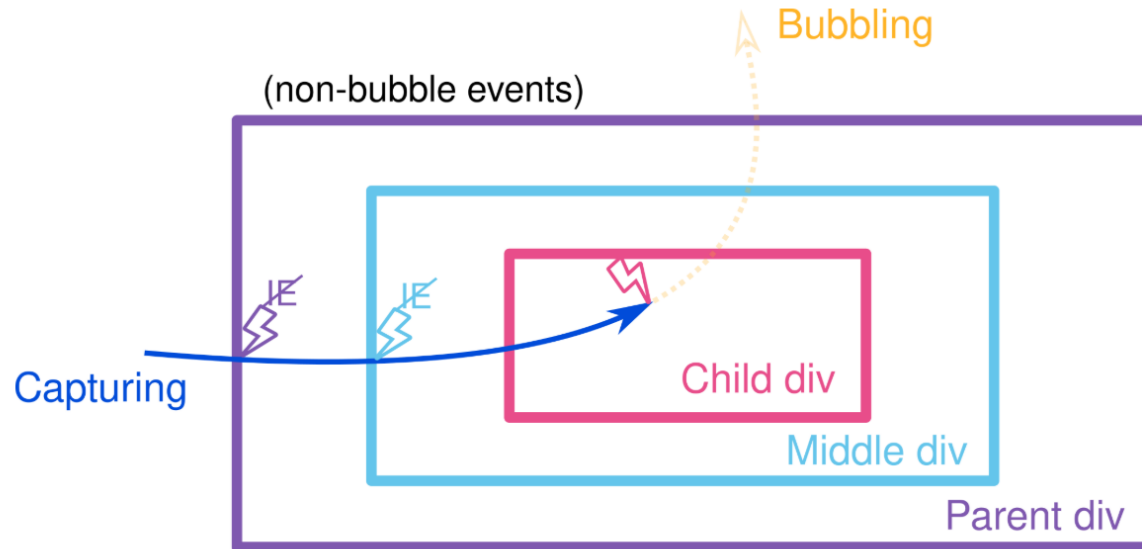
React 에서 Event와 관련된 prop는
항상 on으로 시작한다.

```
function Form() {  
  function handleSubmit(e) {  
    e.preventDefault();  
    console.log('You clicked submit.');  }  
  
  return (  
    <form onSubmit={handleSubmit}>  
      <button type="submit">Submit</button>  
    </form>  
  );  
}
```

PreventDefault() 속성을 활용해서
브라우저의 기본 동작을 방지하고
직접 정의한 동작을 넣을 수 있다.

더 많은 React Event 들은 부록을 참고하세요.

Event propagation - 버블링 & 캡처링



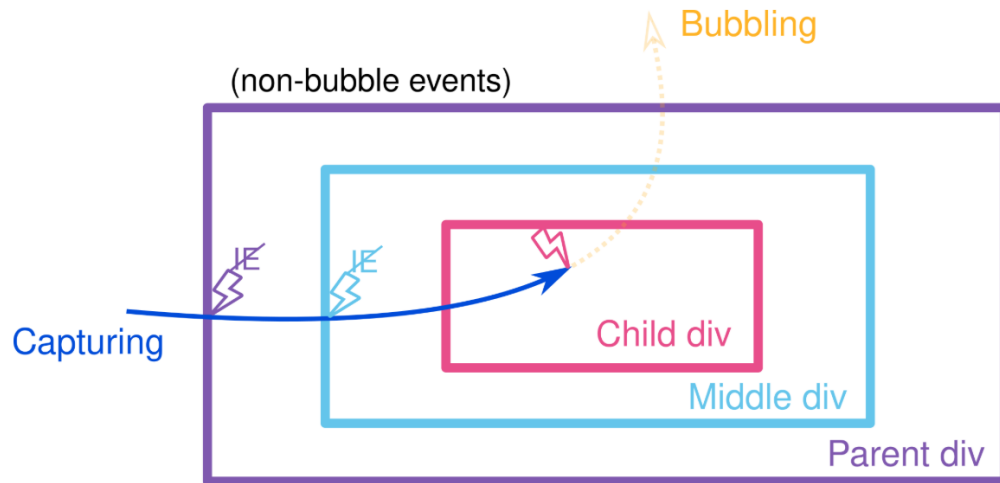
이벤트 버블링 : 발생한 이벤트를 **부모** 요소로 전파

이벤트 캡처링 : 발생한 이벤트를 **자식** 요소로 전파

이벤트 버블링

이벤트 버블링은 대부분의 이벤트에서 발생

버블링이 발생하지 않는 event도 간혹 있으니 주의!



발생한 이벤트가 부모 요소로 전파되며 각각에 정의된 handler가 **모두** 실행된다.

(이벤트 이름).target으로 처음 버블링이 시작된 요소에 접근 가능

이벤트가 진행되어도 변화 X.

이벤트 버블링 - example

Search Bar 컴포넌트

```
<form
  className="w-full max-w-[600px] mx-auto flex justify-center"
  onSubmit={e => {
    e.preventDefault();
    if (inputValue.trim()) {
      // board로 검색 라우팅 (검색어 보존)
      router.push(`/board?search=${encodeURIComponent(inputValue.trim())}`);
    } else {
      router.push("/board");
    }
  }}
>
  <SearchBar
    value={inputValue}
    onChange={e => setInputValue(e.target.value)}
  />
</form>
```

<Page.tsx>

Bubbling

e.target으로
값 접근

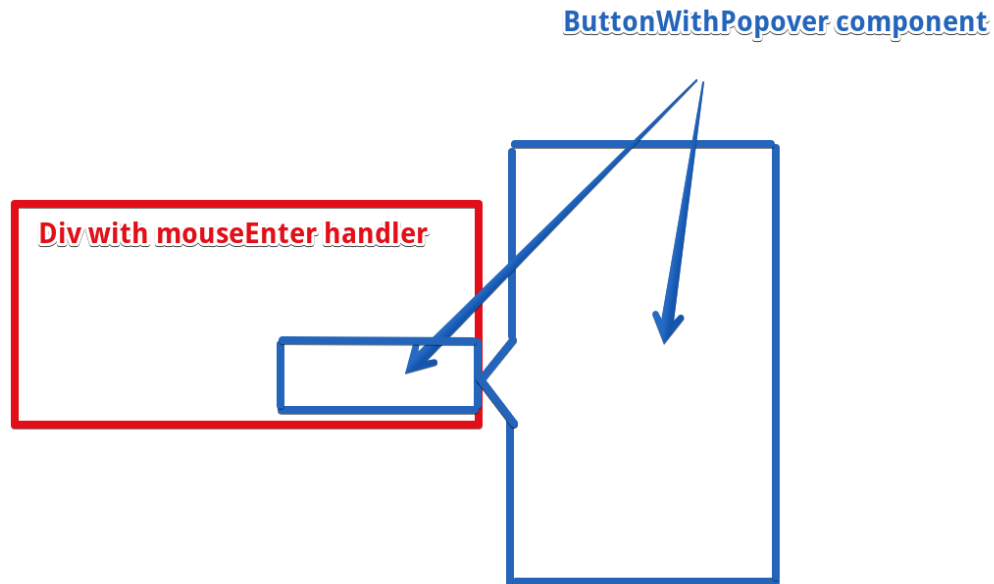
```
return (
  <div className="flex h-[56px] border-[2px] bg-white border-ara
    px-4 py-2 space-x-2 w-[90%] max-w-[650px] shadow-sm shadow-a
  >
    <input
      type="search"
      className="w-full focus:outline-none focus:ring-0 placehol
      placeholder="검색어를 입력하세요."
      value={value}
      onChange={onChange}
      onKeyDown={handleKeyDown}
    />
    <button onClick={handleSearch}>
      <Image src="SearchBar_search.svg" width={28} height={28} a
    </button>
  </div>
```

이벤트 버블링은 상위 컴포넌트에서 이벤트를 모아서 처리할 수 있도록 한다.

이벤트 버블링 – stop Propagation

이벤트가 부모 이벤트로 전파되는 것을 막을때는 `stopPropagation()`을 사용한다.

사용 예시)



부모 컴포넌트에 다른 중요한 이벤트가 있는 경우

부모 컴포넌트로의 전파를 막아준다.

이벤트 캡처링

```
function App() {  
  return (  
    <div  
      onClickCapture={() => console.log("▼ Div 캡처링")}  
      onClick={() => console.log("⬆ Div 버블링")}  
    >  
      <button  
        onClickCapture={() => console.log("▼ Button 캡처링")}  
        onClick={() => console.log("⬆ Button 버블링")}  
      >  
        Click Me  
      </button>  
    </div>  
  );  
}
```

부모 요소에서 발생한 event를 자식 요소에게 전파

기본적인 이벤트 핸들러는 **버블링**에서 실행

1. Div onClickCapture
2. Button onClickCapture
3. Button onClick
4. Div onClick

이벤트 핸들러 실행 순서

자식보다 이벤트를 먼저 잡아서
interrupt 해야 하는 경우.

preventDefault

모든 이벤트는 브라우저의 기본 동작을 갖는다.

기본 이벤트와는 다른 이벤트를 직접 정의하고 싶다면 (이벤트 이름).preventDefault()를 Handler에 등록해야 한다.

```
const onPickImage = (e: React.ChangeEvent<HTMLInputElement>) => {  
  const f = e.target.files?.[0];  
  if (f) pickAndUpload(f, 'IMAGE');  
  e.target.value = '';  
};  
  
const onPickFile = (e: React.ChangeEvent<HTMLInputElement>) => {  
  const f = e.target.files?.[0];  
  if (f) pickAndUpload(f, 'FILE');  
  e.target.value = '';  
};  
  
const handleKeyDown = (e: React.KeyboardEvent<HTMLTextAreaElement>) => {  
  if (e.key === 'Enter' && !e.shiftKey) {  
    e.preventDefault();  
    handleSend();  
  }  
}
```

참고) Event handle는 react Event를 parameter로 받는다.

부록

1. 마우스(Mouse) 이벤트

- `onClick` - 클릭했을 때
- `onDoubleClick` - 더블 클릭했을 때
- `onMouseDown` - 마우스 버튼을 누를 때
- `onMouseUp` - 마우스 버튼을 땄 때
- `onMouseEnter` - 마우스가 요소에 들어올 때 (버블링 없음)
- `onMouseLeave` - 마우스가 요소에서 나갈 때 (버블링 없음)
- `onMouseOver` - 마우스가 요소 위로 올라갔을 때 (버블링 있음)
- `onMouseOut` - 마우스가 요소 밖으로 나갔을 때 (버블링 있음)
- `onMouseMove` - 마우스가 움직일 때
- `onContextMenu` - 오른쪽 클릭 시(컨텍스트 메뉴)

2. 키보드(Keyboard) 이벤트

- `onKeyDown` - 키를 누를 때
- `onKeyUp` - 키를 땄 때
- `onKeyPress` - 키를 눌렀을 때 (단, 최신 React에서는 잘 안 씀 → `onKeyDown` 사용 권장)

3. 폼(Form) 이벤트

- `onChange` - 입력 값이 바뀔 때 (input, select, textarea)
- `onInput` - 입력 시마다 (HTML 기본 이벤트 그대로)
- `onSubmit` - 폼 제출할 때
- `onReset` - 폼 리셋할 때
- `onInvalid` - 유효성 검사 실패할 때

4. 포커스(Focus) 이벤트

- `onFocus` - 요소가 포커스를 얻었을 때
- `onBlur` - 요소가 포커스를 잃었을 때

5. 폼 요소 전용 이벤트

- `onSelect` - 텍스트를 드래그/선택했을 때
- `onBeforeInput` - 입력 전에 발생 (예: 한글 조합 시작 시)
- `onCompositionStart` - 글자 조합(IME 입력) 시작
- `onCompositionUpdate` - 조합 중
- `onCompositionEnd` - 조합 끝

6. 드래그 & 드롭(Drag & Drop) 이벤트

- `onDrag` - 드래그 중
- `onDragStart` - 드래그 시작
- `onDragEnd` - 드래그 종료
- `onDragEnter` - 드래그 대상이 요소에 들어왔을 때
- `onDragLeave` - 드래그 대상이 요소를 벗어났을 때
- `onDragOver` - 드래그 대상이 요소 위에 있을 때
- `onDrop` - 드롭했을 때

7. 터치(Touch) 이벤트 (모바일 전용)

- `onTouchStart` - 터치 시작
- `onTouchMove` - 터치 중 이동
- `onTouchEnd` - 터치 끝
- `onTouchCancel` - 시스템이 터치를 취소했을 때

부록

9. 클립보드(Clipboard) 이벤트

- `onCopy` - 복사할 때
- `onCut` - 잘라낼 때
- `onPaste` - 붙여넣기 할 때

10. 폼 외 기타 이벤트

- `onScroll` - 스크롤할 때
- `onWheel` - 마우스 휠 스크롤할 때
- `onAbort` - 리소스 로딩이 중단될 때 (이미지, 미디어 등)
- `onLoad` - 리소스가 로딩될 때
- `onError` - 에러 발생 시
- `onToggle` - `<details>` 요소가 열리거나 닫힐 때

Homework

Hw3_event 완성하기

Due :

The End

감사합니다