DEVIEW 2021

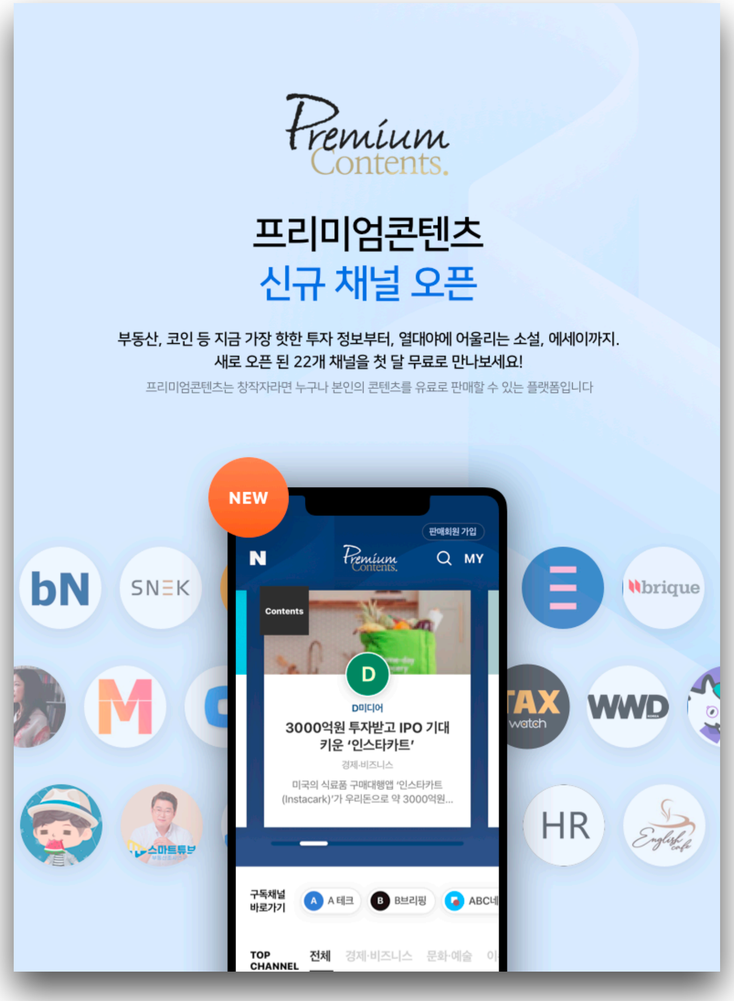
Next.js, Apollo와 함께 리액트 개발의 Next Level로 가자!

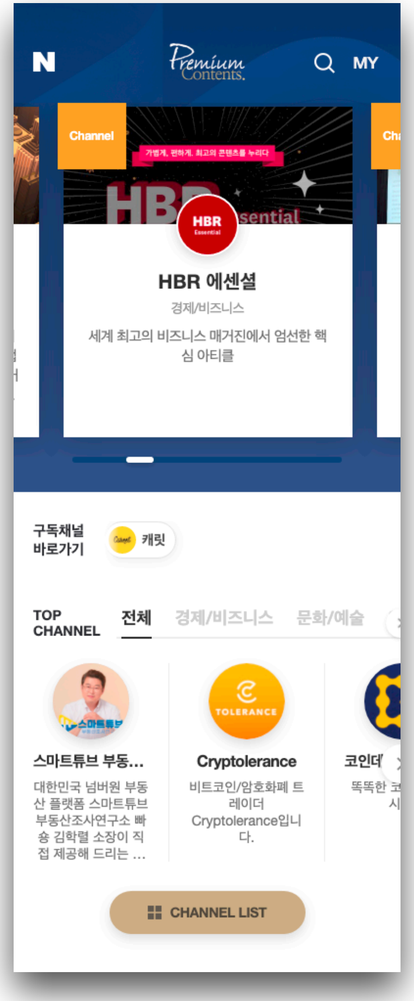
이건 NAVER 프리미엄콘텐츠플랫폼

CONTENTS

1. 프리미엄 콘텐츠

1.1 프리미엄 콘텐츠





프리미엄 콘텐츠 서비스

1.2 프리미엄 콘텐츠 스튜디오

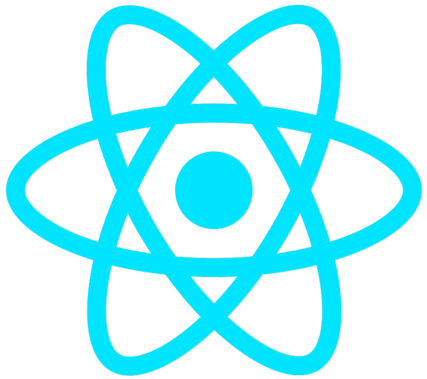
DEVIEW 2021

프리미엄 콘텐츠 스튜디오 PC

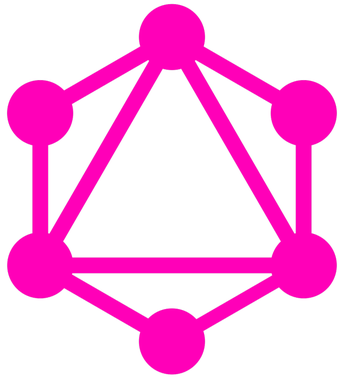
프리미엄 콘텐츠 스튜디오 Mobile

1.2 프리미엄 콘텐츠 스튜디오

요구사항



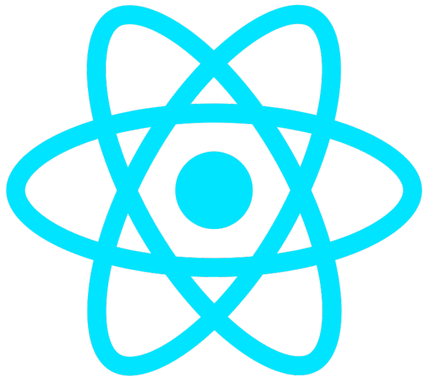
React



GraphQL

1.2 프리미엄 콘텐츠 스튜디오

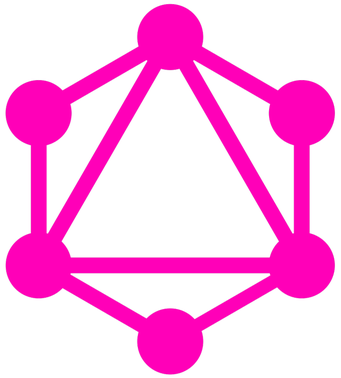
요구사항



React

🤝

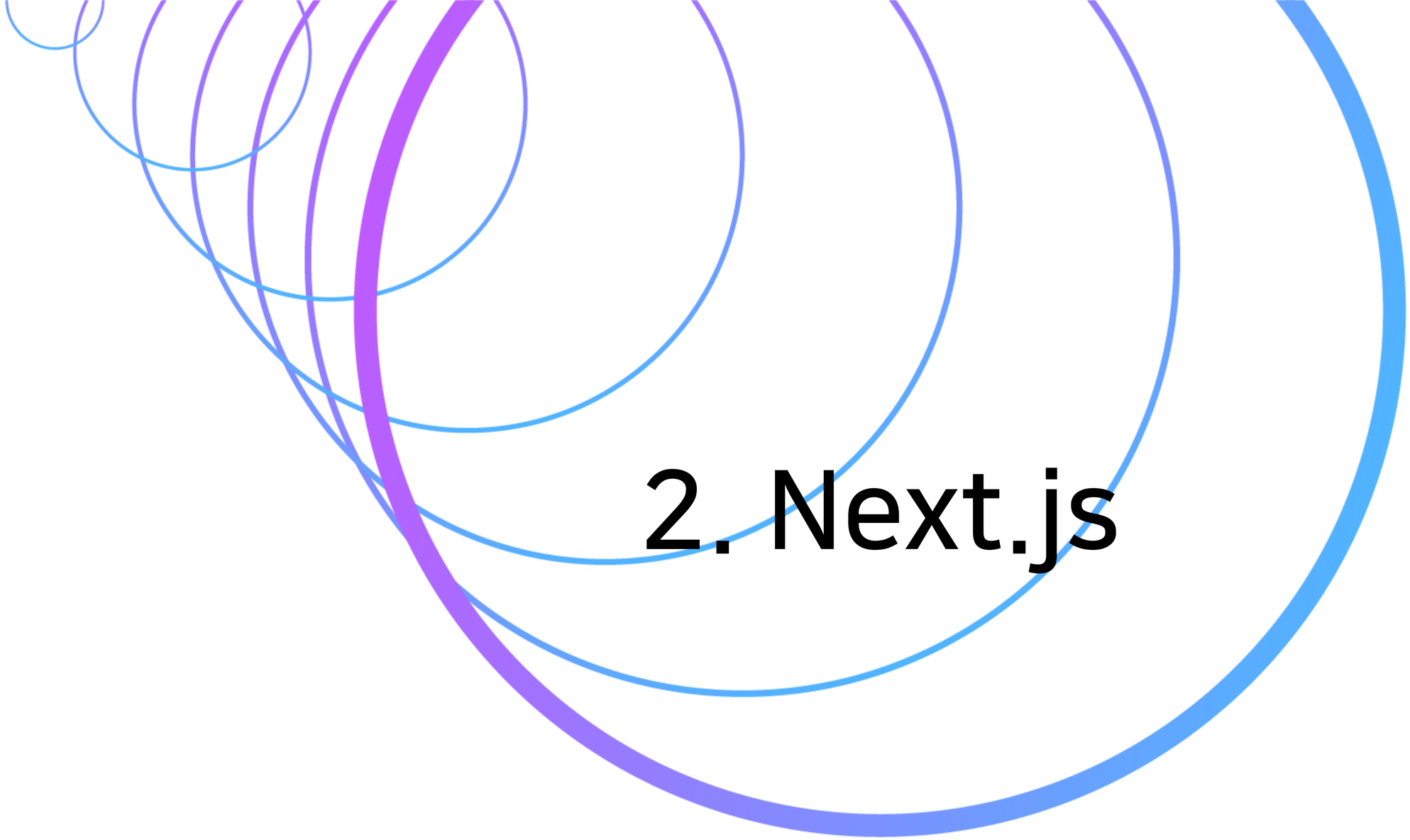
Next.js



GraphQL

🤝

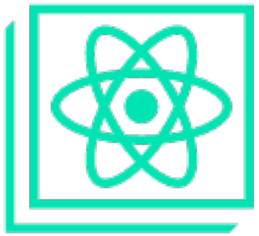
Apollo Client



2. Next.js

2.1 Create React App

DEVIEW 2021



👉

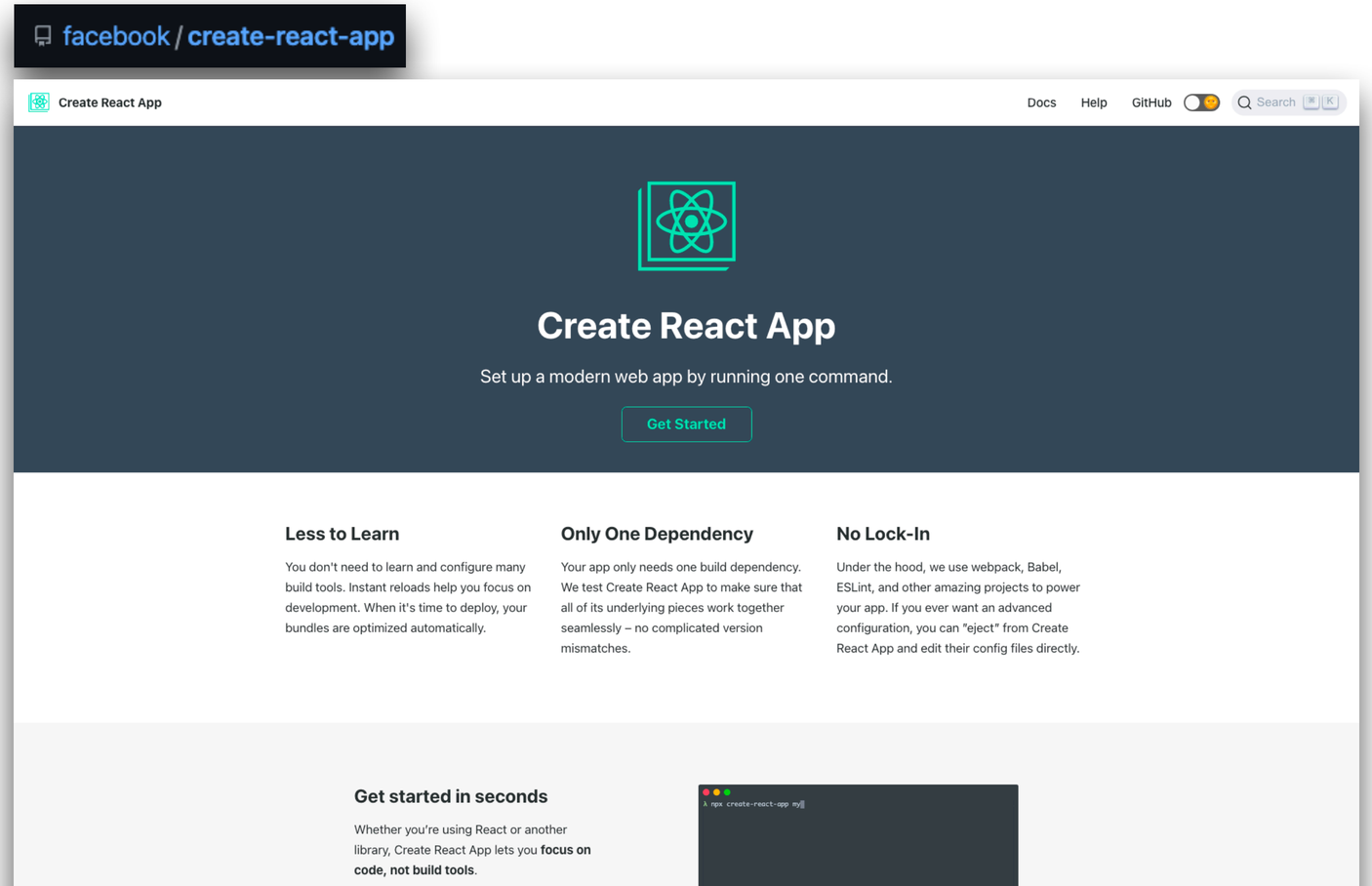
NEXT.JS

2.1 Create React App

DEVIEW 2021

소개

- React 팀에서 생성, 유지, 관리하는 React Framework



Create React App
공식 홈페이지

2.1 Create React App

DEVIEW 2021

npx create-react-app my-app

```
λ npx create-react-app my-app
npx: installed 114 in 4.308s

Creating a new React app in ~/my-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

yarn add v1.2.1
info No lockfile found.
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
```

npx create-react-app my app

2.1 Create React App

DEVIEW 2021

😀 단 하나의 의존성, react-scripts

- 의존성간의 버전 미스매치 해결
- 프로젝트 단순화

```
{
  "name": "test",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.11.4",
    "@testing-library/react": "^11.1.0",
    "@testing-library/user-event": "^12.1.10",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "^1.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Create React App

package.json

2.1 Create React App

DEVIEW 2021

😀 무궁무진한 확장성

`npm run eject`

Note: this is a one-way operation. Once you `eject`, you can't go back!

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc.) into your project as dependencies in `package.json`. Technically, the distinction between dependencies and development dependencies is pretty arbitrary for front-end apps that produce static bundles.

In addition, it used to cause problems with some hosting platforms that didn't install development dependencies (and thus weren't able to build the project on the server or test it right before deployment). You are free to rearrange your dependencies in `package.json` as you see fit.

All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.

You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

Create React App 공식문서
npm run eject

2.1 Create React App

DEVIEW 2021

☹️ 자체 지원 SSR(Server-side Rendering)이 없음

- SSR을 위한 서버 구축
- @loadable-component와 같은 SSR용 Code Splitting 라이브러리 필요
- 복잡한 Webpack & Babel 설정

2.1 Create React App

DEVIEW 2021

☹️ eject시 복잡성 증가

```
yarn run v1.22.5
$ react-scripts eject
NOTE: Create React App 2+ supports TypeScript, Sass, CSS Modules and more without ejecting: https://reactjs.org/blog/2018/10/01/create-react-app-v2.html

? Are you sure you want to eject? This action is permanent. › (y/N)
```

eject시 경고문구

2.1 Create React App

DEVIEW 2021

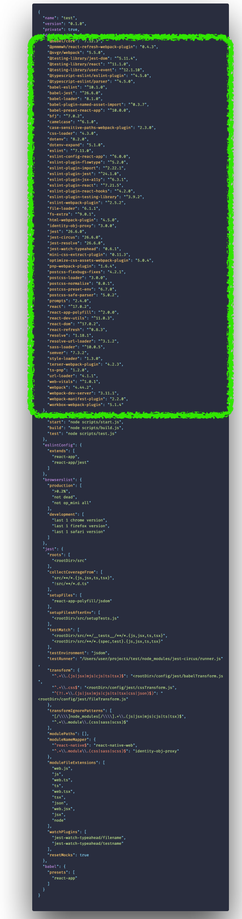
☹ eject시 복잡성 증가

의존성 7개

```
{
  "name": "test",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.11.4",
    "@testing-library/react": "^11.1.0",
    "@testing-library/user-event": "^12.1.10",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "^1.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

eject

eject 전
Create React App
package.json



의존성 64개

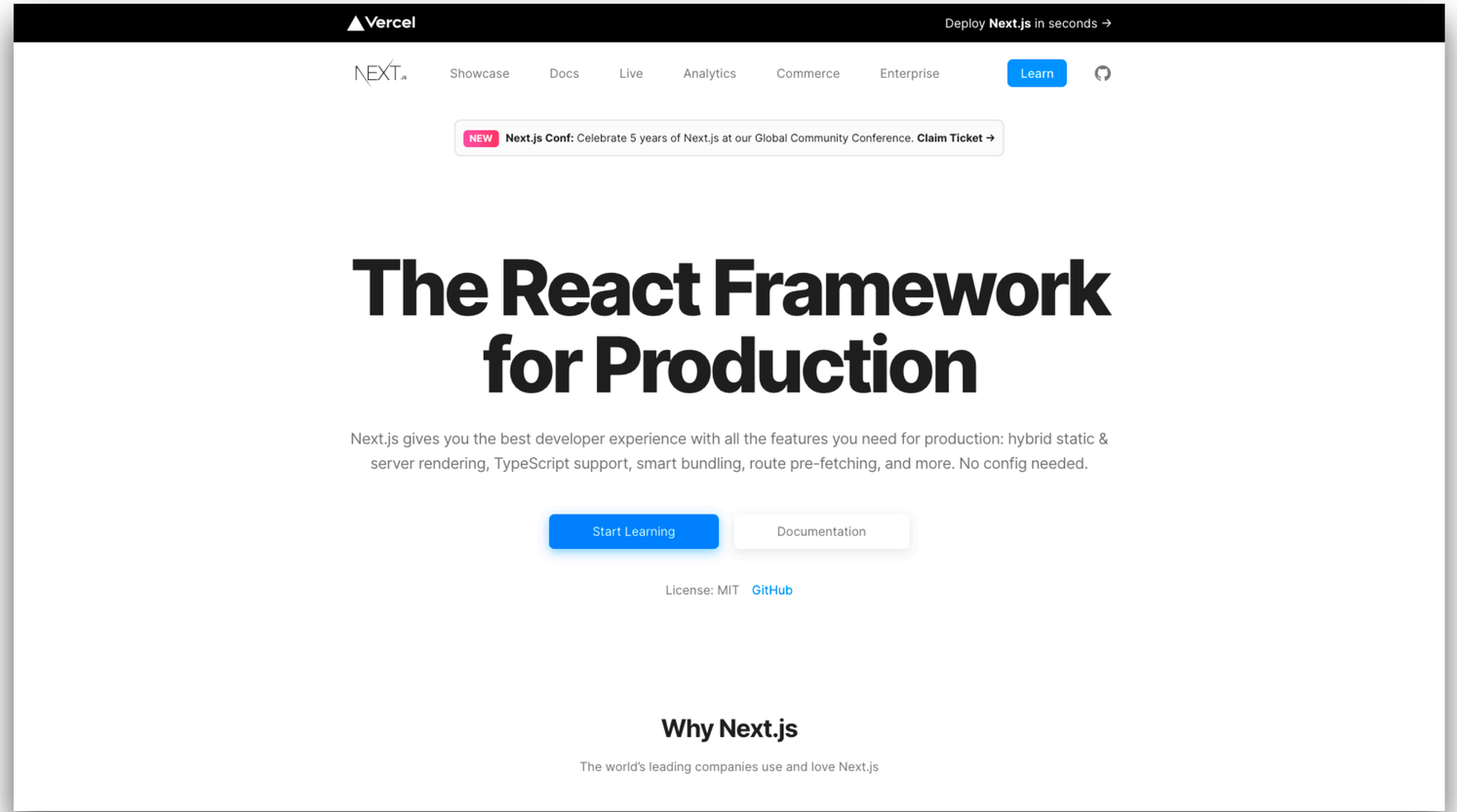
eject 후
Create React App
package.json

2.2 Next.js

DEVIEW 2021

소개

- Vercel 사에서 생성, 유지, 관리하는 React Framework



Next.js
공식 홈페이지

2.2 Next.js

DEVIEW 2021

Pre-rendering

- Static Generation와 Server-side Rendering, 두가지 형태의 Pre-rendering

SSG

getStaticProps

getStaticPaths

SSR

getServerSideProps

2.2 Next.js

DEVIEW 2021

next.config.js

- 추가적으로 세부 설정을 수정하고 싶다면, next.config.js로 간단하게 수정 가능

```
module.exports = {
  webpack(config) {
    config.plugins.push(
      new UitSpritesmithWebpack({
        spriteSrc: path.resolve(__dirname, './public/static/sprites/'),
        spriteDest: path.resolve(__dirname, './public/static/img/sprites/'),
        cssDest: path.resolve(__dirname, './public/static/scss/utils/sprites/'),
        imgURL: '~public/static/img/sprites',
        prefix: 'sp_',
        ratio: 2,
      }),
    )

    config.resolve = {
      ...config.resolve,
      alias: {
        ...config.resolve.alias,
      },
    }

    config.node = {
      fs: 'empty',
      net: 'empty',
    }

    return config
  },
}
```

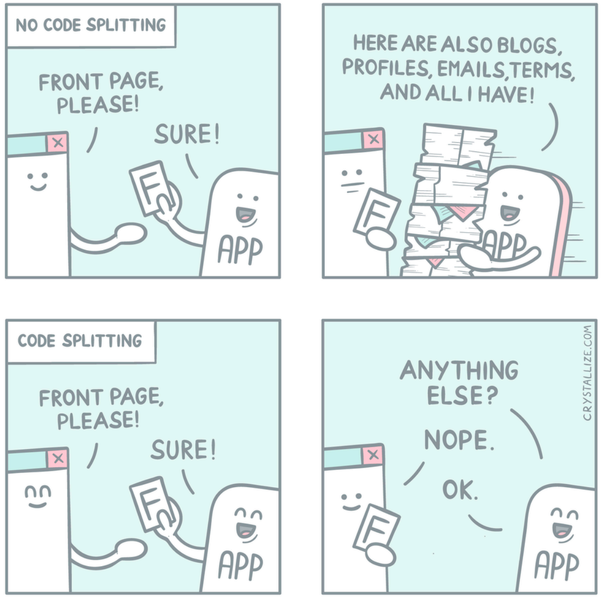
next.config.js

2.2 Next.js

DEVIEW 2021

Code-splitting

- 페이지들이 각각의 JavaScript 번들로 분리
- 파일, React Component Dynamic Import 지원



https://crystallize.com/blog/frontend-performance-optimization-react-code-splitting

```
const SecondAuthModalWithNoSSR = dynamic(() => import('../components/Common/Modal/SecondAuthModal'), { ssr: false })
```

Dynamic Import React Component

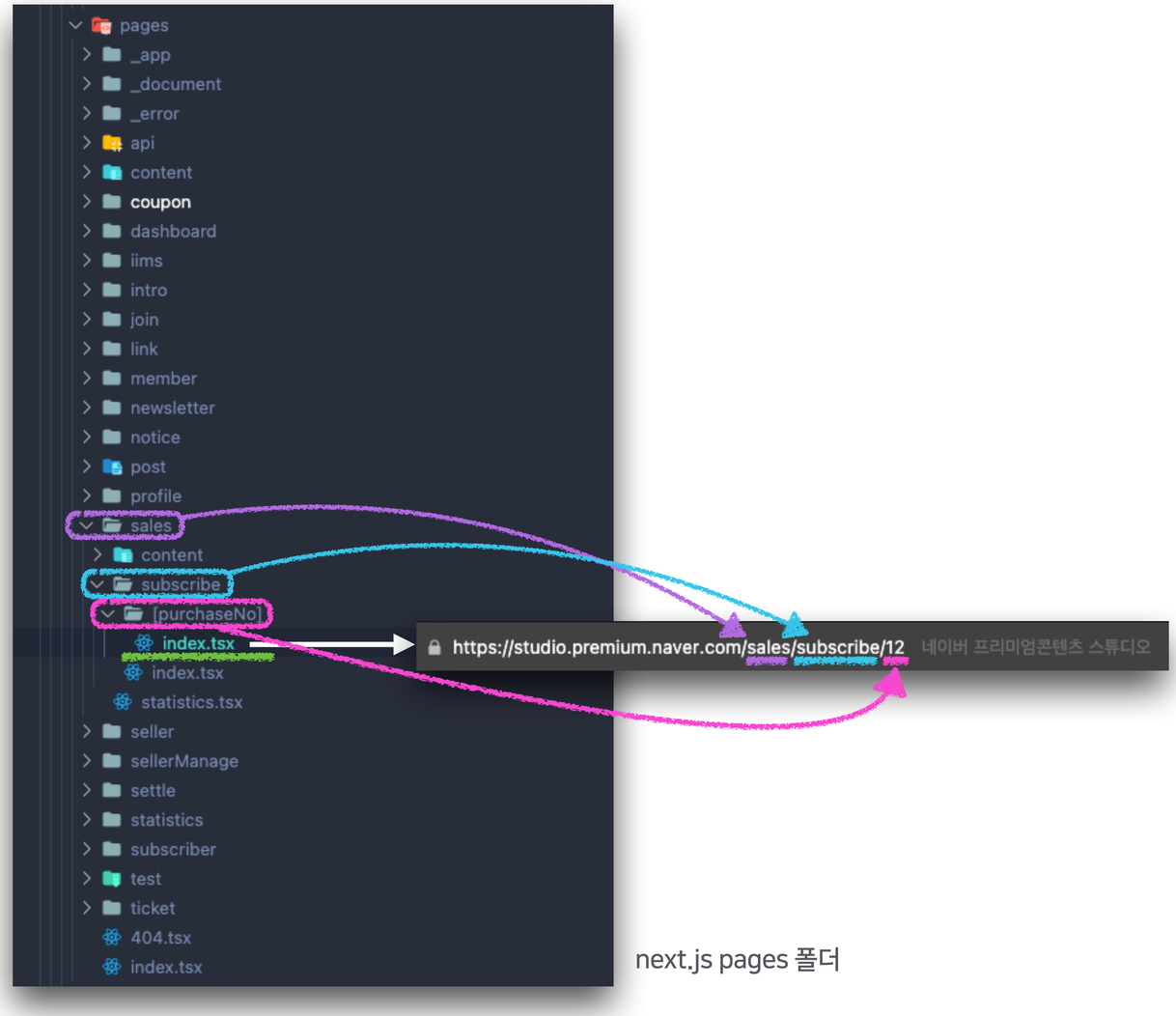
2.2 Next.js

DEVIEW 2021

Routing

- 파일 시스템에 기반한 라우팅
- 정적, 동적, 중첩 라우팅

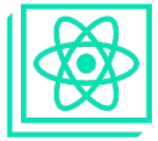
파일 위치		URI
pages/content/manage	=>	/content/manage
pages/member/[id]	=>	/member/1, /member/a
pages/post/[…slug]	=>	/post/1/2, /post/a/b



next.js pages 폴더

2.3 Create React App 👉 Next.js

DEVIEW 2021



자체 지원 SSR 없음

eject시 복잡성 증가

👉

NEXT.JS

Pre-rendering

next.config.js

+ Routing

+ Code-splitting

3. Apollo Client

3.1 REST

DEVIEW 2021

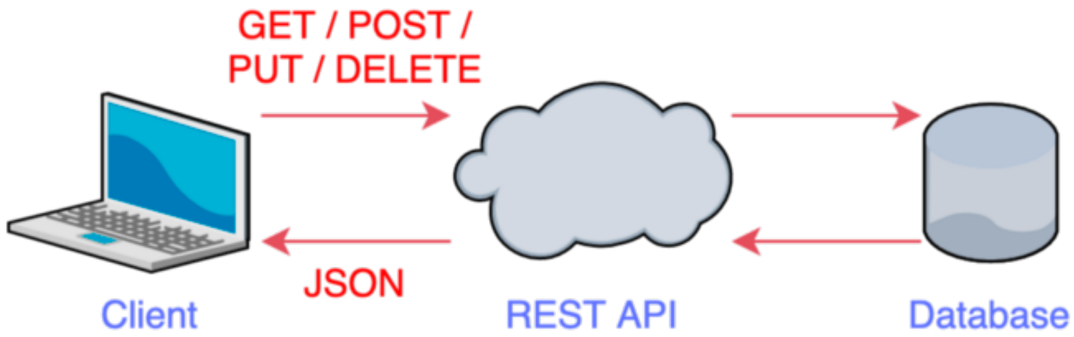
{ REST } 👉 GraphQL

3.1 REST

DEVIEW 2021

소개

- Web API를 설계하는 데 가장 보편적으로 사용하는 아키텍처



3.1 REST

DEVIEW 2021

자원을 표현하는 URI 기반

POST	/{cpType}/{cpName}/{channelName}/products	상품 등록
POST	/{cpType}/{cpName}/{channelName}/tickets	이용권 단건 등록
POST	/{cpType}/coupons	[쿠폰] 발행
GET	/channel	채널 목록 조회

3.1 REST

DEVIEW 2021

☹ Overfetching

- 불필요한 데이터가 있는 경우, 필요한 부분만 거르는 작업 필요

/sessions/29 

Response


```
{
  "session" {
    "id": 29,
    "title": "Apollo, Next.js와 함께 리액트 개발의 Next Level로 가자!",
    "category": "for Juniors",
    "content": "Apollo + Next.js = Next Level",
    "createdAt": "2021.10.01",
    "updatedAt": "2021.11.26",
    "speaker"  {
      "name": "이건",
      "gender": "M",
      "age": 29
    }
  }
}
```

3.1 REST

DEVIEW 2021

☹ Overfetching

- 불필요한 데이터가 있는 경우, 필요한 부분만 거르는 작업 필요

/sessions/29 

Response

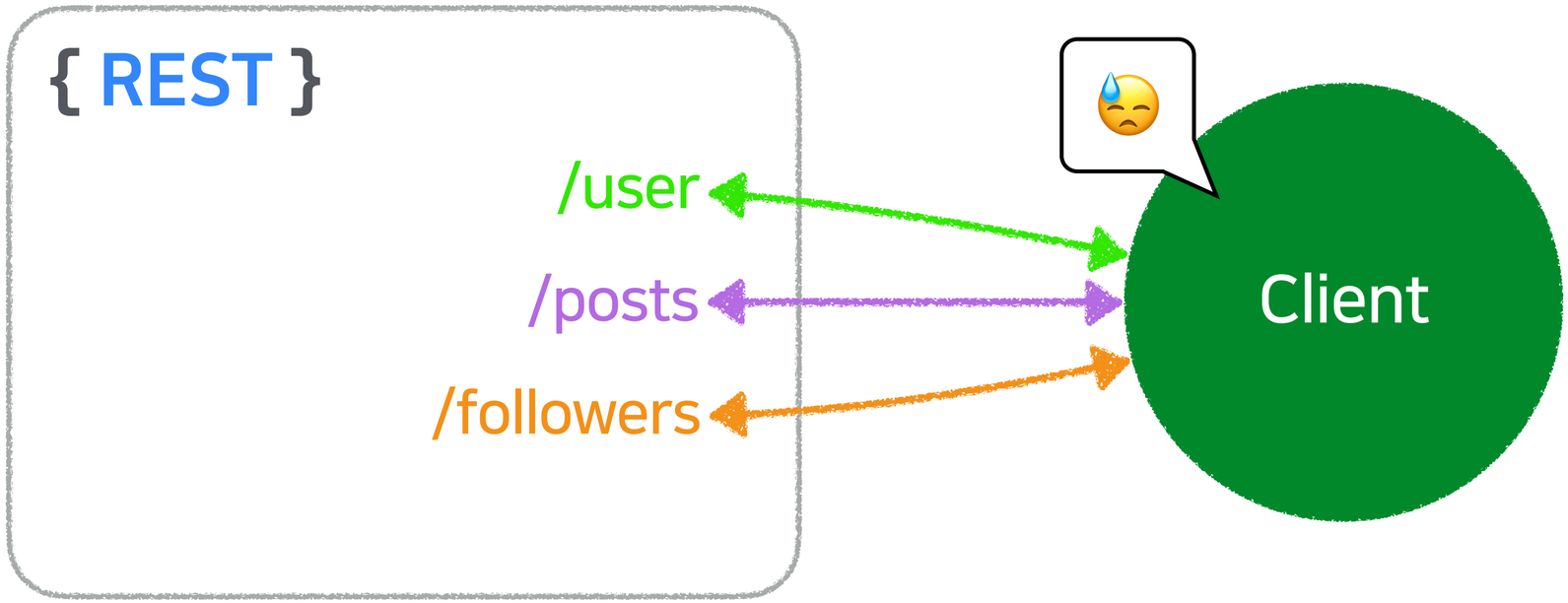

```
{
  "session" {
    "id": 29,
    "title": "Apollo, Next.js와 함께 리액트 개발의 Next Level로 가자!",
    "category": "for Juniors",
    "content": "Apollo + Next.js = Next Level",
    "createdAt": "2021.10.01",
    "updatedAt": "2021.11.26",
    "speaker"  {
      "name": "이건",
      "gender": "M",
      "age": 29
    }
  }
}
```

3.1 REST

DEVIEW 2021

☹️ Underfetching

- 한 페이지에 여러 요청이 필요한 경우 각각의 요청을 따로 보내야한다.

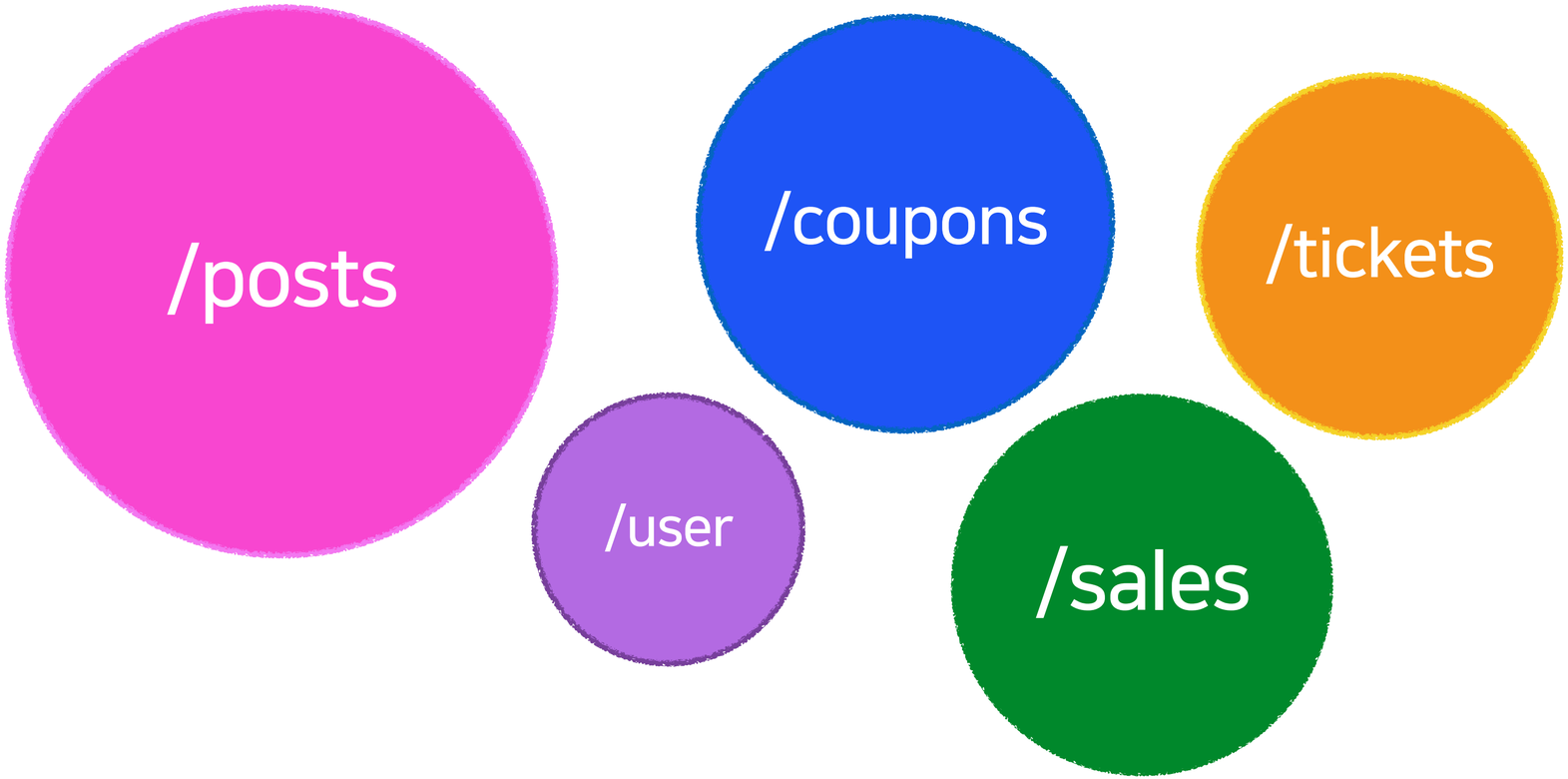


3.1 REST

DEVIEW 2021

☹ 무수한 엔드포인트

- 자원이 많아질수록 더 많아지는 엔드포인트



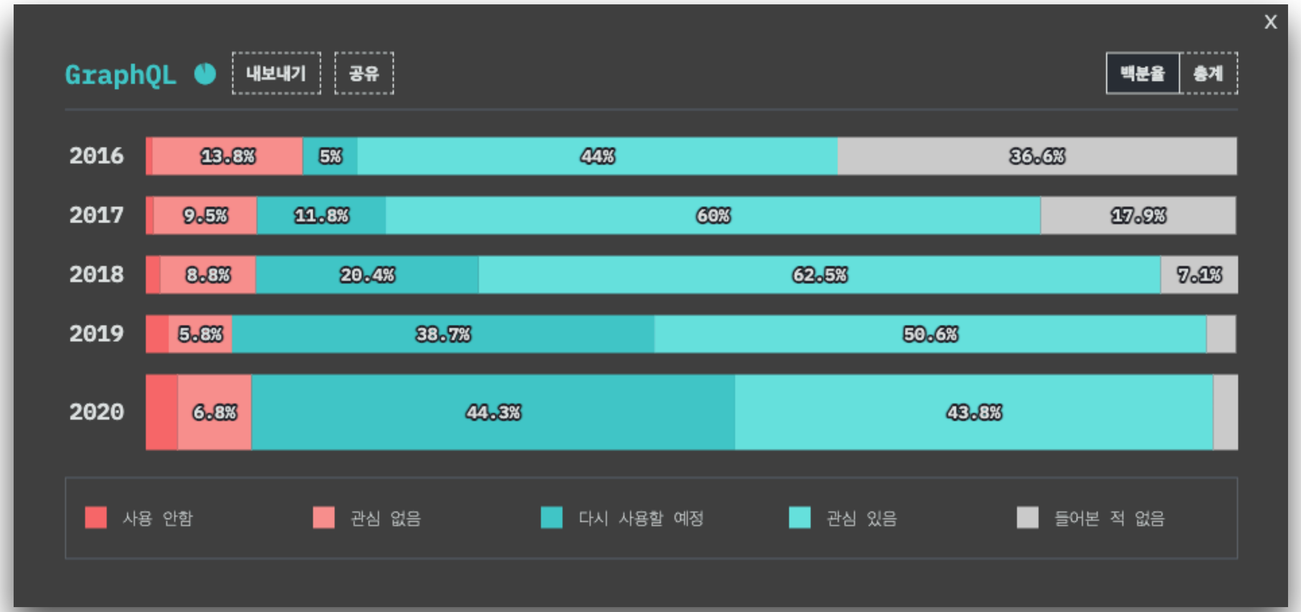
3.2 GraphQL

DEVIEW 2021

소개

- API와 런타임을 위한 쿼리 언어
- 클라이언트가 데이터를 효율적으로 가져오는 것이 목적

GraphQL



State of JS 2020

3.2 GraphQL

😀 필요한 데이터만 쏙쏙

GraphQL query

```
{
  session  {
    title
    category
    content
    speaker  {
      name
    }
  }
}
```

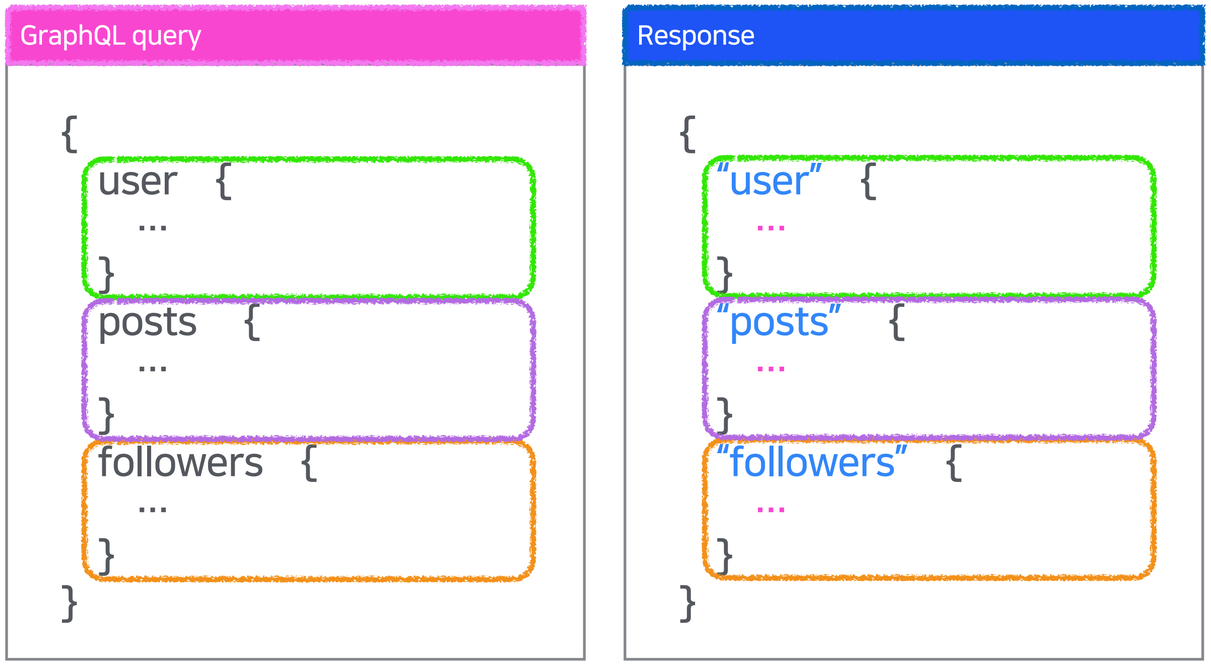
Response

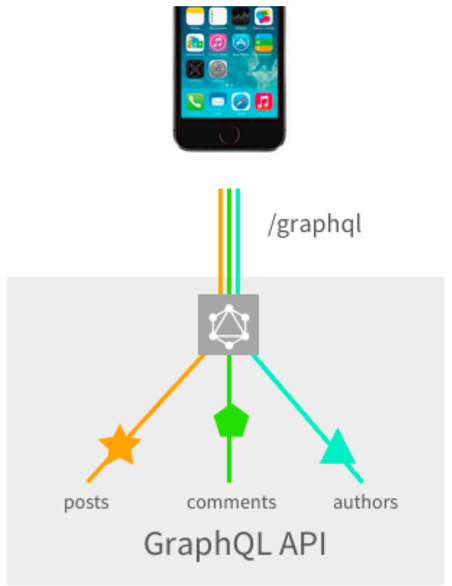
```
{
  "session"  {
    "title": "Apollo, Next.js와 함께 리액트 개발의 Next Level로 가자!",
    "category": "for Juniors",
    "content": "Apollo + Next.js = Next Level",
    "speaker"  {
      "name": "이건"
    }
  }
}
```

3.2 GraphQL

DEVIEW 2021

😀 여러 요청을 한 번에





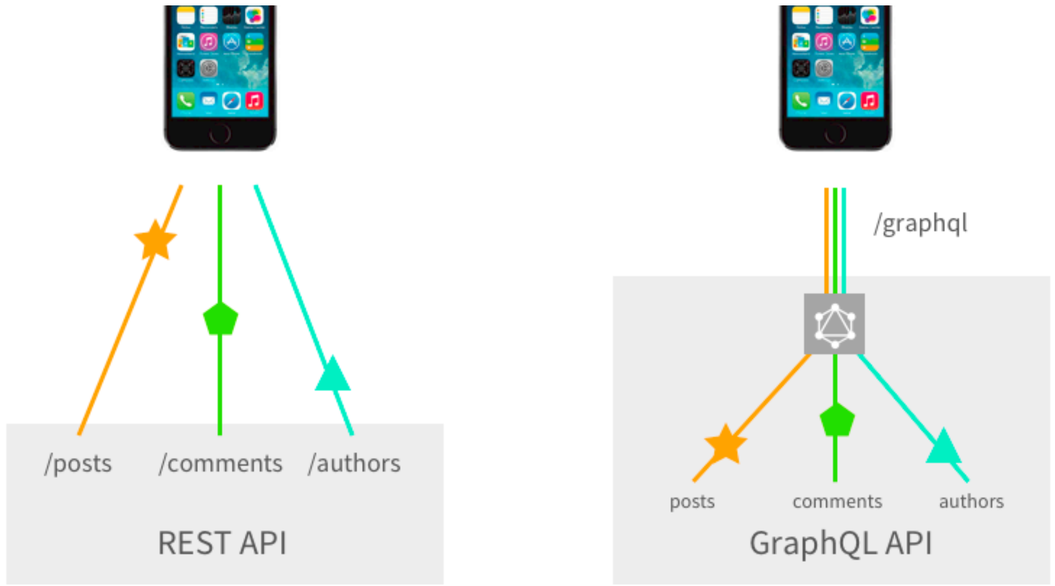
* https://www.apollographql.com/blog/graphql/basics/graphql-vs-rest/

3.2 GraphQL

DEVIEW 2021

😀 단 하나의 엔드포인트

- 하나의 엔드포인트에 POST로 모든 요청 가능



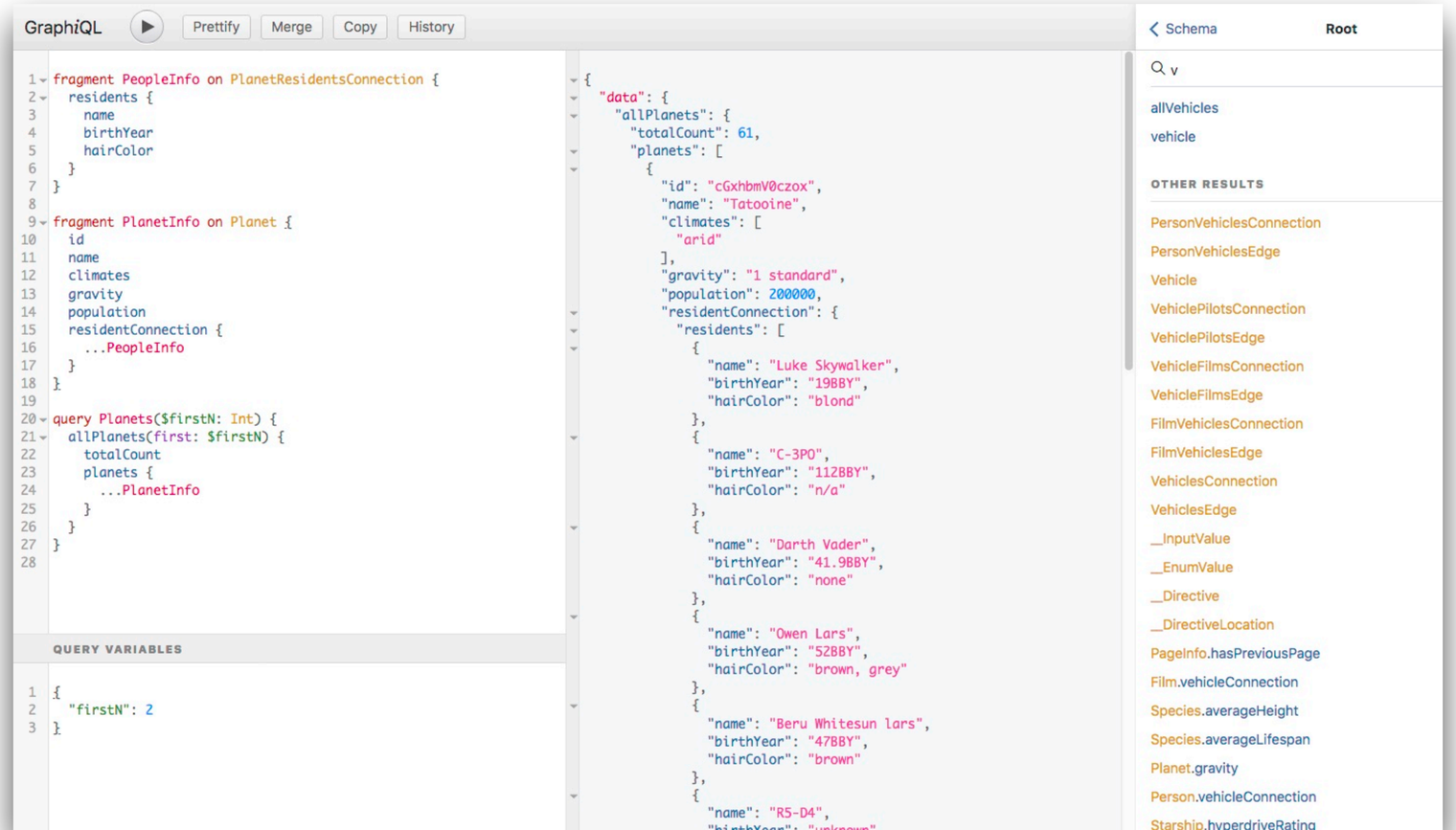
* https://www.apollographql.com/blog/graphql/basics/graphql-vs-rest/

3.2 GraphQL

DEVIEW 2021

😀 GraphiQL

- GraphQL 서버를 자유롭게 테스트 해볼 수 있는 도구



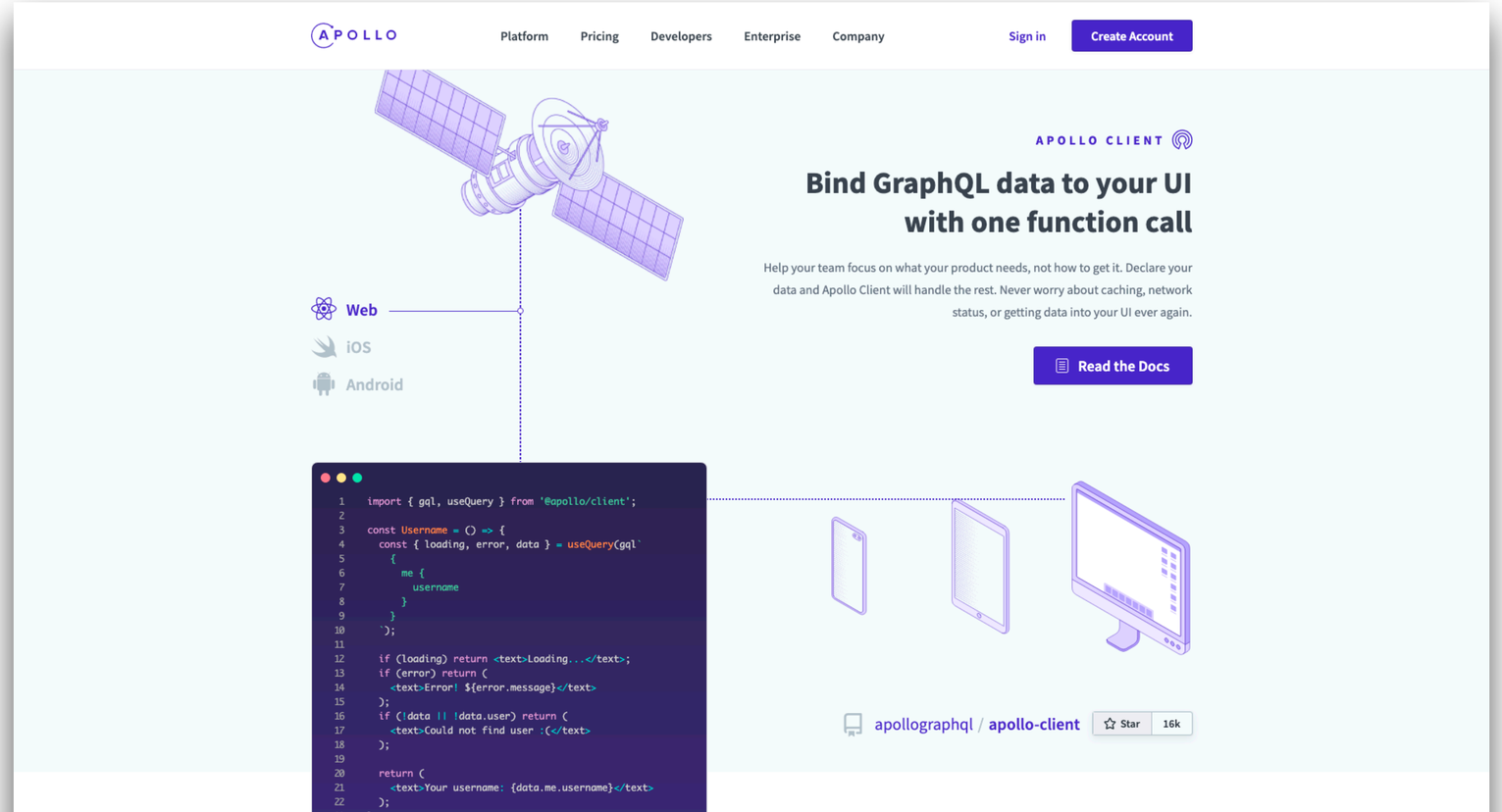
GraphiQL

3.3 Apollo Client

DEVIEW 2021

소개

- Apollo Graph에서 제작한 GraphQL 클라이언트



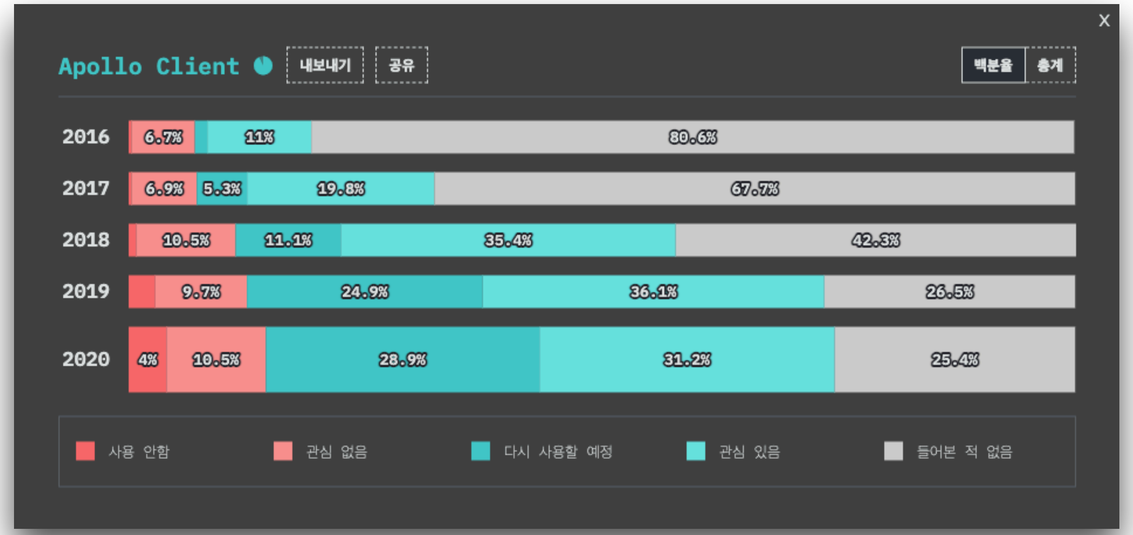
Apollo Client
공식 홈페이지

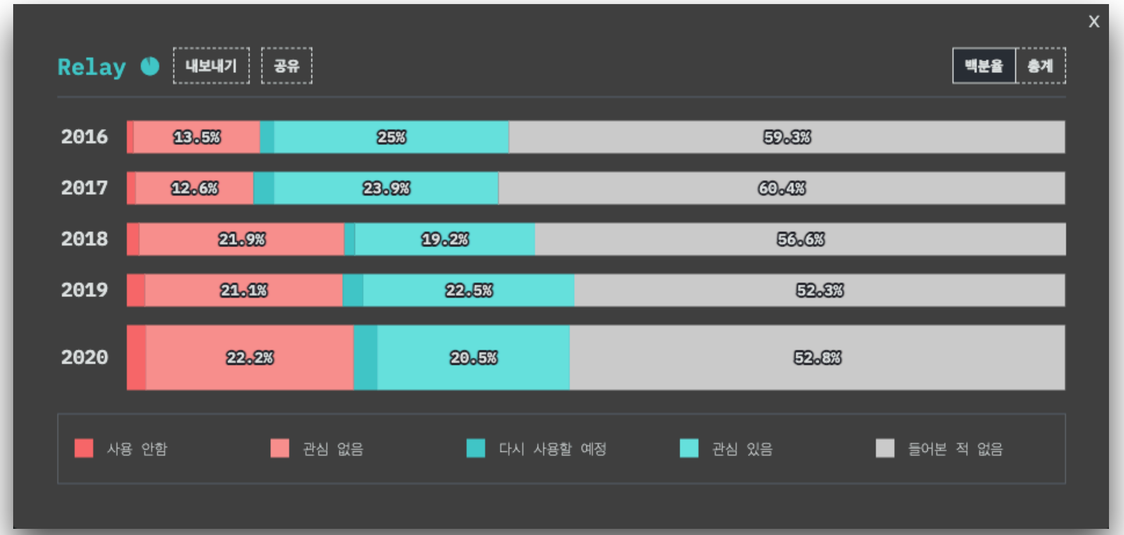
3.3 Apollo Client

DEVIEW 2021

소개

- Apollo Graph에서 제작한 GraphQL 클라이언트





State of JS 2020

3.3 Apollo Client

DEVIEW 2021

😀 최신 리액트를 위해 디자인된 라이브러리

- useQuery
- useMutation

```
const { data, loading, error } = useQuery
  { purchaseDetail: Query['getPurchaseDetail'] },
  QueryGetPurchaseDetailArgs & PremiumAuth & GeneralAuth
>(GET_PURCHASE_DETAIL_QUERY, {
  variables: {
    cpType,
    cpName,
    cpId,
    subId,
    channelName: subId,
    purchaseNo: targetPurchaseNo,
  },
  context,
})
```

useQuery

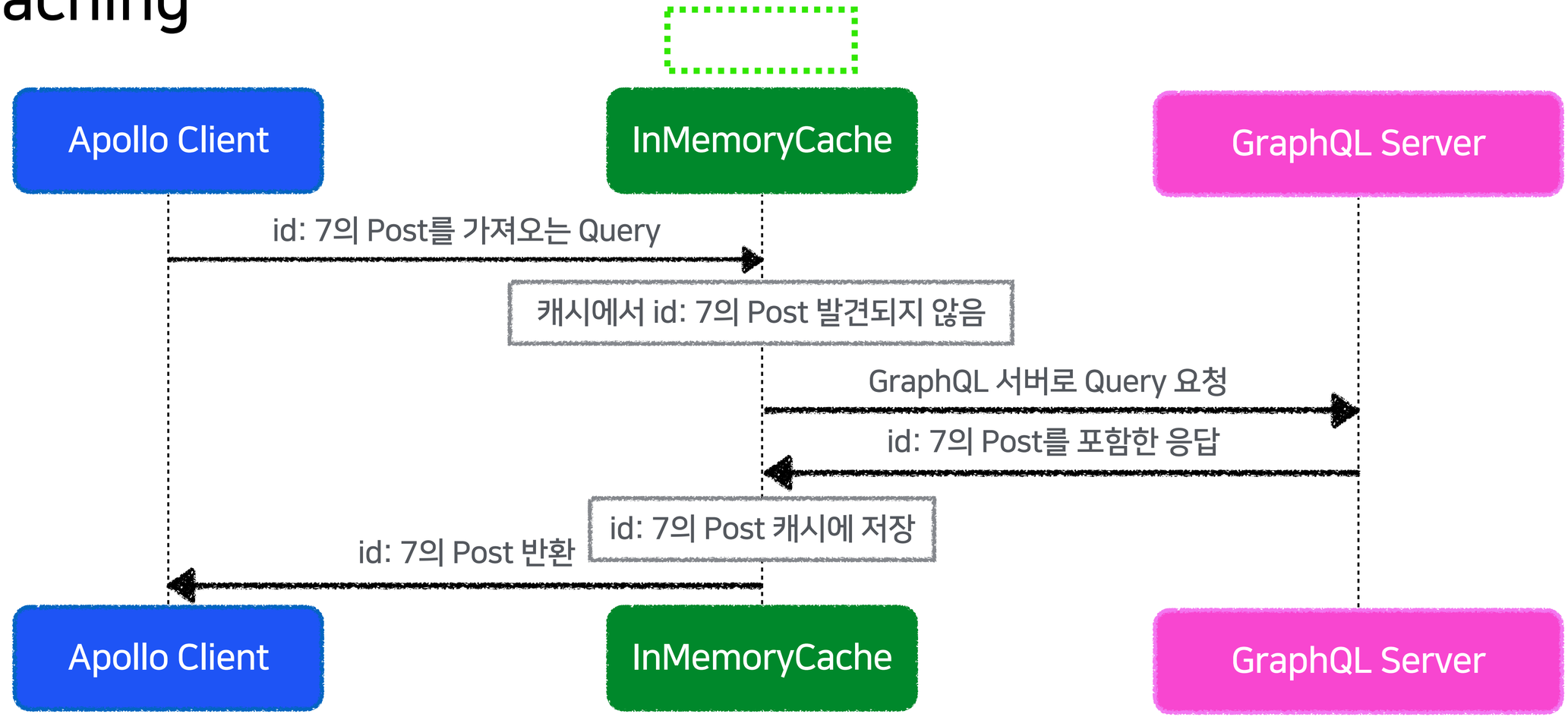
```
const [removeContent] = useMutation
  { removeContent: ContentsHubUpdated },
  { _id: string; body: RemoveContentInputBody } & PremiumAuth & GeneralAuth
>(REMOVE_CONTENT_MUTATION, {
  context,
})
```

useMutation

3.3 Apollo Client

DEVIEW 2021

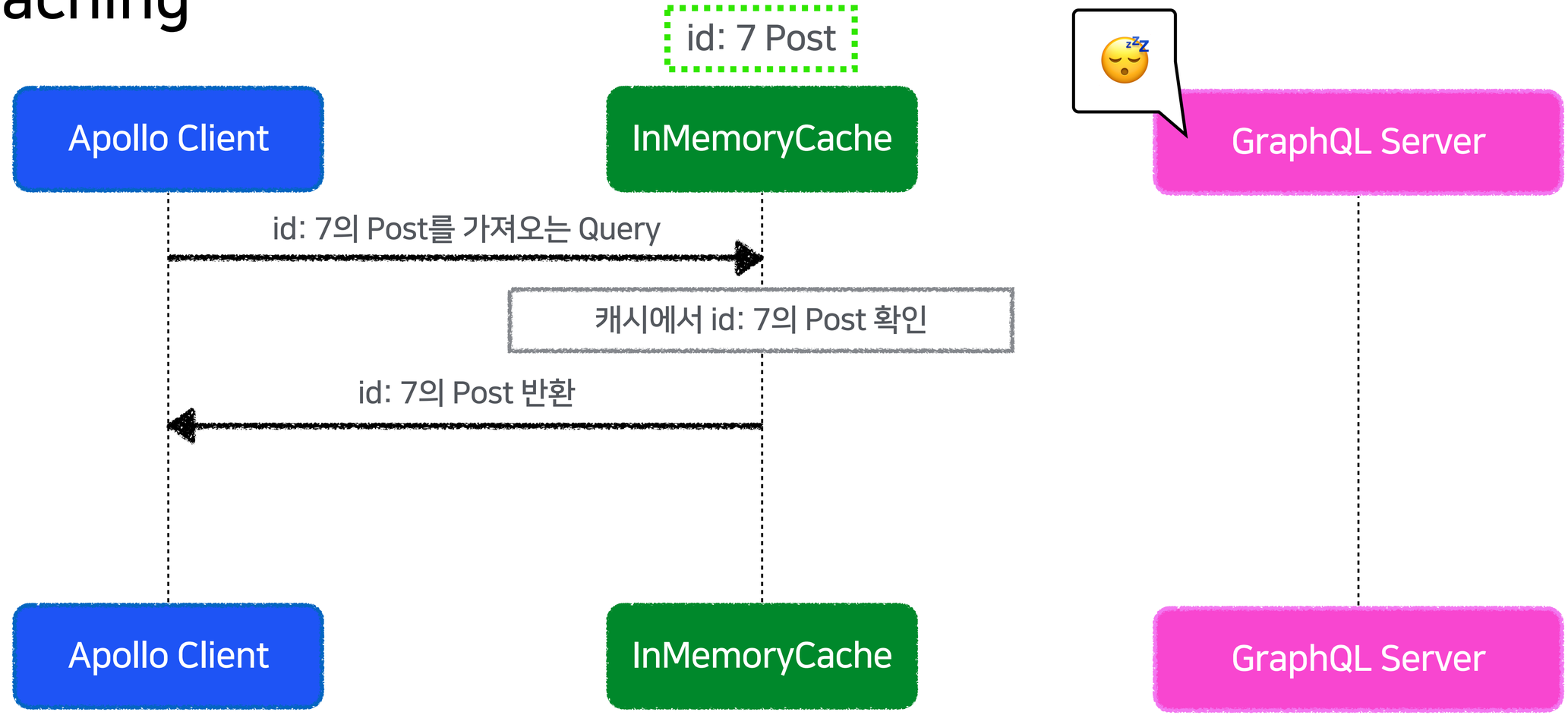
😀 Caching



3.3 Apollo Client

DEVIEW 2021

😀 Caching



3.3 Apollo Client

DEVIEW 2021

😀 Caching

useQuery
fetchPolicy

Supported fetch policies

NAME	DESCRIPTION
`cache-first`	Apollo Client first executes the query against the cache. If *all* requested data is present in the cache, that data is returned. Otherwise, Apollo Client executes the query against your GraphQL server and returns that data after caching it. Prioritizes minimizing the number of network requests sent by your application. This is the default fetch policy.
`cache-only`	Apollo Client executes the query *only* against the cache. It never queries your server in this case. A `cache-only` query throws an error if the cache does not contain data for all requested fields.
`cache-and-network`	Apollo Client executes the full query against both the cache *and* your GraphQL server. The query automatically updates if the result of the server-side query modifies cached fields. Provides a fast response while also helping to keep cached data consistent with server data.
`network-only`	Apollo Client executes the full query against your GraphQL server, *without* first checking the cache. The query's result *is* stored in the cache. Prioritizes consistency with server data, but can't provide a near-instantaneous response when cached data is available.
`no-cache`	Similar to `network-only`, except the query's result *is not* stored in the cache.
`standby`	Uses the same logic as `cache-first`, except this query does *not* automatically update when underlying field values change. You can still *manually* update this query with `refetch` and `updateQueries`.

3.4 REST API 👉 GraphQL

DEVIEW 2021

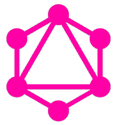
{ REST }

OverFetching

UnderFetcing

무수한 엔드포인트

👉



GraphQL

필요한 데이터만 쏙쏙

여러 요청을 한 번에

단일 엔드포인트

+ 리액트 통합성, Caching (with Apollo)

4. Troubleshooting & Tips

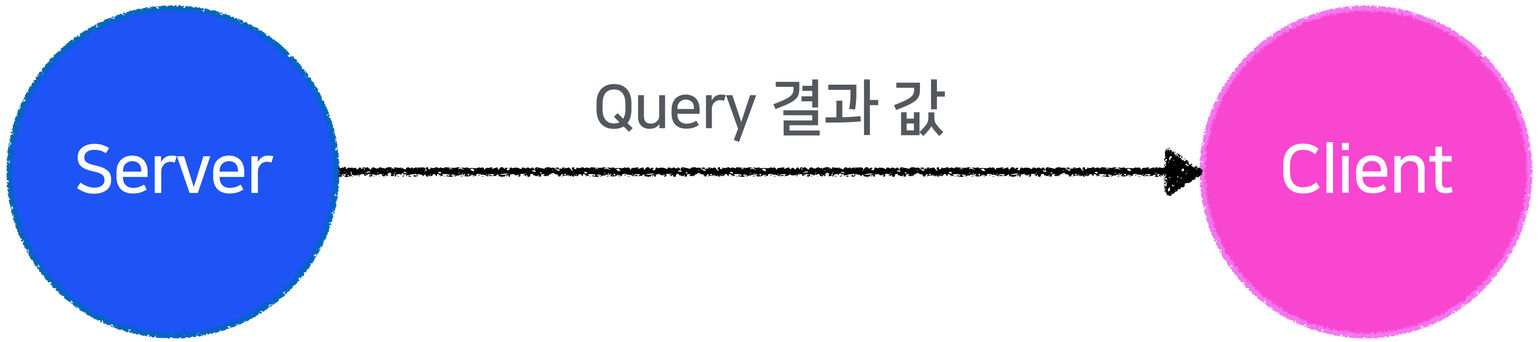
4.1 SSR with Apollo Client

문제 상황

- SSR에서 Apollo Client로 얻은 결과 데이터를 어떻게 클라이언트로 보낼 것인가?

4.1 SSR with Apollo Client

시도1: 결과 값만 보내기



4.1 SSR with Apollo Client

DEVIEW 2021

시도1: 결과 값만 보내기

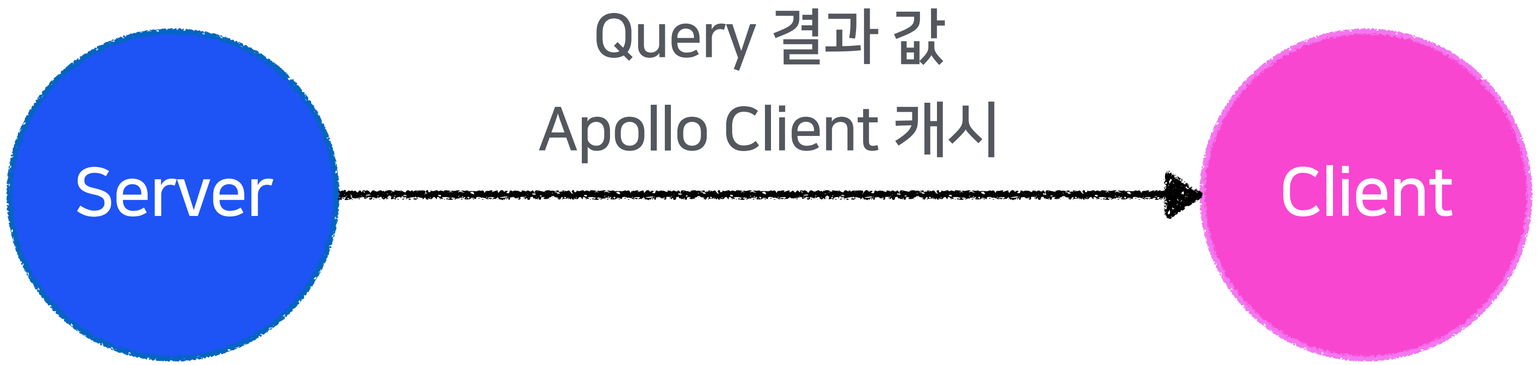
- SSR 실패시 채울 값이 없이 때문에 결국 useQuery 필요
- Mutation 과정 이후 캐시 업데이트가 안 됨

```
export async function getServerSideProps () {
  const apolloClient = initializeApollo()

  const { data } = await apolloClient.query({
    query: ALL_POSTS_QUERY,
    variables: allPostsQueryVars,
  })

  const { allPosts } = data

  return {
    props: {
      initialAllPosts: allPosts,
    },
  }
})
```

4.1 SSR with Apollo Client

DEVIEW 2021

시도2: 결과 값과 캐시 모두 보내기



4.1 SSR with Apollo Client

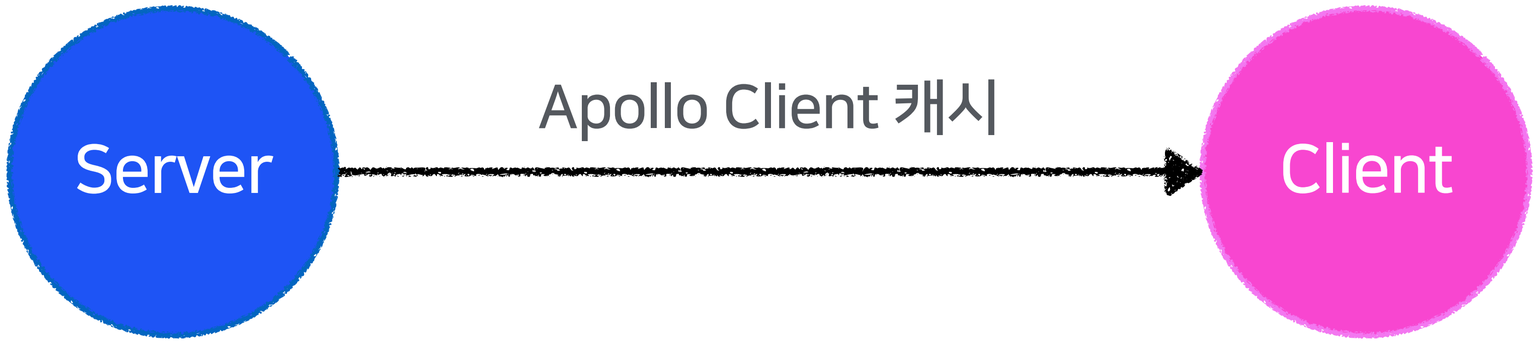
시도2: 결과 값과 캐시 모두 보내기

- 결과 값과 캐시에 중복된 값을 보내서 코드가 더 복잡해지는 느낌

```
export async function getServerSideProps () {
  const apolloClient = initializeApollo()

  const { data } = await apolloClient.query({
    query: ALL_POSTS_QUERY,
    variables: allPostsQueryVars,
  })

  const { allPosts } = data

  return {
    props: {
      [APOLLO_STATE_PROP_NAME]: apolloClient.cache.extract(),
      initialAllPosts: allPosts,
    }
  }
})
```

4.1 SSR with Apollo Client

해결: 캐시만 보내기



4.1 SSR with Apollo Client

DEVIEW 2021

해결: 캐시만 보내기

- 데이터 중복 없이 필요한 정보만 보내짐
- SSR 성공: useQuery에서 로딩 과정 없이 캐시에 있는 데이터를 사용
- SSR 실패: useQuery에서 로딩 ~ 패칭의 과정 진행

```
export async function getServerSideProps () {
  const apolloClient = initializeApollo()

  await apolloClient.query({
    query: ALL_POSTS_QUERY,
    variables: allPostsQueryVars,
  })

  return {
    props: {
      [APOLLO_STATE_PROP_NAME]: apolloClient.cache.extract()
    }
  }
})
```

4.2 인증 구현

DEVIEW 2021

문제 상황

- 인증 정보가 있어야 GraphQL 요청을 보낼 수 있다.
- 매번 인증을 위한 코드를 복사 붙여넣기하는 반복작업을 하고싶지 않다.

4.2 인증 구현

해결: 페이지 컴포넌트를 위한 HOC, SSR 메소드를 위한 HOF

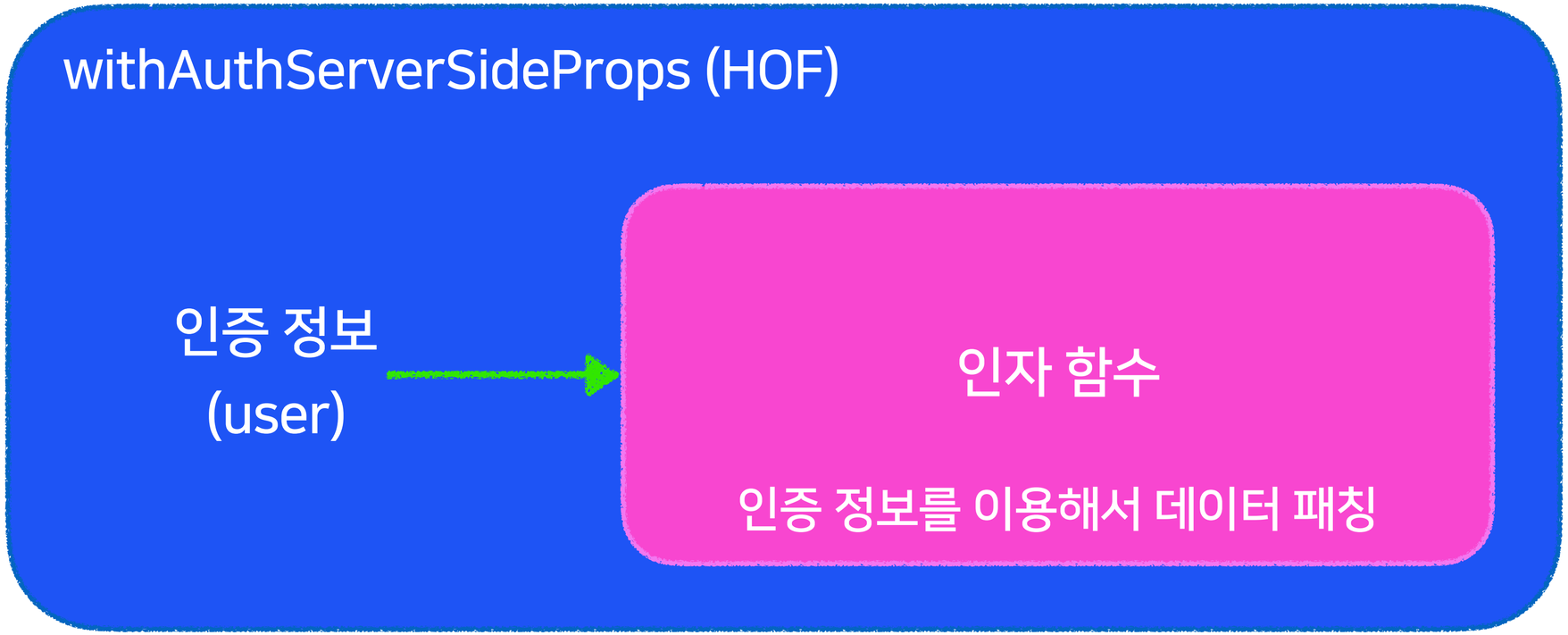
- getServerSideProps HOF - withAuthServerSideProps
- 페이지 컴포넌트 HOC - withAuthComponent

4.2 인증 구현

DEVIEW 2021

withAuthServerSideProps

- SSR 메소드인 getServerSideProps의 인증 로직을 처리할 HOF



4.2 인증 구현

DEVIEW 2021

withAuthServerSideProps

- SSR 메소드인 getServerSideProps의 인증 로직을 처리할 HOF

sps-premium-cms - meteredPaywall.tsx

```tsx
export const getServerSideProps: GetServerSideProps<{}, {}> = withAuthServerSideProps(
  async (ctx, apolloClient, user) => {
    const { authId, cpName, cpType, cpId, subId } = user

    await apolloClient.query<
      { channelMeteredPaywallConfig: Query['getChannelMeteredPaywallConfig'] },
      QueryGetChannelMeteredPaywallConfigArgs & PremiumAuth & GeneralAuth
    >({
      query: GET_CHANNEL_METERED_PAYWALL_CONFIG,
      variables: { cpName, cpType, cpId, subId },
      context: { authId, svcType: SvcType.PREMIUM },
    })

    return {
      props: {
        initialApolloState: apolloClient.cache.extract(),
      },
    }
  },
)
```

4.2 인증 구현

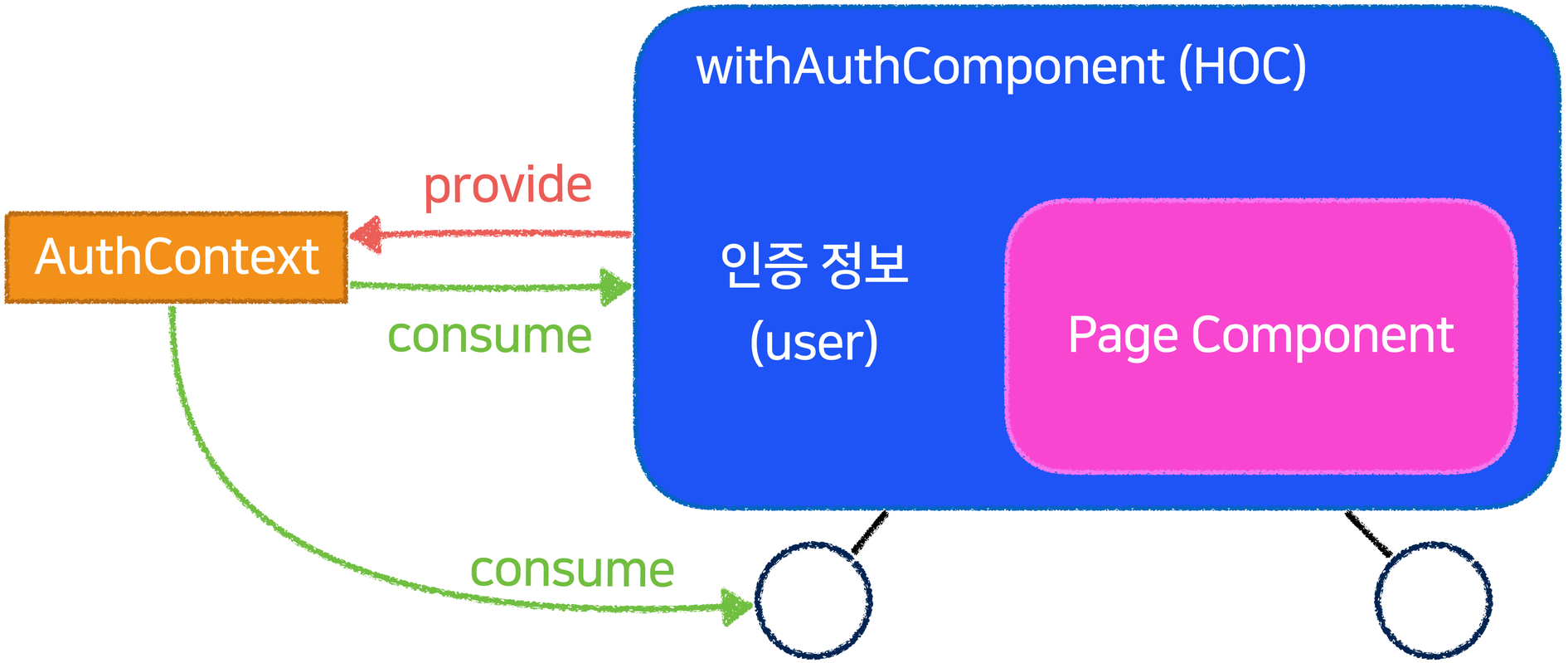
DEVIEW 2021

withAuthComponent

- 페이지 컴포넌트의 인증 로직을 처리할 HOC

sps-premium-cms - meteredPaywall.tsx

```
1  export default withAuthComponent(MeteredPaywall)
2
```



4.3 서버 스키마 동기화

문제 상황

- 클라이언트의 스키마 인터페이스와 서버의 스키마의 불일치

AS-IS

Query **getTicketCount**

이용권 수 조회

TYPE

TicketCountResult

ARGUMENTS

cpType: CpType!

cpName: String!

channelName: String

categoryTypes: [TicketCategoryType] = [CHANNEL_NORMAL]
디폴트 값: CHANNEL_NORMAL

startYmdt: String
패턴: yyyyMMddHHmmss

endYmdt: String
패턴: yyyyMMddHHmmss

TO-BE

Query **getTicketCount**

이용권 수 조회

TYPE

TicketCountResult

ARGUMENTS

cpType: CpType!

cpName: String!

channelName: String!

categoryTypes: [TicketCategoryType] = [CHANNEL_NORMAL]
디폴트 값: CHANNEL_NORMAL

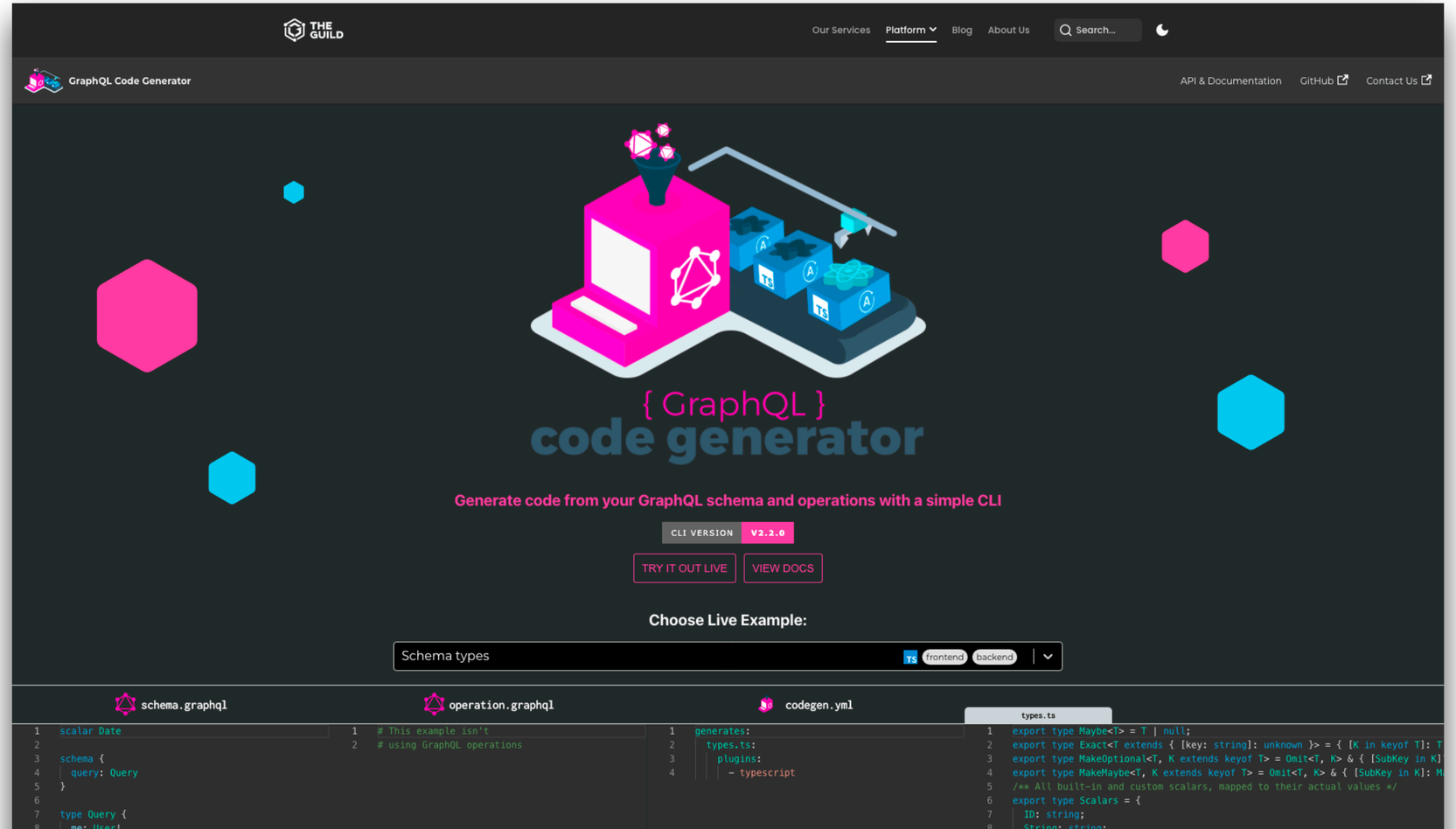
startYmdt: String
패턴: yyyyMMddHHmmss

endYmdt: String
패턴: yyyyMMddHHmmss

saleStatuses: [SaleStatus]
디폴트 값: SALE

4.3 서버 스키마 동기화

해결: graphql-code-generator

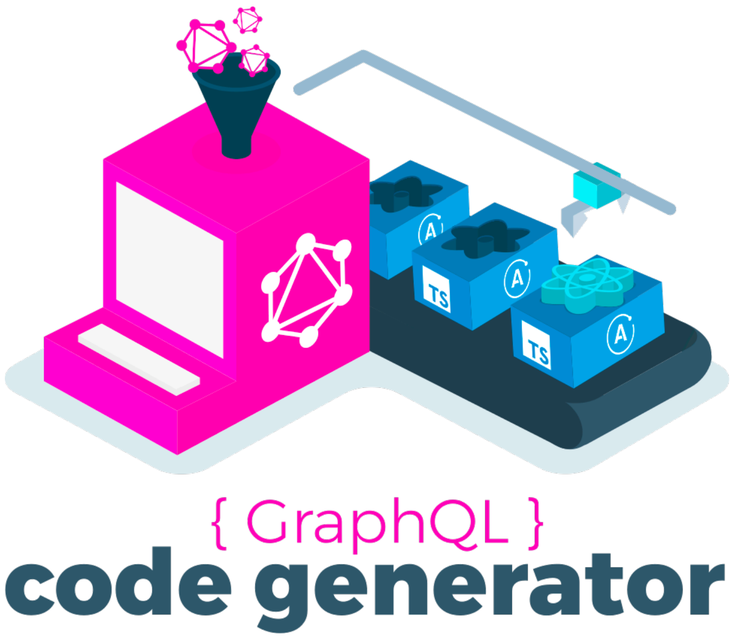


graph-code-generator
공식 홈페이지

4.3 서버 스키마 동기화

DEVIEW 2021

해결: graphql-code-generator



codegen.yml

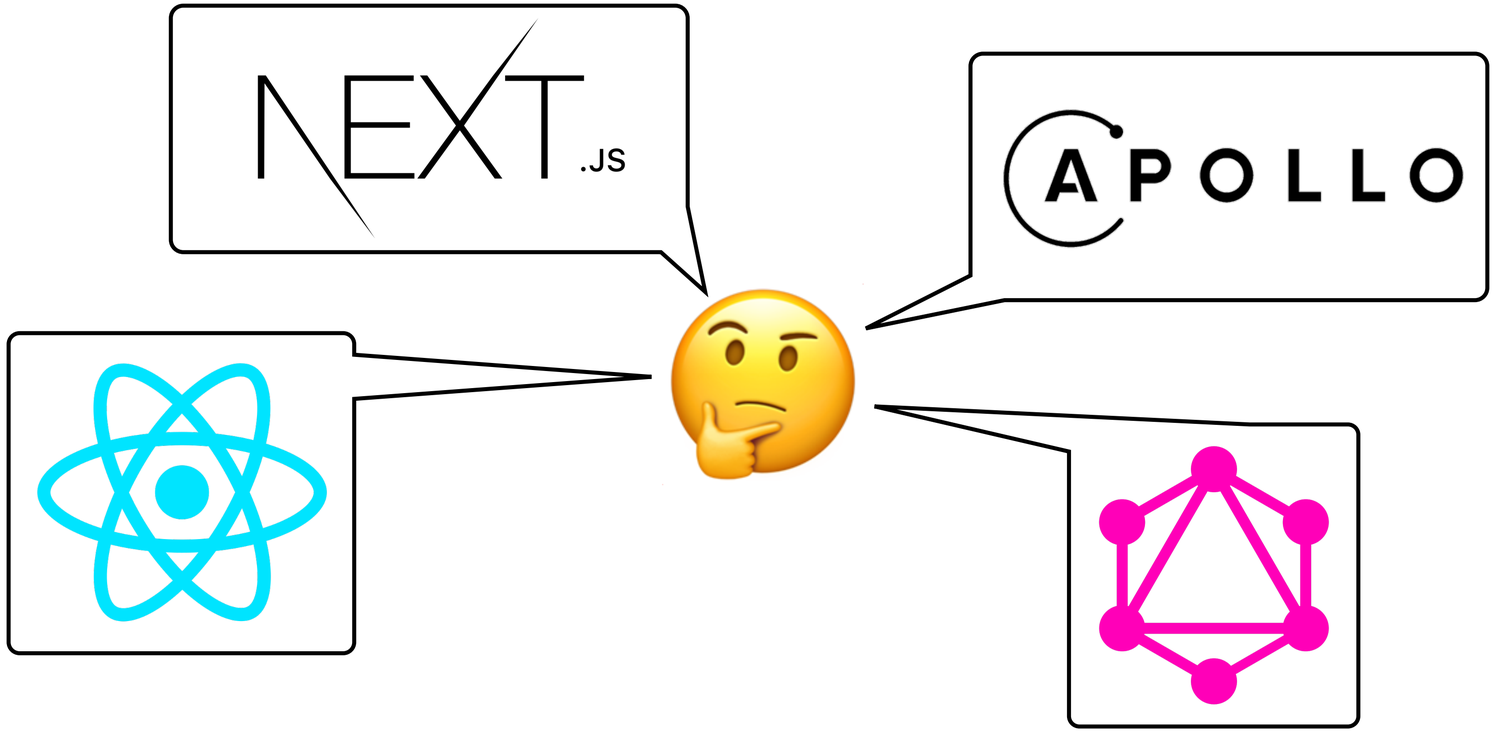
```yaml
generates:
  ./src/interfaces/types.ts:
    schema: GraphQL 서버의 graphql 주소
    plugins:
      - typescript
```

```
const { data, loading, error } = useQuery<
  { posts: Query['getTickets'] }, QueryGetTicketsArgs
  >(GET_TICKETS_QUERY)
```

5. Conclusion

5.1 새로운 기술 스택 학습

DEVIEW 2021



Thank You