Cypress로 첫 번째 E2E 테스트 작성하기

실무에 바로 적용하는 프런트엔드 테스트

사용 브랜치

shopping-mall-e2e-test-example

$ git checkout shopping-mall-e2e-test-example

단위·통합에서 작성한 테스트를 E2E에서도 작성?

✔ 컴포넌트 일부가 아닌 앱 자체에 대한 기능이 동작하는지 검증

→ 어디까지나 단위·통합에서는 일부 컴포넌트의 조합을 테스트

→ 앱이 구동 되었을 때도 동일하게 동작한다면 안정성이 더욱 높아짐

단위·통합에서 작성한 테스트를 E2E에서도 작성?

✔ 컴포넌트 일부가 아닌 앱 자체에 대한 기능이 동작하는지 검증

→ 어디까지나 단위·통합에서는 일부 컴포넌트의 조합을 테스트

→ 앱이 구동 되었을 때도 동일하게 동작한다면 안정성이 더욱 높아짐

✔ 통합 테스트로 검증한 기능을 파악하는 노력을 해서 분리해야 한다면 DX(Developer Experience)에 큰 장점이 없음

→ 워크 플로우가 클 수록 파악하는데 많은 시간이 소요됨

Cypress의 단언

- ✔내부적으로 Chai, Chai-jQuery, Sinon-Chai의 문법을 사용할 수 있도록 확장되어 있음
- ✔ Chai-jQuery는 주로 DOM 객체에 대한 단언
- ✔ Sinon-Chai는 주로 Stub이나 Spy에 대한 단언

자주 쓰이는 Cypress 단언

더 많은 단언은 링크로!

```
// 1. length
// ex> list 요소가 3개 인지 검증
cy.get('li').should('have.length', 3);

// 2. class
// ex> list 요소가 disabled class가 없는지 검증
cy.get('li').should('not.have.class', 'disabled');

// 3. value
// ex> textarea 요소가 Hello 값을 가지고 있는지 검증
cy.get('textarea').should('have.value', 'Hello');

// 4. visibility
// ex> list 요소가 보이는지 검증
cy.get('li').should('be.visible');

// 5. existance
// ex> list 요소가 존재하는지 검증
cy.get('li').should('exist');

// 6. state
// ex> radio 요소가 선택되었는지 검증
cy.get(':radio').should('be.checked')
```

정리

단위·통합 테스트와 E2E 테스트의 중복 기능 검증

- 전체 앱을 띄웠을 때도 정상 동작하는지 검증 → 안정성 향상
- 통합 테스트에서 검증된 기능을 찾아 필터링 하는 과정 → 불필요한 시간 소요

활용한 Cypress 주요 문법

- beforeEach, beforeAll을 통한 setup
- **'cy.'** prefix로 내장 API를 사용할 수 있음
    - visit() 함수를 통한 특정 페이지 방문
- Cypress Testing Library를 통해 Testing Library에서 제공하는 DOM 노드 쿼리 사용
- **should()** 함수를 통해 단언. Cypress는 내부적으로 chai, chai-jQuery, sinon-chai를 확장해 사용

E2E 테스트의 효과

- 단위·통합 테스트에서 처럼 별도 모킹 과정이 없기 때문에 페이지 이동, API 연동, 로그인 후 처리 등 실제 앱에서 발생하는 모든 과정을 검증할 수 있음