userEvent를 사용한 사용자 상호작용 테스트

실무에 바로 적용하는 프런트엔드 테스트

이재성

fireEvent

✔ DOM 이벤트를 시뮬레이션 하기 위해 제공

✔ 내장되어 있기 때문에 별도의 설치 없이 사용 가능

```
import { fireEvent } from '@testing-library/react';

fireEvent.click(getByText(container, 'Submit'));
```

fireEvent

✔ fireEvent의 동작은 단순하게 해당 이벤트만 디스패치

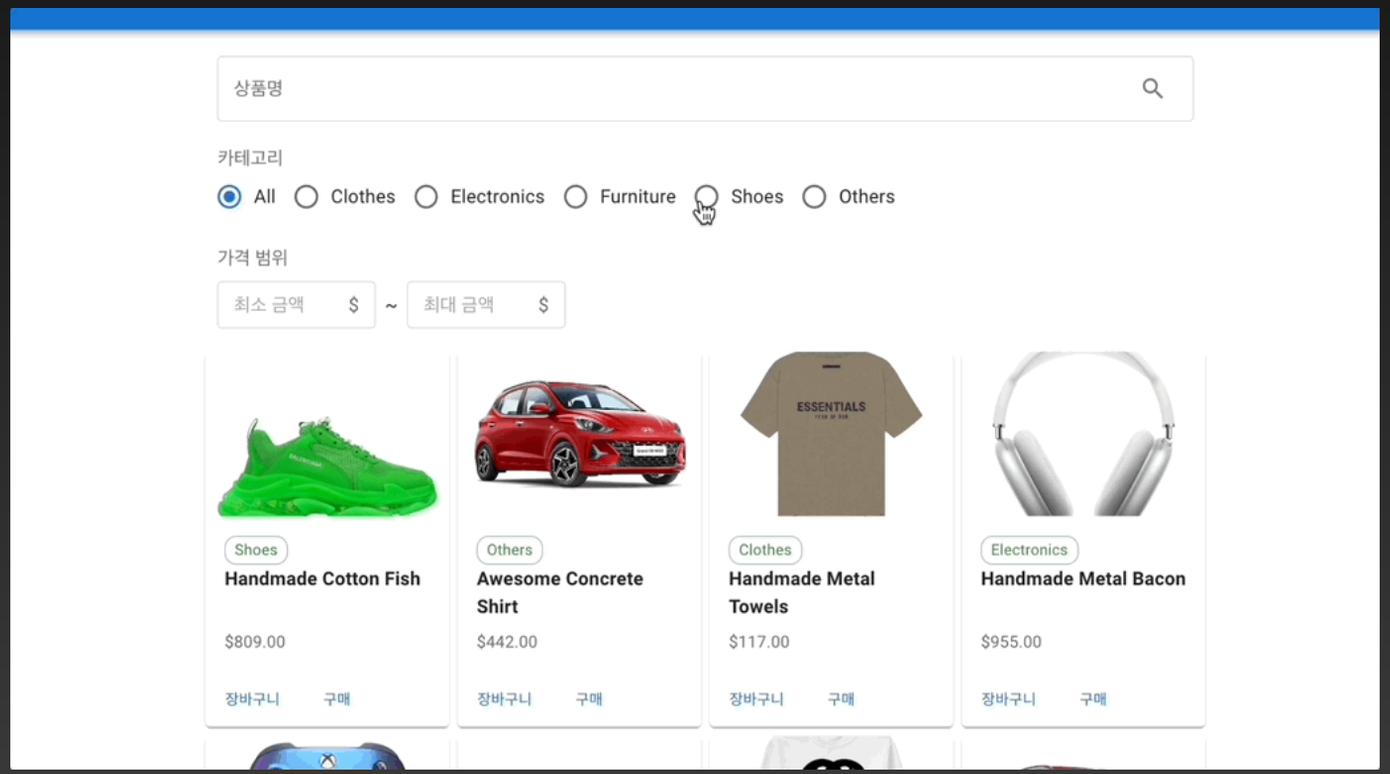
```
const clickEvent = new MouseEvent('click', {
  bubbles: true,
  cancelable: true,
});

button.dispatchEvent(clickEvent);
```

fireEvent는 실제 상황과 거리가 있다

하지만, 사용자가 실제 요소를 클릭할 때

pointerdown, mousedown, pointerup, mouseup, click, focus 등의 이벤트가 연쇄적으로 발생



userEvent

✔ disabled, display 상태도 고려하여 실제와 더 유사하게 테스트 가능

disabled 된 텍스트 필드는 입력이 불가

```
import userEvent from '@testing-library/user-
event';
const user = userEvent.setup();

user.click(screen.getByText('구매하기'));
```

userEvent

✔ 실제 사용자가 사용하는 것처럼 이벤트가 연쇄적으로 발생

✔ disabled, disabled 상태도 고려하여 실제 상황과 유사하게 테스트 가능

테스트의 신뢰성 향상

하지만, userEvent로 불가능하다면..

✔ 대부분의 사례는 userEvent로 모두 가능!

✔ 이벤트(ex. scroll)을 지원하지 않는다면, 꼭 테스트해야 한다면! fireEvent로!

```
fireEvent.scroll(container, {
  target: {
    scrollTop: 10000,
  },
});
```

정리

fireEvent

- @testing-library/react 모듈에 내장되어 제공
- 특정 요소에서 원하는 이벤트만 쉽게 발생시킬 수 있음

fireEvent vs userEvent

- fireEvent는 DOM이벤트만 발생시키는 반면, userEvent는 다양한 상호 작용을 시뮬레이션 할 수 있음
    - 클릭 이벤트가 발생한다면.. pointerdown, mousedown, pointerup, mouseup, click, focus가 연쇄적으로 발생
    - 실제 상황처럼 disabled된 버튼이나 인풋 입력이 불가능함
- 테스트 코드 작성 시에는 userEvent를 활용해 실제 상황과 유사한 코드로 테스트의 신뢰성을 높이자
- userEvent에서 지원하지 않는 부분이 있을 때, fireEvent 활용을 고민하자