Journey of State Management

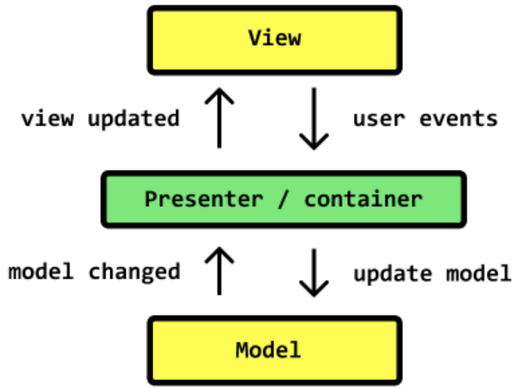
Casting프로젝트를 Angular에서 Next로 전환하면서 느낀 경험을 공유합니다.

- Wilson

FrontEnd 에서 하는 일

Model View Presenter

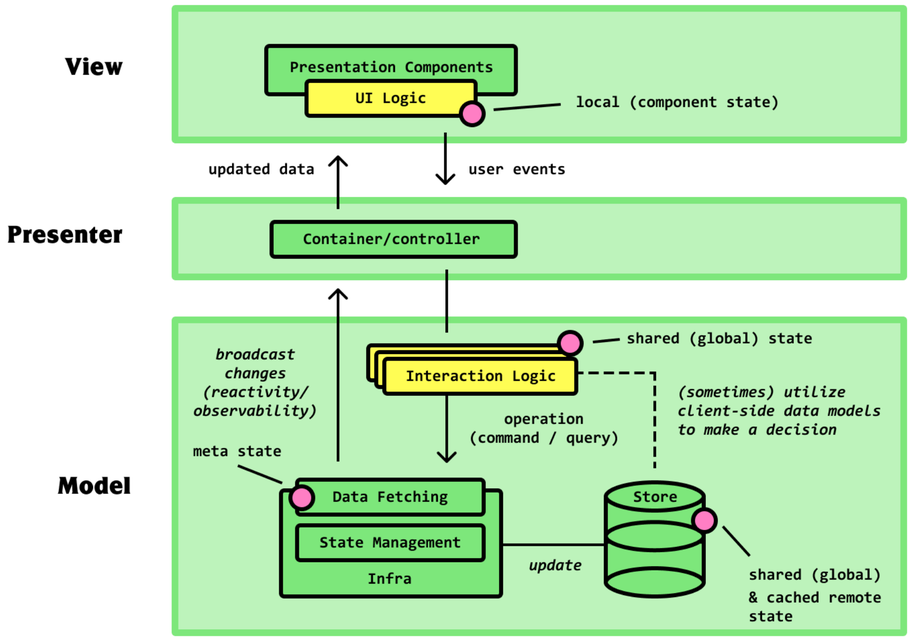
used for client-side apps

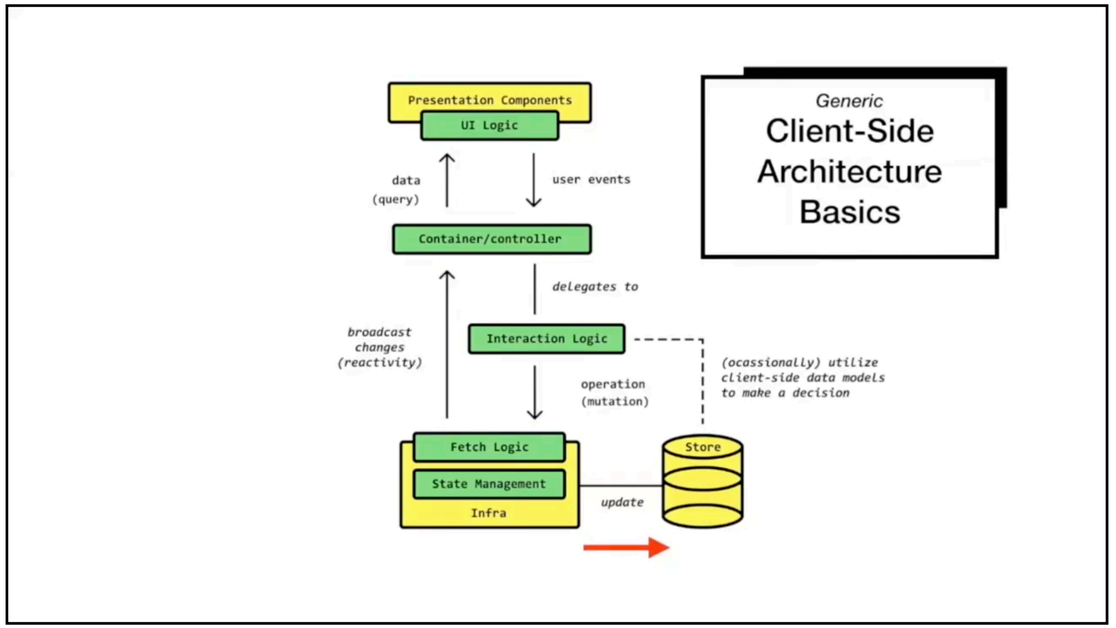


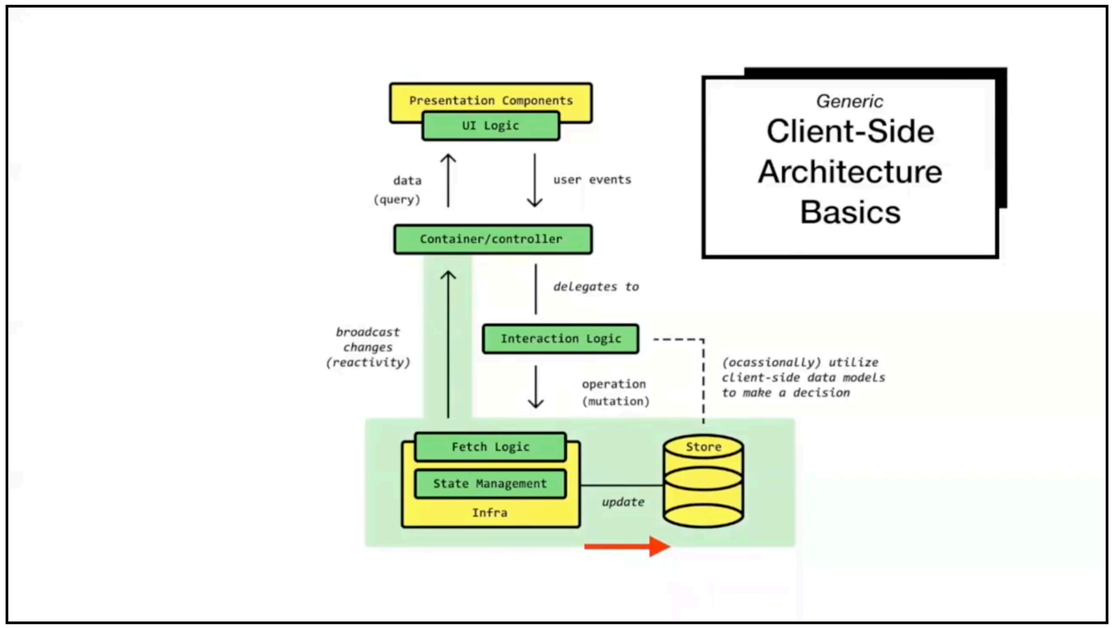
Modern

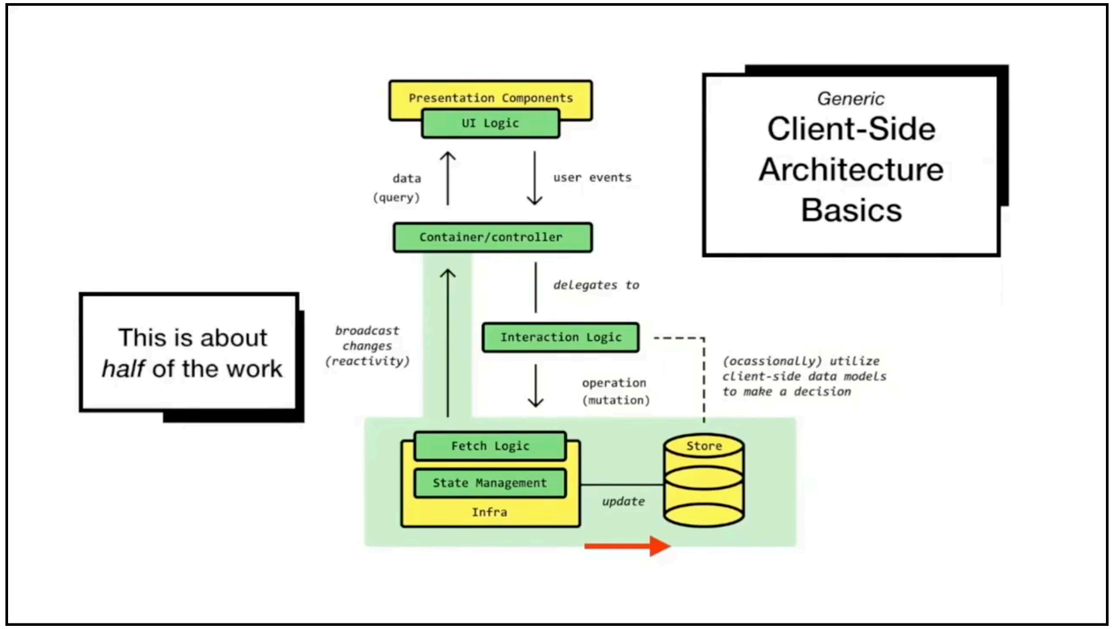
Model View Presenter

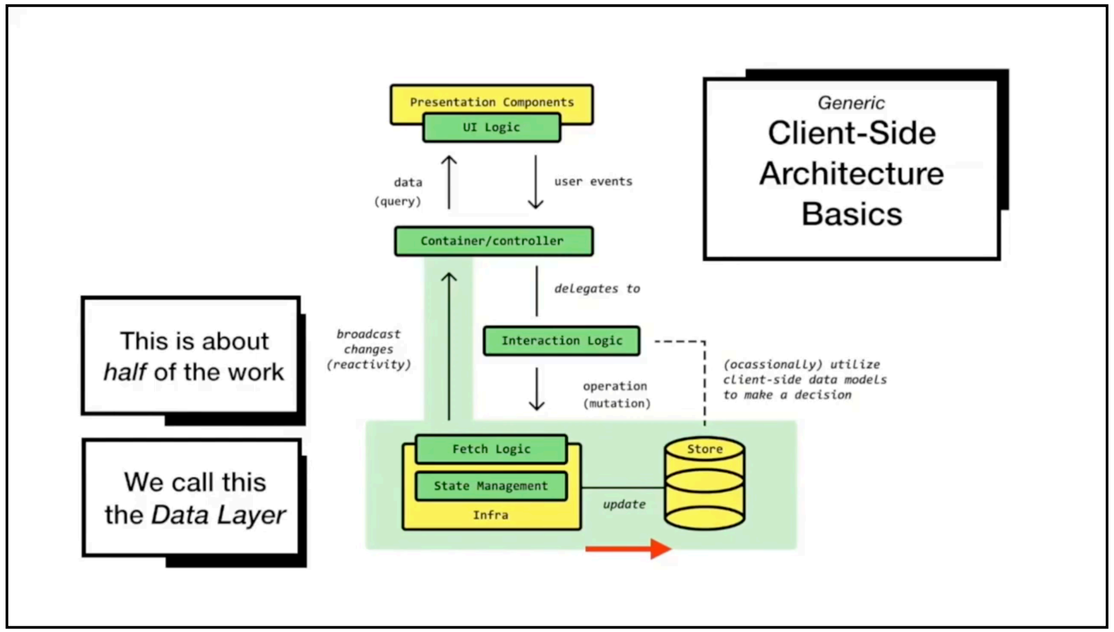
Client-side architecture basics




Generic
Client-Side Architecture Basics
Presentation Components
UI Logic
data (query)
user events
Container/controller
delegates to
broadcast changes (reactivity)
Interaction Logic
(ocassionally) utilize client-side data models to make a decision
operation (mutation)
Fetch Logic
State Management
Infra
update
Store


Generic
Client-Side Architecture Basics
Presentation Components
UI Logic
data (query)
user events
Container/controller
delegates to
broadcast changes (reactivity)
Interaction Logic
(ocassionally) utilize client-side data models to make a decision
operation (mutation)
Fetch Logic
State Management
Infra
update
Store


Generic
Client-Side Architecture Basics
Presentation Components
UI Logic
data (query)
user events
Container/controller
delegates to
Interaction Logic
broadcast changes (reactivity)
operation (mutation)
(ocassionally) utilize client-side data models to make a decision
Fetch Logic
State Management
Infra
update
Store
This is about half of the work


Generic
Client-Side Architecture Basics
Presentation Components
UI Logic
data (query)
user events
Container/controller
delegates to
Interaction Logic
broadcast changes (reactivity)
operation (mutation)
(ocassionally) utilize client-side data models to make a decision
Fetch Logic
State Management
Infra
update
Store
This is about half of the work
We call this the Data Layer

Data Layer

State Management

오늘 이야기는

Apollo Data Layer

익숙한 프로젝트 구성

React + Redux + Redux-Saga

신규 Casting 프로젝트 구성은

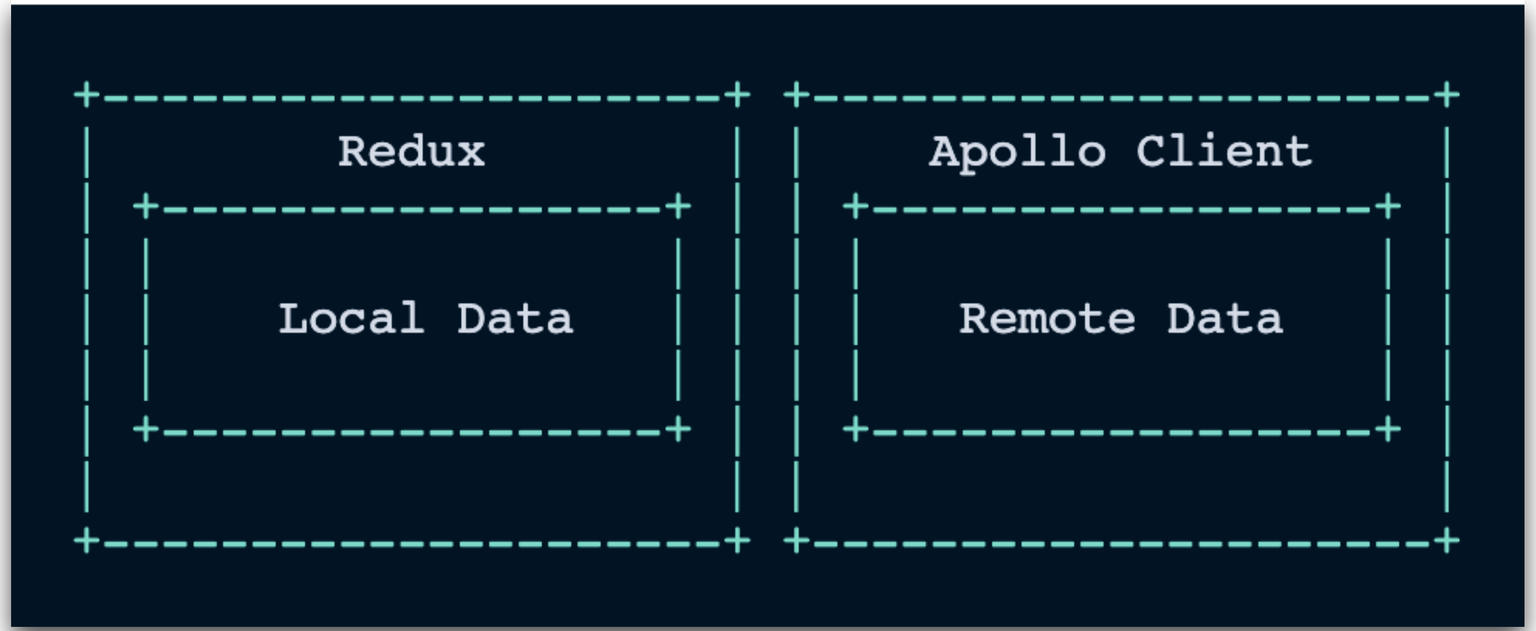
Next + Redux + Redux-Saga + Apollo

해당 스택으로 하는 상태관리

- Redux에서 정의하는 3가지 State
  - Raw(Remote) State
  - Application State
  - UI State

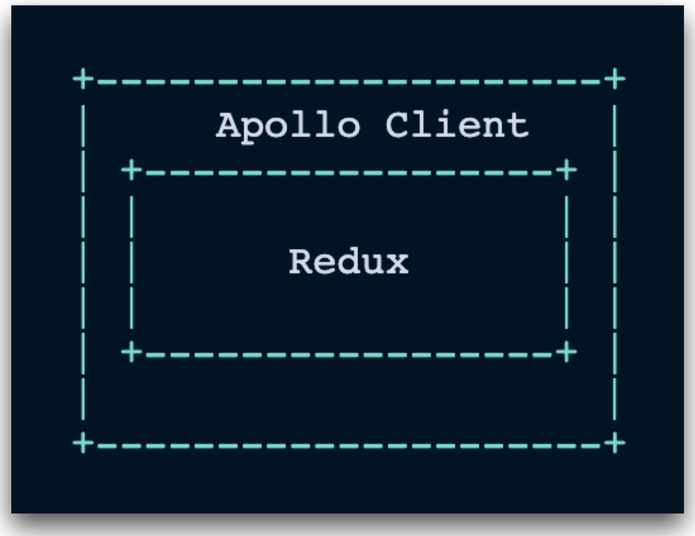
해당 스택으로 하는 상태관리

- Redux
  - Application State
  - UI State
- ***Apollo***
  - ***Remote State***


Redux
Local Data
Apollo Client
Remote Data

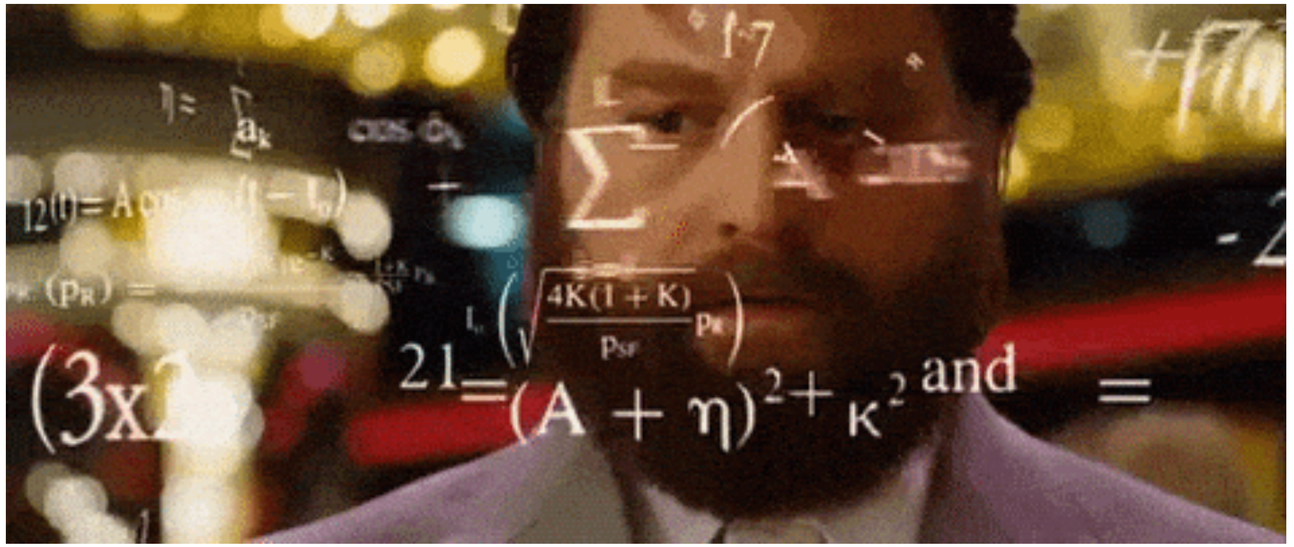
프로젝트를 조금 진행해보니...

Or



Problem to Solve

- ***Domain Logic*** - Redux / React Hook / Redux-Saga
- ***Networking / API*** - Apollo / Redux-Saga
- ***State management*** - Redux / Context Api / Redux-Saga


$(3x$
$21=(A+\eta)^2+\kappa^2$ and
$=$

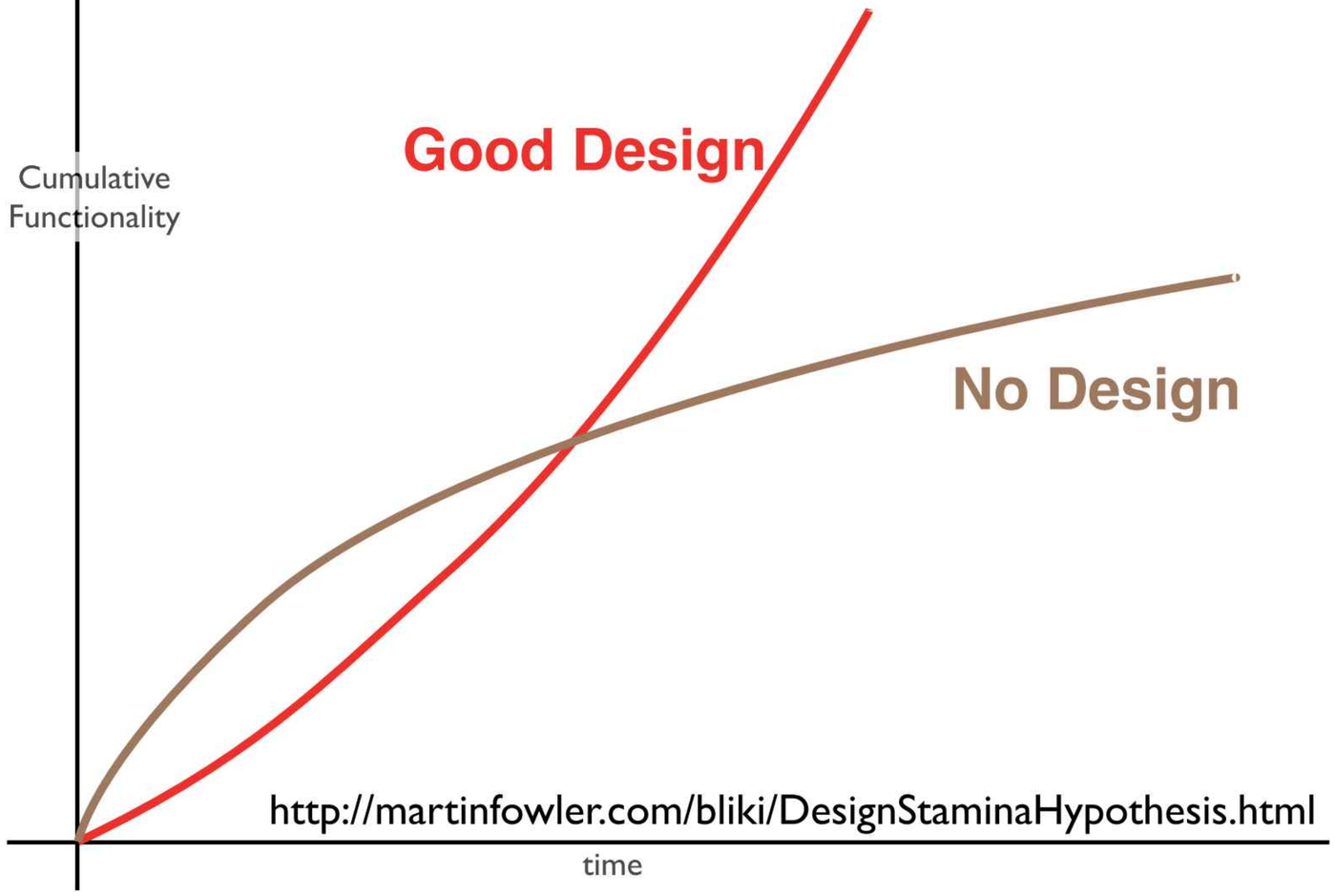
원칙에 혼동이 오기 시작

한때 학자들은 농업혁명이 인간성을 향한 위대한 도약이라고 생각했다. 이들은 두뇌의 힘을 연료로 하는 진보의 이야기를 지어냈다. …

이 이야기는 환상이다.

- 사피엔스

Design Stamina Hypothesis



진보된 아키텍쳐라고 생각했지만
모호한 규칙의 집합

임의의 규칙을 부과하는 대신
구조화된 아키텍쳐를 생각해보자

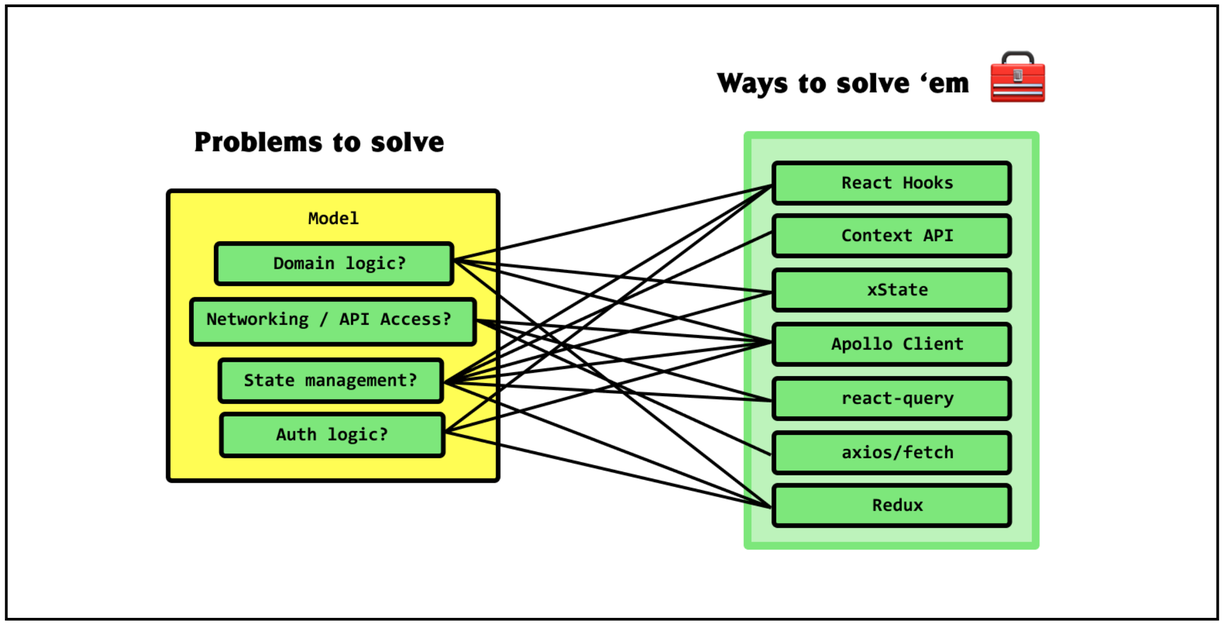
새로운 대안에 대해서 탐색

원하는 아키텍쳐

- 프로젝트를 진행할 수록 개발에 가속도가 생겨야 한다.
- 보일러 플레이트가 적어야한다. State 설정이 간단해야 한다.
- Tool 에서 제공하는 명확한 규칙이 있어야 한다.
- Remote, Local, Shared 상태를 관리 할 수 있어야 한다.
- 마이너한 Tool은 선택하지 않는다.

2020 React State management

- React Hook
- Redux
- Context Api
- Apollo Client
- xState
- React-Query

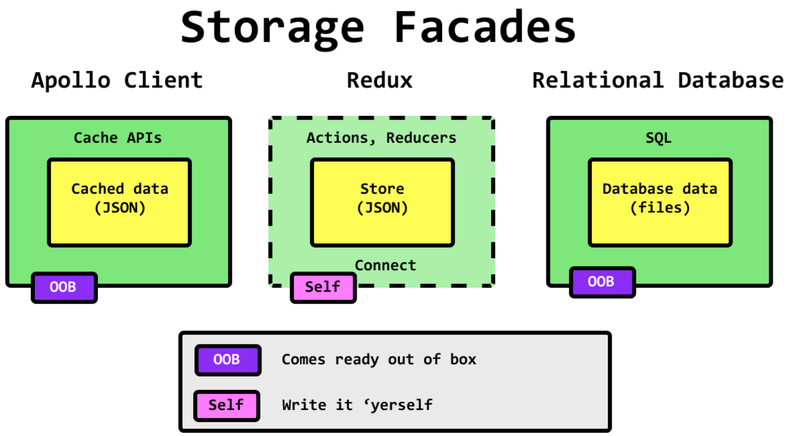

Ways to solve ‘em
Problems to solve
Model
Domain logic?
Networking / API Access?
State management?
Auth logic?
React Hooks
Context API
xState
Apollo Client
react-query
axios/fetch
Redux

Redux + Redux-Saga

너무 많은 보일러플레이트로 **생산성에 의문**

Apollo Client가 마음에 드는점

- 보일러 플레이트가 적다.
- Apollo가 제공하는 Best Practice



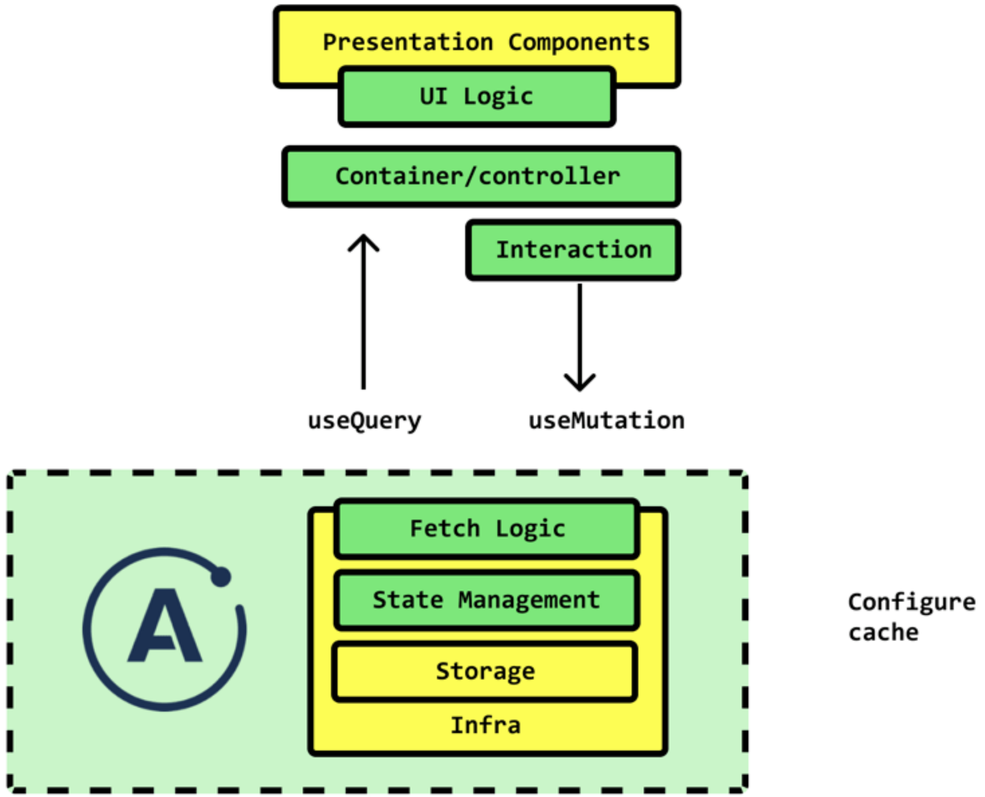
wilson-jung commented 15 days ago • edited

- redux-saga 에서 graphql api를 호출하는 방법
- redux, redux-saga -> apollo-client 전환이 가능한가?

Redux + Redux-Saga = Apollo Client 가능할까?

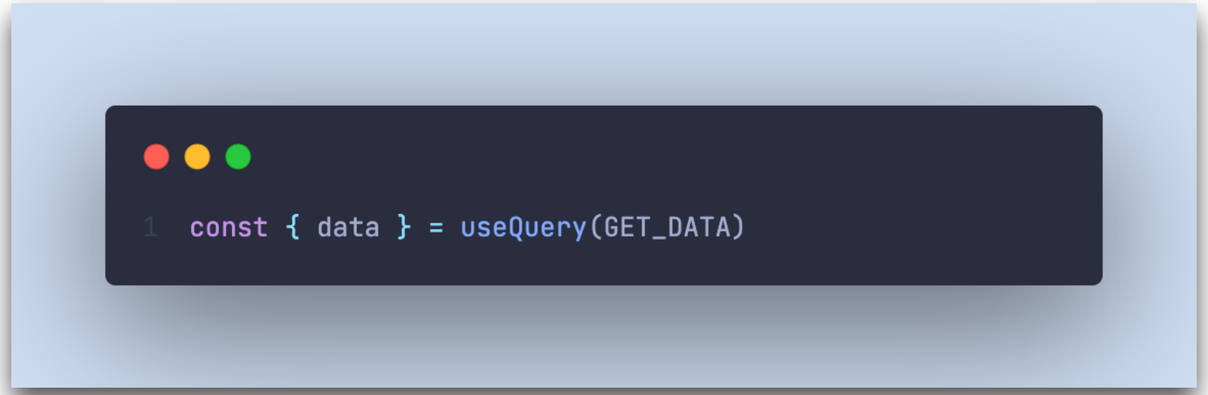
Apollo Client Data Layer 관리

- Redux: State management
- Redux-Saga: SideEffect / Networking



Remote State

State retrieved from backend services



Remote State

useQuery for ***Fetch***

```
const GET_TODO = gql`
  query GetTodos {
    todos {
      id
      text
      completed
    }
  }
`

const MyApp = () => {
  // data 값이 변경되면 자동으로 컴포넌트 re-render
  const { data } = useQuery(GET_TODO)
  return (
    <div>
      ...
      <ul>
        {data.todos.map(todo => <li>{todo}</li>)}
      </ul>
      ...
    </div>
  )
}
```

Remote State

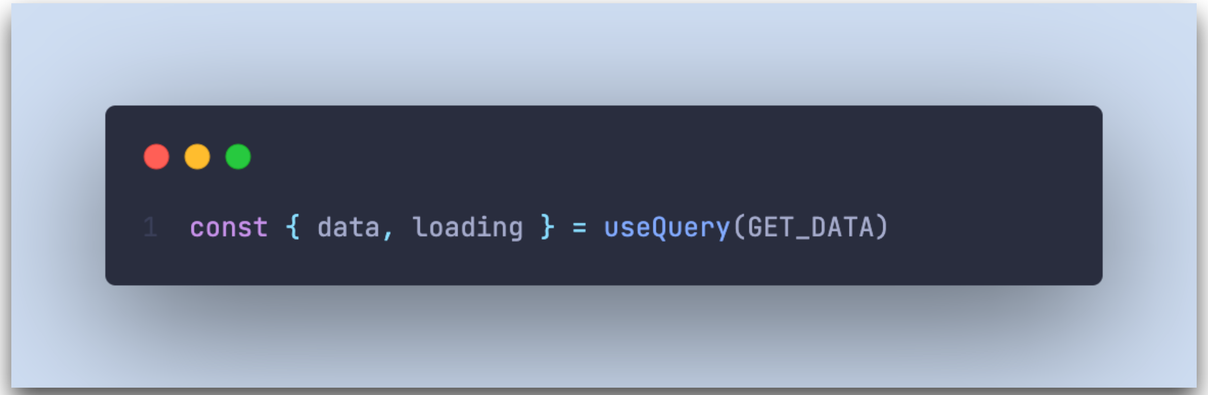
useMutation for ***Update***

```
const ADD_TODO = gql`
  mutation AddTodo($text: String!) {
    addTodo (text: $text) {
      todo {
        id
        text
        completed
      }
    }
  }
`

const MyApp = () => {
  // data 값이 변경되면 자동으로 컴포넌트 re-render
  const { data } = useQuery(GET_TODO)
  const [mutate] = useMutation(ADD_TODO)
  return (
    <div>
      <button onClick={() => mutate('new Todo')} />
      ...
      <ul>
        {data.todos.map(todo => <li>{todo}</li>)}
      </ul>
      ...
    </div>
  )
}
```

Meta State

State *about* state



Meta State

State *about* state

```
const MyApp = () => {
  // loading 값이 변경되면 자동으로 컴포넌트 re-render
  // loading: boolean
  const { data, loading: fetchTodoLoading } = useQuery(GET_TODO)
  const [mutate, { loading: addTodoLoading }] = useMutation(ADD_TODO)
  return (
    <div>
      <button onClick={() => mutate('new Todo')} loading={addTodoLoading} />
      ...
      {
        fetchTodoLoading
          ? <p>loading...</p>
          : <ul>
              {data.todos.map(todo => <li>{todo}</li>)}
            </ul>
      }
      ...
    </div>
  )
}
```

Local State

State belonging to a single component

Shared Local State

State used by multiple components

Shared Local State

- client-side only state
- combination remote and client-side

Client-side only state

```
const todosVar = makeVar<Todos>(todoInitialValue) // initialize

const cache: InMemoryCache = new InMemoryCache({
  typePolicies: {
    Query: {
      fields: {
        darkMode: {
          read () {
            return darkModeVar();
          }
        }
      }
    }
  }
})

const GET_DARK_MODE = gql`
  query GetAllTodos {
    darkMode @client
  }
`

const { data, loading } = useQuery(GET_DARK_MODE)
```

Combination remote and client-side

```
const cache: InMemoryCache = new InMemoryCache({
  typePolicies: {
    Query: {
      fields: {
        darkMode: {
          read () {
            return darkModeVar();
          }
        }
      }
    }
  }
})

const GET_TODO_AND_DARK_MODE = gql`
  query GetTodosAndDarkMode {
    todos {
      id
      text
      completed
    }
    darkMode @client
  }
`

const { data, loading } = useQuery(GET_TODO_AND_DARK_MODE)
```

End