1부를 마무리하며

실무에 바로 적용하는 프런트엔드 테스트

이재성

테스트란

앱의 품질과 안정성을 높이기 위해
사전에 결함을 찾아내고 수정하기 위한 일련의 행위

테스트 코드의 효과

- ✔ **리팩토링** - 거대한 범위의 리팩토링도 안정성 있게 진행할 수 있음
- ✔ **문서** - 테스트 코드는 앱의 이해를 돕는 좋은 문서
- ✔ **좋은 설계** - 테스트 단위에 대한 고민은 좋은 설계에 대한 사고로 이어짐

테스트 작성 시 중요한 규칙

- ✔ **인터페이스 기준으로 테스트를 작성하자**
  - ✔ 내부 구현에 대한 의존성이 없어야 한다
  - ✔ UI와 이벤트를 기준으로 동작을 확인하자
- ✔ **의미 있는 테스트인지 고민하자**
  - ✔ 커버리지 100% 보다는 유의미한 기능을 검증할 수 있는지 확인하자
- ✔ **테스트 코드의 가독성을 생각하자**
  - ✔ 테스트 디스크립션을 명확하게 작성하자
  - ✔ 하나의 테스트에서는 가급적 하나의 동작만 검증하자

무슨 테스트를 배웠을까?

- ✔ **단위 테스트**: 앱에서 테스트 가능한 가장 작은 소프트웨어를 실행해 예상대로 동작하는지 확인하는 테스트
  - ✔ 공통 컴포넌트, 리액트 훅, 유틸 함수
    → 특정 도메인에 종속되기 보다는 모듈 자체 만으로 독립적으로 동작하는 모듈
- ✔ **통합 테스트**: 두 개 이상의 모듈이 상호 작용하여 발생하는 상태를 검증해 실제 앱의 비즈니스 로직과 가깝게 기능을 검증
  - ✔ 나누어진 비즈니스 로직을 기준으로 컴포넌트 조합을 검증
  - ✔ 상태 관리나 API 호출은 상위 컴포넌트로 응집
    → 테스트 범위를 나누고 코드를 관리하기 좋음

그 과정에서 무엇을 배웠을까?

✔ Vitest 프레임워크

✔ 단언(assertion)

✔ 매처(matcher)

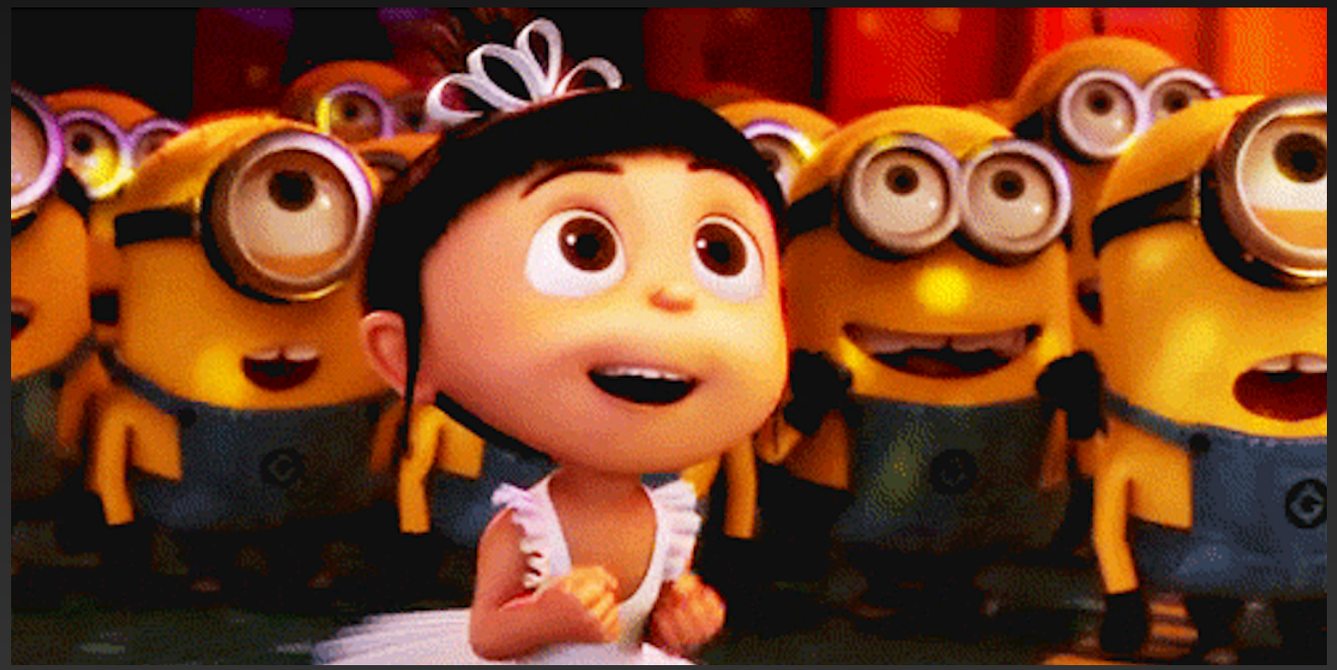
✔ Setup, Teardown

✔ Testing Library

✔ 모킹과 msw

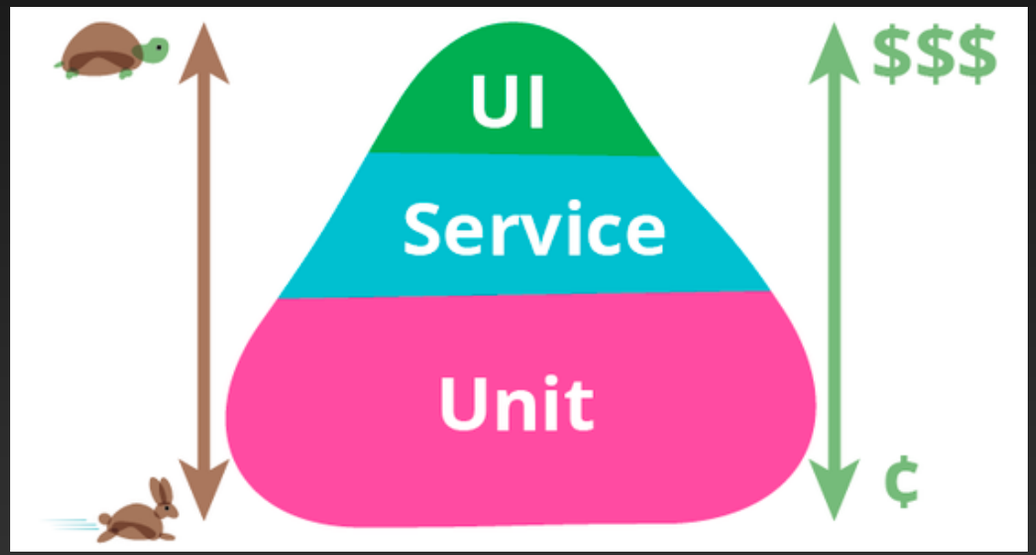
테스트의 한계를 배운 이유는 뭘까?

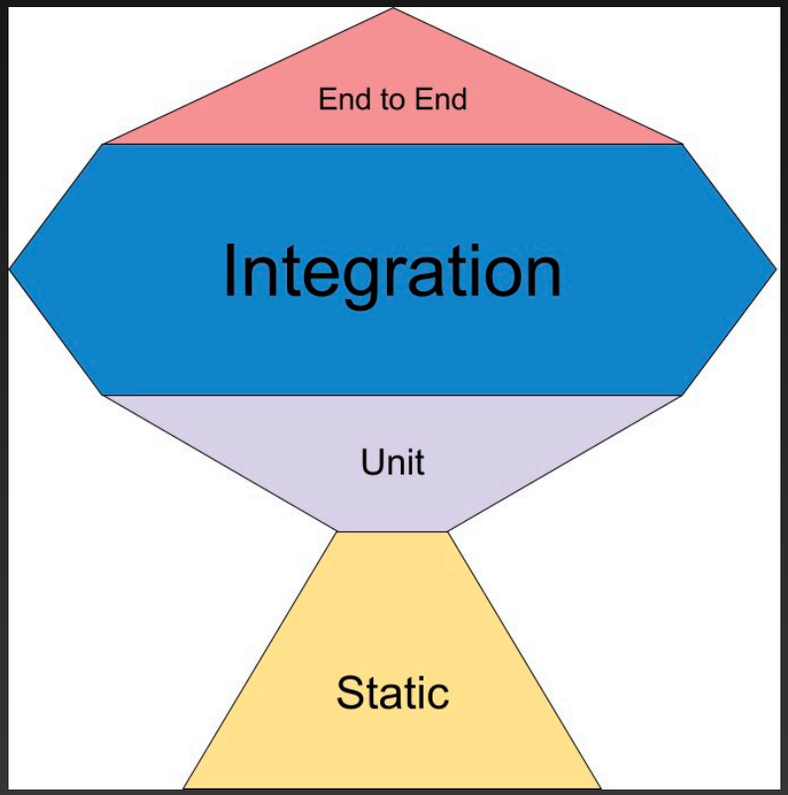
‘모든 것을 다 해결해줄 테스트가 있으니 기다려라?’



테스트 피라미드, 트로피 등..

테스트 작성 및 운영 비용과 속도를 기준으로 테스트를 설계하는 방법





출처: https://martinfowler.com/bliki/TestPyramid.html

출처: https://kentcdodds.com/blog/the-testing-trophy-and-testing-classifications

결국 중요한 건,
각 테스트가 검증하는 내용과 한계를 정확하게 이해하고
개개인의 프로젝트에 맞게 설계할 수 있는 능력.
그에 따라오는 내가 작성한 코드에 대한 신뢰감.

2부에서는..

✔ **E2E 테스트**

✔ **시각적 회귀 테스트**

그럼, 2부에서도 잘 부탁드립니다. 💪