포트폴리오 작성을 위한 리액트 프로젝트 고도화
1회차

지역변수 vs 전역변수

Agenda

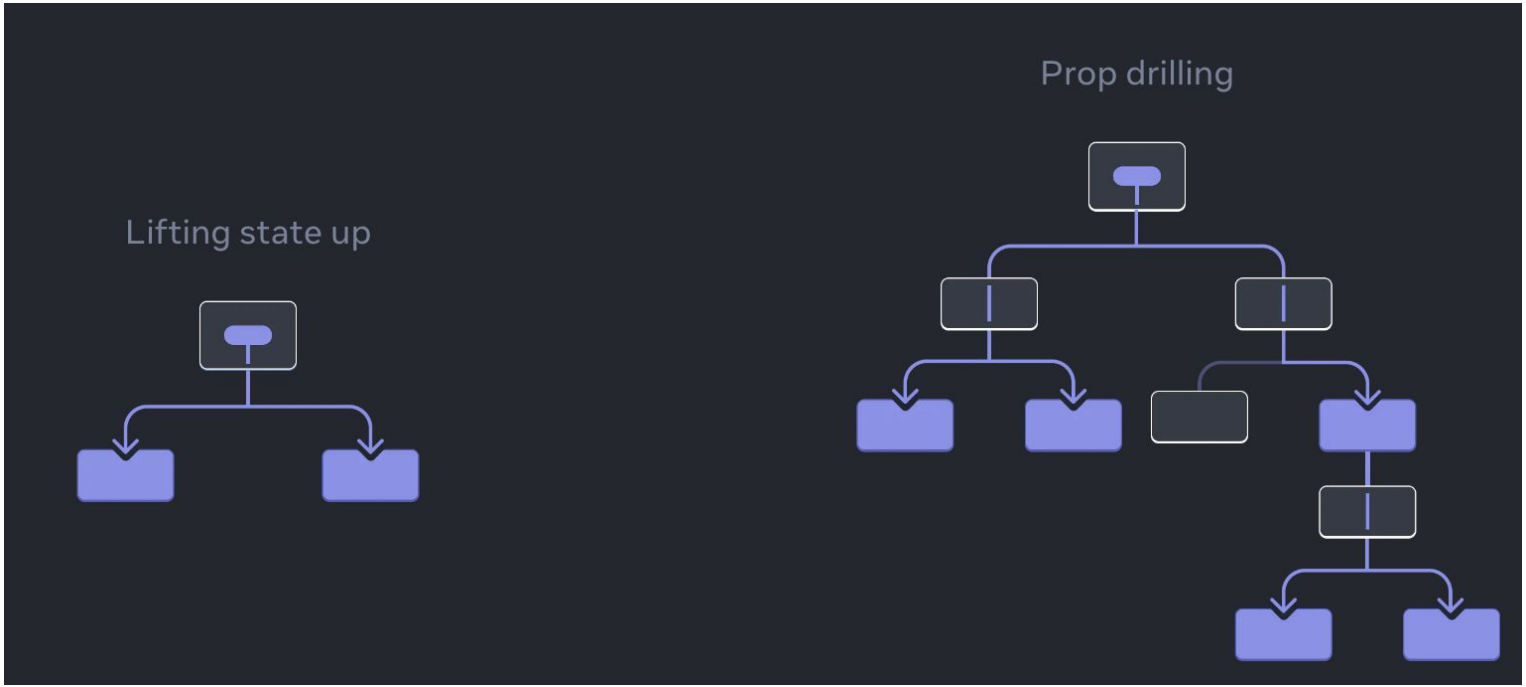
1. 지역변수와 전역변수 비교
   a. 언제 어떤것을 사용할 것인지?
   b. 꼭 필요한지?
2. 다양한 툴 비교
   a. Context API, Redux, Recoil, Zustand
3. React Query
4. 렌더링 효율 비교

지역변수 vs 전역변수

1. state를 어떻게 관리할까?
    a. 리액트에 한정짓지 않고 프로그래밍 관점에서 비교한다면?
        i. Scope
        ii. Lifespan
        iii. Namepsace collision
    b. var vs let vs const
2. 해당 변수를 어디에서 사용하는가?
3. 전역변수 설정하는 기준?

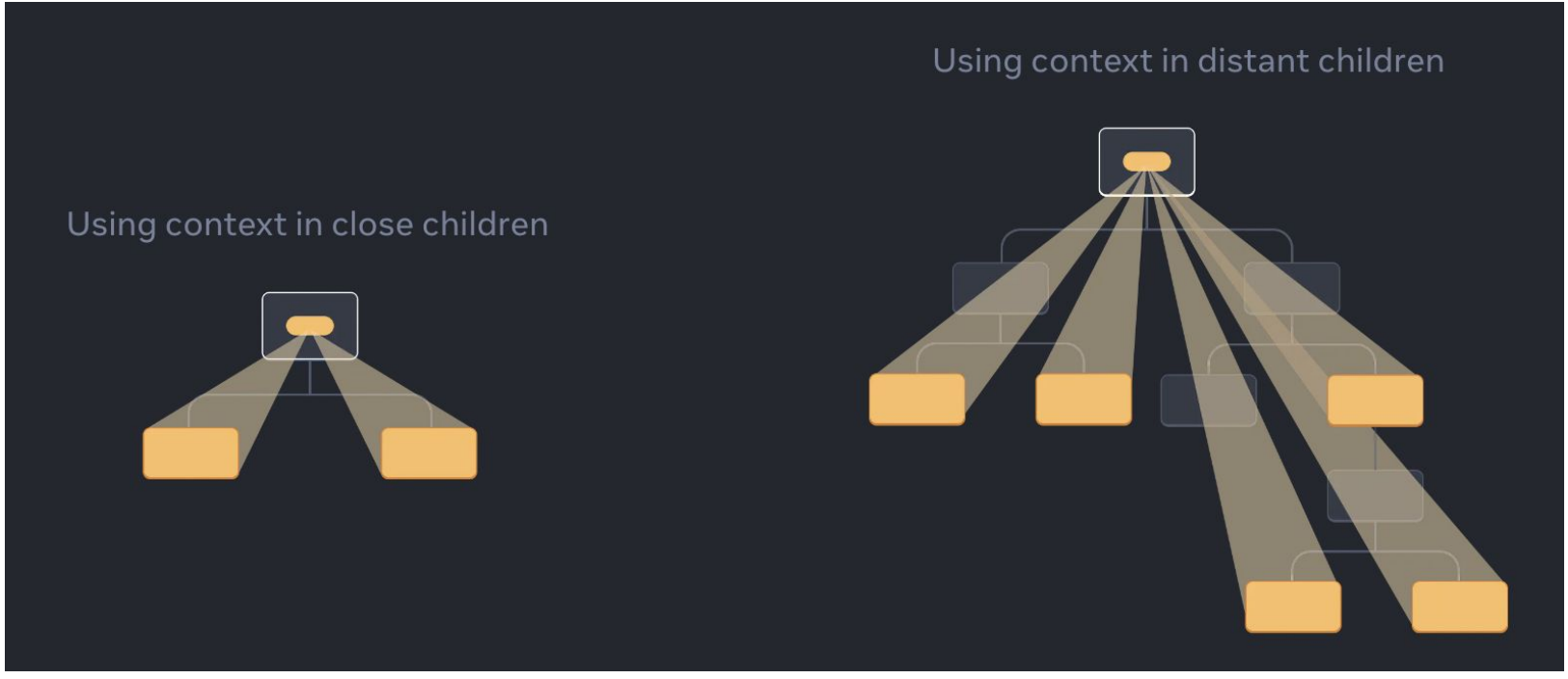
전역 상태 관리의 필요성

1. Container - Presenter 방식



전역 상태 관리의 필요성

1. Flux && Redux



전역변수 툴 비교 - Context API (feature/context_api)

1. 특징
    a. `State` 를 `Provide` 하는 방식
    b. React 내장 기능이라 별도 라이브러리 설치가 필요하지 않음
    c. HTTP Request도 Context에서 모두 관리
        i. `context`라는 단어에 맞게 해당 관심사와 관련된 모든 것을 context가 처리

전역변수 툴 비교 - Context API (feature/context_api)

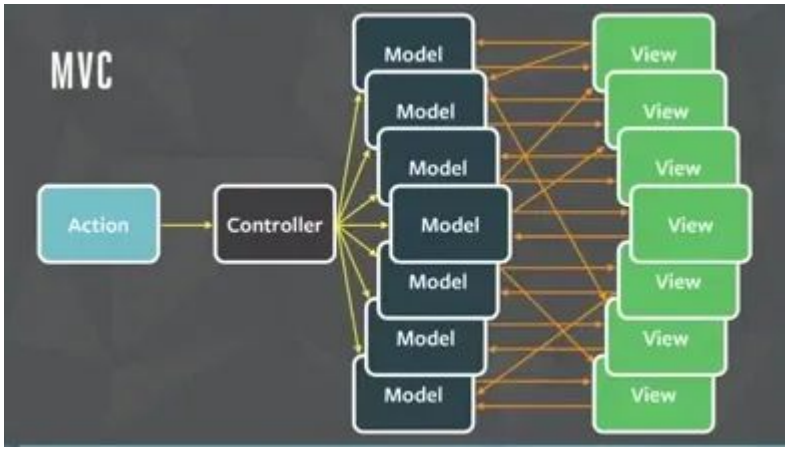
1. 특징
    a. `State` 를 `Provide` 하는 방식
    b. React 내장 기능이라 별도 라이브러리 설치가 필요하지 않음
    c. HTTP Request도 Context에서 모두 관리
        i. `context`라는 단어에 맞게 해당 관심사와 관련된 모든 것을 context가 처리
2. 단점
    a. 비지니스 로직이 복잡해지는 경우 Provider 다수 생성 필요
    b. Provider 1개의 경우에도 설정할 것들이 많음

전역변수 툴 비교 - Redux (feature/redux)

1. 특징
    a. FLUX 패턴을 적용함

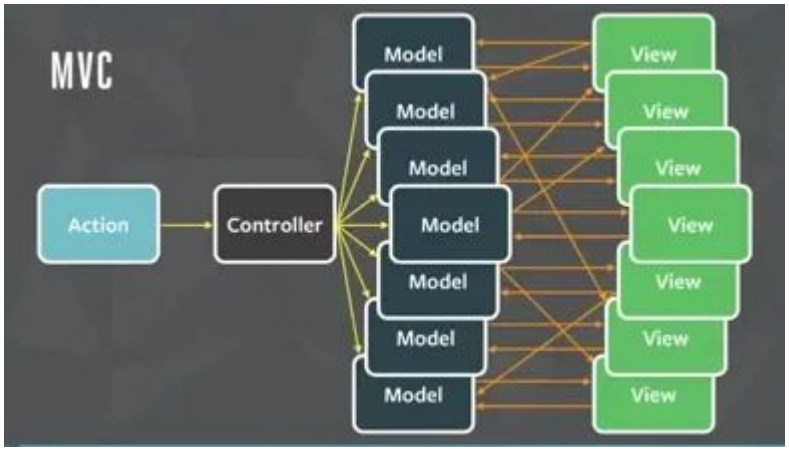
전역변수 툴 비교 - Redux (feature/redux)

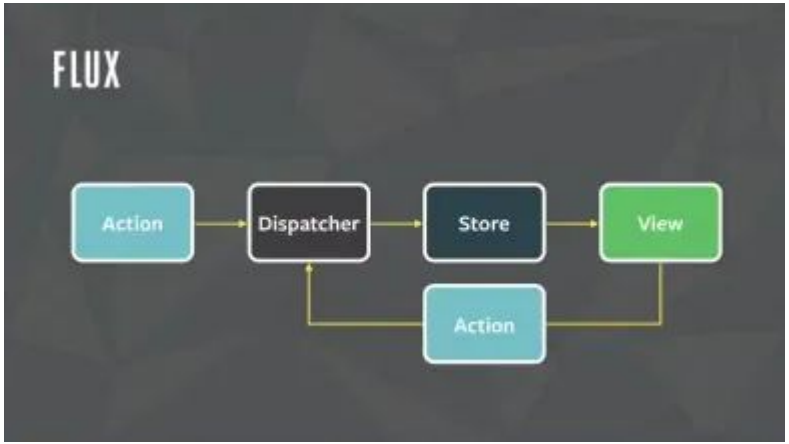
1. 특징
    a. FLUX 패턴을 적용함



전역변수 툴 비교 - Redux (feature/redux)

1. 특징
    a. FLUX 패턴을 적용함





전역변수 툴 비교 - Redux (feature/redux)

1. 특징
    a. FLUX 패턴을 적용함
    b. Store 하나에서 변수를 관리하기 때문에 여러 Provider를 선언하지 않아도 됨
    c. Thunk, saga 등의 Middleware를 통한 비동기처리
    d. Redux ToolKit 나오고 설정이 비교적 수월해짐
2. 단점
    a. 설정할 것이 엄청 많음
    b. FLUX 패턴을 적용했기 때문에 효율이 떨어질 수도 있음
    c. 요즘 새로 시작하는 프로젝트에서는 잘 안씀
        i. 애자일이 일반화(?) 되면서 그런 것 같은데
        ii. 요즘은 작게 시도하고 빠르게 실패하고 재도전 하는게 추세
        iii. 처음부터 redux를 도입하기는 쉽지않음

전역변수 툴 비교 - Recoil (feature/recoil)

1. 특징
    a. 내가 필요한 값만 `subscribe`하는 느낌
    b. `useXXXX`와 같이 리액트 개발자들에게 익숙한 문법
    c. 구조가 간단해서 적용하기 쉬움
    d. Concurrent mode 지원
    e. 렌더링 효율이 좋다는 일반적인 의견 - 케바케
2. 단점
    a. 커뮤니티 서포트가 많지 않음
    b. atom이 엄청 많아질수도
    c. 미들웨어 지원 없음
        i. HTTP Request 처리 별도로 해야함

전역변수 툴 비교 - Zustand (feature/zustand)

1. 특징
    a. 지금까지 본것들중에 설정이 제일 간단함
    b. 별다른 패턴 없이 자유롭게 적용 가능함
    c. Shallow comparison을 통한 효율 개선
    d. 함수형, 클래스형 다 지원함
2. 단점
    a. Recoil과 마찬가지로 커뮤니티 지원 부족
    b. 미들웨어 없음

Server State - React Query

1. 비즈니스 로직들을 대부분 백엔드에서 관리
    a. 많은 변수들이 서버에서 불러온 값을 관리하기 위해 사용
2. 서버에서 불러온 값들을 클라이언트가 관리하지 말자
3. 코드를 다시 보면…

React Query 최적화

1. Cache된 데이터의 상태
    a. Fresh
    b. Stale
    c. Inactive
2. staleTime vs cacheTime
    a. useRestaurantList를 통해서 확인

가장 성능이 좋은 툴은?

1. 사전질문에 redux, recoil, zustand 등등 비교했을 때 뭐가 제일 성능이 좋은지
    a. 사실 크게 의미 없음
    b. 굳이 따지자면 Context API가 성능에는 제일 안좋을 것
        i. 상태 변경 시 동시에 너무 많은 부분을 re-rendering하기 때문
        ii. 그런데 이마저도 케바케

그래서 뭘 써야하나?

1. 포트폴리오 만드는 신입/취준 개발자
    a. 아무거나 하나 잡아서 하나를 진득하게
    b. 지금은 “다양한 경험" 보다는 “깊이 들어갈 수 있는 가능성"을 증명하는 것이 중요함
    c. 어떤 라이브러리를 사용하는지는 크게 상관없음
        i. 다만 “해당 라이브러리를 선택한 이유”를 설명할 수 있어야 함
        ii. “왜" 가 중요하다
2. 일하고 있는 주니어 개발자
    a. 회사에서 사용하는 툴을 깊게 공부하는 것을 추천
    b. 새로나온 / 사용해보지 못한 툴이 궁금하다면 가볍게 토이프로젝트 해보고 블로그 작성
3. 그래도 추천해달라고 하시면
    a. react-query
    b. recoil 또는 zustand
        i. redux는 개인프로젝트에서 사용하기에는 너무 부담
        ii. context api는 현업에서 사용하지 않음

전역변수를 사용하지 않을 수는 없을까?

1. Toss SLASH 23
   a. 페이지 routing을 하지 않고 component로 관리
   b. 관련성 있는 코드의 “응집도"를 높이는 방식
2. 항상 이런 방식을 택할수는 없음
   a. 개발팀의 컨벤션 중요함
   b. 컨벤션을 지키는 코드가 가독성에 유리

Recap

1. 지역변수 vs 전역변수
2. 다양한 툴들 비교
3. React-Query의 장점