

React Testing Library와 컴포넌트 테스트

실무에 바로 적용하는 프론트엔드 테스트

이재성

Testing Library



<https://testing-library.com/docs/>

UI 컴포넌트 테스트를 도와주는 라이브러리

Testing Library



React



Cypress



Testcafe



Svelte



Vue



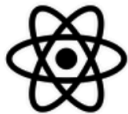
Angular



ReasonReact



Puppeteer



React Native



Preact



Nightwatch



And more...


Testing Library의 핵심 철학

UI 컴포넌트를
사용자가 사용하는 방식으로 테스트 하자

Testing Library의 핵심 철학

UI 컴포넌트를

사용자가 사용하는 방식으로 테스트 하자



DOM 노드를 쿼리(조회)하고
사용자와 비슷한 방식으로 이벤트를 발생시키자

인터페이스를 기준으로 테스트를 작성하자

올바른 테스트 코드

```
// 🧑  
it('버튼을 누르면 모달을 띄운다.', () => {  
  // 유저의 동작과 비슷하도록 클릭 이벤트를 발생  
  user.click(screen.getByRole('button'));  
  
  // ...  
});
```

- ✓ 내부 구현과 종속성이 없으며 캡슐화에 위반되지 않음
 - ✓ DOM, 이벤트 인터페이스 기반으로 검증
 - ✓ 테스트 코드를 직관적으로 이해할 수 있음

Testing Library 쿼리

<https://testing-library.com/docs/queries/about>

vi.fn()

<https://vitest.dev/api/vi.html#vi-fn>

정리

React Testing Library는

- UI 컴포넌트를 사용자가 사용하는 방식으로 테스트
 - 사용자가 앱을 사용하는 방식과 테스트 방식이 유사할수록 신뢰성은 향상
- DOM과 이벤트 인터페이스를 기반으로 요소를 조회하고, 다양한 동작을 시뮬레이션 할 수 있음
 - 요소 조회를 위한 쿼리는 다양하며, 우선 순위가 존재
 - <https://testing-library.com/docs/queries/about>
- 인터페이스 기반의 직관적인 코드와 내부 구현에 종속되지 않는 견고한 테스트 코드

Spy 함수는

- 함수의 호출 여부, 인자, 반환 값 등 함수 호출에 관련된 다양한 값을 저장
- 콜백 함수나 이벤트 핸들러가 올바르게 호출 되었는지 검증할 수 있음
 - toHaveBeenCalledWith, toHaveBeenCalled 매치

테스트 코드: unit-test-example-with-answer 브랜치