스토리 작성하기

실무에 바로 적용하는 프런트엔드 테스트

스토리 작성

✔ 일반적으로 .stories.(tsx|ts|jsx|js) 로 끝나는 파일에 작성

✔ ES6 모듈 기반의 CSF(Component Story Format)으로 작성

CSF의 핵심 구성 요소

✔ **메타 데이터**: 이름이나 동작에 대한 정보를 설명 (default export)

✔ **스토리**: 각각의 시나리오를 나타냄 (named export)

play 함수

✔ 스토리를 렌더링 한 뒤 사용자의 상호작용을 시뮬레이션

✔ 내부적으로 Testing Library를 통해 구현

✔ DOM과 상호작용을 하거나 어설션을 작성해 테스트도 가능

스토리 작성 대상

props를 받아 UI만 렌더링하는 컴포넌트

렌더링 결과를 빠르게 확인할 수 있고
스토리를 상세화하면 설정 없이 편하게 확인할 수 있다.

스토리 작성 대상

props를 받아 UI만 렌더링하는 컴포넌트

렌더링 결과를 빠르게 확인할 수 있고
스토리를 상세화하면 설정 없이 편하게 확인할 수 있다.

❌ 복잡한 비즈니스 로직이 있는 컴포넌트를 스토리로 작성할 경우
몇 가지 UI 확인을 위해 과한 모킹이 필요해짐

스토리 작성 대상

props를 받아 UI만 렌더링하는 컴포넌트

렌더링 결과를 빠르게 확인할 수 있고
스토리를 상세화하면 설정 없이 편하게 확인할 수 있다.

⭕ 비즈니스 로직이 모여있는 컴포넌트는
하위 컴포넌트를 대상으로 스토리를 **상세화**해 작성하자

스토리 작성 대상

props를 받아 UI만 렌더링하는 컴포넌트

렌더링 결과를 빠르게 확인할 수 있고
스토리를 상세화하면 설정 없이 편하게 확인할 수 있다.

스토리와 통합 테스트의 대상을 적절하게 나누기 위해서는
비즈니스 로직과 UI를 검증하기 위한 구조가 잘 나누어져 있어야 함
👉 **자연스럽게 좋은 컴포넌트 설계 구조에 대한 고민으로!**

정리

스토리

- CSF(Component Story Format)으로 작성
- .stories.(tsx|ts|jsx|js)로 끝나는 파일에 작성
- 메타 데이터와 각각의 시나리오를 보여주는 스토리로 구성
    - **메타 데이터**: default export로 정의, 제목, 필드 정보, 매개변수, 데코레이터 등에 대한 정보를 정의
    - **스토리**: named export로 정의
        - args: 각 스토리 별로 인자 값을 동적으로 변경해 UI에 반영할 수 있음

Play

- 스토리를 렌더링한 후 사용자 상호 작용을 시뮬레이션 할 수 있도록 도와줌

스토리 작성 대상

- 단순하게 UI만 렌더링 하는 컴포넌트의 시나리오를 상세화
- 비즈니스 로직이 응집되어 있는 컴포넌트의 경우 별도 준비 과정이 필요해 작성이 어려울 수 있다.
- 스토리 작성 대상과 통합 테스트 대상을 구분해 비즈니스 로직과 UI를 구분하기 좋은 설계를 하자