

LUMOS

이현진

23.10.05

Agenda. 내가 느꼈던 것들 순서

- 패키지 관리자 문제 + node_modules
- hooks와 queries
- Widget Driven Development(WDD)

패키지 관리자 문제

- @tossteam/* 패키지를 nexus로 다운받는 것으로 변경됨
- 현재 .yarnrc 파일은 변경된 nexus 레지스트리를 지정
- 하지만, yarn.lock은 기존 레지스트리를 가르킴 => lock을 지우고 패키지 업데이트 필요

lock 파일을 지우고 새로 설치해보자

- install만 5분이 걸렸다.

```
success Saved lockfile.  
✨ Done in 621.86s.
```

- install 후 tsc 컴파일 할 때 특정 TDS 컴포넌트에서 새로운 prop crossOrigin을 무조건 요구하는 문제

```
src/pages/LoansFlowControl/LoansFlowControl.tsx:761:26 - error TS2741: Property 'crossOrigin' is missing in type '{ checked: boolean; css: SerializedStyles; name: string; label: string; value: string; onChange: (event: ChangeEvent<HTMLInputElement>) => { ...; }; }' but required in type 'Pick<Props, "size" | "title" | "form" | "label" | "slot" | "style" | "pattern" | "color" | "height" | "translate" | "width" | "accept" | "alt" | "autoComplete" | "capture" | ... 273 more ... | "labelFontWeight">'.
```

761

<Radio

~~~~~

Found 159 errors in 59 files.

- 왜 같은 버전을 설치했음에도 다른 결과가 있을까?

Phantom Dependency 때문인가? Type을 다른 곳에서 불러오는 것인가?

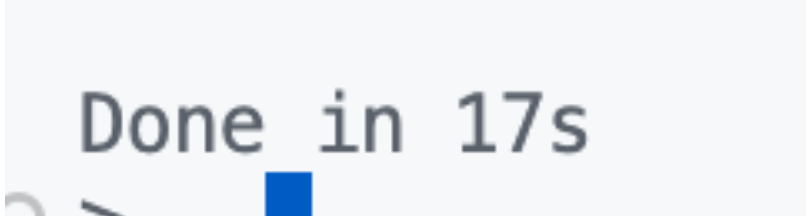
현재는 @tossteam 패키지를 압축해서 전달받아서 갈아끼우는 것으로 해결하고 있다.

꼭 고치고 싶다..

# npm, yarn classic의 문제

1. 매우 큰 공간차지, 현재 프로젝트에서 install만 5분 소요

- pnpm으로 바꿔서 테스트 -> 17초로 단축



Done in 17s

- yarn berry로 바꿔서 테스트 -> 18초로 단축



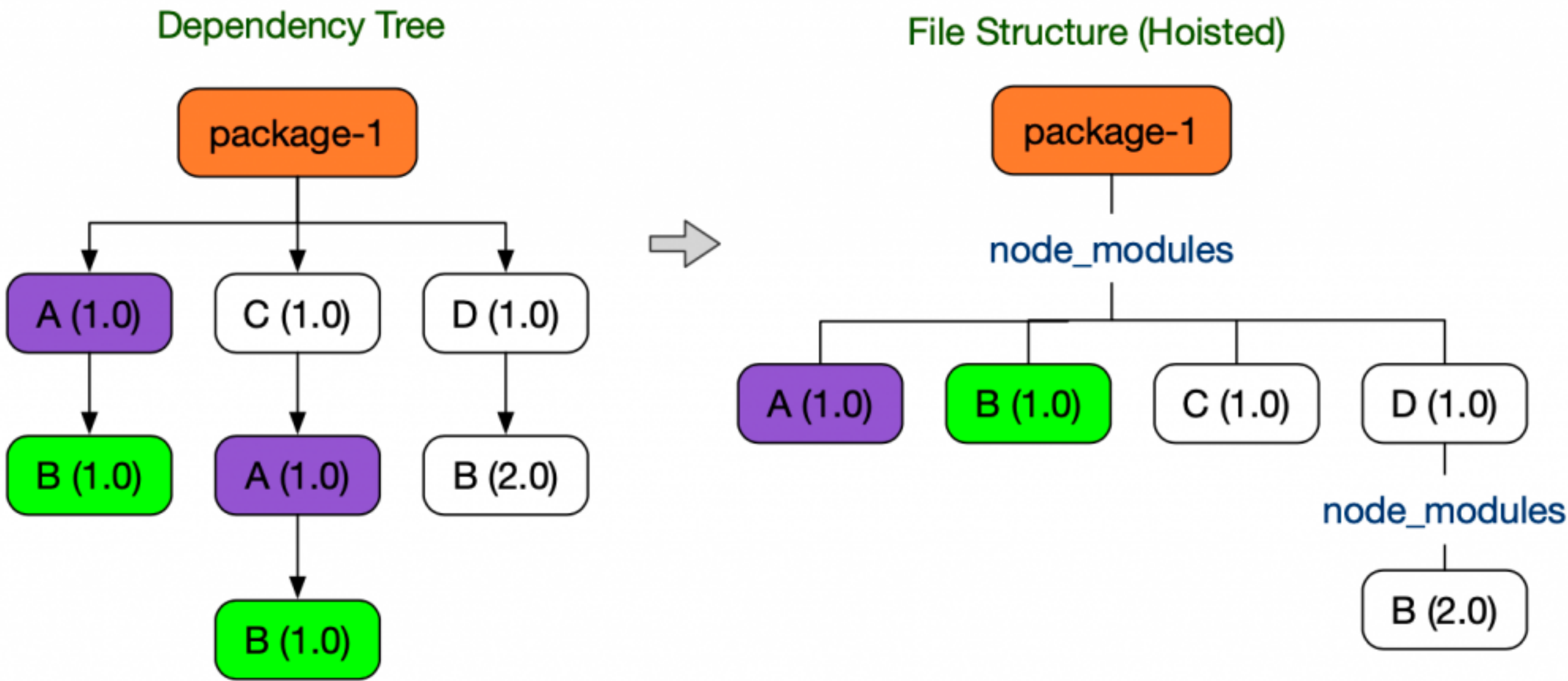
YN0000: Completed  
YN0000: Done with warnings in 18s 442ms

- 패키지 관리자를 바꾸고싶음 앞의 TDS 문제나, CI, CD에서 발생할 수 있는 Side Effect 체크 필요

2. npm 및 yarn classic에서는 중복해서 설치되는 node\_modules를 위해 끌어올리기(hoisting)을 사용하는데 이는 **유령 의존성, Phantom Dependency** 문제를 발생시킨다. 유령의존성은 package.json에 명시되지 않은 라이브러리를 사용할 수 있는 현상을 발생시킨다.

3. 비효율적인 의존성 검색(재귀적 탐색)

# Phantom Dependency 우리 프로젝트에서는..



```
src/pages/DueControl/LegalAction/Detail/DetailPage.tsx
```

```
1 + import { ReactElement } from 'react';
2 + import styled from '@emotion/styled';
3 + import { Spacing } from '@tossteam/emotion-utils';
4 + import { useRouter, useQueryParam } from '@tossteam/use-query-param';
```

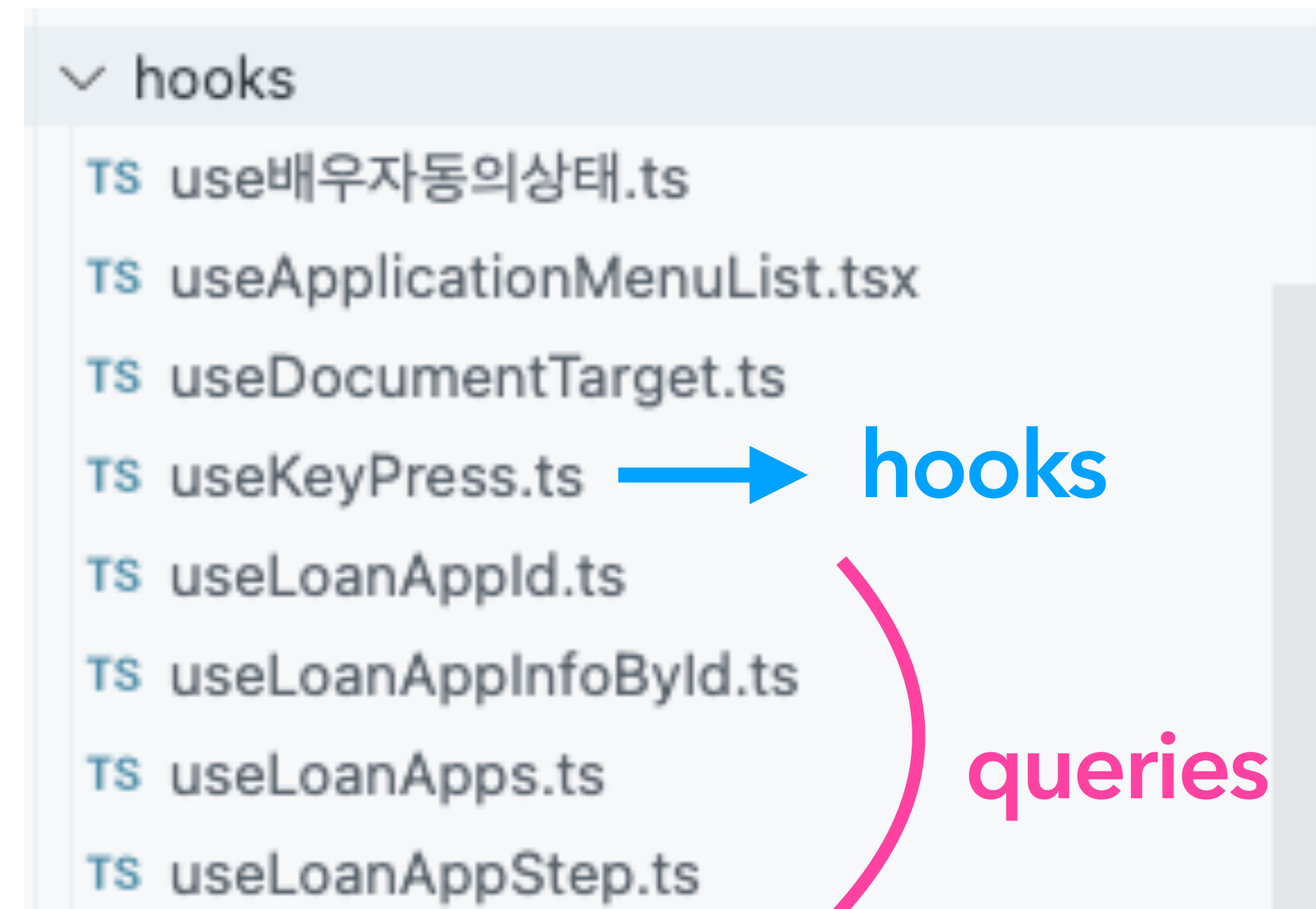
asst2309083 6 days ago  
이거 package.json에 없는 것 같아요. @tossteam/use-query-param

asst2308065 6 days ago  
아 ! 어쩐지 autoComplete가 불안정하더니 언제 빠졌나요?ㅋㅋ

```
import { Container } from '@mui/system';
```

# hooks와 queries

- 애플리케이션에서 공통 기능을 가지는 hook은 최상단에 위치시켜 어디서든 가져올 수 있게 한다
- hook과 query를 구분한다 (queries 폴더 제안)



# hooks와 queries

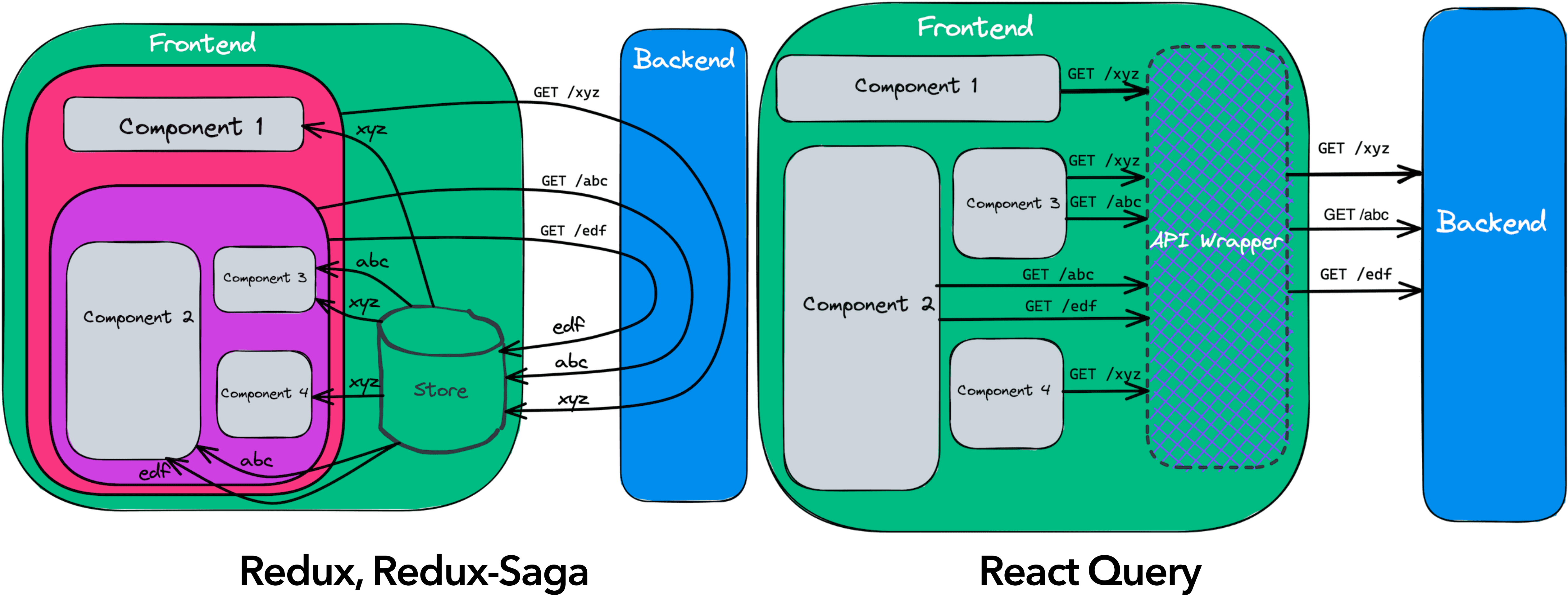
분리하는 이유는?

1. 애플리케이션 내부의 어떤 기능을 선언적으로 정의하는 것과 애플리케이션 외부의 `server state`를 가져오는 것은 명확히 역할이 다르다고 생각
2. 프로젝트를 둘러보면 실제로 대부분의 `queries`에 해당한다  
쿼리가 아니라 애플리케이션의 기능을 담당하는 거라면 따로보는게 구분하기 쉽다
3. 페이지 단위로 실제 쿼리들이 어느 레벨에 영향을 미치는지에 따라 `query`들을 배치할 수 있다

예를 들어 `page` 단위에 영향을 미친다면 `page` 하위의 `queries` 폴더에 두고, 각각의 위젯 단위로 영향을 미친다면 각각의 위젯의 범위에 배치시켜서 폴더 단위에서 해당 쿼리가 어디에 쓰이는지 제한시킨다.  
폴더 `scope`로 쿼리를 격리

# 3000미터 위에서 내려다보기

LOMOS는.. 2가지 방법을 모두 사용한다



# Redux, React Query

Redux와 같은 스토어는 데이터를 컴포넌트의 외부에 놓고 Dispatch할 수 있는 액션을 정의해 데이터 Flow를 관리한다

서버 상태를 가져오기 위해 스스로 로딩상태, 에러 상태를 정의할 필요가 있을까?

데이터를 가져오는 query가 목적이라면 React Query로 점진적으로 대체하는게 좋다고 생각

Redux는 제거해하는가?

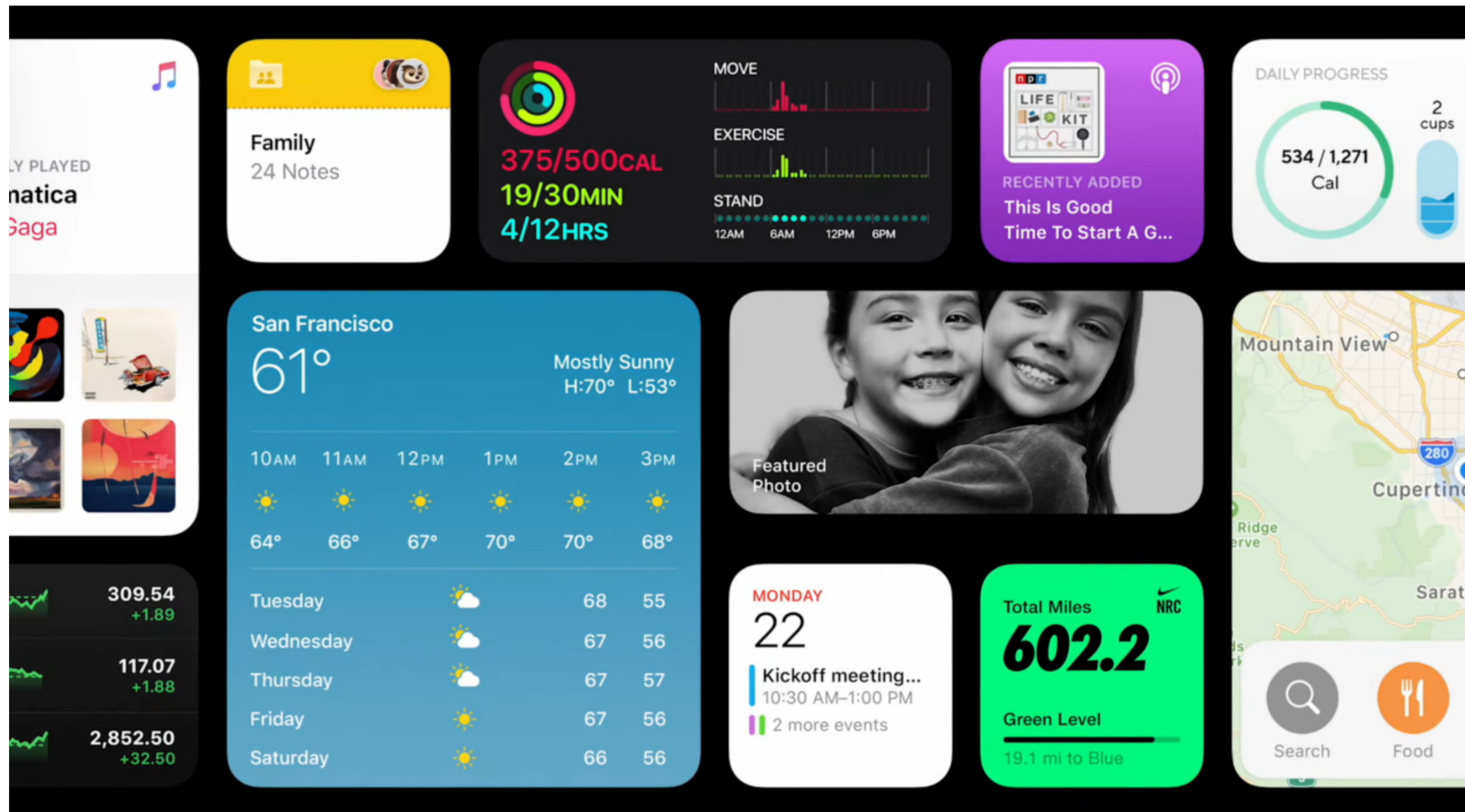
동시에 바뀌는 로컬 state가 거의 없다,, 제거할 수 있다고 생각

물론 지금 프로젝트에서 잘 돌아가고 있으므로 점진적 개선.

# Widget Driven Development

LUMOS에 특화된 프로젝트 아키텍처가 있을까?

React Query를 잘 사용하고 그 범위를 잘 제어할 수 있도록..



위젯은 각각은 자신의 역할이 있고 외부 데이터를 가져온다.

# Widget Driven Development

☰

toss bank

김종인 님

전세대출 신청 업무분배

담당자별

고객별

Widget

페이지 크기

30

고객번호

전월세신청번호

권리조사신청번호

계정계 고객번호

담당자 이름

조회기간

2023-07-07

→

2023-10-05

조회

Widget

보고싶은 열

진행 상태

최종 심사 단계

미완료 단계

신청번호 외 15개

외부정보 적재 요청됨 외 11건

최종심사단계

미완료단계

조회된 목록

| 순번 | 신청번호 | 고객명 | 고객번호        | 상품명    | 진행 상태     | 최종 심사 단계 | 업무 상태 | 신청일        | 서류제출완료시각         | 대출실행예정일    | 잔금일        | 업무배정일      | 유효기간       | 담당자(ID)  | 비고 | 자세히 | 선택                       |
|----|------|-----|-------------|--------|-----------|----------|-------|------------|------------------|------------|------------|------------|------------|----------|----|-----|--------------------------|
| 1  | 3567 | 정현우 | 11035826118 | HF청년전세 | 어드민 심사 대기 | 없음       | 진행중   | 2023-10-04 | 2023-10-04 19:54 | -          | -          | -          | 2024-01-04 | -        |    | 🔍   | <input type="checkbox"/> |
| 2  | 3559 | 박신건 | 11032909620 | HF일반전세 | 어드민 심사 대기 | 없음       | 진행중   | 2023-09-25 | 2023-09-25 21:29 | -          | -          | -          | 2023-12-25 | -        |    | 🔍   | <input type="checkbox"/> |
| 3  | 3534 | 정현우 | 11035826118 | HF일반전세 | 실행성공      | 권리조사2차   | 진행중   | 2023-09-21 | 2023-09-21 12:41 | 2023-09-21 | 2023-09-21 | 2023-10-04 | 2023-12-21 | 김채연(166) |    | 🔍   | <input type="checkbox"/> |
| 4  | 3355 | 유소진 | 11042771813 | HF일반전세 | 어드민 심사 대기 | 기보중물건지   | 진행중   | 2023-08-29 | 2023-08-29 17:52 | -          | 2023-09-11 | 2023-10-04 | 2023-11-29 | 박신건(282) |    | 🔍   | <input type="checkbox"/> |
| 5  | 3271 | 조민국 | 11032908234 | HF일반전세 | 취소 완료     | 이미지 제출완료 | 진행중   | 2023-08-22 | 2023-08-22 21:09 | -          | -          | 2023-10-04 | 9999-12-31 | 박신건(282) |    | 🔍   | <input type="checkbox"/> |

전체 선택

전체 해제

<

1

>

선택 업무 회수

선택 업무 배정

위젯은 데이터 페칭의 주체가 되며  
여러 컴포넌트들로 구성된다

위젯을 구성하는 로직을  
Colocate 시킨다면, 폴더와 파일을 넘나들 필요가 없다

# 조회목록

보고싶은 열

진행 상태

최종 심사 단계

미완료 단계

신청번호 외 15개

외부정보 적재 요청됨 외 11건

최종심사단계

미완료단계

## 조회된 목록

| 순번 | 신청번호 | 고객명 | 고객번호        | 상품명    | 진행 상태     | 최종 심사 단계 | 업무 상태 | 신청일        | 서류제출완료시각         | 대출실행예정일    | 잔금일        | 업무배정일      | 유효기간       | 담당자(ID)  | 비고 | 자세히 | 선택                       |
|----|------|-----|-------------|--------|-----------|----------|-------|------------|------------------|------------|------------|------------|------------|----------|----|-----|--------------------------|
| 1  | 3567 | 정현우 | 11035826118 | HF청년전세 | 어드민 심사 대기 | 없음       | 진행중   | 2023-10-04 | 2023-10-04 19:54 | -          | -          | -          | 2024-01-04 | -        |    | 🔍   | <input type="checkbox"/> |
| 2  | 3559 | 박신건 | 11032909620 | HF일반전세 | 어드민 심사 대기 | 없음       | 진행중   | 2023-09-25 | 2023-09-25 21:29 | -          | -          | -          | 2023-12-25 | -        |    | 🔍   | <input type="checkbox"/> |
| 3  | 3534 | 정현우 | 11035826118 | HF일반전세 | 실행성공      | 권리조사2차   | 진행중   | 2023-09-21 | 2023-09-21 12:41 | 2023-09-21 | 2023-09-21 | 2023-10-04 | 2023-12-21 | 김채연(166) |    | 🔍   | <input type="checkbox"/> |
| 4  | 3355 | 유소진 | 11042771813 | HF일반전세 | 어드민 심사 대기 | 기보중물건지   | 진행중   | 2023-08-29 | 2023-08-29 17:52 | -          | 2023-09-11 | 2023-10-04 | 2023-11-29 | 박신건(282) |    | 🔍   | <input type="checkbox"/> |
| 5  | 3271 | 조민국 | 11032908234 | HF일반전세 | 취소 완료     | 이미지 제출완료 | 진행중   | 2023-08-22 | 2023-08-22 21:09 | -          | -          | 2023-10-04 | 9999-12-31 | 박신건(282) |    | 🔍   | <input type="checkbox"/> |

전체 선택

전체 해제

선택 업무 회수

선택 업무 배정

widgets/  
조회목록/  
보고싶은열필터.tsx  
진행상태필터.tsx ...  
조회된목록Table.tsx  
선택업무회수Button.tsx  
선택업무배정Button.tsx  
queries/  
use선택업무배정.ts  
use대출목록조회.ts  
use선택업무회수.ts

widgets/  
검색창/

# 검색창.tsx

계정계고객번호.tsx

## 조회기간.tsx

queries/

## useSearch.ts

## 조회목록

# 보고싶은열필터.tsx

## 진행상태필터.tsx ...

## 조회된 목록Table.tsx

## 선택업무회수Button.ts

## 선택업무배정Button.ts

queries/

use선택업무배정.t

use대출목록조회.ts

## use선택업무회수.ts

queries/

페이지 크기

30

고객번호

전월세신청번호

권리조사신청번호

계정계 고객번호

담당자 이름

조회기간

2023-07-07 → 2023-10-05

조회

보고싶은 열      진행 상태      최종 심사 단계      미완료 단계  
 🔼 신청번호 외 15개 ▼    🔼 외부정보 적재 요청됨 외 11건 ▼    🔼 최종심사단계 ▼    🔼 미완료단계 ▼

## 조회된 목록

| 순번 | 신청번호 | 고객명 | 고객번호        | 상품명    | 진행 상태     | 최종 심사 단계 | 업무 상태 | 신청일        | 서류제출완료시각         | 대출실행예정일    | 잔금일        | 업무배정일      | 유효기간       | 담당자(ID)  | 비고 | 자세히 | 선 |
|----|------|-----|-------------|--------|-----------|----------|-------|------------|------------------|------------|------------|------------|------------|----------|----|-----|---|
| 1  | 3567 | 정현우 | 11035826118 | HF청년전세 | 어드민 심사 대기 | 없음       | 진행중   | 2023-10-04 | 2023-10-04 19:54 | -          | -          | -          | 2024-01-04 | -        |    |     |   |
| 2  | 3559 | 박신건 | 11032909620 | HF일반전세 | 어드민 심사 대기 | 없음       | 진행중   | 2023-09-25 | 2023-09-25 21:29 | -          | -          | -          | 2023-12-25 | -        |    |     |   |
| 3  | 3534 | 정현우 | 11035826118 | HF일반전세 | 실행성공      | 권리조사2차   | 진행중   | 2023-09-21 | 2023-09-21 12:41 | 2023-09-21 | 2023-09-21 | 2023-10-04 | 2023-12-21 | 김채연(166) |    |     |   |
| 4  | 3355 | 유소진 | 11042771813 | HF일반전세 | 어드민 심사 대기 | 기보증물건지   | 진행중   | 2023-08-29 | 2023-08-29 17:52 | -          | 2023-09-11 | 2023-10-04 | 2023-11-29 | 박신건(282) |    |     |   |
| 5  | 3271 | 조만국 | 11032908234 | HF일반전세 | 취소 완료     | 이미지 제출완료 | 진행중   | 2023-08-22 | 2023-08-22 21:09 | -          | -          | 2023-10-04 | 9999-12-31 | 박신건(282) |    |     |   |

전체 선택

전체 해

< 1 >

선택 업무 회수

선택 업무 배정

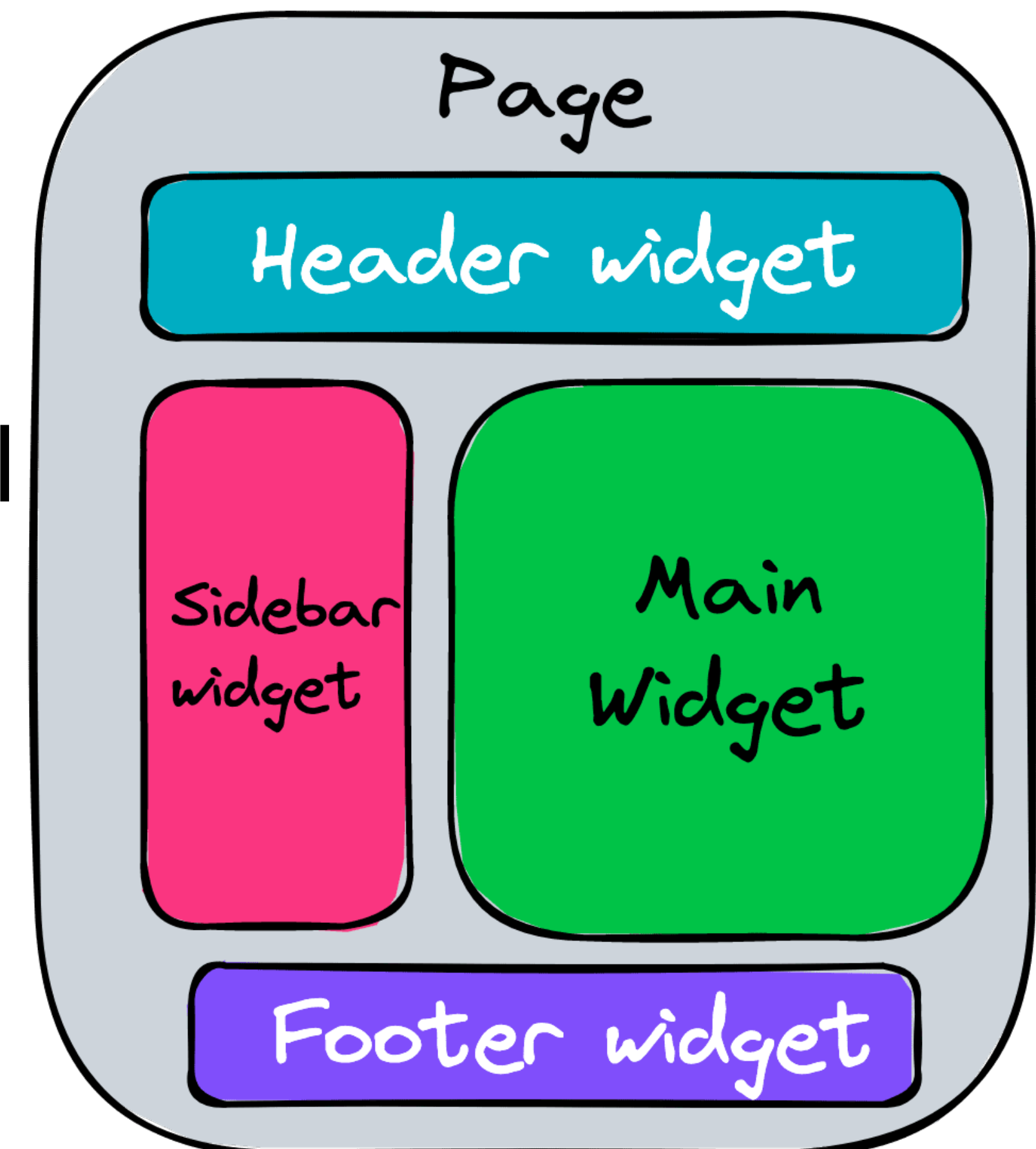
# Widget Driven Development

widget들을 조합한다면 page가 완성된다

Page 컴포넌트는 레이아웃, 위젯간의 간격 스타일링. 위젯에 전달할 props정의 같은 역할을 한다

모든 큰 덩어리를 위젯 취급할 수 있다.

위젯 폴더 내부에는 그 위젯을 구성하는 서로 연관된 컴포넌트들 배치



# Widget Driven Development

widgets/  
조회목록/

조회목록Suspense.ts

조회목록 ErrorBoundary.ts

보고싶은열필터.tsx

진행상태필터.tsx ...

조회된목록Table.tsx

선택업무회수Button.tsx

선택업무배정Button.tsx

queries/

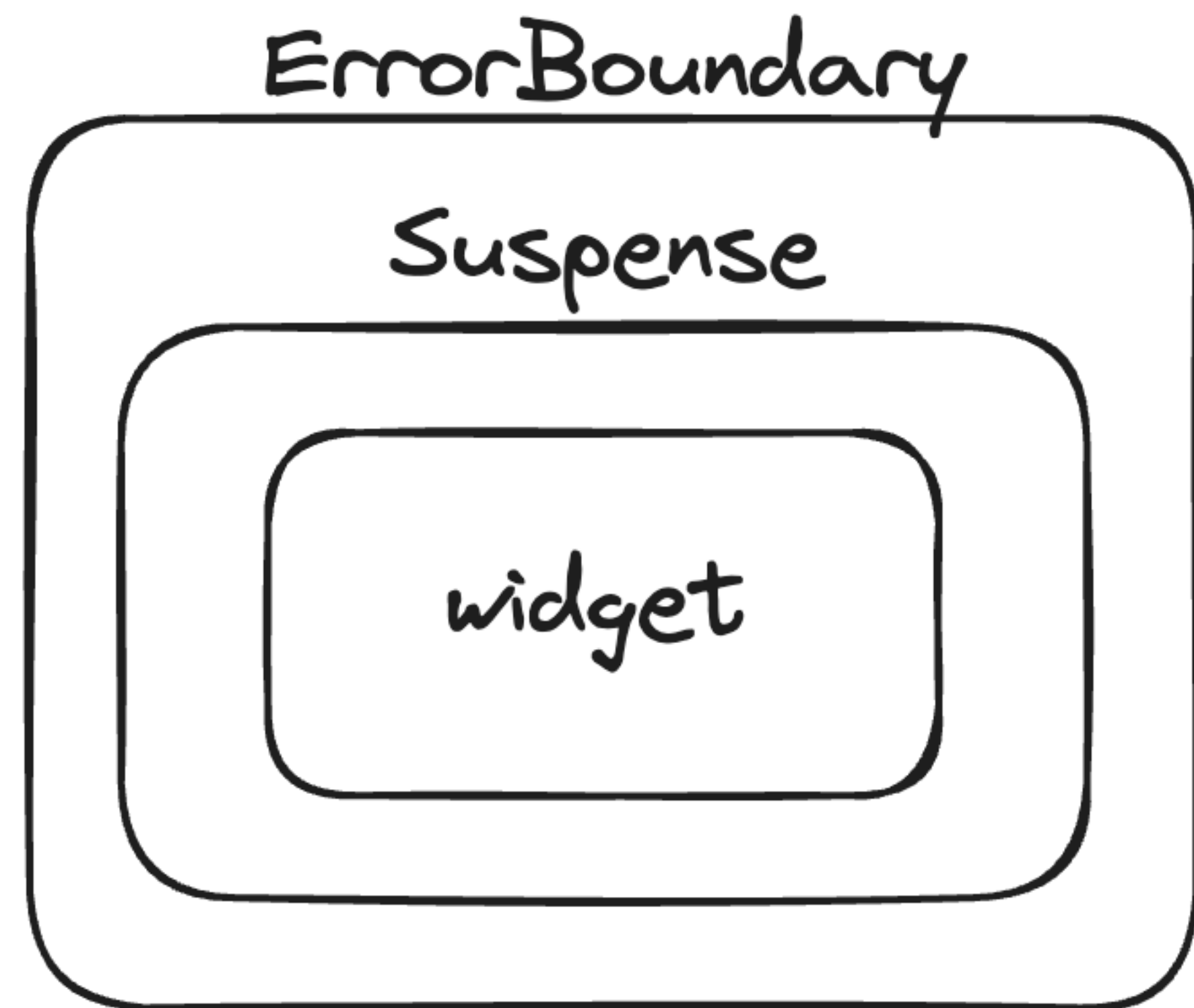
use선택업무배정.ts

use대출목록조회.ts

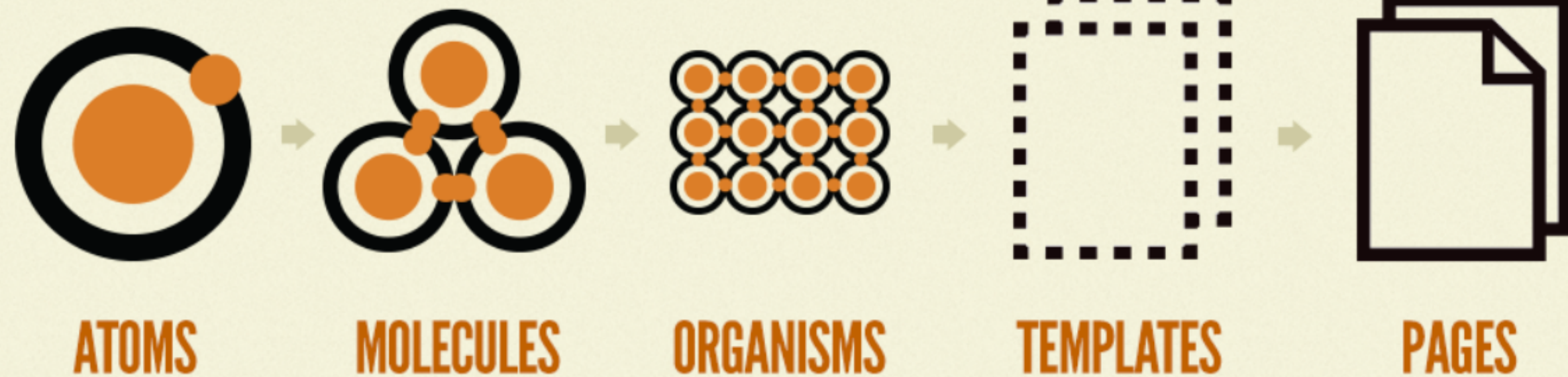
use선택업무회수.ts

queries/

각각의 위젯은 그 기능의 주체이기 때문에  
기능이 동작하지 않았을 때 각각의 바운더리를  
쉽게 정의할 수 있다.



# Atomic Design Pattern



TDS에 감사하며,,

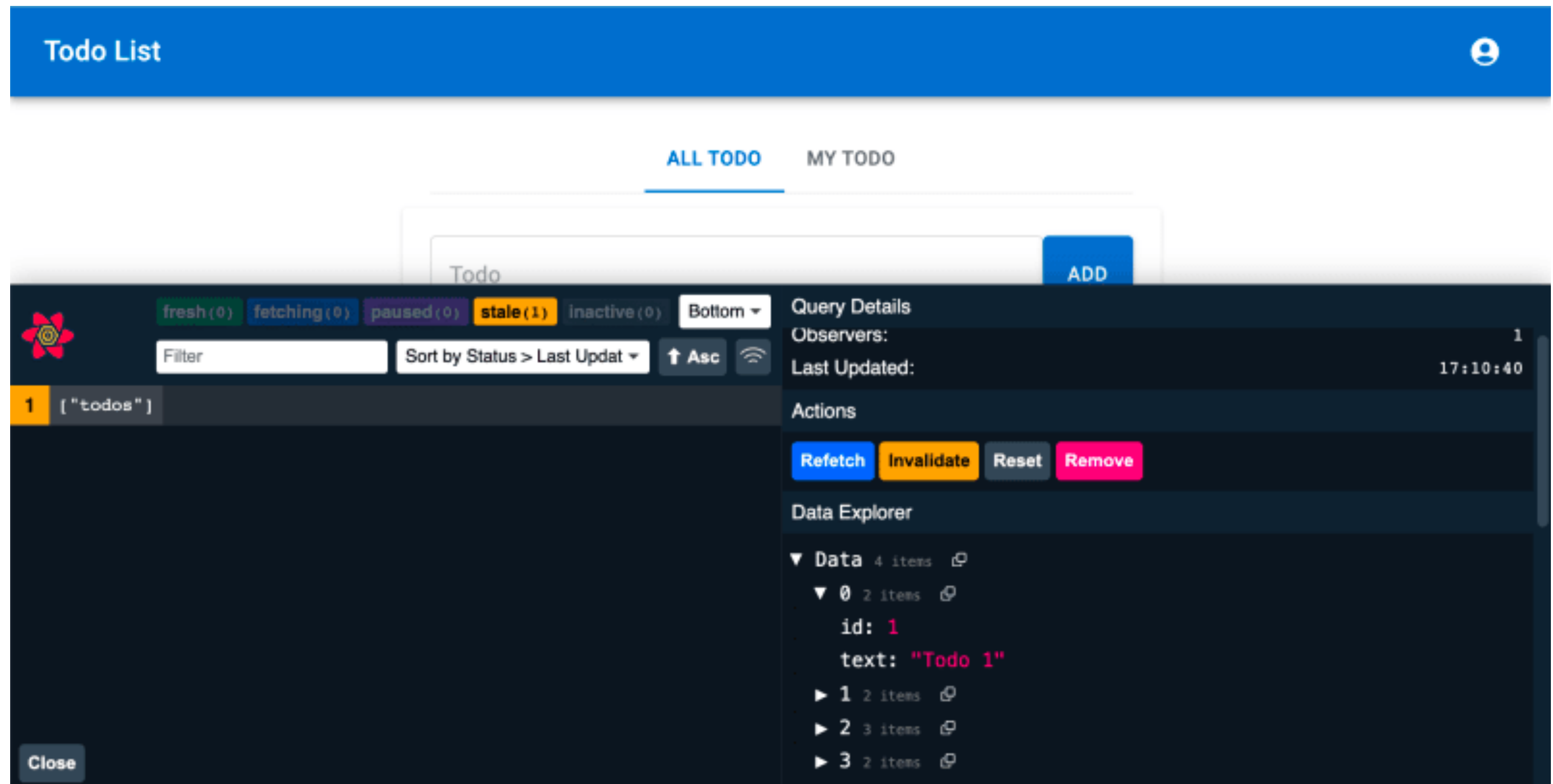
우리는 Atom을 건드릴 필요가 없다

Molecules, Organisms, Pages와 같은  
좀더 복잡한 컴포넌트를 다루는데에  
관심이 있다

결국 위젯은 molecules 이상의 조합

# React Query Devtools의 필요성

쿼리들을 직접 invalidate, refetch시키면서 더 쉽게 테스트할 수 있다



# 코드 컨벤션

- 함수 컴포넌트 내부에 함수 키워드로 함수 정의 vs arrow function으로 함수 정의

# LUMOS

이현진

23.10.11

# 패키지 의존성 문제

- yarn install 과정에서 의존성을 맞추면서 생기는 문제
- ^ (캐럿)은 라이브러리들이 semantic versioning을 따른다는 가정하에 동작
  - $^1.0.2$ 은  $1.0.2 \leq x < 2.0$
  - $^1.0$ 은  $1.0.0 \leq x < 2.0$
- package.json의 react 패키지를 "react": "16.8.2"라고 설정하면 캐럿의 범위는 16.8.2 이상 17.0.0 미만의 최신 버전을 적용한다

# 패키지 의존성 문제

기존 : @types/react에 대해 각각 어떤 버전을 쓸지 따로 정의

```
"@types/react@" :
  version "18.2.22"
  resolved "https://nexus.tossbank.bz/repository/npm-group/@types/react/-/react-18.2.22.tgz"
  integrity sha512-60fLT0LqzarLED203UQImc/lsNRgG0jE/a1mPW9KjMemY0LMITWEsbS4VvZ
  dependencies:
    "@types/prop-types" "*"
    "@types/scheduler" "*"
    csstype "^3.0.2"

"@types/react@^16":
  version "16.14.46"
  resolved "https://nexus.tossbank.bz/repository/npm-group/@types/react/-/react-16.14.46.tgz"
  integrity sha512-Am4pyXMrr6cWw/TN3oqHtEZl0j+G6Up/08m65+xF/3ZaUgkv1GAtTPWw4y
  dependencies:
    "@types/prop-types" "*"
    "@types/scheduler" "*"
    csstype "^3.0.2"

"@types/react@^17", "@types/react@^17.0.11", "@types/react@^17.0.3": You,
  version "17.0.65"
  resolved "https://nexus.tossbank.bz/repository/npm-group/@types/react/-/react-17.0.65.tgz"
  integrity sha512-oxur785xZYHvnI7TRS61dXbkIhDPnGfsXKv0cNXR/0mL4SipRIFpSMzA7HMI
  dependencies:
    "@types/prop-types" "*"
    "@types/scheduler" "*"
    csstype "^3.0.2"
```

```

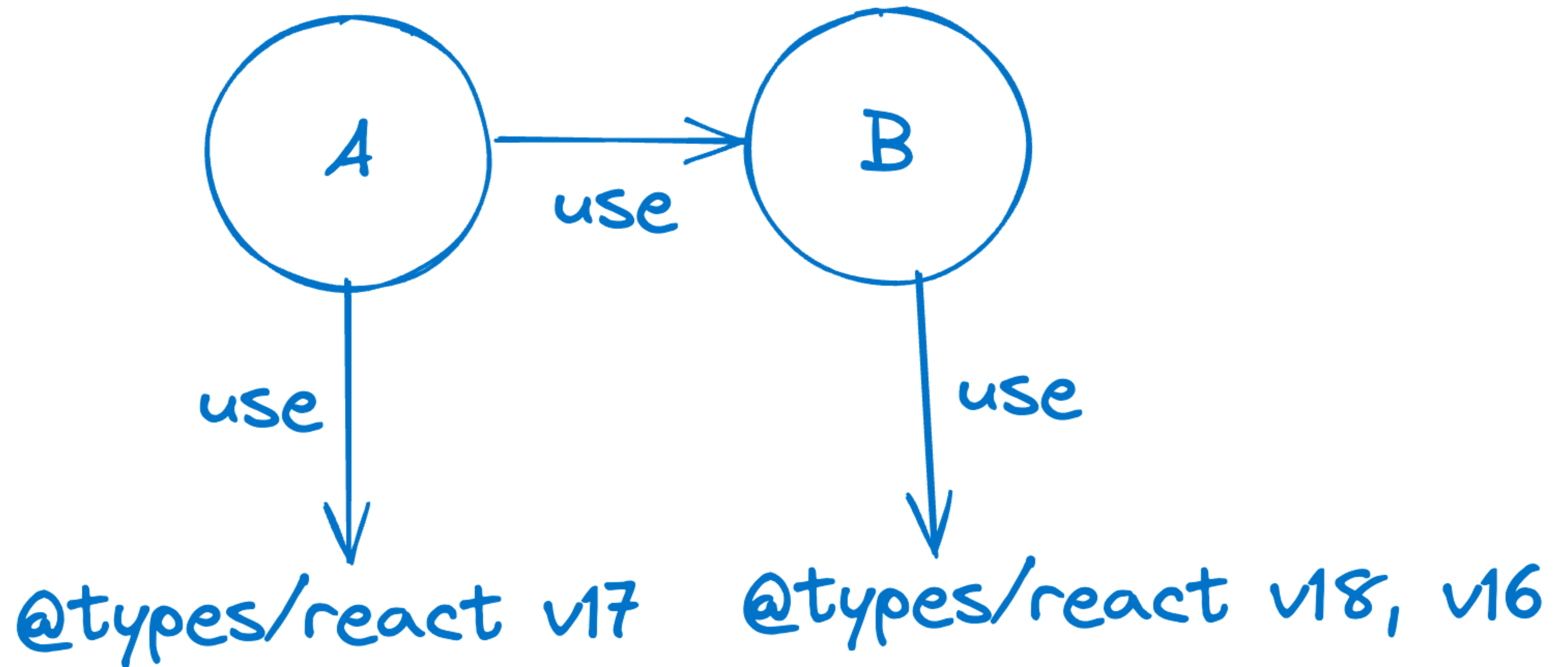
  version "17.0.65"
  resolved "https://nexus.tossbank.bz/repository/npm-group/@types/react/-/react-17.0.65.tgz"
  integrity sha512-oxur785xZYHvnI7TRS61dXbkIhDPnGfsXKv0cNXR/0mL4SipRIFpSMzA7HMI
  dependencies:
    "@types/prop-types" "*"
    "@types/scheduler" "*"
    csstype "^3.0.2"

```

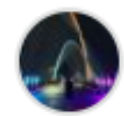
변경후: 모든 버전을 17.0.11로 고정

```
"@types/react*", "@types/react@17.0.11", "@types/react@^16", "@types/react@^17", "@types/
  version "17.0.11"
  resolved "https://nexus.tossbank.bz/repository/npm-group/@types/react/-/react-17.0.11.tgz"
  integrity sha512-yFRQbD+whVonItSk7ZzP/L+gPTJVBkL/7shLEF+i9GC/1cV3JmUxEQz6+9ylhUpWSDuqo1N
  dependencies:
    "@types/prop-types" "*"
    "@types/scheduler" "*"
    csstype "^3.0.2"
```

# 새로운 type 에러는 왜 발생하나요?



# Due Control 프로젝트에서 타입오류가 발생한 이유



asst2309083 2 days ago • edited ▾

Author



...

@asst2308065 호민님, 듀컨트롤에서 사용중인 컴포넌트에서 ts에러가나서 타입을 바꿨습니다. 타입정의 괜찮은지 한번만 체크부탁드립니다! 🙏



asst2308065 2 days ago



...

@asst2309083 해당 타입은 `react-hook-form` 내부에 정의된 타입이라 라이브러리 내부를 러프하게 봤는데요! 큰 문제 없어 보이네요!

hook-form 버전은 바뀌지 않은 것 같은데 어디서 ts에러가 났는지 궁금하네요! 위클리 때 공유해 주시면 좋을 것 같아요!



1

`react-hook-form@^7.18.0:`

version "7.36.1"

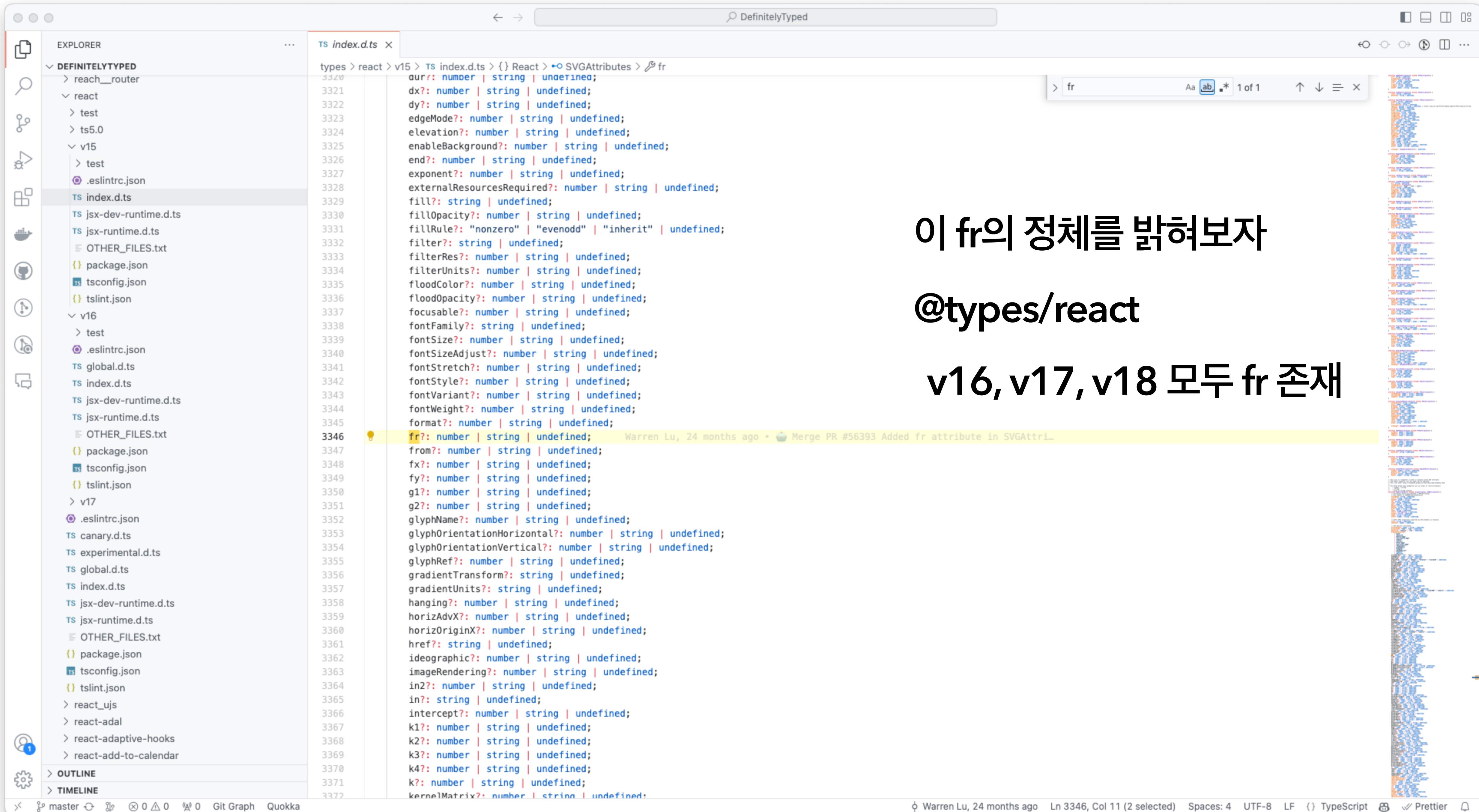
resolved "https://registry.yarnpkg.com/react-hook-form/-/react-hook-form-7.36.1.tgz#82a311fe8cbe75e689fd4529f083b7c983da6520"

integrity sha512-EbYYkCG2p8ywe7ik0H2l02lAFMrrrsLZi1I8fqd8ifDGNakhomHZQzQsP6ksvzrWBKntRe8b5L5L7Zsd+Gm02Q==

version "7.47.0"

resolved "https://nexus.tossbank.bz/repository/npm-group/react-hook-form/-/react-hook-form-7.47.0.tgz#a42f07266bd297ddf1f914f08f4b5f9783262f31"

integrity sha512-F/TroLjTICipmHeF\MrLtNLce02xr1jU3CyINla5zdwsGUGu2U0xxR4UyJgLlhMwLW/Wzp4cpJ7CPfgJIeKdSg==



이 fr의 정체를 밝혀보자

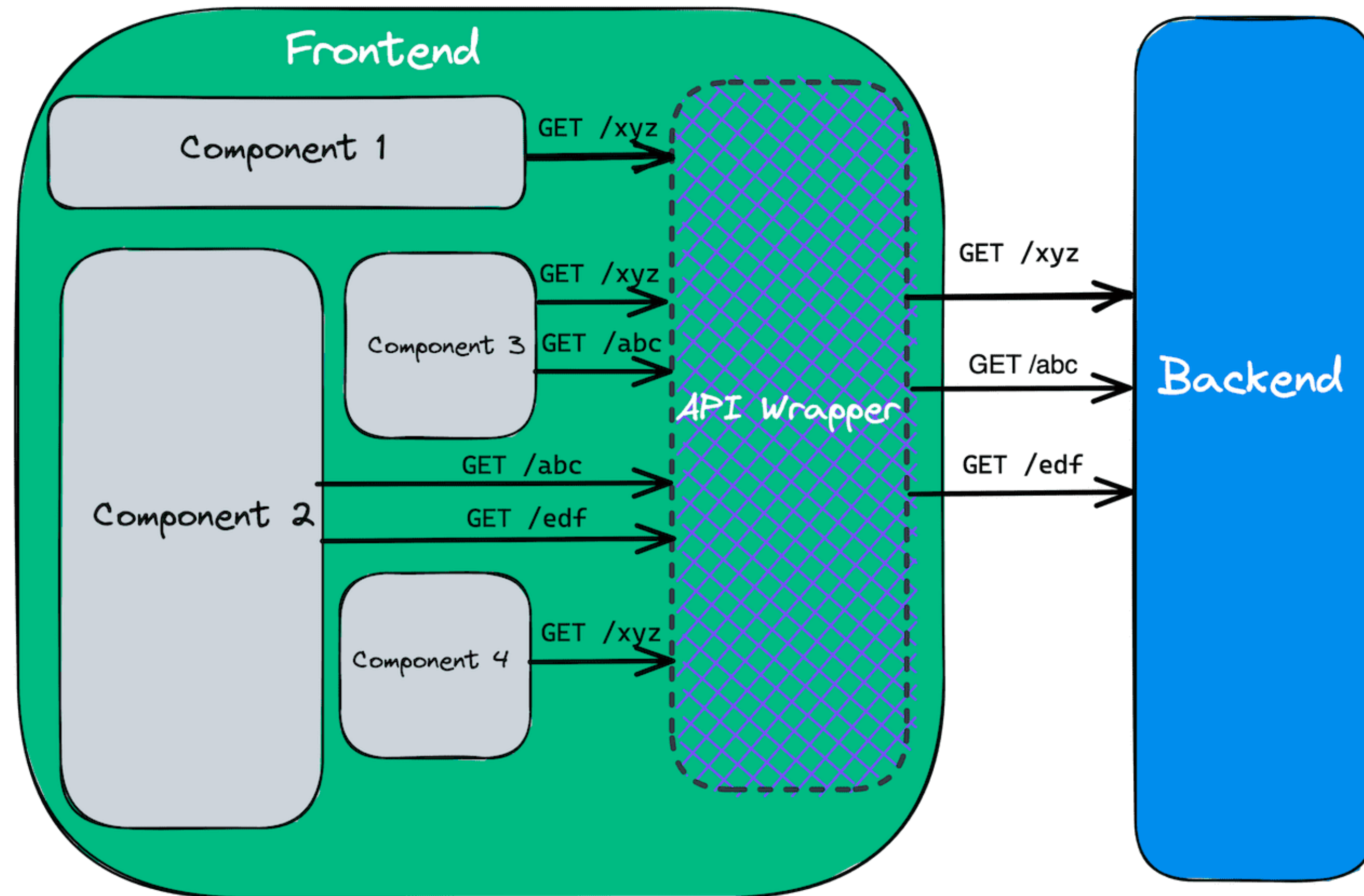
@types/react

v16, v17, v18 모두 fr 존재

# staleTime에 대하여

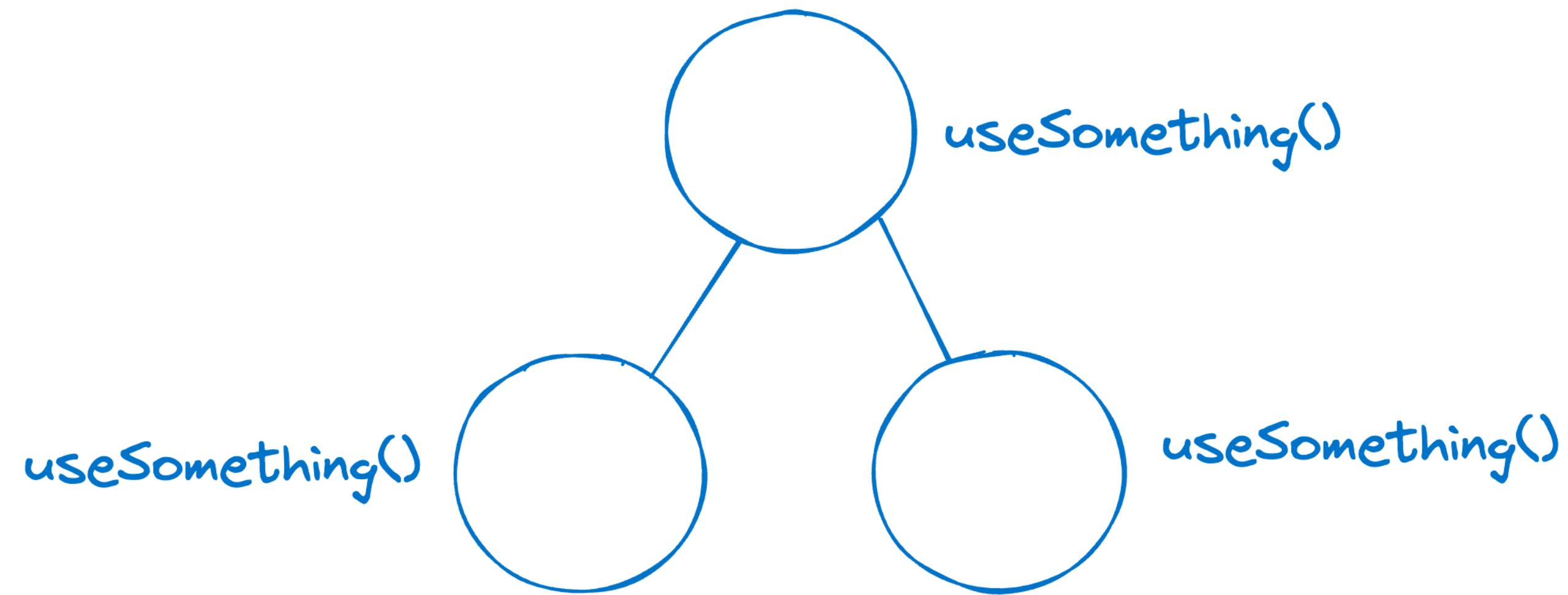
여기서 동일한 API Call은 한 번 일어난다

```
const { data: 첫째 } = useJeonseLoanSidebarStatus();  
const { data: 둘째 } = useJeonseLoanSidebarStatus();
```

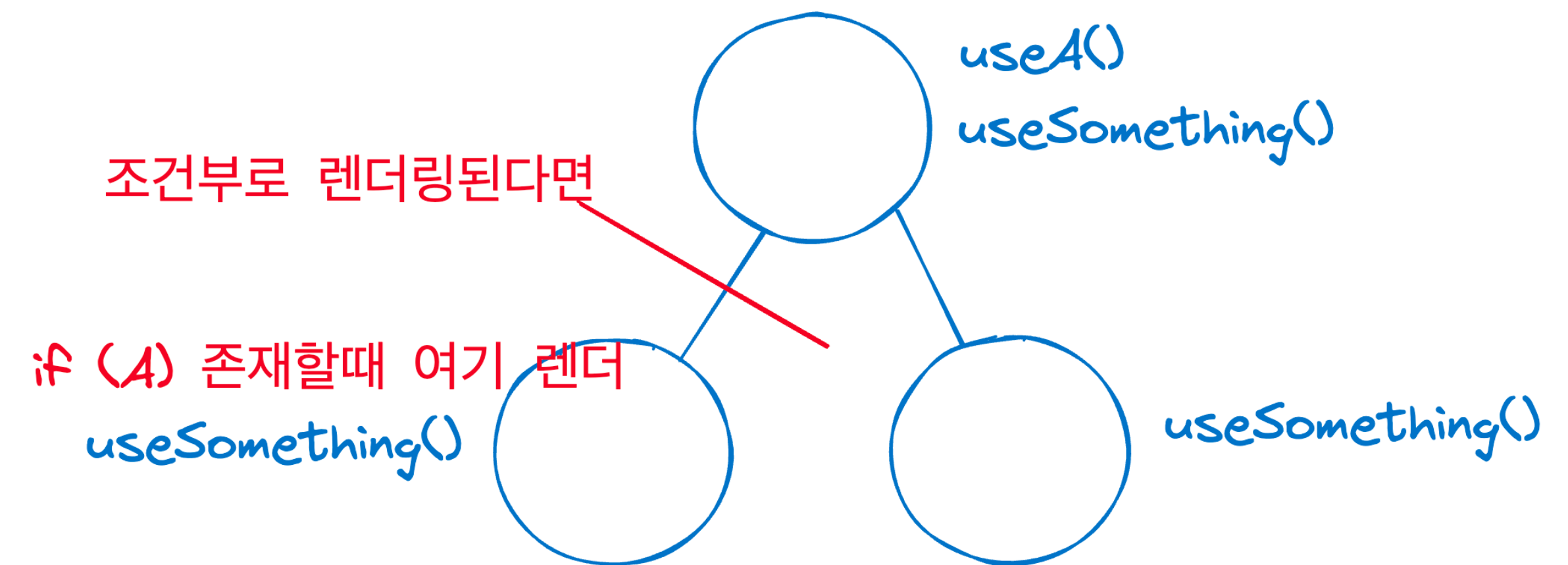


# staleTime이 0일 때

1개의 API Call useSomething



2개의 API Call useSomething



# 선부른 추상화하지말기

- 대출신청
- 지킴보증
- 실행 후 서류 확인
- 주기별 점검

컴포넌트가 얼추 다 비슷하다. 하지만 구현되는 내부 로직은 다를 수 있다.

선부르게 공통 컴포넌트로 통일하다가 나중에 각 페이지마다 필요한 로직을 넣으면서 컴포넌트가 비대해질 수 있다.

각 page 폴더 별로 비슷한 컴포넌트라도 복사해서 나눠놓고 나중에 공통부분을 묶어서 추상화하는게 변경사항에 더 빠르게 대응할 수 있을거라고 생각

# 코드 수정에 확신을 갖을 수 있는 방법이 있을까?

- 라이브로 배포가 되어있다
- 각 페이지에서 할 수 있는 동작을 모두 정의한다면

예) 1. A페이지에서는 고객번호로 대출을 조회할 수 있다

예) 2. A페이지에서는 조회된 목록을 전체 선택/전체 해제 할 수 있다

- 각 페이지에서 할 수 있는 동작을 문구로라도 표현해놓는다면.. 체크할 수 있지않을까?

테스트코드를 모두 커버하는 것은 어려우니, 테스트 케이스를 만들어 놓는 것만으로도 체크하는데 도움이 됨



# type assertion

<https://tkdodo.eu/blog/type-safe-react-query>

```
  
type Todo = { id: number; name: string; done: boolean }
```

```
//  직접 리턴 타입 명시
```

```
const fetchTodo = async (id: number): Promise<Todo> => {  
  const response = await axios.get(`/todos/${id}`)  
  return response.data  
}
```

```
//  이러면 데이터에 대한 타입 추론
```

```
const query = useQuery({  
  queryKey: ['todos', id],  
  queryFn: () => fetchTodo(id),  
})
```

```
query.data
```

```
//      ^?(property) data: Todo | undefined
```

# type assertion



```
const fetchTodo = async (id: number) => {  
  const response = await axios.get<Todo>(`/todos/${id}`)  
  return response.data  
}
```



```
function get<T = any>(url: string): Promise<{ data: T, status: number}>
```

# Runtime 타입 체크

```
import { z } from 'zod'

// 👁 schema 정의
const todoSchema = z.object({
  id: z.number(),
  name: z.string(),
  done: z.boolean(),
})

const fetchTodo = async (id: number) => {
  const response = await axios.get(`/todos/${id}`)
  // 🌟 parsing
  return todoSchema.parse(response.data)
}

const query = useQuery({
  queryKey: ['todos', id],
  queryFn: () => fetchTodo(id),
})
```



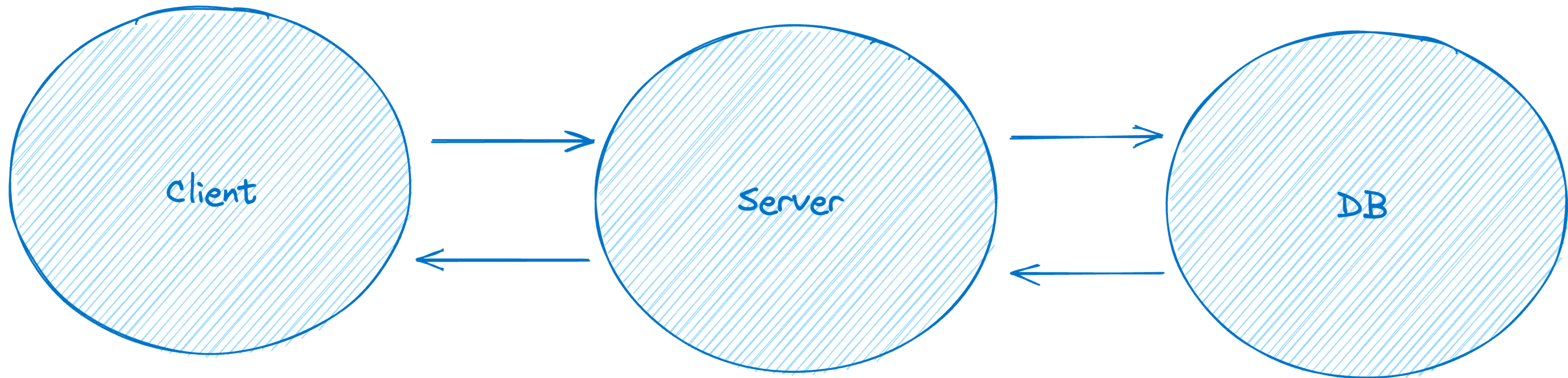
# Zod

- validation: 데이터가 valid 한가?
- parsing: 데이터가 valid한지 체크하고 파싱된 데이터를 반환

# Zod

- 스키마를 정의한 다음 들어오는 타입이 해당 스키마를 준수하는지 런타임에 체크한다
- 스키마를 통과한다면 그 시점에 스키마의 타입이 되는 것이고 통과하지 못한다면 에러를 던진다
- fetch 해서 가져오는 데이터에 대하여 **end to end type-safety**를 보장할 수 있다
- 백엔드로 부터 합의된 구조로 데이터를 받지 않으면 에러를 던진다. (Safe parse도 가능)
- 처음엔 유연한 스키마를 정의 => 점차 fit한 스키마로

- cannot read property of undefined



# SuspenseList

- UI의 순서 제어
- revealOrder (forwards, backwards, together)

```
<SuspenseList revealOrder="forwards">  
  <React.Suspense fallback={<p>loading...</p>}>  
    <Child resource={resource1} />  
  </React.Suspense>  
  <React.Suspense fallback={<p>loading...</p>}>  
    <Child resource={resource2} />  
  </React.Suspense>  
  <React.Suspense fallback={<p>loading...</p>}>  
    <Child resource={resource3} />  
  </React.Suspense>  
</SuspenseList>
```



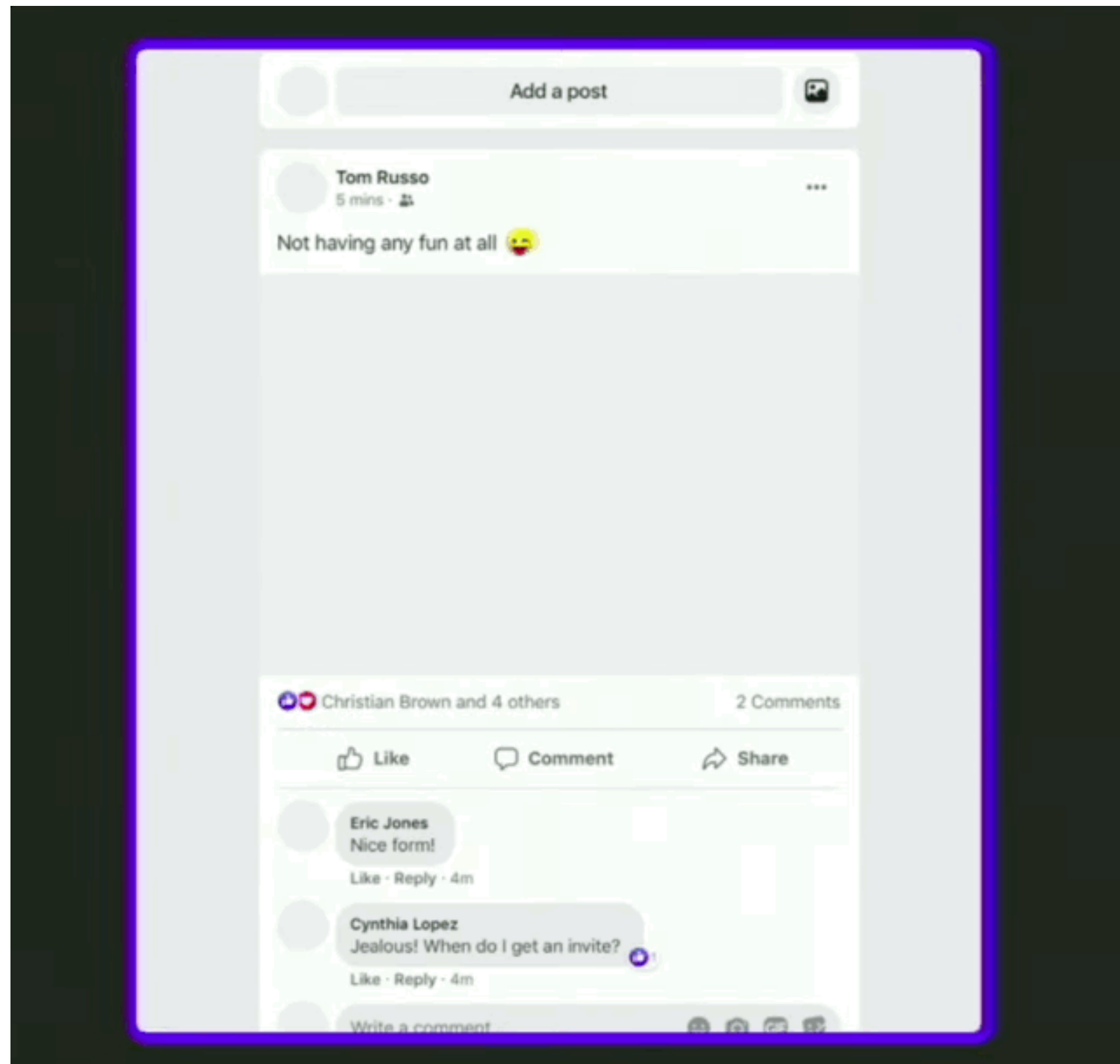
# Build 시간 줄이기

- yarn berry

# const enum

- enum (열거형): 상수의 집합. 제한된 값 집합을 나타내며 주로 정수 또는 문자열과 연결된 상수를 가짐

# Image Suspense



이미지를 기다릴 수 없을까?

# Image Suspense

`<Image/>` (user defined component 의 조건)

- src 를 로딩할 때 suspend 되게한다
- 에러가 발생하면 error를 던짐
- img 가 로드 되면 딱 뜨게 해야함

```
<img src={img.url} />
```

```
=> <ImageWithSuspense />
```

```

// A Resource is an object with a read method returning the payload
interface Resource<Payload> {
  read: () => Payload;
}

type status = "pending" | "success" | "error";

// this function let us get a new function using the asyncFn we pass
// this function also receives a payload and return us a resource with
// that payload assigned as type
function createResource<Payload>(
  asyncFn: () => Promise<Payload>
): Resource<Payload> {
  // we start defining our resource is on a pending status
  let status: status = "pending";
  // and we create a variable to store the result
  let result: any;
  // then we immediately start running the `asyncFn` function
  // and we store the resulting promise
  const promise = asyncFn().then(
    (r: Payload) => {
      // once it's fulfilled we change the status to success
      // and we save the returned value as result
      status = "success";
      result = r;
    },
    (e: Error) => {
      // once it's rejected we change the status to error
      // and we save the returned error as result
      status = "error";
      result = e;
    }
  );
  // lately we return an error object with the read method
  return {
    read(): Payload {
      // here we will check the status value
      switch (status) {
        case "pending":
          // if it's still pending we throw the promise
          // throwing a promise is how Suspense know our component is not ready
          throw promise;
        case "error":
          // if it's error we throw the error
          throw result;
        case "success":
          // if it's success we return the result
          return result;
      }
    },
  };
}

```

```

// First we need a type of cache to avoid creating resources for images
// we have already fetched in the past
const cache = new Map<string, any>();

// then we create our loadImage function, this function receives the source
// of the image and returns a resource
function loadImage(source: string): Resource<string> {
  // here we start getting the resource from the cache
  let resource = cache.get(source);
  // and if it's there we return it immediately
  if (resource) return resource;
  // but if it's not we create a new resource
  resource = createResource<string>(
    () =>
      // in our async function we create a promise
      new Promise((resolve, reject) => {
        // then create a new image element
        const img = new window.Image();
        // set the src to our source
        img.src = source;
        // and start listening for the load event to resolve the promise
        img.addEventListener("load", () => resolve(source));
        // and also the error event to reject the promise
        img.addEventListener("error", () =>
          reject(new Error(`Failed to load image ${source}`))
        );
      })
  );
  // before finishing we save the new resource in the cache
  cache.set(source, resource);
  // and return return it
  return resource;
}

```

```
function SuspenseImage(  
  props: React.ImgHTMLAttributes<HTMLImageElement>  
) : JSX.Element {  
  loadImage(props.src).read();  
  return <img {...props} />;  
}
```

```
interface User {  
  fullName: string;  
  avatar: string;  
}  
  
function User({ fullName, avatar }: User) {  
  return (  
    <div>  
      <SuspenseImage src={avatar} />  
      <h2>{fullName}</h2>  
    </div>  
  );  
}  
  
function UserList({ users }: { users: User[] }) {  
  return (  
    <React.Suspense fallback={<>Loading users...</>}>  
      {users.map((user) => <User key={user.id} {...user} />)}  
    </React.Suspense>  
  )  
}
```