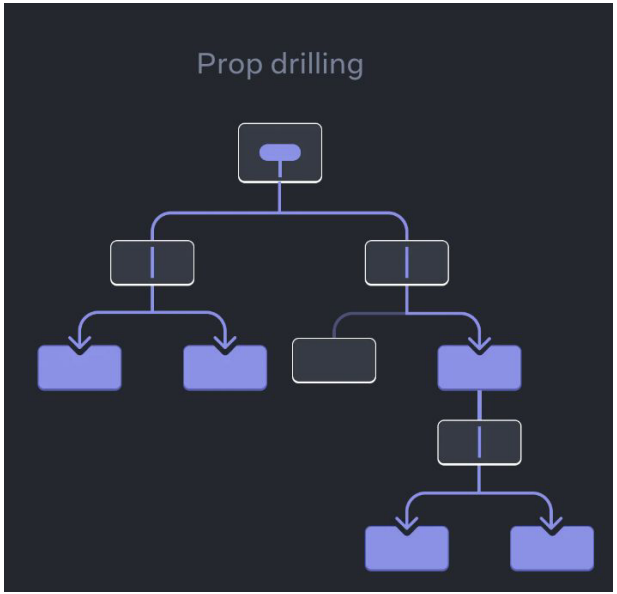
원티드 프론트엔드 챌린지
8월 2회차

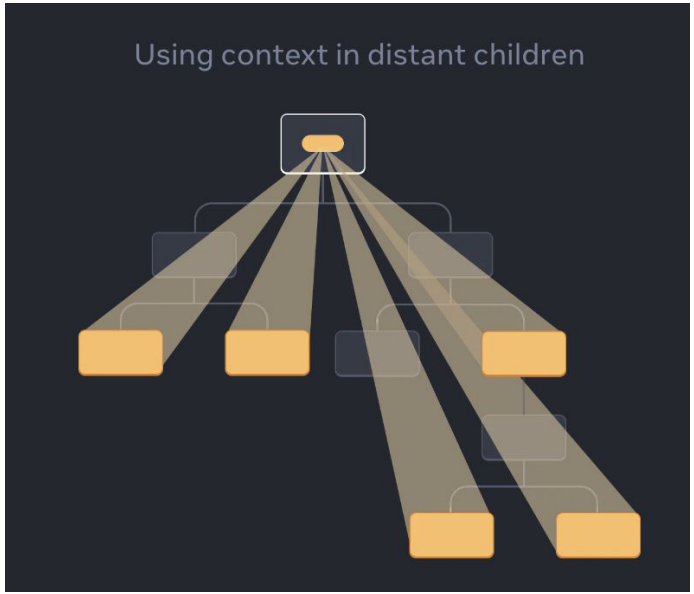
Recap

1. 지역변수 vs 전역변수
2. Context API
3. Recoil
4. React Query

지역변수 vs 전역변수

1. 가급적 지역변수 사용을 권장
2. 전역변수는 꼭 필요한 경우에만





Context API

1. 리액트 내장 패키지로 추가 라이브러리 불필요
2. 관심사가 여러개라면 숫자에 맞는 Provider 선언 필요
3. 불필요한 rendering 발생 가능성

Recoil

1. 리액트와 유사한 syntax
2. atom이라는 state
3. atom을 조작한 selector
4. 요즘 회사들에서 많이 사용

Recoil

1. atom을 공통함수에서 사용이 어렵다는 질문 - 그러면 안됩니다

```
const calculateTotalPrice = (price: number) => {
  const count = useRecoilValue(countState);
  return price * count;
};
```

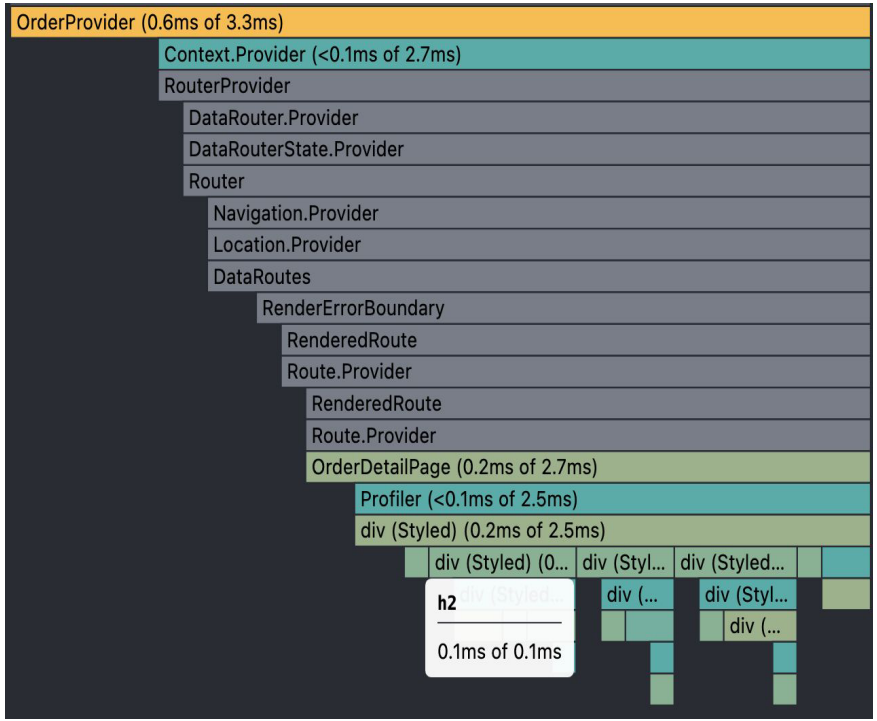
Recoil

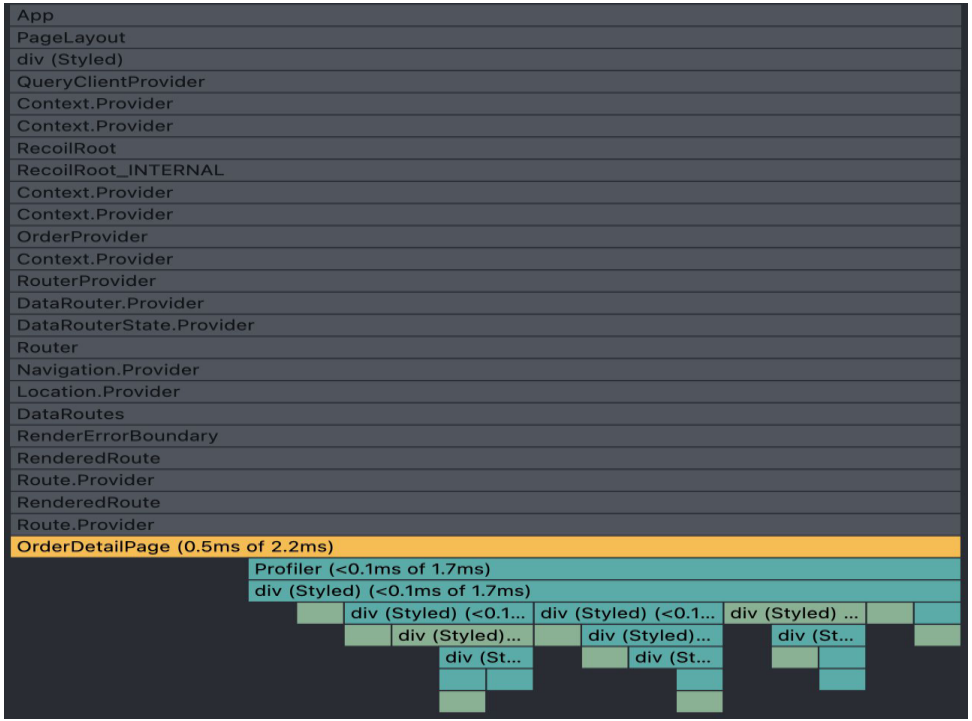
1. atom을 공통함수에서 사용이 어렵다는 질문 - 그러면 안됩니다
    a. 함수가 Pure Function이 아님
    b. 같은 input이 들어오더라도 항상 같은 output을 리턴할 수 없는 구조임

```
let count = 1;

const calculateTotalPrice= (price: number) => {
  return price * count;
}
```

Context API vs Recoil





React Query

1. 서버 상태 관리
2. staleTime, cacheTime 사용
3. optimistic update
    a. https://www.youtube.com/watch?v=rnN5ng6aoAc&ab_channel=Codevolution
    b. https://uncertainty.oopy.io/optimistic-ui-with-react-query

Agenda

1. 무엇을 테스트 할 것인가?
   a. 어디까지 테스트 할 것인가?
   b. 어떻게 테스트 할 것인가?
2. 유닛테스트
3. 통합테스트
4. E2E 테스트
5. 테스트 자동화
6. 프론트엔드 TDD

무엇을 테스트 할 것인가?

1. UI 테스트는 진행하지 않음
    a. 로그인 화면에서 비밀번호 input과 버튼 사이의 거리를 재는 것은 무의미
        i. 기기의 크기에 따라 달라질 것
    b. 컴포넌트 렌더링 순서도 무의미
        i. JSX는 선언한대로 렌더링 된다
        ii. 인풋이 버튼 위에 존재하는지는 테스트할 필요가 없는 것
2. 요구사항의 “사용자 시나리오"에 집중해야 한다
    a. 아이디 비밀번호를 입력했을 때 버튼이 활성화 되는가?
    b. 하지만 위보다 중요한것은
        i. 로그인이 성공하는가?
        ii. 로그인이 실패하는가?
    c. “기능”에 중점을 두고 테스트를 해야한다

무엇을 테스트 할 것인가?

1. 로그인 화면에서 버튼 테스트
2. 테스트 시나리오?
    a. 버튼이 잘 보이는가
    b. 버튼이 요구조건에 맞게 활성화 되는가
    c. 활성화된 버튼을 클릭하면 이벤트가 잘 발생하는가

무엇을 테스트 할 것인가?

1. 로그인 화면에서 버튼 테스트
2. 테스트 시나리오?
    a. 버튼이 잘 보이는가
    b. 버튼이 요구조건에 맞게 활성화 되는가
    c. 활성화된 버튼을 클릭하면 이벤트가 잘 발생하는가
3. 위에 3개 다 아님
    a. 제대로 된 계정정보로 로그인이 잘 되는가?
    b. 잘못된 계정정보로 로그인이 실패하는가?
        i. 로그인 실패 여부가 “에러메세지" 로 확인이 가능하다면
        ii. 해당 항목은 확인 가능
4. **테스트를 위한 테스트를 하지 말 것.**

무엇을 테스트 할 것인가?

1. 다시 로그인으로 돌아가면
2. 로그인이 잘 되는지 확인하려면
    a. 이벤트 핸들러가 잘 동작하는지 봐야하고
    b. 결국은 버튼이 잘 그려지는지를 봐야한다.
3. 하지만 난 “로그인 성공" 이라는 간단한 기능에 유닛테스트를 하는 중인데?
    a. 그런데 로그인 성공을 테스트하려면
        i. 인풋도 잘 작동하고
        ii. 버튼도 잘 작동하고
        iii. HTTP request도 잘 이루어져야함
    b. 이정도면 통합테스트 아닌가?
        i. 프론트엔드는 유닛테스트와 통합테스트의 경계가 애매함 (개인적의견)

어떻게 테스트 할 것인가?

1. Given - When - Then
2. True 또는 False를 리턴
3. 결과값을 expectation과 비교
4. 테스트는 한번에 하나씩만

단위테스트

1. 코드의 가장 작은 단위를 테스트
    a. 각각의 컴포넌트
    b. 각각의 함수
    c. Jest를 많이 사용함
2. 로그인으로 예를 들자면
    a. 아이디와 비밀번호 입력 필드가 존재하는지.
    b. 아이디와 비밀번호 입력 필드가 적절한 속성(type, name 등)을 가지고 있는지
    c. 로그인 버튼이 존재하고, 활성 상태인지
    d. 아이디 또는 비밀번호가 없을 때 로그인 버튼을 클릭하면 에러 메시지가 표시되는지
    e. 로그인 함수(예: handleLogin())가 예상대로 동작하는지
3. 언급한 것처럼 단위테스트와 통합테스트의 경계는 모호함

통합테스트

1. 여러 컴포넌트나 모듈이 함께 작동하는 경우를 테스트
    a. 컴포넌트간 상호작용을 보는 것
    b. 단위테스트보다 넓은 범위
2. 로그인으로 예를 들자면
    a. 아이디와 비밀번호를 입력하고 로그인 버튼을 클릭했을 때, 올바른 요청이 서버로 전송되는지
    b. 서버에서 응답이 오면 적절하게 UI가 업데이트되는지
    c. 로그인이 성공한 후에 사용자를 적절한 페이지로 리디렉션하는지

E2E 테스트

1. 사용자 관점에서 어플리케이션 플로우를 전체적으로 테스트
    a. 사용자의 실제 흐름을 테스트하는 것
2. 실제 환경에서 테스트
    a. 모든 컴포넌트
    b. 데이터베이스
    c. 네트워크
    d. 사용자 인터페이스
3. 로그인 기능을 예로 들면
    a. 사용자가 실제 환경과 같은 상황에서 웹사이트를 방문하여 아이디와 비밀번호를 입력하고, 로그인 버튼을 클릭하는 과정을 시뮬레이션
4. 애자일에 부적합하다고 생각함
    a. 빠른 대응/수정이 불가능함

프론트엔드 TDD

1. 완벽한 TDD는 불가능
2. Component UI를 먼저 작성한다면
   a. 비즈니스 로직에 대한 테스트를 먼저 작성하고
   b. 비즈니스 로직을 작성하는 것은 가능할듯
3. 하지만 완벽한 TDD라고 보기는 어려움

[아하!모먼트] 요즘 관심있는 분야는 어떤것인가요?

1. 무엇을 공부할 것인가?
    a. 최신동향 파악
    b. 채용공고 분석
2. 기술면접 대비
    a. 프로그래머스
    b. Leetcode
    c. 면접 스터디
    d. 실전 면접 경험