Code Generator와 TypeScript로 멋지게 API 응답 처리하기

30 April 2020 @ TypeScript Korea Online Meetup

@tonyfromundefined

발표자 소개 👨‍💻

원지혁

당근마켓 Software Engineer

- Full-Stack Web Engineer
- 당근마켓 Software Engineer
- *GraphQL Korea* Organizer
- *AWSKRUG GraphQL Group* Organizer

> *Vue.js*로 *FE* 개발을 처음 시작해,
> 현재는 *TypeScript*와 *React.js*를 주로 사용합니다.
> 높은 퀄리티의 앱을 빠르게 개발할 수 있는
> 소프트웨어 개발 방법에 대해 끊임없이 고민하고 있습니다.

목차

우리가 API를 처리하는 방법

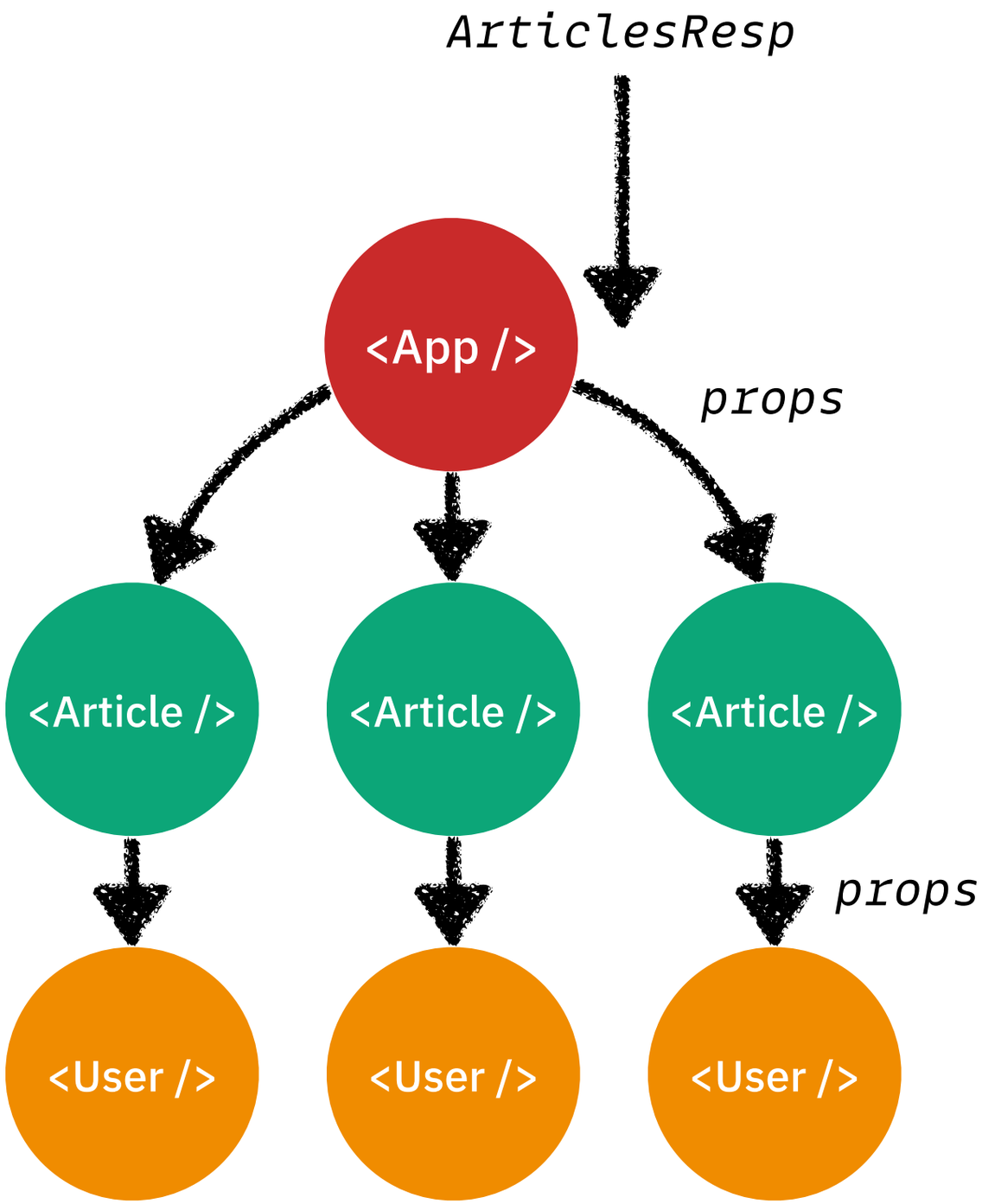
Code Generator와 TypeScript로 멋지게 API 응답 처리하기

Interface, Type, Class로 한땀한땀

```ts
interface ArticlesResp {
  data: {
    articles: Array<{
      id: string
      user: {
        id: string
        displayName: string
        username: string
      }
      title: string
      content: string
      commentNum: string
    }>
  }
}

export async function getArticles() {
  const { data } = await client.get<ArticlesResp>('/articles')
  return data
}
```

```ts
interface UsersResp {
  data: {
    users: Array<{
      id: string
      displayName: string
      username: string
    }>
  }
}

export async function getUsers() {
  const { data } = await client.get<UsersResp>('/users')
  return data
}
```

컴포넌트로 데이터 전달



- ArticlesResp 타입에 따라서 각 컴포넌트에 한땀 한땀 Props를 파야함
- 각 컴포넌트는 ArticlesResp의 Subset을 가져가는 구조
- 전역 상태 관리를 사용하더라도, <Article />, <User />의 Props는 ArticlesResp에 의존
- 컴포넌트로 분리해서 개발한다지만... 급하게 개발하다보면, 결국 의존성이 `화면` 단위로 묶이게 됨

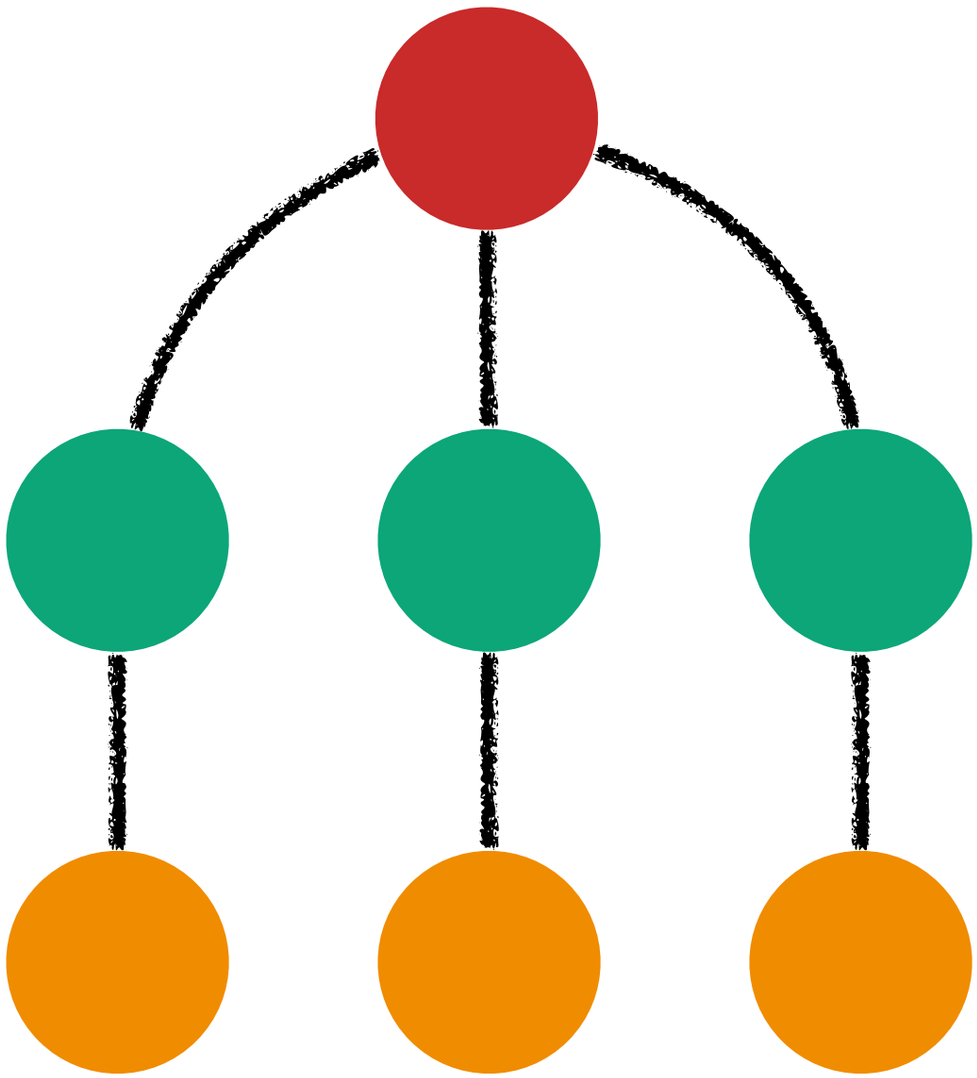
어떻게 해결할까?

- 수 많은 해결방법이 있겠지만...
- 오늘 세션에서는 GraphQL과 Relay를 통해
  데이터 처리 관련 로직과 View 관련 로직을 분리하는 해결방법 소개

클라이언트 GraphQL 스키마

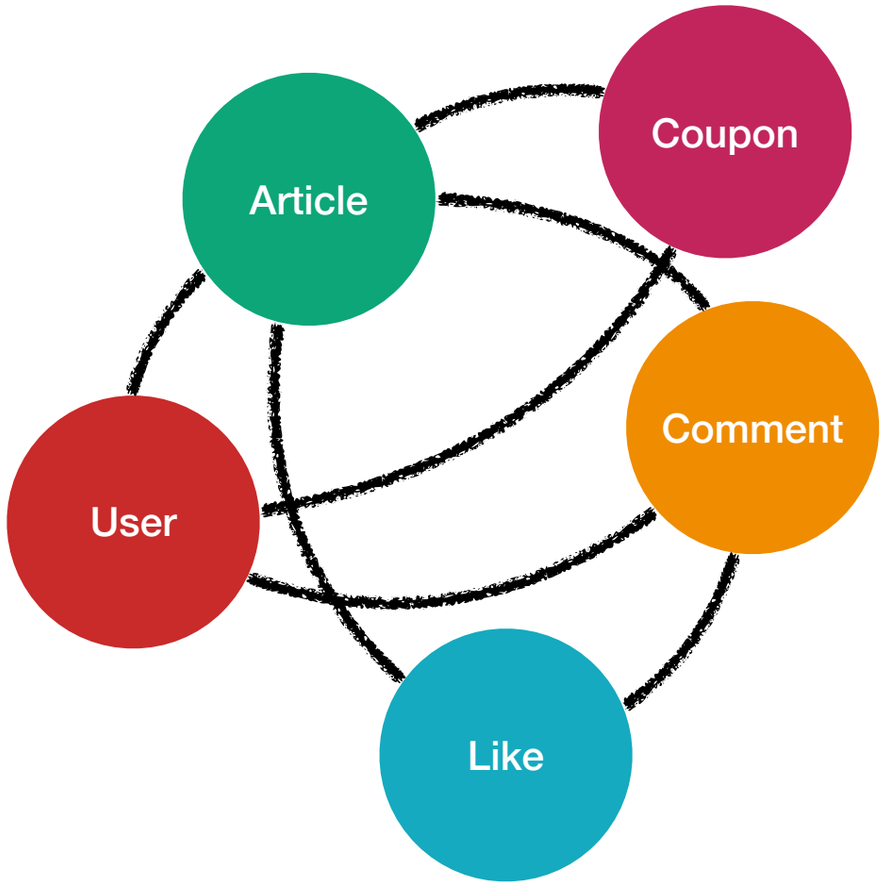
Code Generator와 TypeScript로 멋지게 API 응답 처리하기

우리가 만드는 일반적인 앱



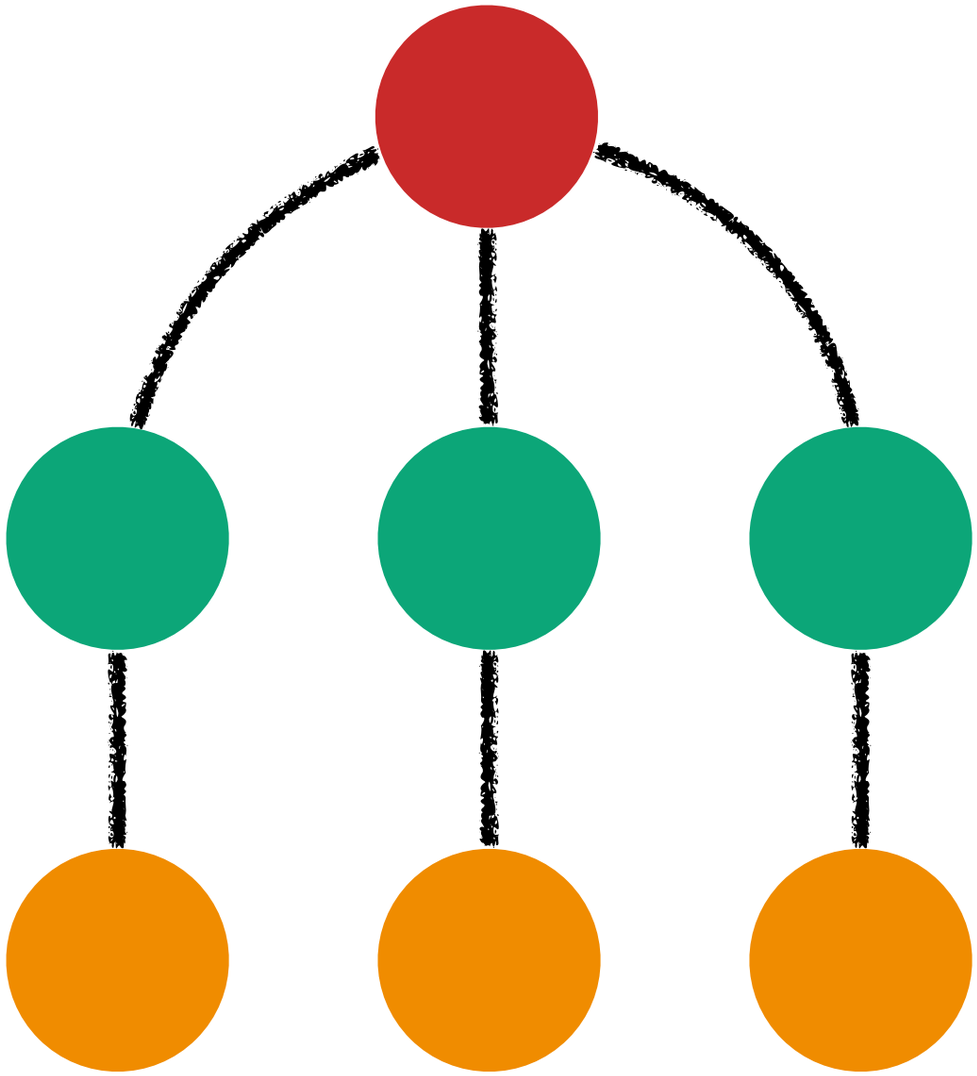
DOM

+

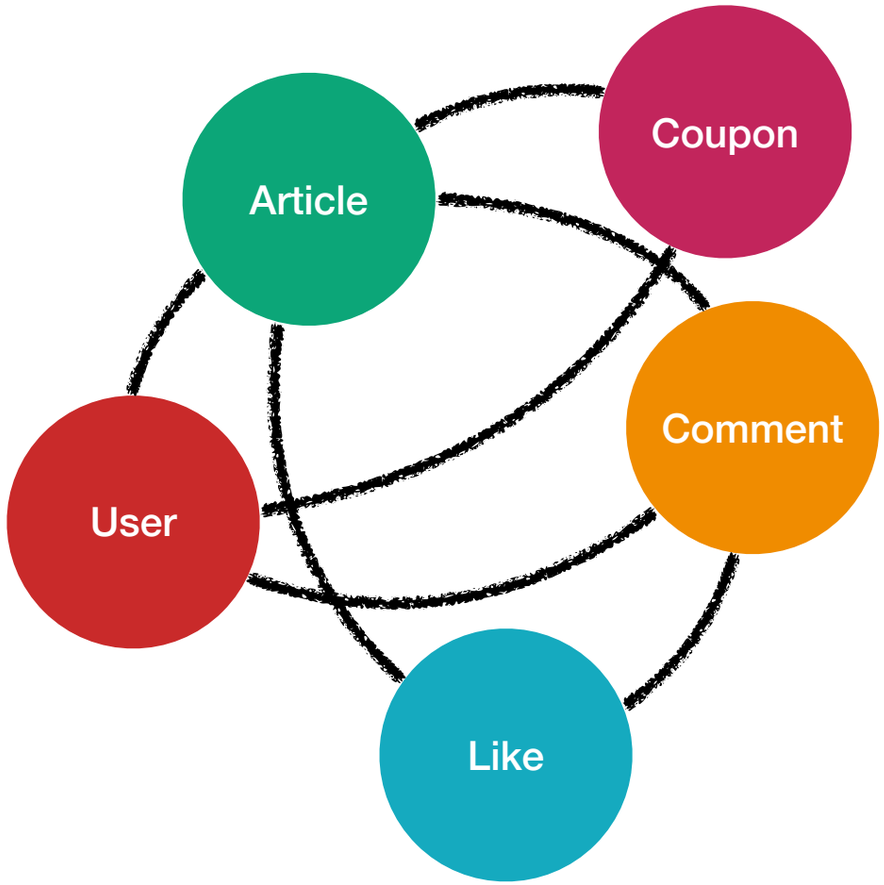


복잡한 데이터 구조

컴포넌트와 스키마를 분리하자

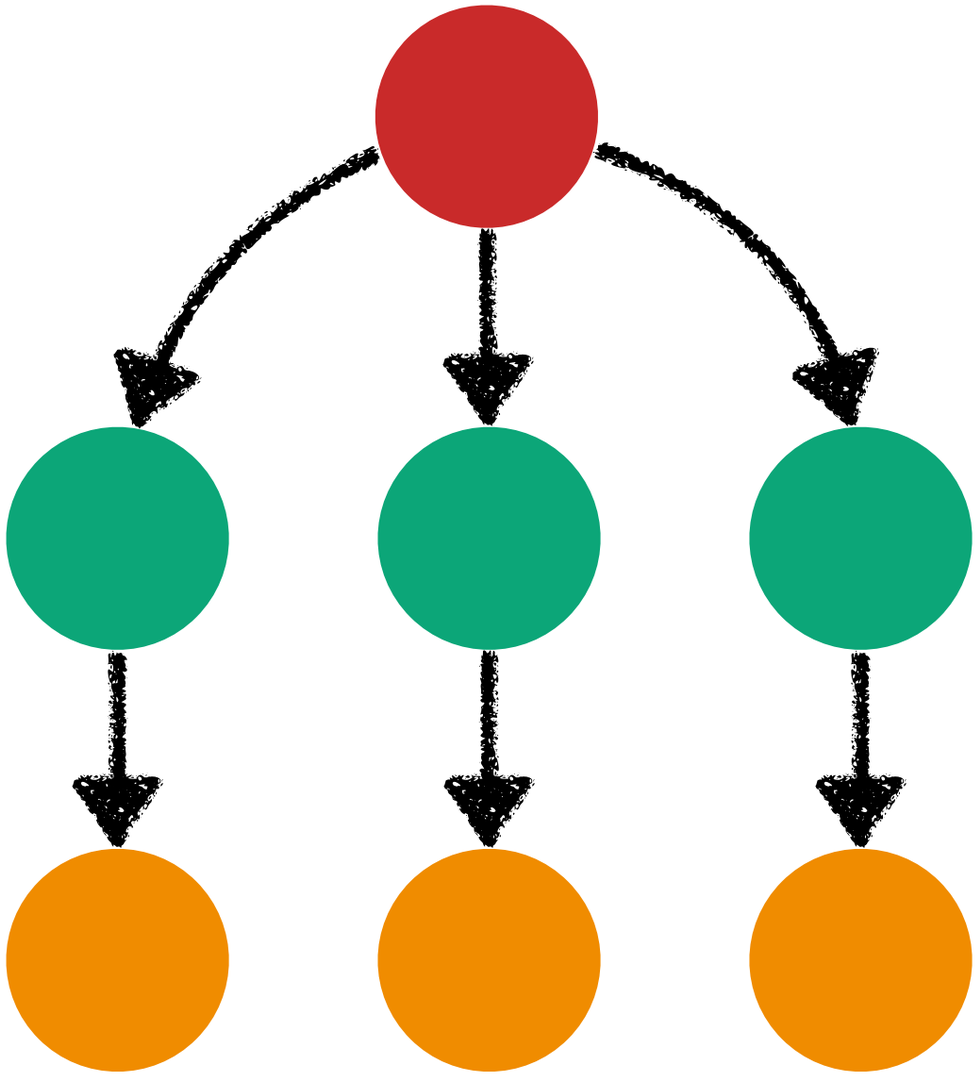


DOM

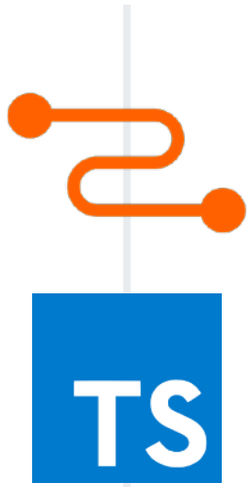


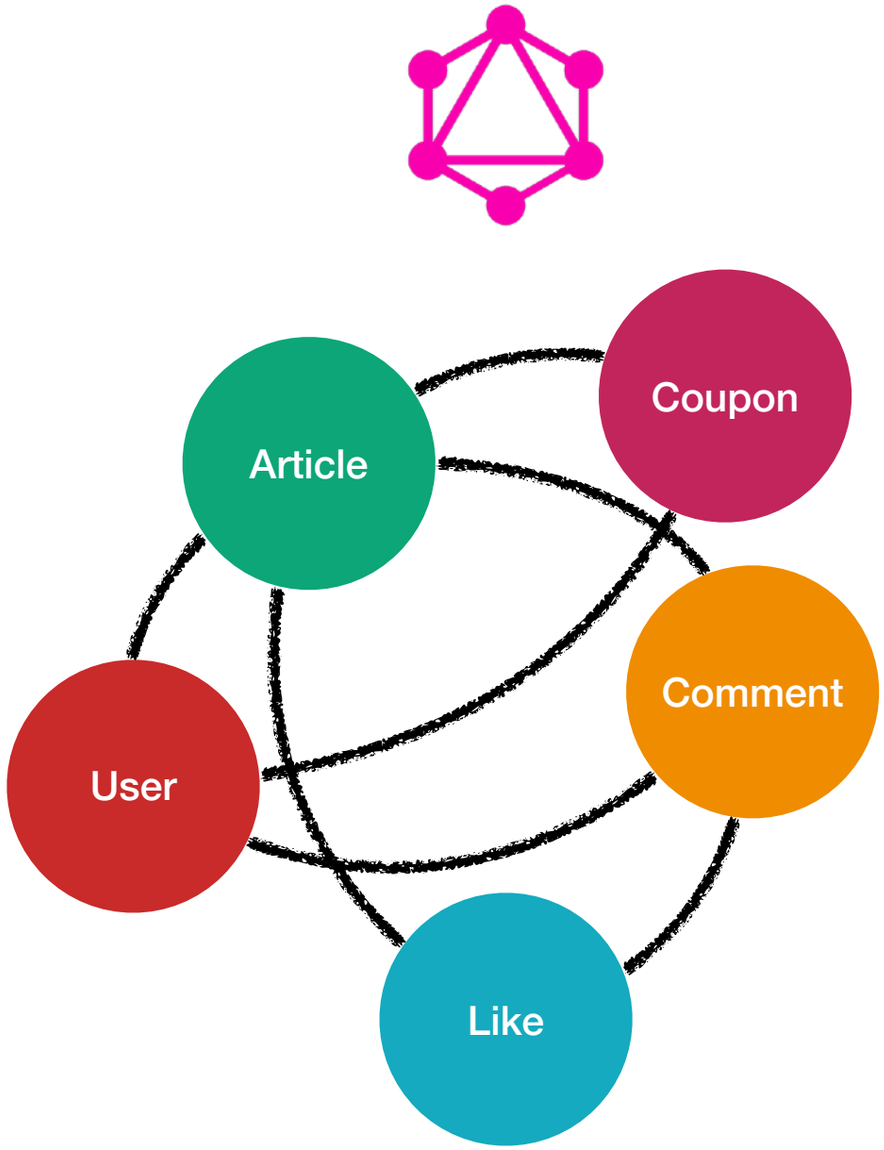
복잡한 데이터 구조

GraphQL로 스키마를 관리



Component Tree





Graph

GraphQL SDL

Schema Definition Language

```
type User {
  articles: [Article!]!
}

type Article {
  author: User!
  likes: [Like!]!
  comments: [Comment!]!
  coupons: [Coupon!]!
}
```

```
type Comment {
  user: User!
  article: Article!
  comments: [Comment!]!
  like: [Like!]!
}

type Like {
  user: User!
}

type Coupon {}
```

Code Generator로 타이핑하기

Code Generator와 TypeScript로 멋지게 API 응답 처리하기

작업 순서

Demo

What is Relay?

Code Generator와 TypeScript로 멋지게 API 응답 처리하기

Declarative Data Fetching

<Article />

“나는 Article 타입의 title, content 필드가 필요해”

<User />

“나는 User 타입의 displayName, username 필드가 필요해”

Declarative Data Fetching

<Article />

“나는 *Article* 타입의 *title, content* 필드가 필요해”

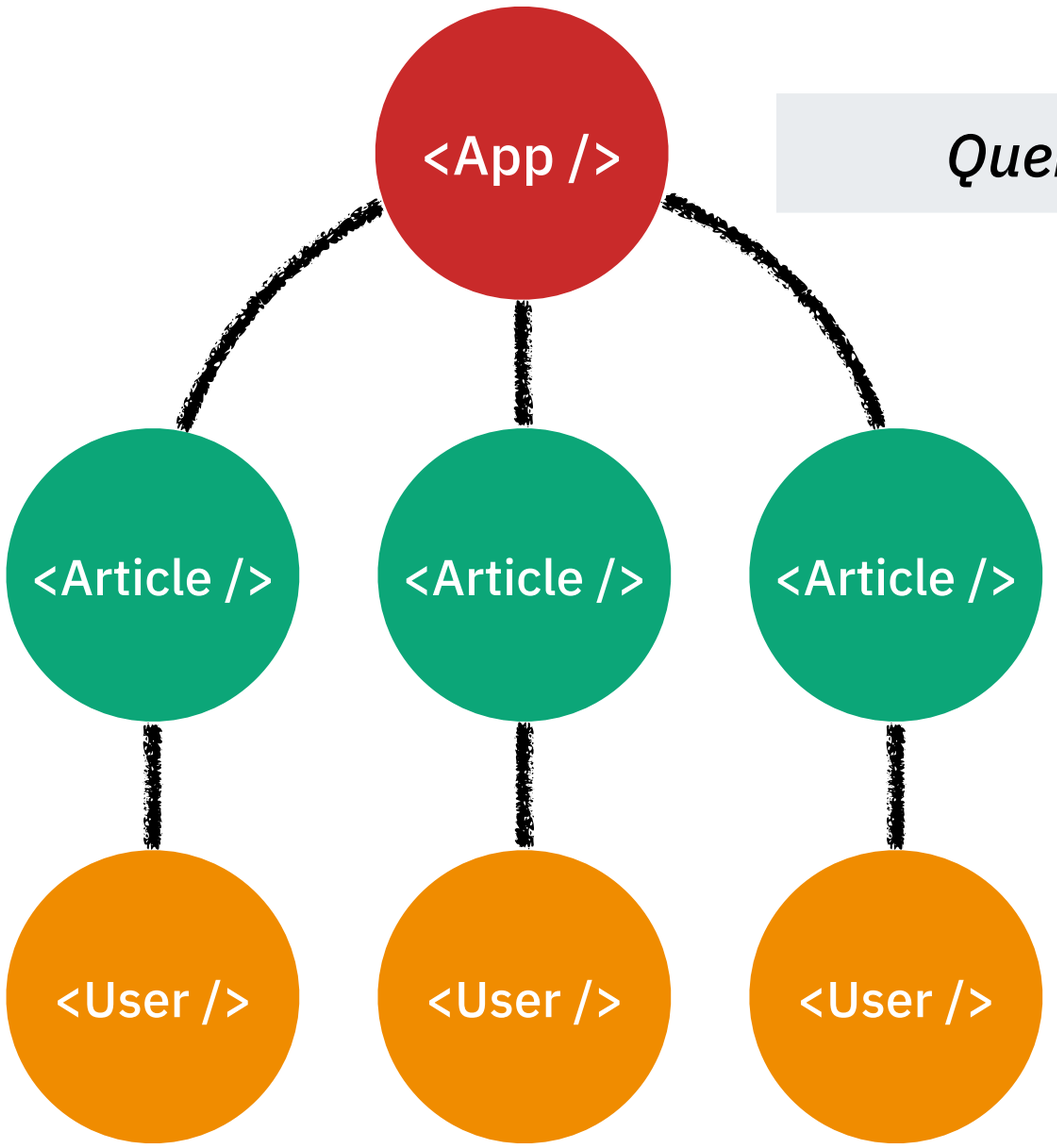
<User />

“나는 *User* 타입의 *displayName, username* 필드가 필요해”

<App />

“나는 내 아래 컴포넌트들에게 필요한 정보를 모아서 *GraphQL* 쪽에 물어볼꺼야”

Relay가 해주는 일



Query

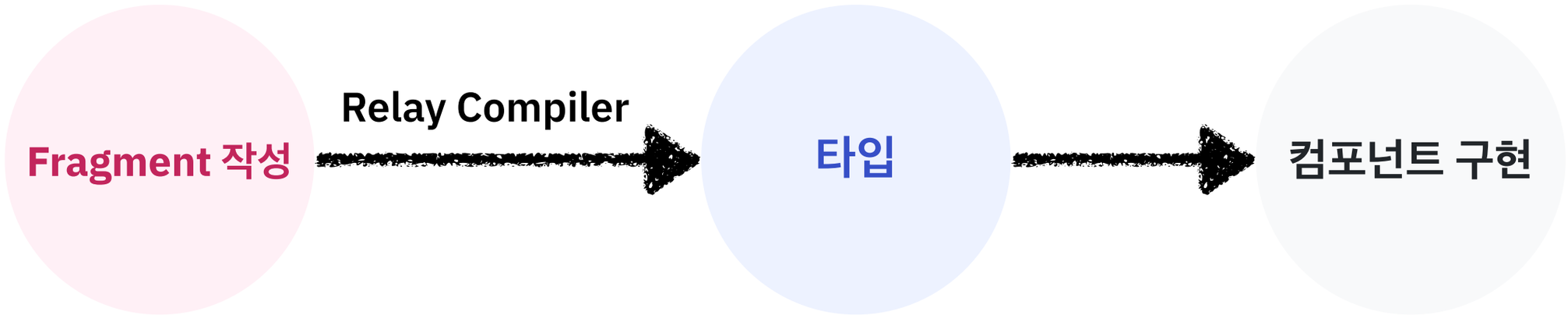
```
query AppQuery {
  articles {
    id
    ...Article_article
  }
}

fragment Article_article on Article {
  title
  content
  user {
    id
    ...User_user
  }
}

fragment User_user on User {
  username
  displayName
}
```

Relay Compiler

Code Generator와 TypeScript로 멋지게 API 응답 처리하기

Code Generator 와 *TypeScript* 로 멋지게 *API* 응답 처리하기

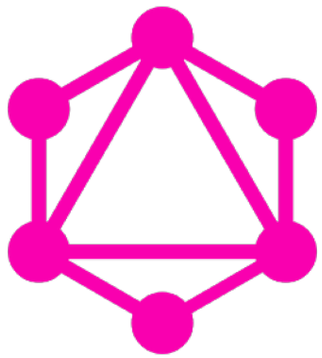
또 하나의 Codegen



Demo

Code Generator 와 *TypeScript* 로 멋지게 *API* 응답 처리하기

Welcome to GraphQL Korea!



GraphQL Korea

토론할 주제가 차고 넘치는 *GraphQL* 생태계로 여러분들을 초대합니다 😆

https://www.facebook.com/groups/graphql-kr/

https://bit.ly/graphql-korea-slack

One more thing...

저희랑 같이 개발하실래요?

당근마켓에서 *TypeScript* 개발자 *(Node.js, Front-end)*를 모시고 있어요*!*

팀 소개 및 지원하기 → team.daangn.com

질문과 답변