

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

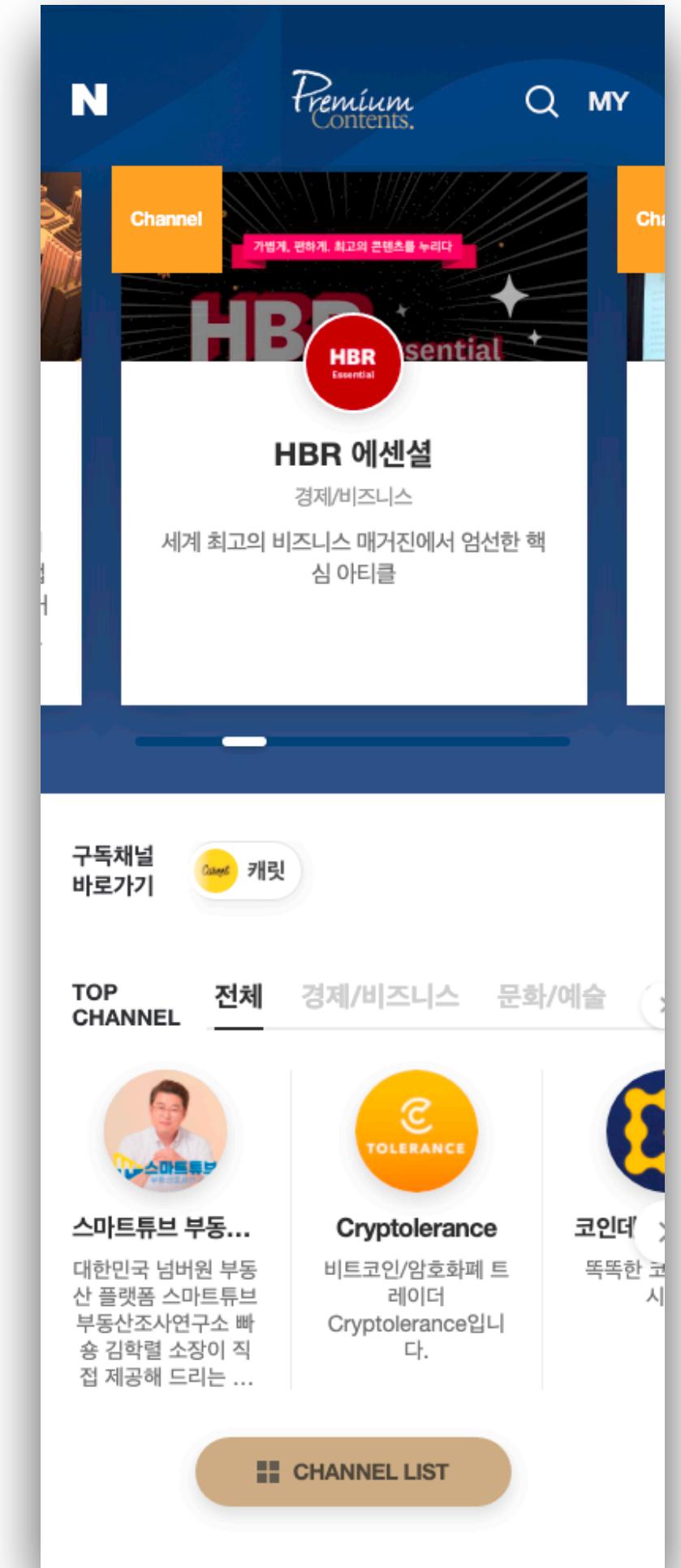
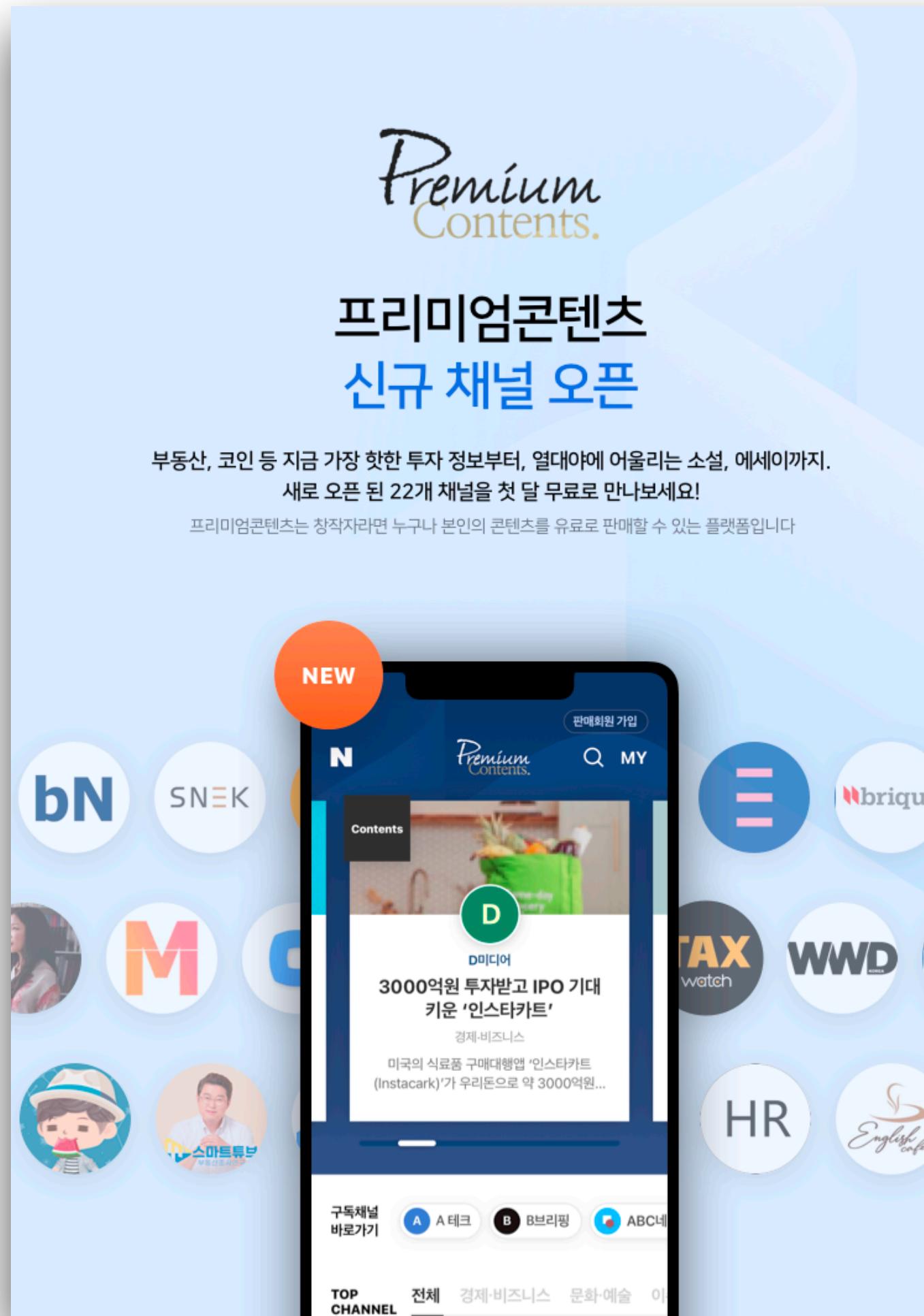
이건 NAVER 프리미엄콘텐츠플랫폼

# CONTENTS

1. 프리미엄 콘텐츠
2. Next.js
3. Apollo Client
4. Troubleshooting & Tips
5. Conclusion

# 1. 프리미엄 콘텐츠

# 1.1 프리미엄 콘텐츠



프리미엄 콘텐츠 서비스

# 1.2 프리미엄 콘텐츠 스튜디오

Premium Contents. 프리미엄콘텐츠 스튜디오

## 방문분석

- 조회수 : 선택한 기간 동안 우리 채널에 방문한 사용자가 콘텐츠를 열람한 횟수입니다.
- 조회수 순위 : 선택한 기간 동안 조회수가 가장 많은 콘텐츠를 순서대로 최대 100개까지 제공합니다.
- 순방문자수 : 선택한 기간 동안 우리 채널에 1회 이상 방문한 중복되지 않은 사용자수입니다.

조회수	조회수 순위	순방문자수																
일간	주간	월간																
2021.09.26. 일																		
<span style="color: orange;">●</span> 전체 <span style="color: lightblue;">○</span> 유료사용자 <span style="color: lightgreen;">○</span> 무료사용자																		
<b>일간 전체 조회수 16,940</b>																		
<table border="1"> <thead> <tr> <th>날짜</th> <th>전체</th> <th>유료 구독자</th> <th>무료 사용자</th> </tr> </thead> <tbody> <tr> <td>2021.09.26. (일)</td> <td>16,940</td> <td>947</td> <td>15,993</td> </tr> <tr> <td>2021.09.25. (토)</td> <td>6,960</td> <td>1,081</td> <td>5,879</td> </tr> <tr> <td>2021.09.24. (금)</td> <td>6,816</td> <td>1,453</td> <td>5,363</td> </tr> </tbody> </table>			날짜	전체	유료 구독자	무료 사용자	2021.09.26. (일)	16,940	947	15,993	2021.09.25. (토)	6,960	1,081	5,879	2021.09.24. (금)	6,816	1,453	5,363
날짜	전체	유료 구독자	무료 사용자															
2021.09.26. (일)	16,940	947	15,993															
2021.09.25. (토)	6,960	1,081	5,879															
2021.09.24. (금)	6,816	1,453	5,363															

프리미엄 콘텐츠 스튜디오 PC

Premium Contents. 프리미엄콘텐츠 스튜디오

## 구독자 수

자세히보기 >

총 구독자수	결제자	이용자
<b>1,131</b>	<b>1,128</b>	<b>3</b>

## 결제자 수

자세히보기 >

일 평균 (최근 30일)	주 평균 (최근 15주)	월 평균 (최근 6개월)
<b>35</b>	<b>183</b>	<b>483</b>

## 판매현황

자세히보기 >

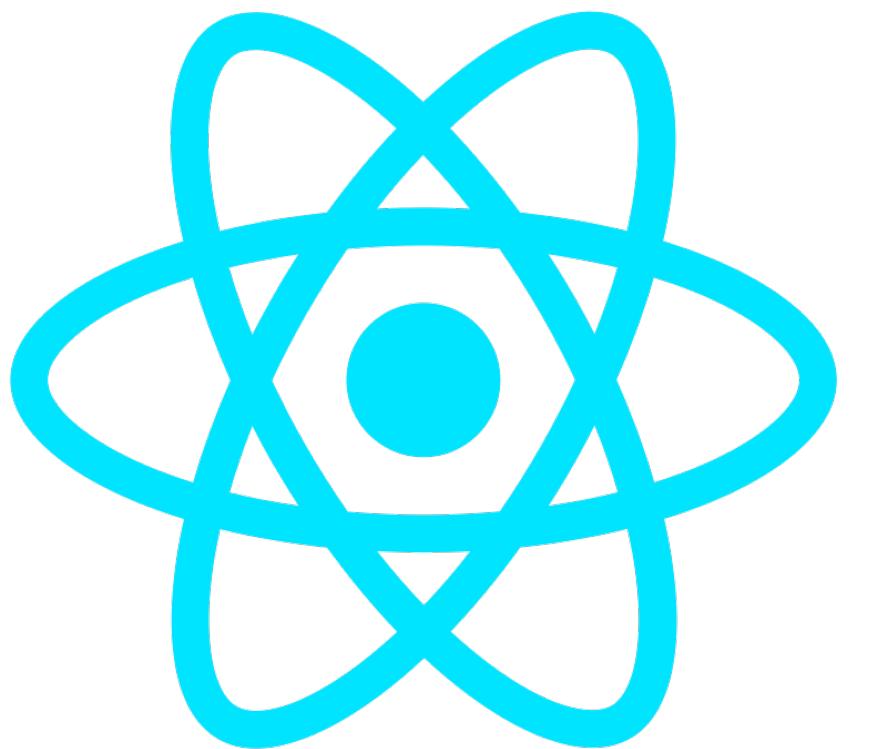
날짜	판매량	매출액 (원)
9/16	~	~
9/17	~	~
9/18	~	~
9/19	~	~
9/20	~	~
9/21	~	~
9/22	~	~
9/23	~	~
9/24	~	~
9/25	~	~
9/26	~	~
9/27	~	~
9/28	~	~

프리미엄 콘텐츠 스튜디오 Mobile

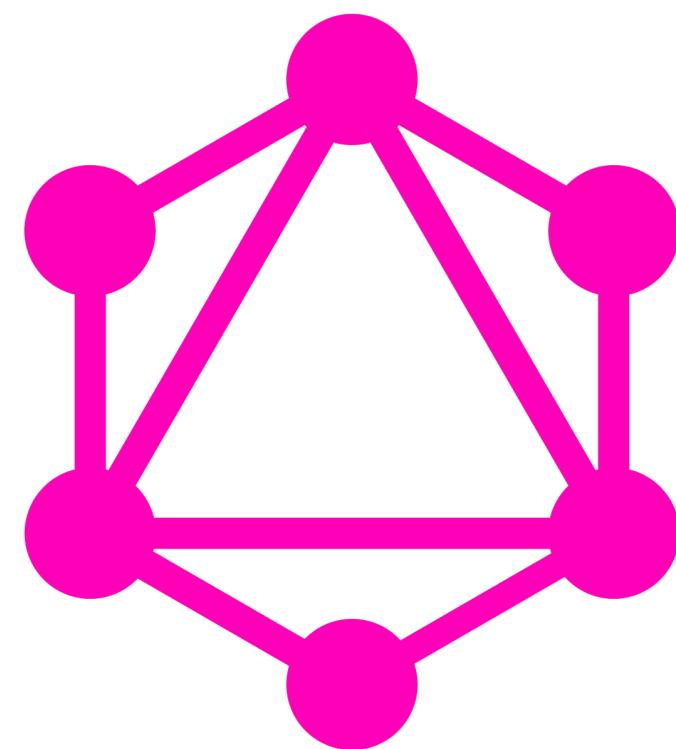
# 1.2 프리미엄 콘텐츠 스튜디오

N DEVIEW  
2021

## 요구사항



React

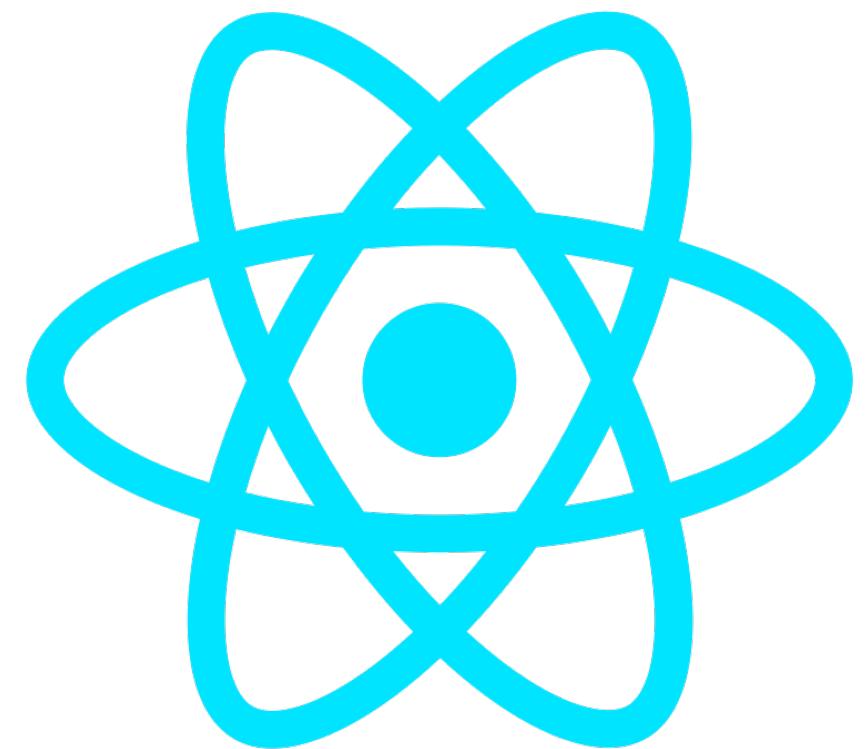


GraphQL

# 1.2 프리미엄 콘텐츠 스튜디오

N DEVIEW  
2021

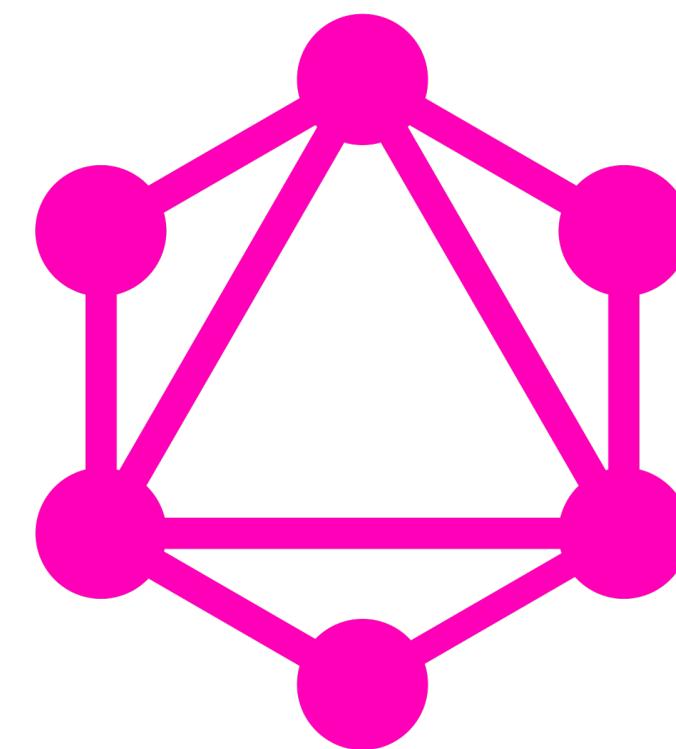
## 요구사항



React



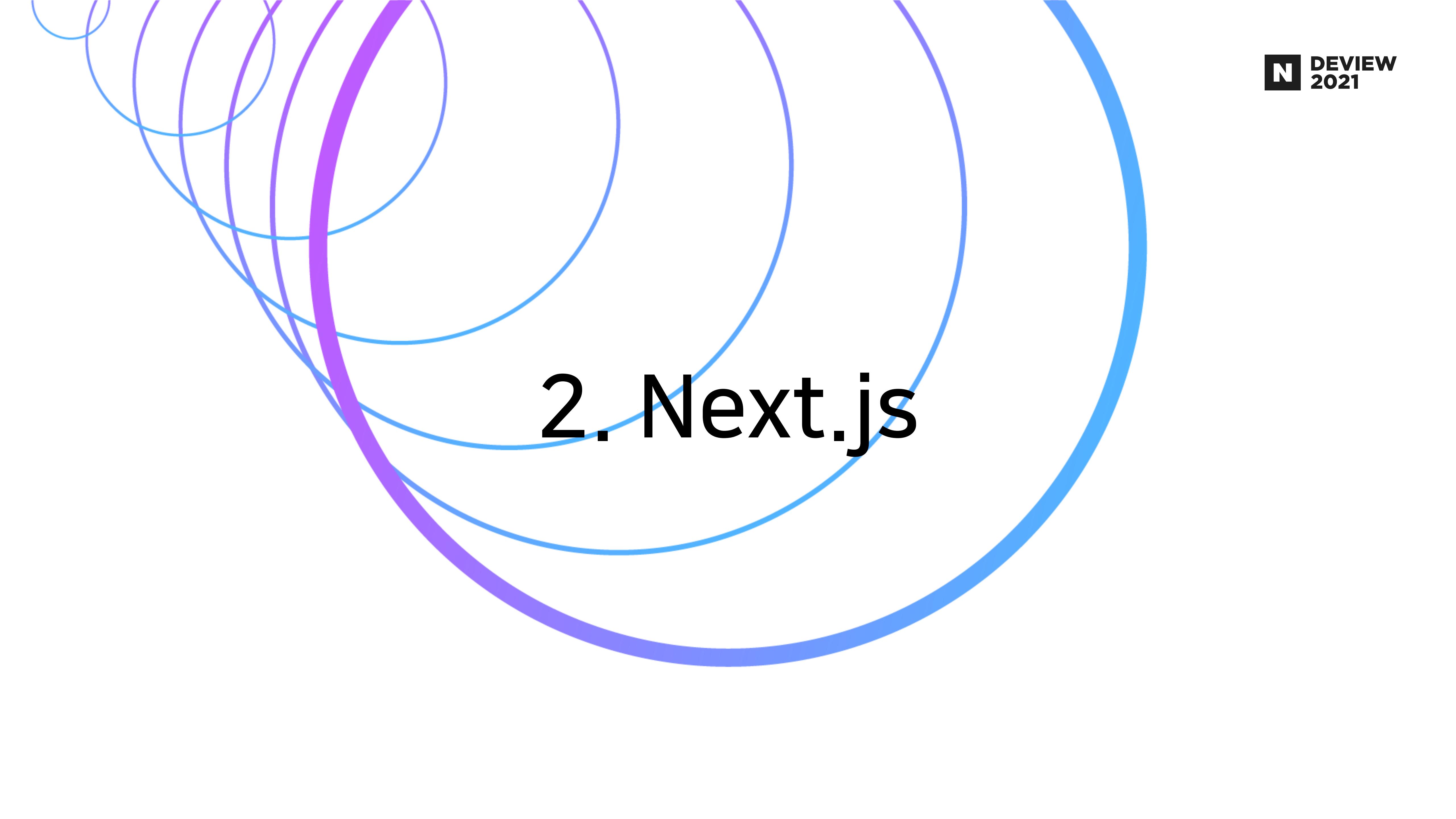
Next.js



GraphQL

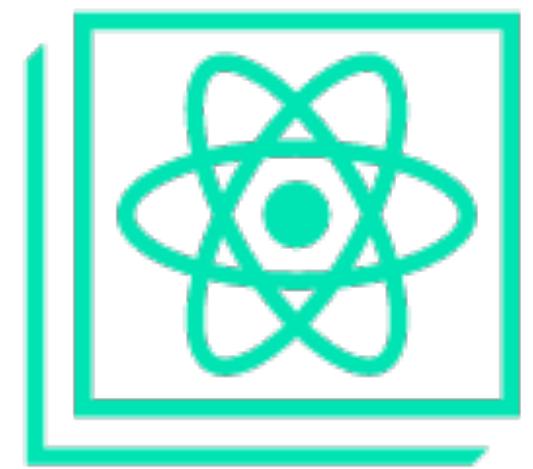


Apollo Client



## 2. Next.js

# 2.1 Create React App

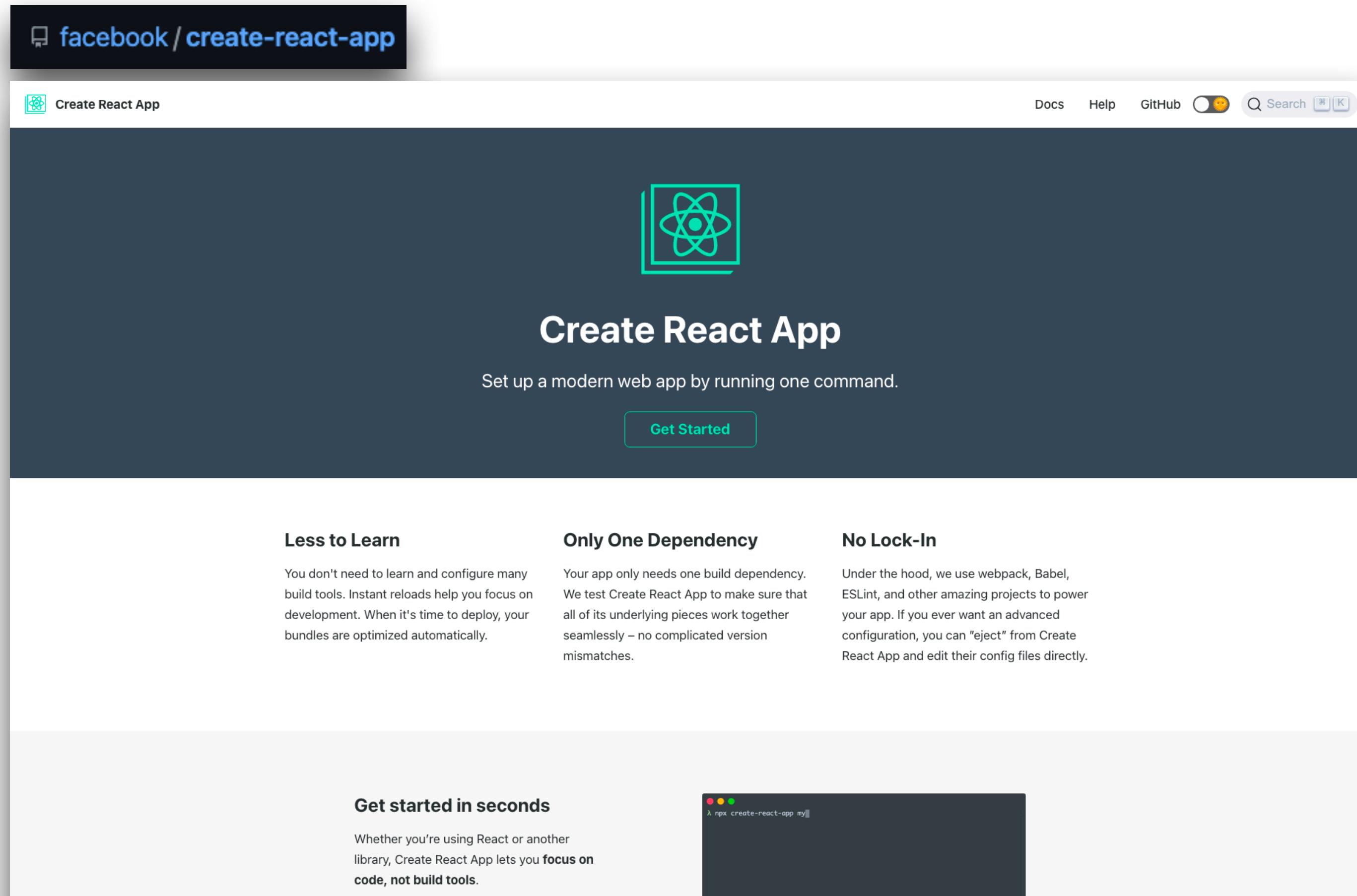


NEXT.JS

# 2.1 Create React App

## 소개

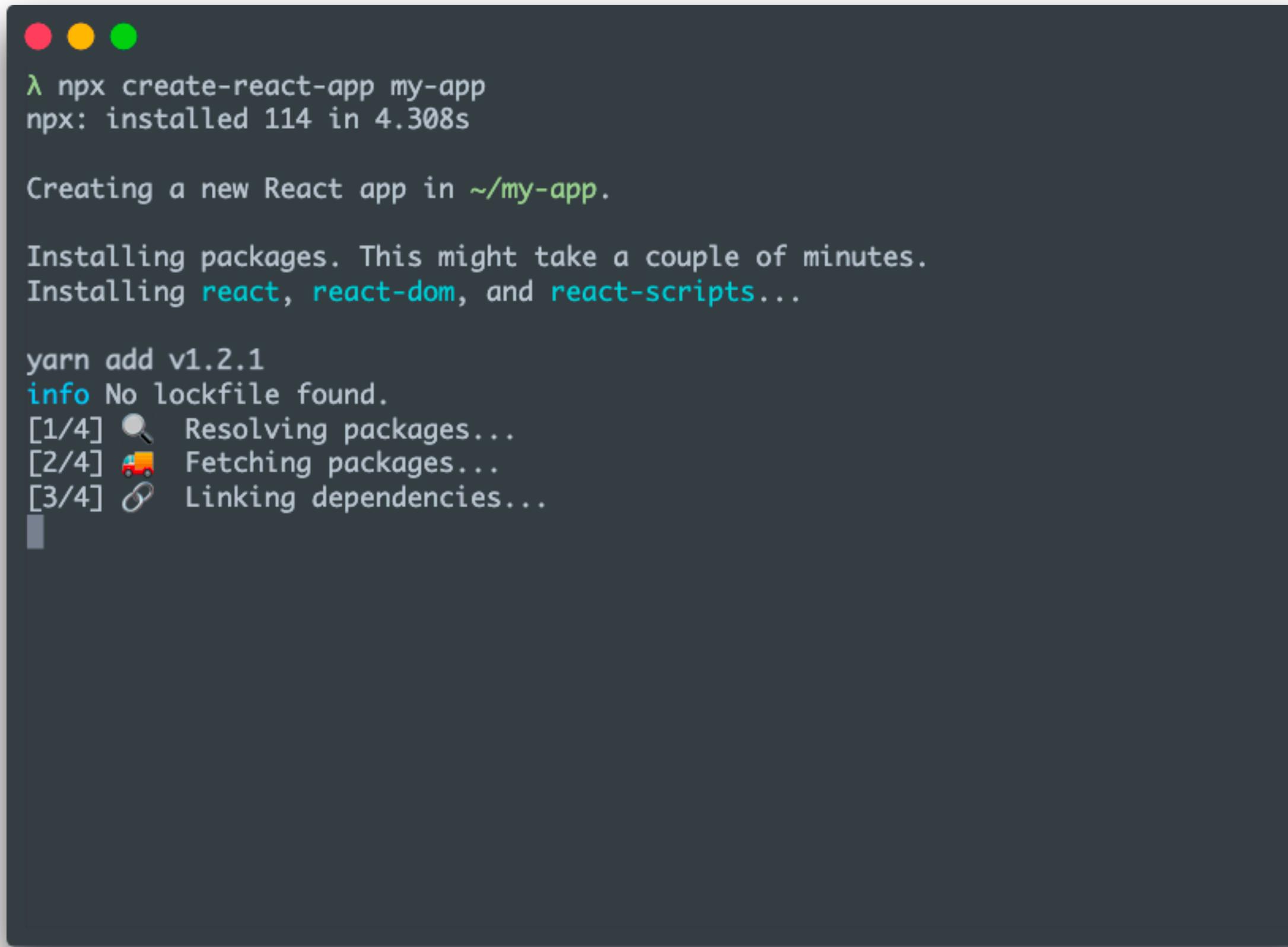
- React 팀에서 생성, 유지, 관리하는 React Framework



The screenshot shows the official Create React App website. At the top, there's a header with the URL "facebook/create-react-app". Below the header, the main title "Create React App" is displayed with a large React logo icon. A subtext says "Set up a modern web app by running one command." followed by a "Get Started" button. The page then lists three benefits: "Less to Learn", "Only One Dependency", and "No Lock-In". Under "Less to Learn", it says: "You don't need to learn and configure many build tools. Instant reloads help you focus on development. When it's time to deploy, your bundles are optimized automatically." Under "Only One Dependency", it says: "Your app only needs one build dependency. We test Create React App to make sure that all of its underlying pieces work together seamlessly – no complicated version mismatches." Under "No Lock-In", it says: "Under the hood, we use webpack, Babel, ESLint, and other amazing projects to power your app. If you ever want an advanced configuration, you can "eject" from Create React App and edit their config files directly." At the bottom left, there's a section titled "Get started in seconds" with a note about focusing on code. On the right side, there's a "Create React App" logo and the text "공식 홈페이지".

# 2.1 Create React App

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

## 😊 단 하나의 의존성, react-scripts

- 의존성간의 버전 미스매치 해결
- 프로젝트 단순화

Create React App  
package.json

```
{  
  "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"  
    ]  
  }  
}
```

# 2.1 Create React App

## 😊 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.

## 2.1 Create React App

 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

😢 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

😢 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 전  
Create React App  
package.json

eject



의존성 64개

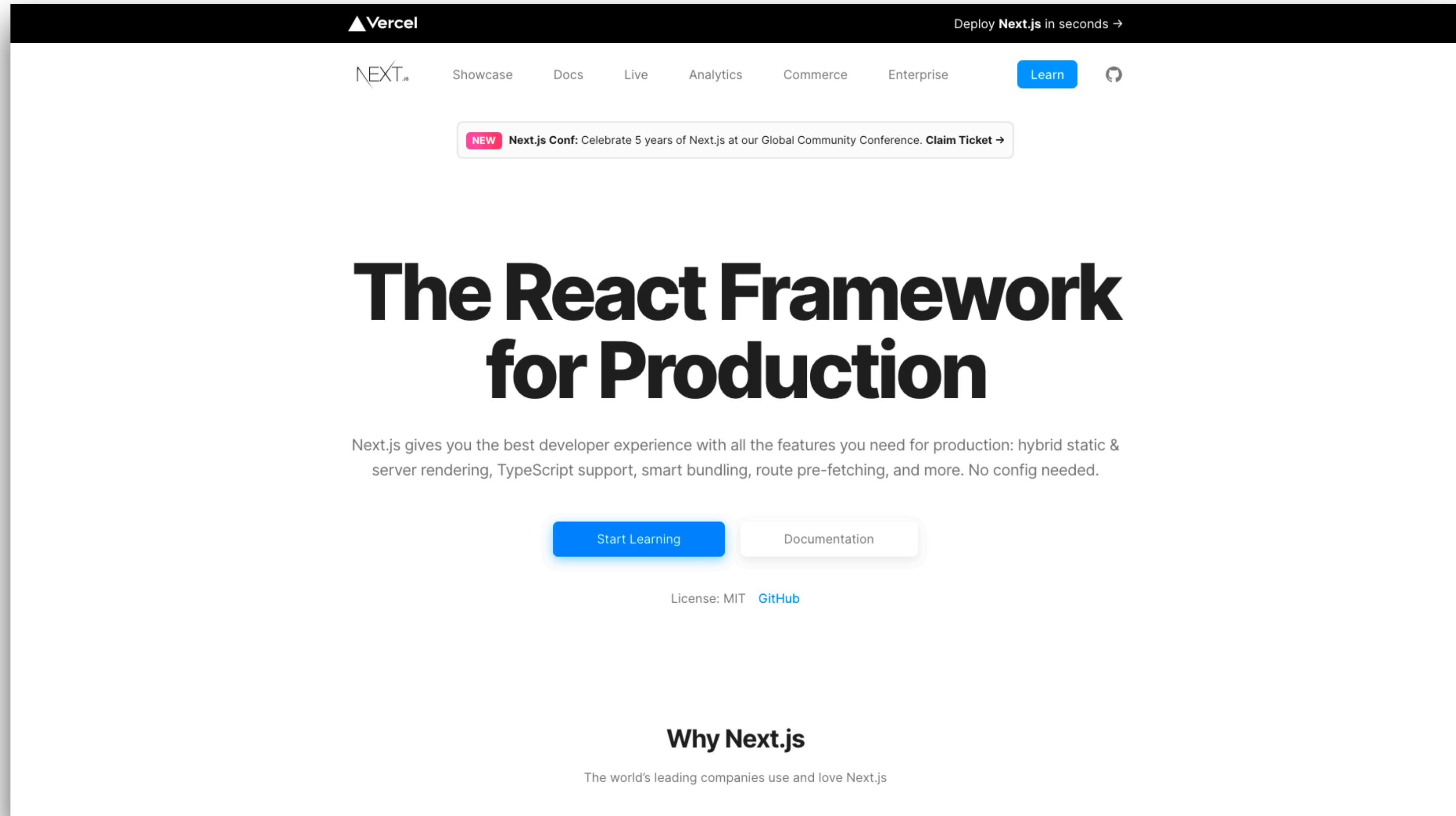
```
{
  "name": "test",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "17.0.2",
    "react-dom": "17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "1.0.1"
  },
  "scripts": {
    "start": "node scripts/start.js",
    "build": "node scripts/build.js",
    "test": "node scripts/test.js"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "resolutions": {
    "react": "17.0.2",
    "react-dom": "17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "1.0.1"
  },
  "devDependencies": {
    "react": "17.0.2",
    "react-dom": "17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "1.0.1"
  },
  "scripts": {
    "start": "node scripts/start.js",
    "build": "node scripts/build.js",
    "test": "node scripts/test.js"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

eject 후  
Create React App  
package.json

# 2.2 Next.js

## 소개

- Vercel 사에서 생성, 유지, 관리하는 React Framework



The screenshot shows the official Next.js website. At the top, there's a black header bar with the Vercel logo and a 'Deploy Next.js in seconds' button. Below the header, the Next.js logo is on the left, followed by navigation links: Showcase, Docs, Live, Analytics, Commerce, Enterprise, Learn (which is highlighted in blue), and a GitHub icon. A red 'NEW' badge with the text 'Next.js Conf: Celebrate 5 years of Next.js at our Global Community Conference. Claim Ticket →' is visible. The main title 'The React Framework for Production' is prominently displayed in large, bold, black font. Below the title, a subtitle explains: 'Next.js gives you the best developer experience with all the features you need for production: hybrid static & server rendering, TypeScript support, smart bundling, route pre-fetching, and more. No config needed.' Two buttons are present: 'Start Learning' (blue) and 'Documentation' (white). At the bottom, it says 'License: MIT' and has a link to 'GitHub'. The footer contains the text 'Why Next.js' and 'The world's leading companies use and love Next.js'. On the right side of the footer, there's a vertical column with the text 'Next.js' and '공식 홈페이지'.

# 2.2 Next.js

## Pre-rendering

- Static Generation와 Server-side Rendering, 두 가지 형태의 Pre-rendering

**SSG**

`getStaticProps`

`getStaticPaths`

**SSR**

`getServerSideProps`

# 2.2 Next.js

## next.config.js

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

```
module.exports = {
  webpack(config) {
    config.plugins.push(
      new UitSpritesmithWebpackPlugin({
        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',
  }

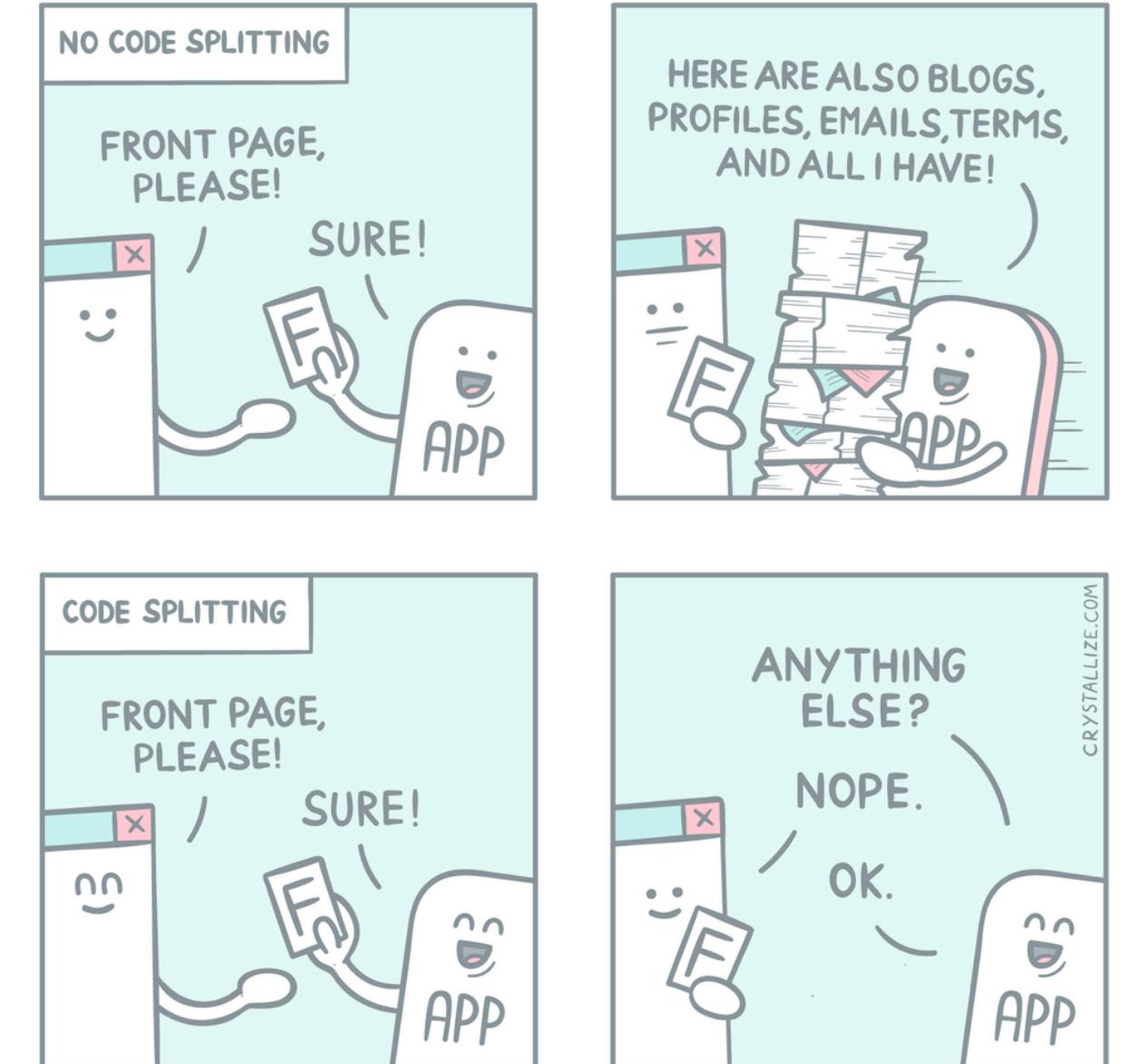
  return config
},
}
```

next.config.js

# 2.2 Next.js

## 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 })
```

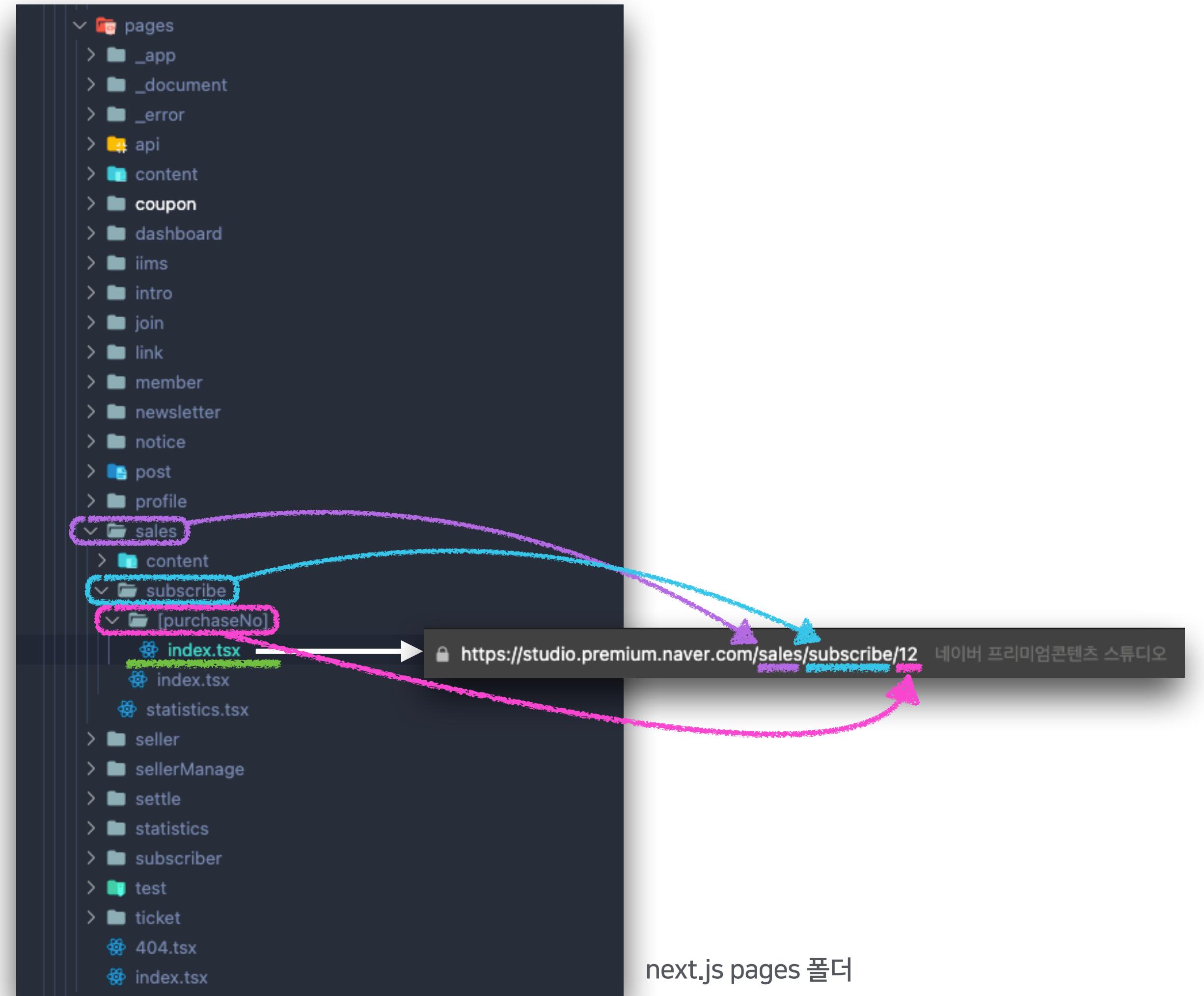
Dynamic Import React Component

# 2.2 Next.js

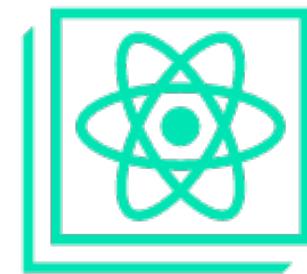
## Routing

- 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



# 2.3 Create React App ➡️ Next.js



자체 지원 SSR 없음  
eject시 복잡성 증가



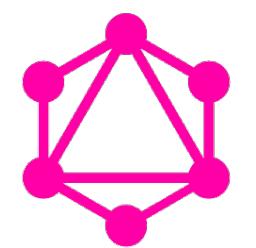
~~NEXT~~.JS

Pre-rendering  
next.config.js  
+ Routing  
+ Code-splitting

## 3. Apollo Client

# 3.1 REST

{ REST }



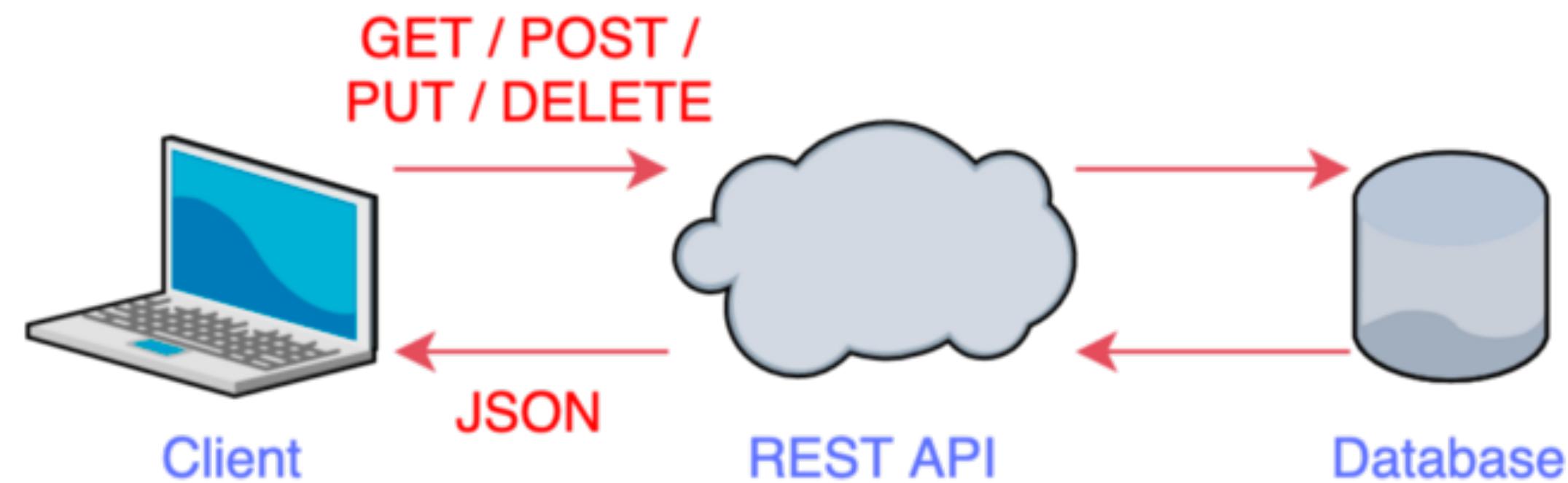
GraphQL

A pink icon representing a GraphQL schema, showing a pentagonal shape with nodes and edges.

# 3.1 REST

## 소개

- Web API를 설계하는 데 가장 보편적으로 사용하는 아키텍처



# 3.1 REST

## 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

## 😢 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

## 😢 Overfetching

- 불필요한 데이터가 있는 경우, 필요한 부분만 거르는 작업 필요

/sessions/29



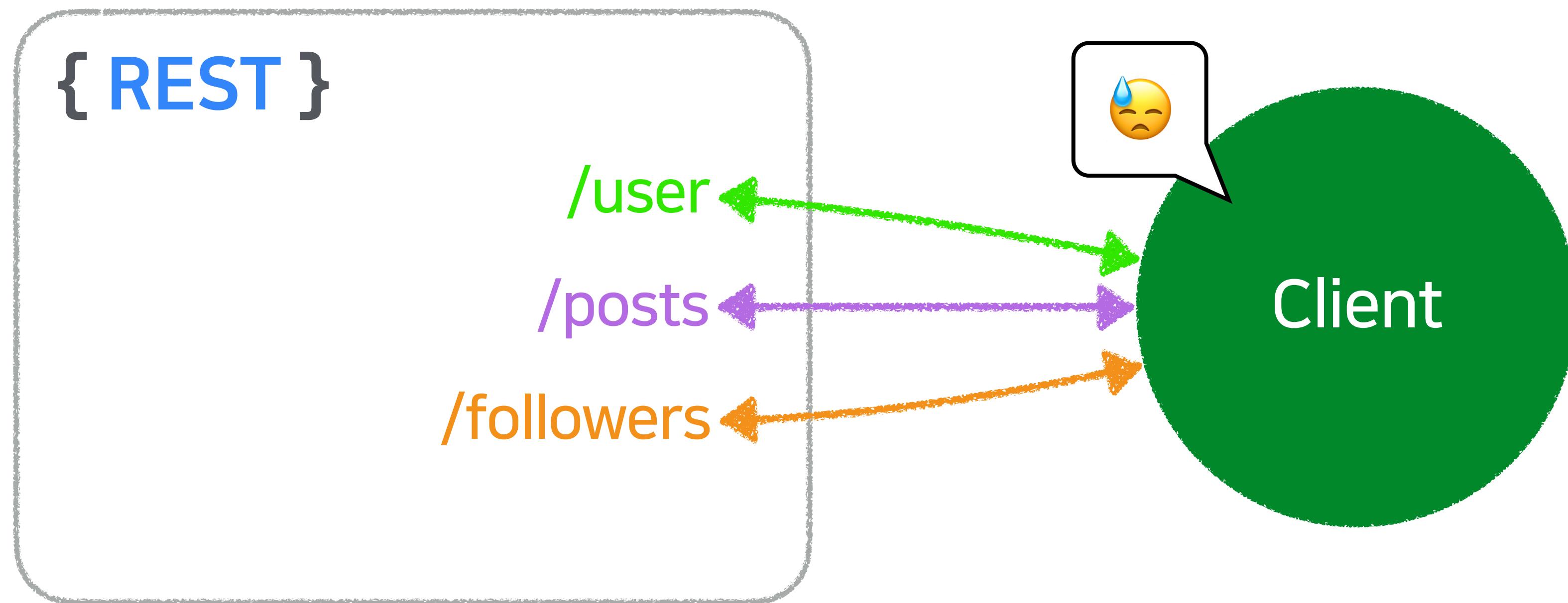
```
Response
```

```
{  
  "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

## 😢 Underfetching

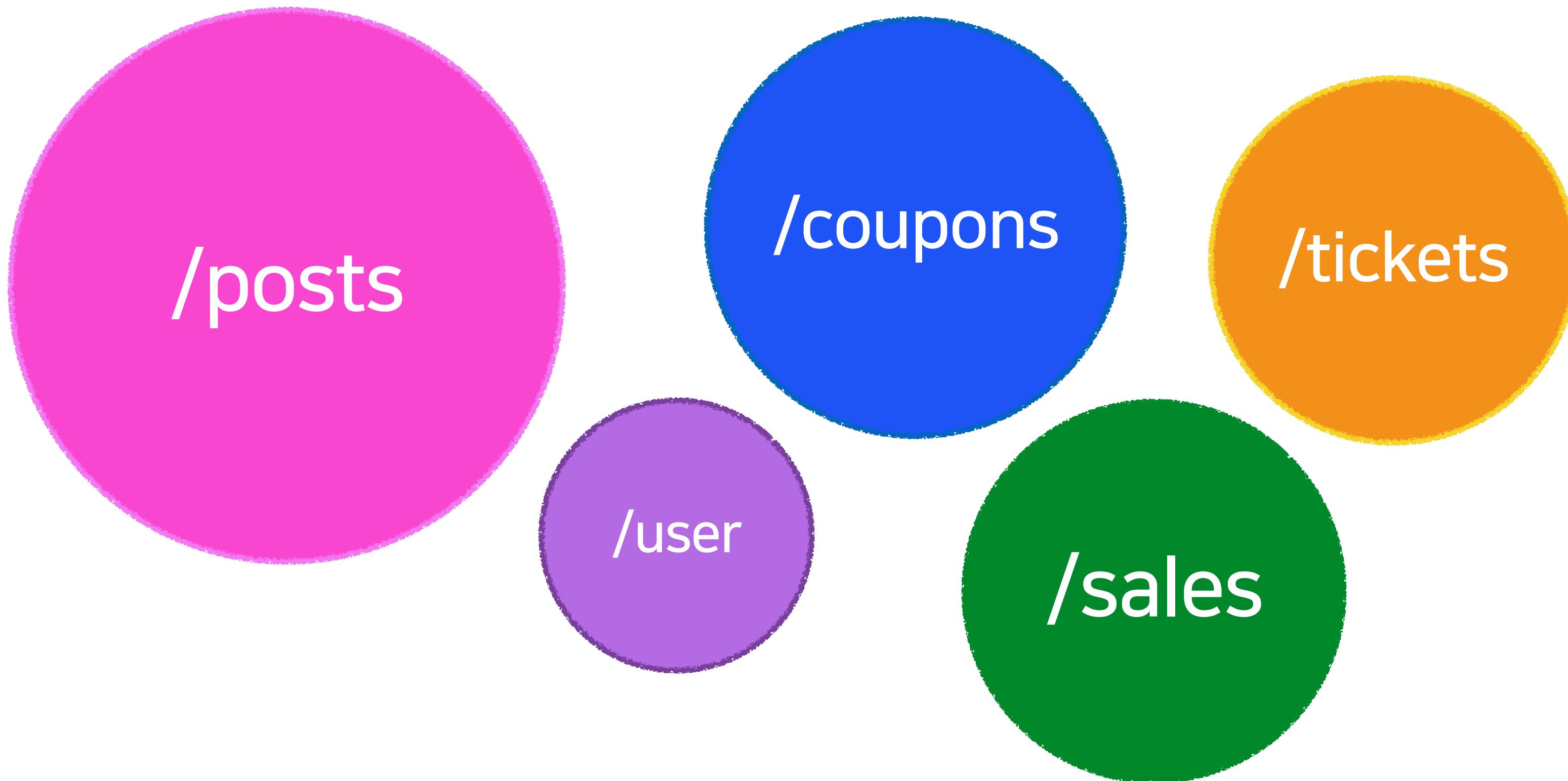
- 한 페이지에 여러 요청이 필요한 경우 각각의 요청을 따로 보내야한다.



# 3.1 REST

## 😢 무수한 엔드포인트

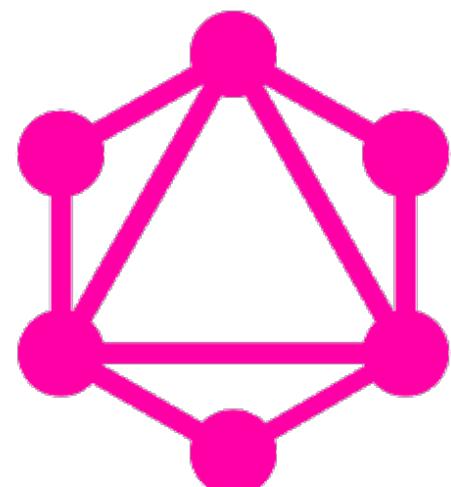
- 자원이 많아질수록 더 많아지는 엔드포인트



# 3.2 GraphQL

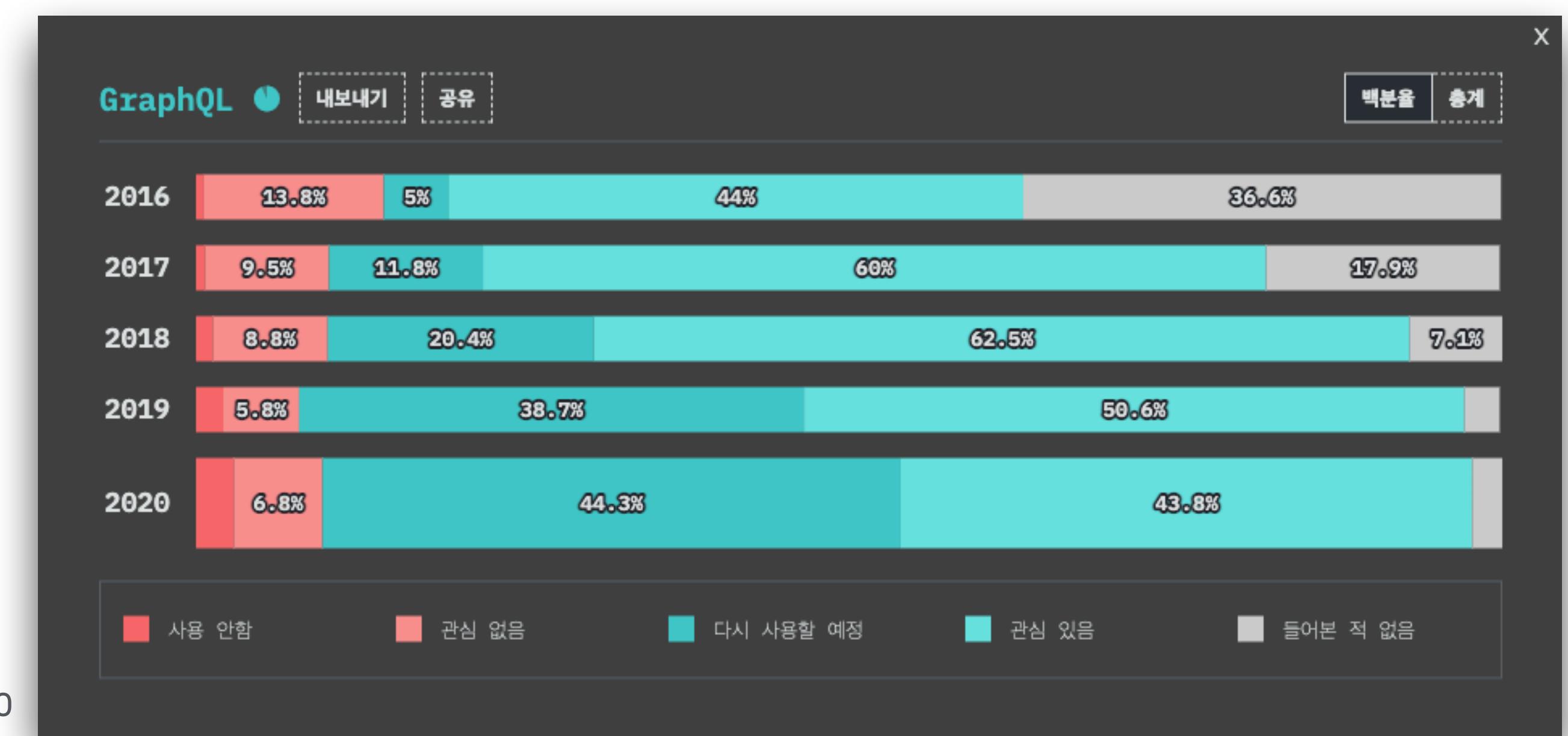
## 소개

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



# GraphQL

State of JS 2020



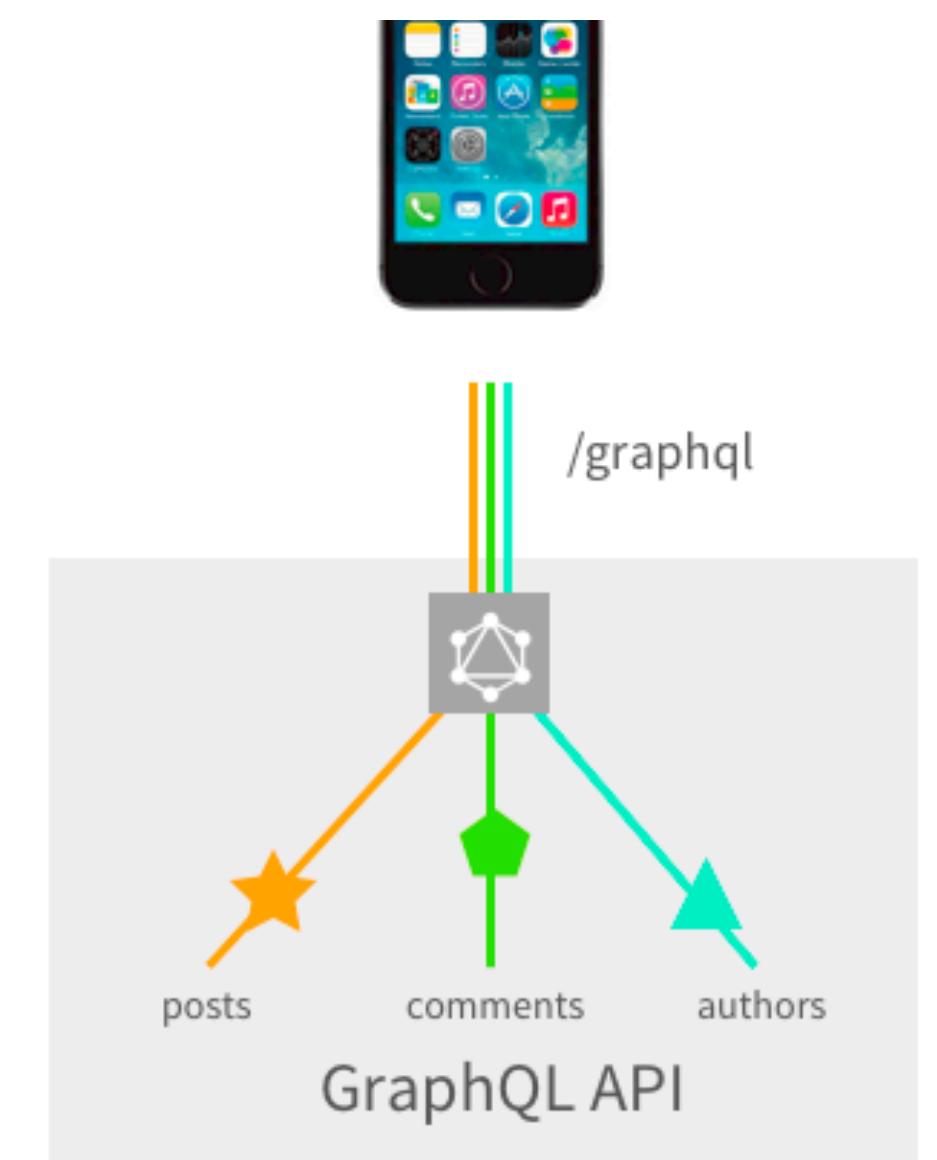
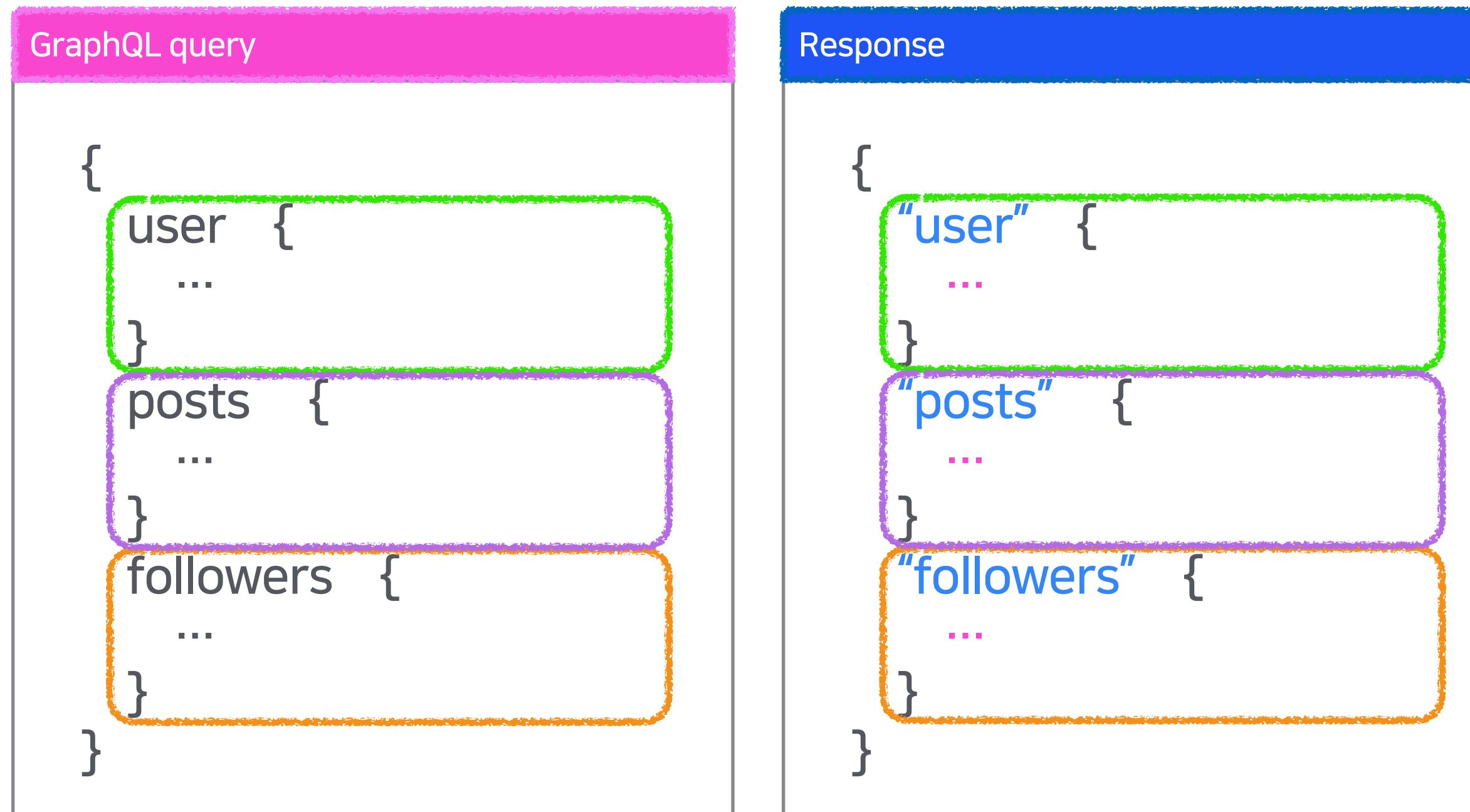
# 3.2 GraphQL

## 😊 필요한 데이터만 쑥쑥

GraphQL query	Response
<pre>{   session {     title     category     content     speaker {       name     }   } }</pre>	<pre>{   "session": {     "title": "Apollo, Next.js와 함께 리액트 개발의 Next Level로 가자!",     "category": "for Juniors",     "content": "Apollo + Next.js = Next Level",     "speaker": {       "name": "이건"     }   } }</pre>

# 3.2 GraphQL

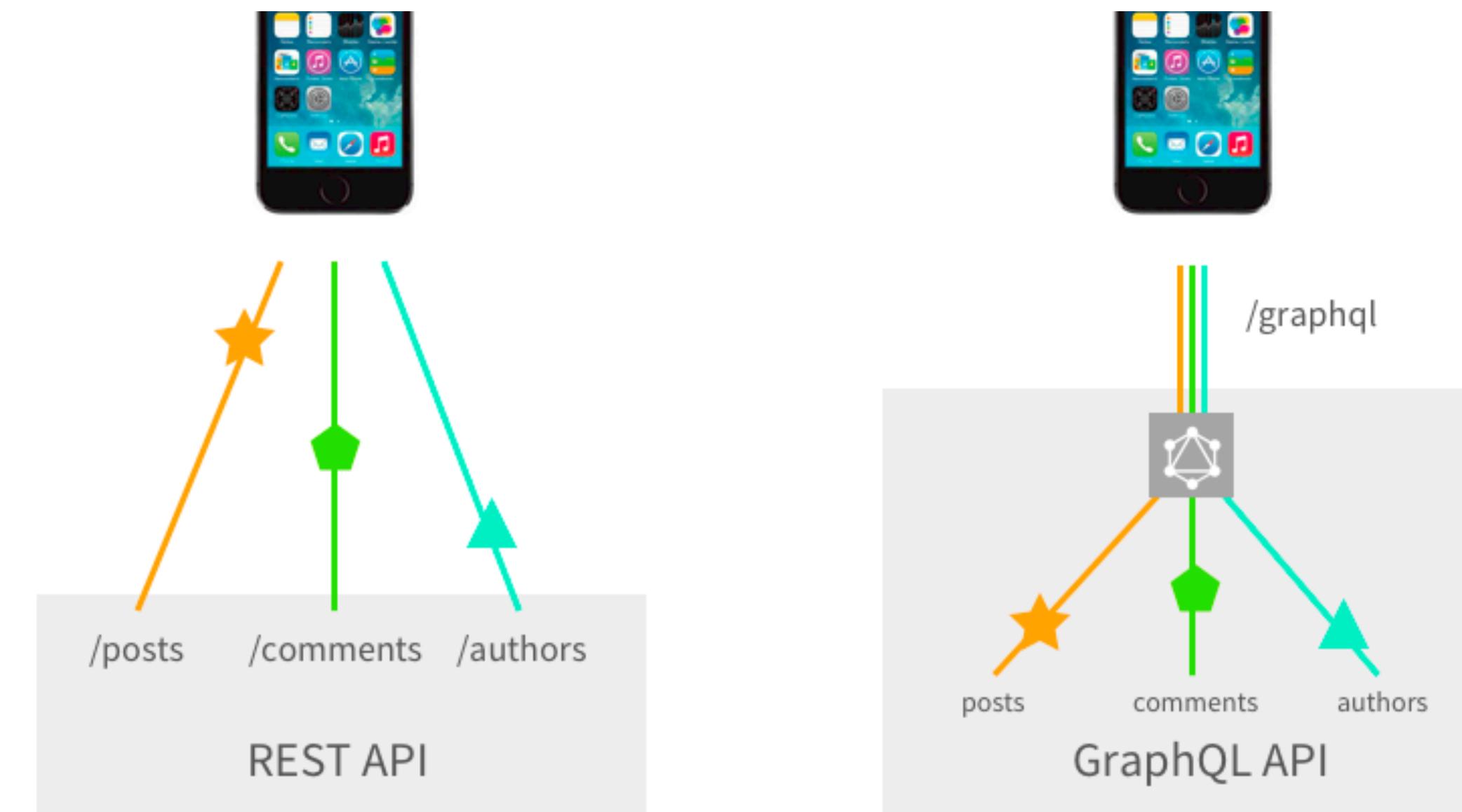
😊 여러 요청을 한 번에



# 3.2 GraphQL

## 😊 단 하나의 엔드포인트

- 하나의 엔드포인트에 POST로 모든 요청 가능



# 3.2 GraphQL

## 😊 GraphiQL

- GraphQL 서버를 자유롭게 테스트 해볼 수 있는 도구

The screenshot shows the GraphiQL interface with the following components:

- GraphQL Editor:** On the left, a code editor displays a GraphQL query with syntax highlighting for fragments, variables, and schema fields.
- Results Panel:** The main area shows the JSON response from the GraphQL server. It includes a list of planets with their names, climates, and a list of residents (people) for each planet, including their names, birth years, and hair colors.
- Schema Browser:** On the right, a sidebar titled "Root" lists various GraphQL schema fields and connections, such as "allVehicles", "vehicle", "PersonVehiclesConnection", and "Vehicle".
- Query Variables:** At the bottom left, there is a section for "QUERY VARIABLES" containing the value for the variable "firstN".

```
graph TD; subgraph Query [GraphQL Editor]; query1["query Planets($firstN: Int){\n    allPlanets(first: $firstN){\n        totalCount\n        planets{\n            ...PlanetInfo\n        }\n    }\n}"]; end; subgraph Results [Results Panel]; results1["{\\n        \"data\": {\\n            \"allPlanets\": {\\n                \"totalCount\": 61,\\n                \"planets\": [\\n                    {\\n                        \"id\": \"cGxhbmV0czox\",\\n                        \"name\": \"Tatooine\",\\n                        \"climates\": [\\n                            \"arid\"\\n                        ],\\n                        \"gravity\": \"1 standard\",\\n                        \"population\": 200000,\\n                        \"residentConnection\": {\\n                            \"residents\": [\\n                                {\\n                                    \"name\": \"Luke Skywalker\",\\n                                    \"birthYear\": \"19BBY\",\\n                                    \"hairColor\": \"blond\"\\n                                },\\n                                {\\n                                    \"name\": \"C-3PO\",\\n                                    \"birthYear\": \"112BBY\",\\n                                    \"hairColor\": \"n/a\"\\n                                },\\n                                {\\n                                    \"name\": \"Darth Vader\",\\n                                    \"birthYear\": \"41.9BBY\",\\n                                    \"hairColor\": \"none\"\\n                                },\\n                                {\\n                                    \"name\": \"Owen Lars\",\\n                                    \"birthYear\": \"52BBY\",\\n                                    \"hairColor\": \"brown, grey\"\\n                                },\\n                                {\\n                                    \"name\": \"Beru Whitesun lars\",\\n                                    \"birthYear\": \"47BBY\",\\n                                    \"hairColor\": \"brown\"\\n                                },\\n                                {\\n                                    \"name\": \"R5-D4\",\\n                                    \"birthYear\": \"unknown\"\\n                                }\\n                            ]\\n                        }\\n                    }\\n                ]\\n            }\\n        }\\n    }\\n}"]; end; subgraph Schema [Schema Browser]; schema1["< Schema Root\\n\\nQ v\\n\\nallVehicles\\nvehicle\\n\\nOTHER RESULTS\\n\\nPersonVehiclesConnection\\nPersonVehiclesEdge\\nVehicle\\nVehiclePilotsConnection\\nVehiclePilotsEdge\\nVehicleFilmsConnection\\nVehicleFilmsEdge\\nFilmVehiclesConnection\\nFilmVehiclesEdge\\nVehiclesConnection\\nVehiclesEdge\\n__InputValue\\n__EnumValue\\n__Directive\\n__DirectiveLocation\\nPageinfo.hasPreviousPage\\nFilm.vehicleConnection\\nSpecies.averageHeight\\nSpecies.averageLifespan\\nPlanet.gravity\\nPerson.vehicleConnection\\nStarship.hyperdriveRating\\n\\nGraphiQL"]; end;
```

GraphQL Editor Content:

```
1 fragment PeopleInfo on PlanetResidentsConnection {  
2   residents {  
3     name  
4     birthYear  
5     hairColor  
6   }  
7 }  
8  
9 fragment PlanetInfo on Planet {  
10  id  
11  name  
12  climates  
13  gravity  
14  population  
15  residentConnection {  
16    ...PeopleInfo  
17  }  
18 }  
19  
20 query Planets($firstN: Int){  
21   allPlanets(first: $firstN){  
22     totalCount  
23     planets {  
24       ...PlanetInfo  
25     }  
26   }  
27 }  
28 }
```

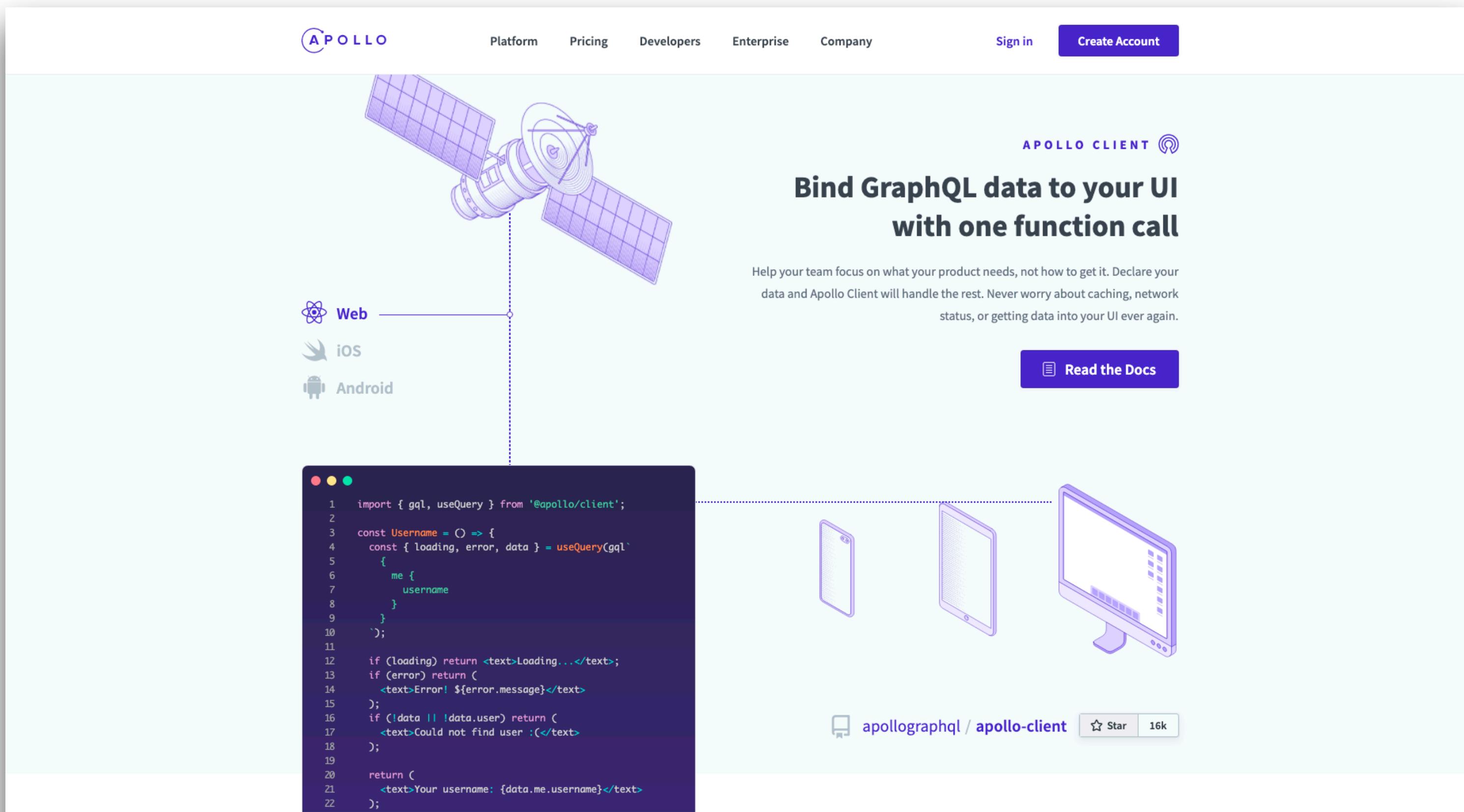
Query Variables:

```
1 {  
2   "firstN": 2  
3 }
```

# 3.3 Apollo Client

## 소개

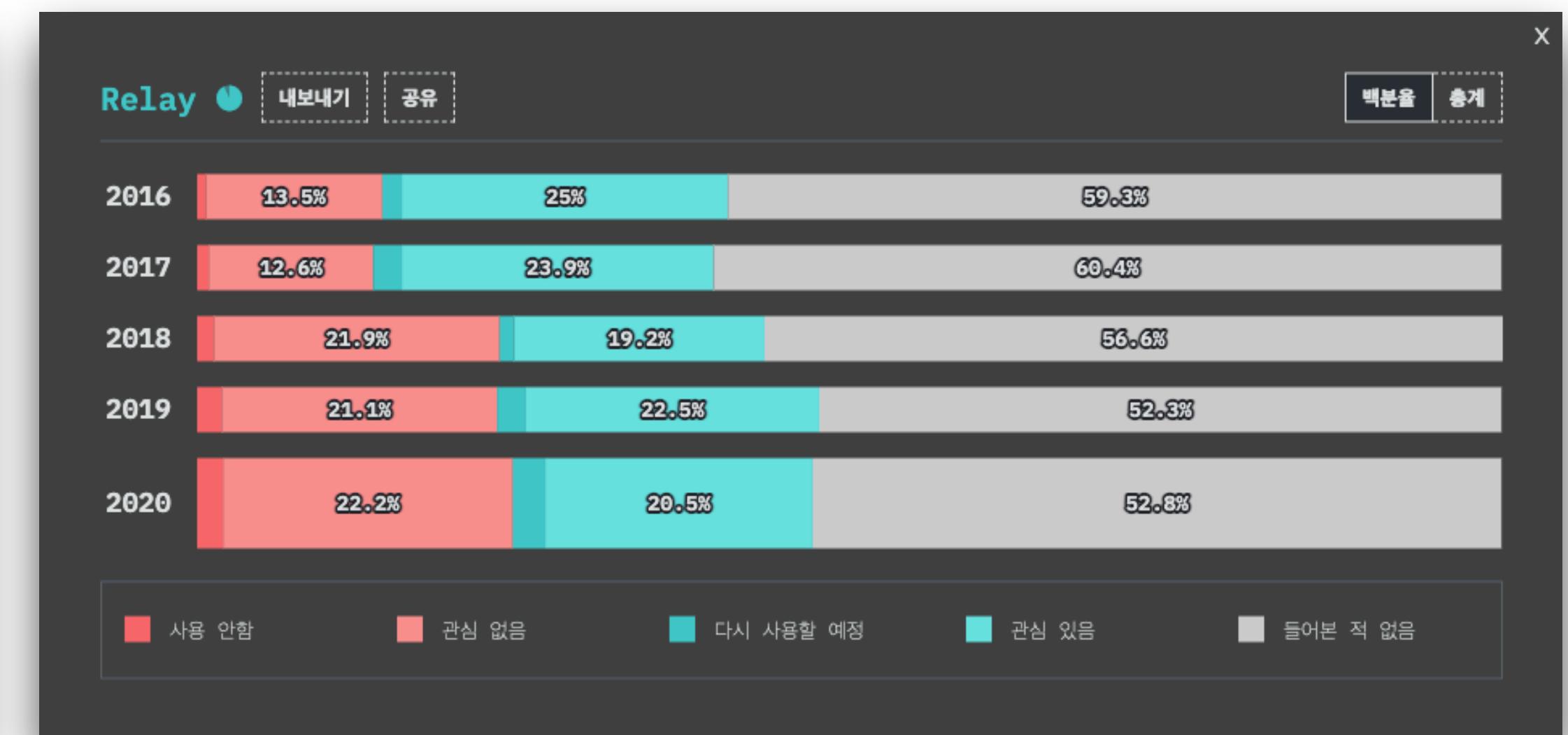
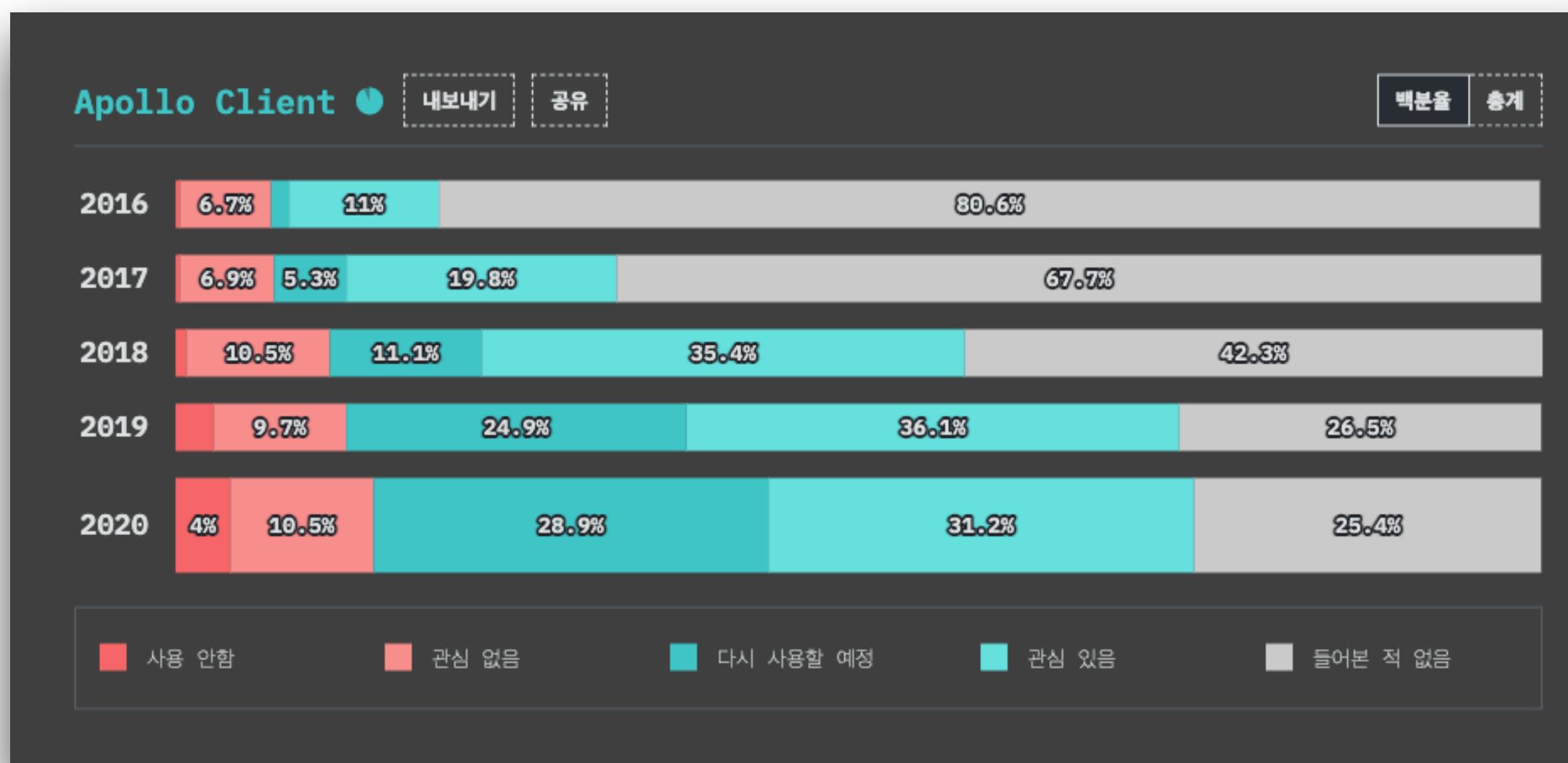
- Apollo Graph에서 제작한 GraphQL 클라이언트



# 3.3 Apollo Client

## 소개

- Apollo Graph에서 제작한 GraphQL 클라이언트



State of JS 2020

# 3.3 Apollo Client

😊 최신 리액트를 위해 디자인된 라이브러리

- useQuery
- useMutation

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

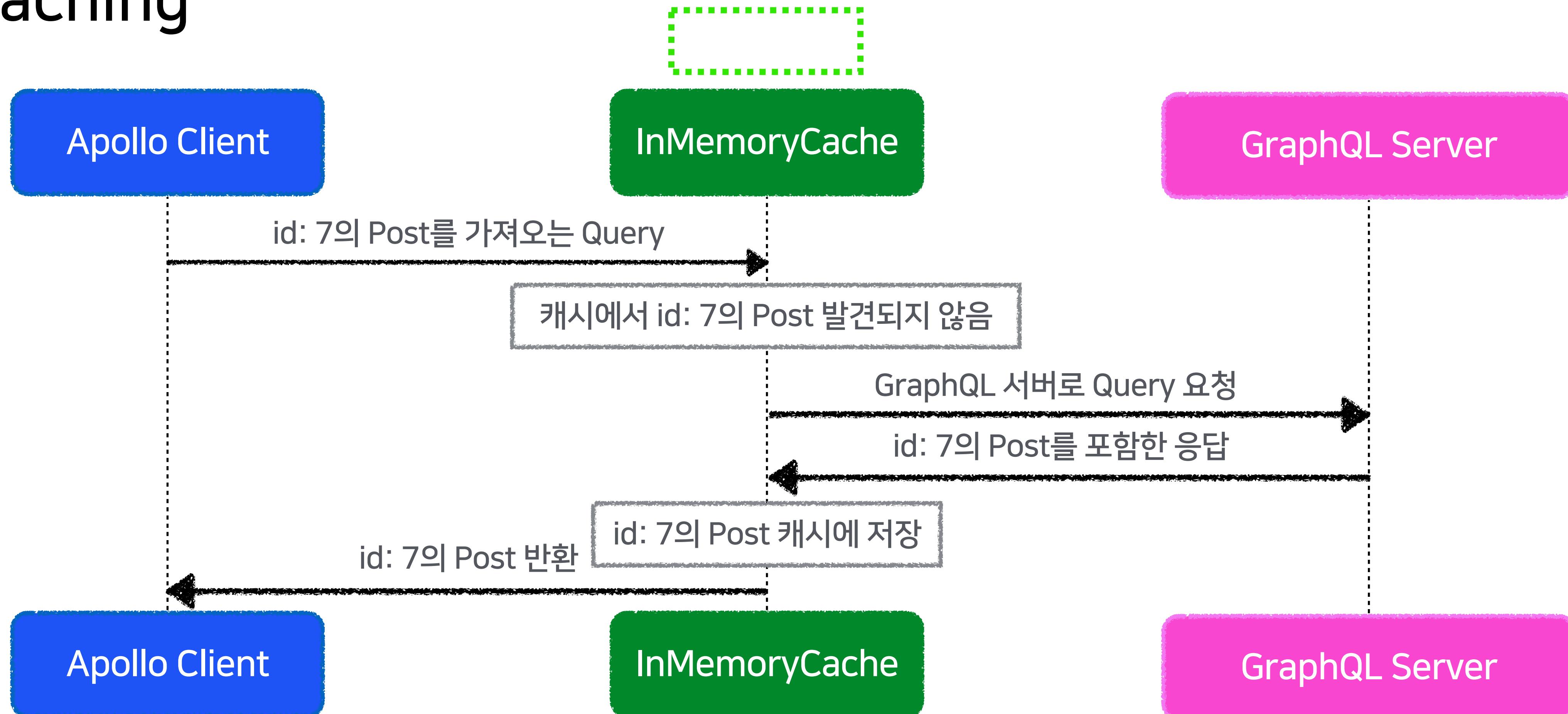
useQuery

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

useMutation

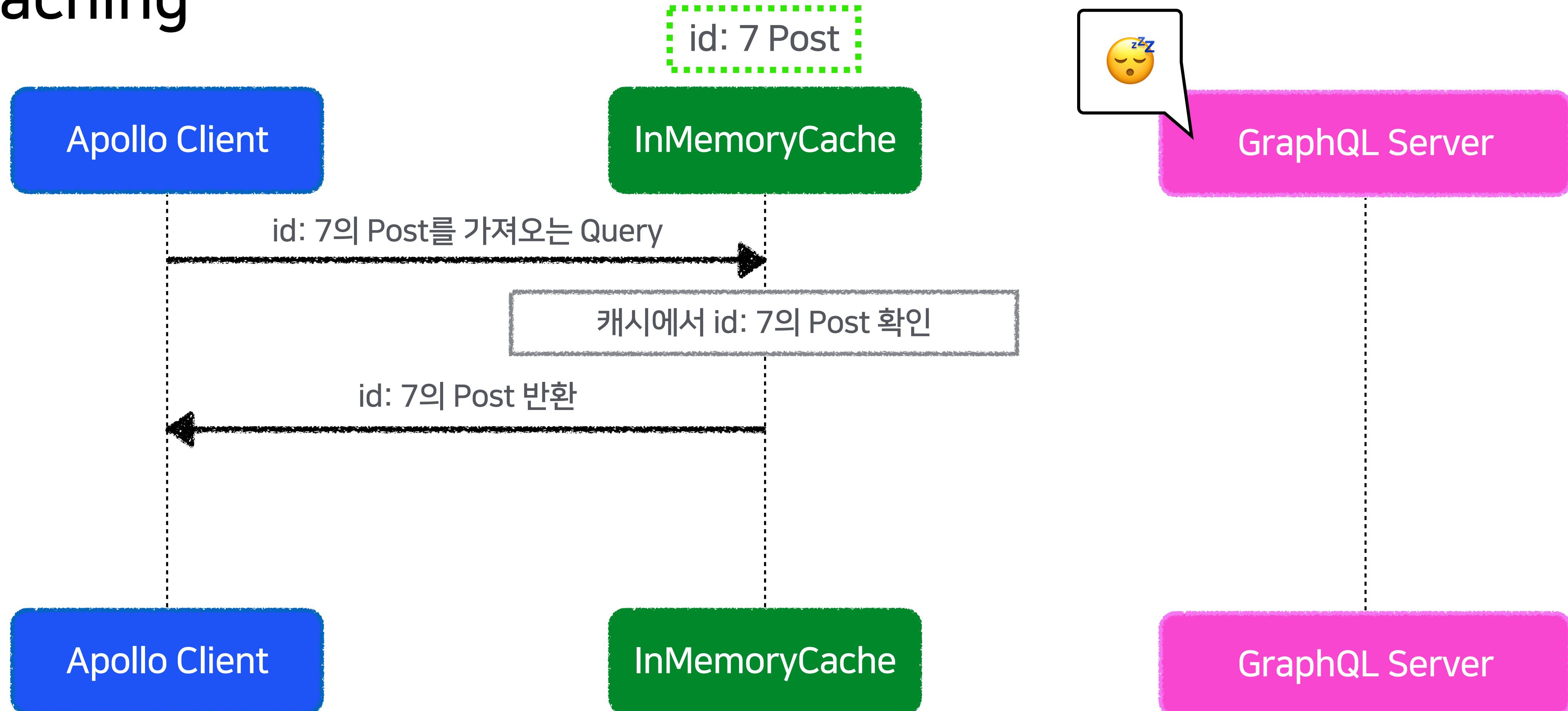
# 3.3 Apollo Client

## 😊 Caching



# 3.3 Apollo Client

## 😊 Caching



# 3.3 Apollo Client

## 😊 Caching

### Supported fetch policies

NAME	DESCRIPTION
<code>cache-first</code>	Apollo Client first executes the query against the cache. If <i>all</i> 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.
<code>cache-only</code>	Apollo Client executes the query <i>only</i> against the cache. It never queries your server in this case.  A <code>cache-only</code> query throws an error if the cache does not contain data for all requested fields.
<code>cache-and-network</code>	Apollo Client executes the full query against both the cache <i>and</i> 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.
<code>network-only</code>	Apollo Client executes the full query against your GraphQL server, <i>without</i> first checking the cache. The query's result <i>is</i> stored in the cache.  Prioritizes consistency with server data, but can't provide a near-instantaneous response when cached data is available.
<code>no-cache</code>	Similar to <code>network-only</code> , except the query's result <i>is not</i> stored in the cache.
<code>standby</code>	Uses the same logic as <code>cache-first</code> , except this query does <i>not</i> automatically update when underlying field values change. You can still <i>manually</i> update this query with <code>refetch</code> and <code>updateQueries</code> .

`useQuery`  
`fetchPolicy`

# 3.4 REST API ➡️ GraphQL

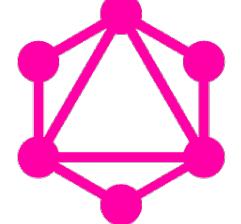
{ REST }

OverFetching

UnderFetching

무수한 엔드포인트



 GraphQL

필요한 데이터만 쑥쑥  
여러 요청을 한 번에  
단일 엔드포인트

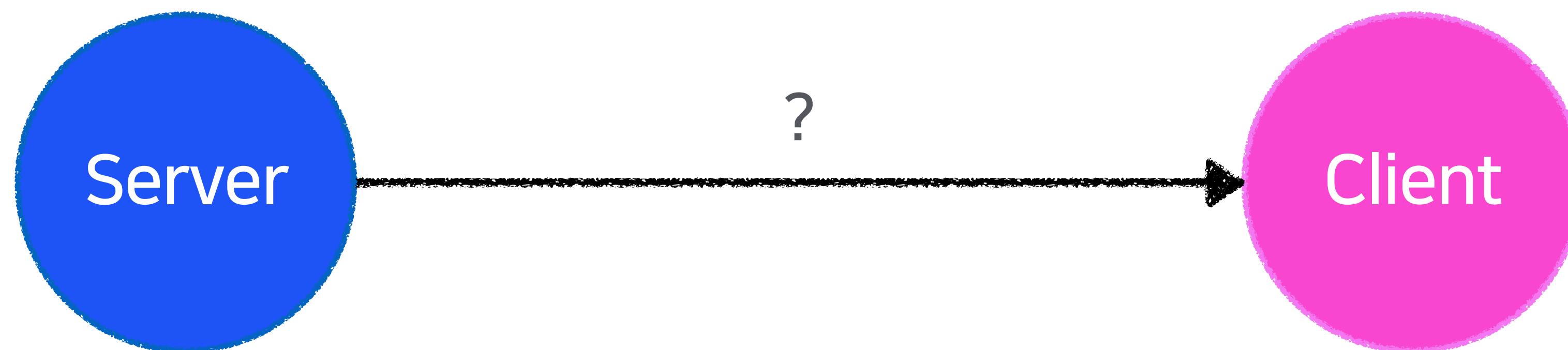
+ 리액트 통합성, Caching (with Apollo)

# 4. Troubleshooting & Tips

# 4.1 SSR with Apollo Client

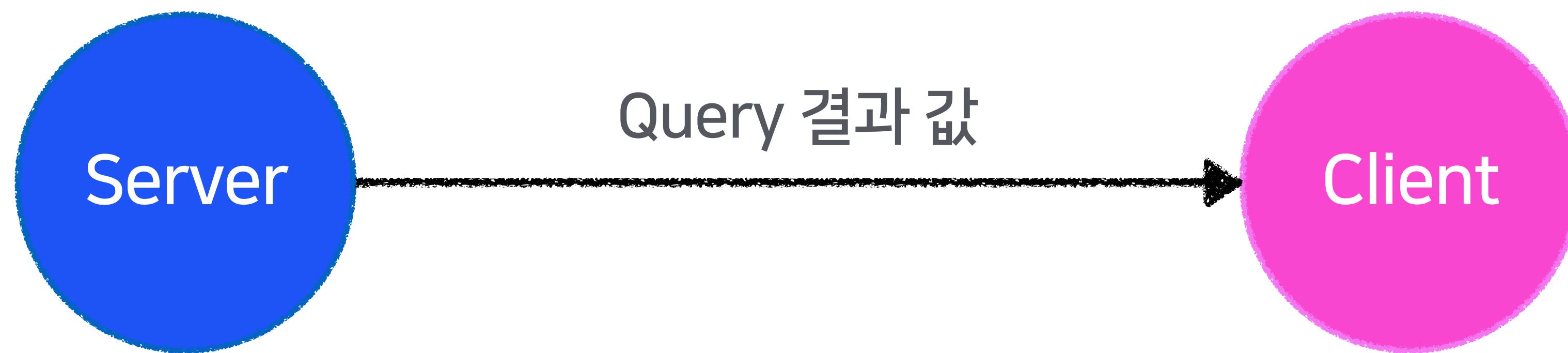
## 문제 상황

- 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

## 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()

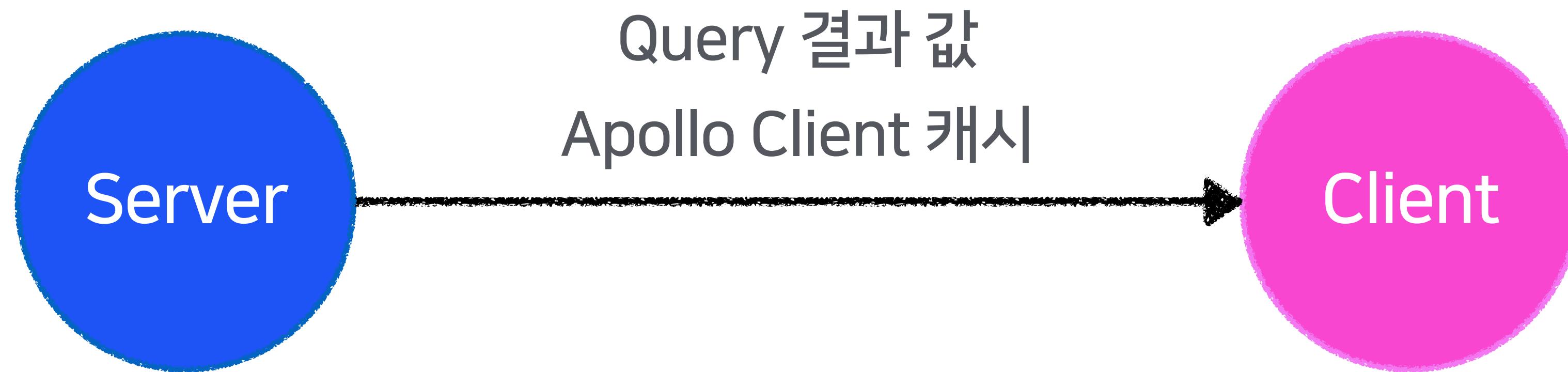
  const { data } = await apolloClient.query({
    query: ALL_POSTS_QUERY,
    variables: allPostsQueryVars,
  })

  const { allPosts } = data

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

# 4.1 SSR with Apollo Client

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()

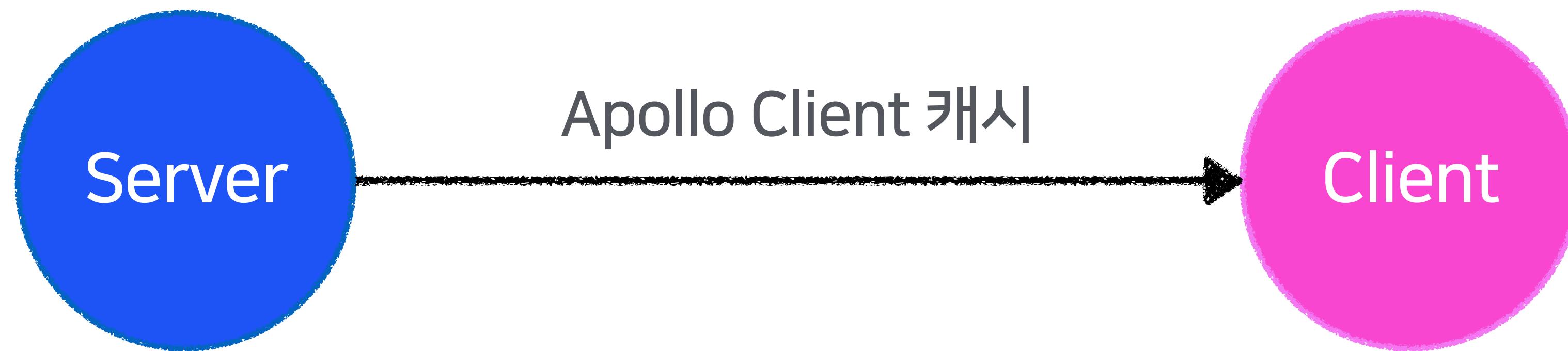
  const { data } = await apolloClient.query({
    query: ALL_POSTS_QUERY,
    variables: allPostsQueryVars,
  })

  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

## 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

## 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

### withAuthServerSideProps

- SSR 메소드인 getServerSideProps의 인증 로직을 처리할 HOF



# 4.2 인증 구현

## withAuthServerSideProps

- SSR 메소드인 getServerSideProps의 인증 로직을 처리할 HOF

```
● ● ● sps-premium-cms - meteredPaywall.tsx

1  export const getServerSideProps: GetServerSideProps<{}, {}> = withAuthServerSideProps(
2    async (ctx, apolloClient, user) => {
3      const { authId, cpName, cpType, cpId, subId } = user
4
5      await apolloClient.query<
6        { channelMeteredPaywallConfig: Query['getChannelMeteredPaywallConfig'] },
7        QueryGetChannelMeteredPaywallConfigArgs & PremiumAuth & GeneralAuth
8      >({
9        query: GET_CHANNEL_METERED_PAYWALL_CONFIG,
10       variables: { cpName, cpType, cpId, subId },
11       context: { authId, svcType: SvcType.PREMIUM },
12     })
13
14     return {
15       props: {
16         initialApolloState: apolloClient.cache.extract(),
17       },
18     }
19   },
20 )
```

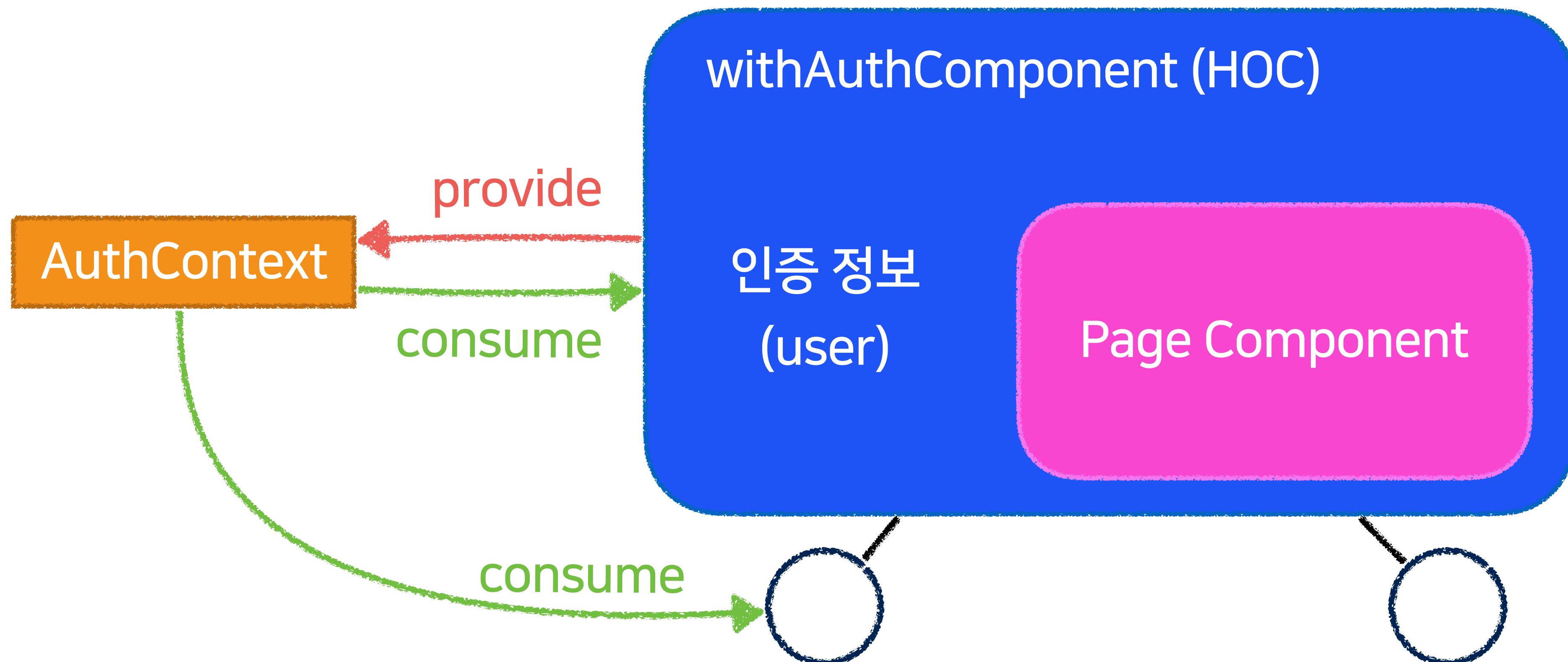
# 4.2 인증 구현

## 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   X
  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   X
  이용권 수 조회

  TYPE
  TicketCountResult

  ARGUMENTS
    cpType: CpType!
    cpName: String!
    channelName: String! (Red box)
    categoryTypes: [TicketCategoryType] = [CHANNEL_NORMAL]
      디폴트 값: CHANNEL_NORMAL
    startYmdt: String
      패턴: yyyyMMddHHmmss
    endYmdt: String
      패턴: yyyyMMddHHmmss
    saleStatuses: [SaleStatus] (Red box)
      디폴트 값: SALE
```

# 4.3 서버 스키마 동기화

해결: graphql-code-generator

The screenshot shows the homepage of The Guild's GraphQL Code Generator. The page features a central illustration of a pink laptop displaying a network graph, with blue blocks labeled 'A' and 'TS' representing code generation. Below the illustration, the text '{ GraphQL } code generator' is displayed. A subtext reads 'Generate code from your GraphQL schema and operations with a simple CLI'. A 'CLI VERSION V2.2.0' button is shown, along with 'TRY IT OUT LIVE' and 'VIEW DOCS' buttons. At the bottom, there's a 'Choose Live Example:' section with dropdown menus for 'Schema types' (set to 'Schema') and 'ts frontend backend' (set to 'ts'). Below this are three code editor panes: 'schema.graphql', 'operation.graphql', and 'codegen.yml'. The 'codegen.yml' pane shows configuration for 'types.ts' with code like:

```
generates:
  types.ts:
    plugins:
      - typescript
```

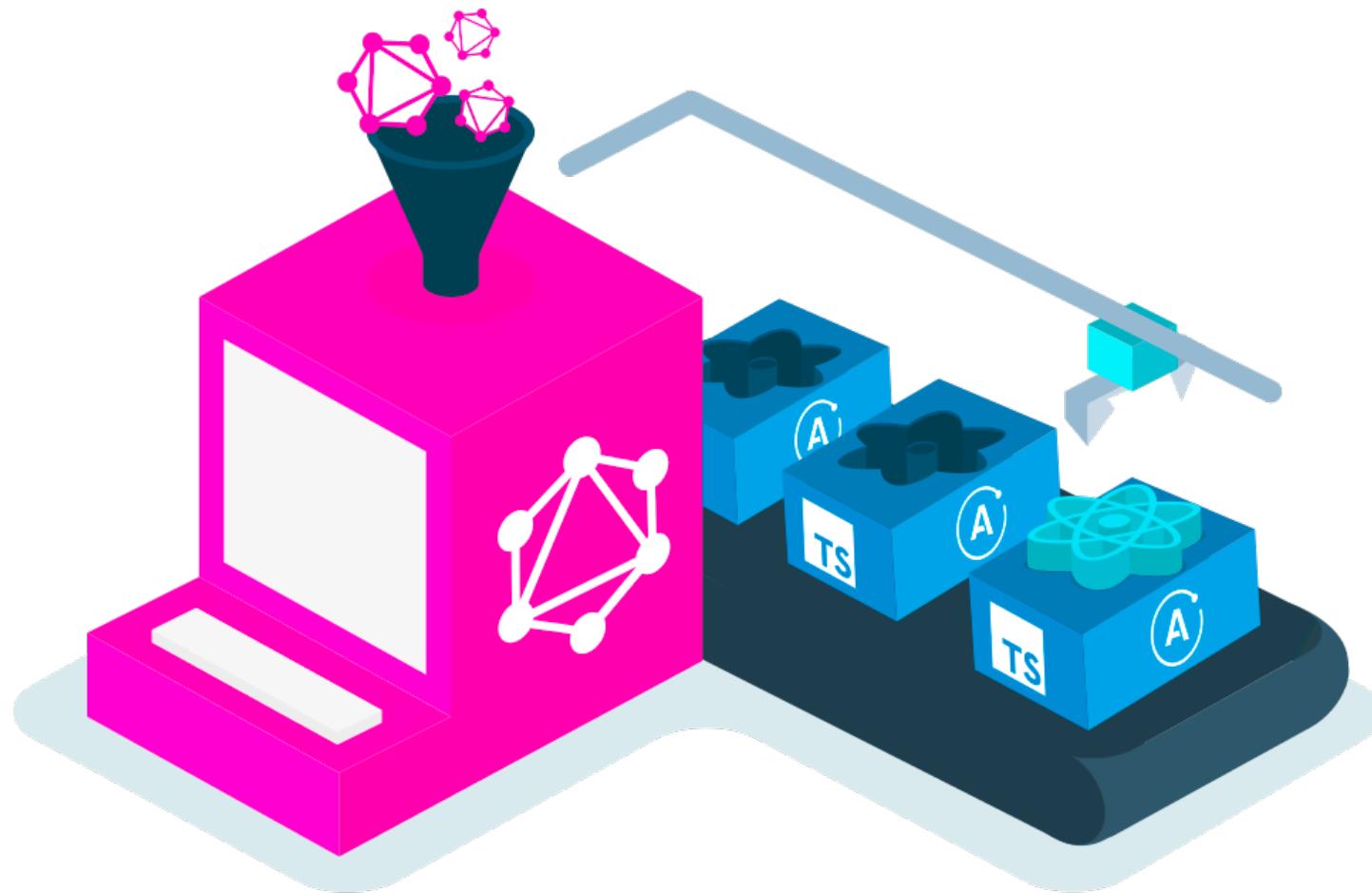
The 'types.ts' file itself contains code such as:

```
export type Maybe<T> = T | null;
export type Exact<T extends { [key: string]: unknown }> = { [K in keyof T]: T[K] extends object ? Exact<T[K]> : T[K]; }
export type MakeOptional<T, K extends keyof T> = Omit<T, K> & { [SubKey in K]: Nullable<T[SubKey]> };
export type MakeMaybe<T, K extends keyof T> = Omit<T, K> & { [SubKey in K]: Nullable<T[SubKey]> };
/** All built-in and custom scalars, mapped to their actual values */
export type Scalars = {
  ID: string;
  String: string;
```

On the right side of the screenshot, there is a vertical text overlay: 'graph-code-generator' and '공식 홈페이지'.

# 4.3 서버 스키마 동기화

해결: graphql-code-generator



{ GraphQL }  
**code generator**

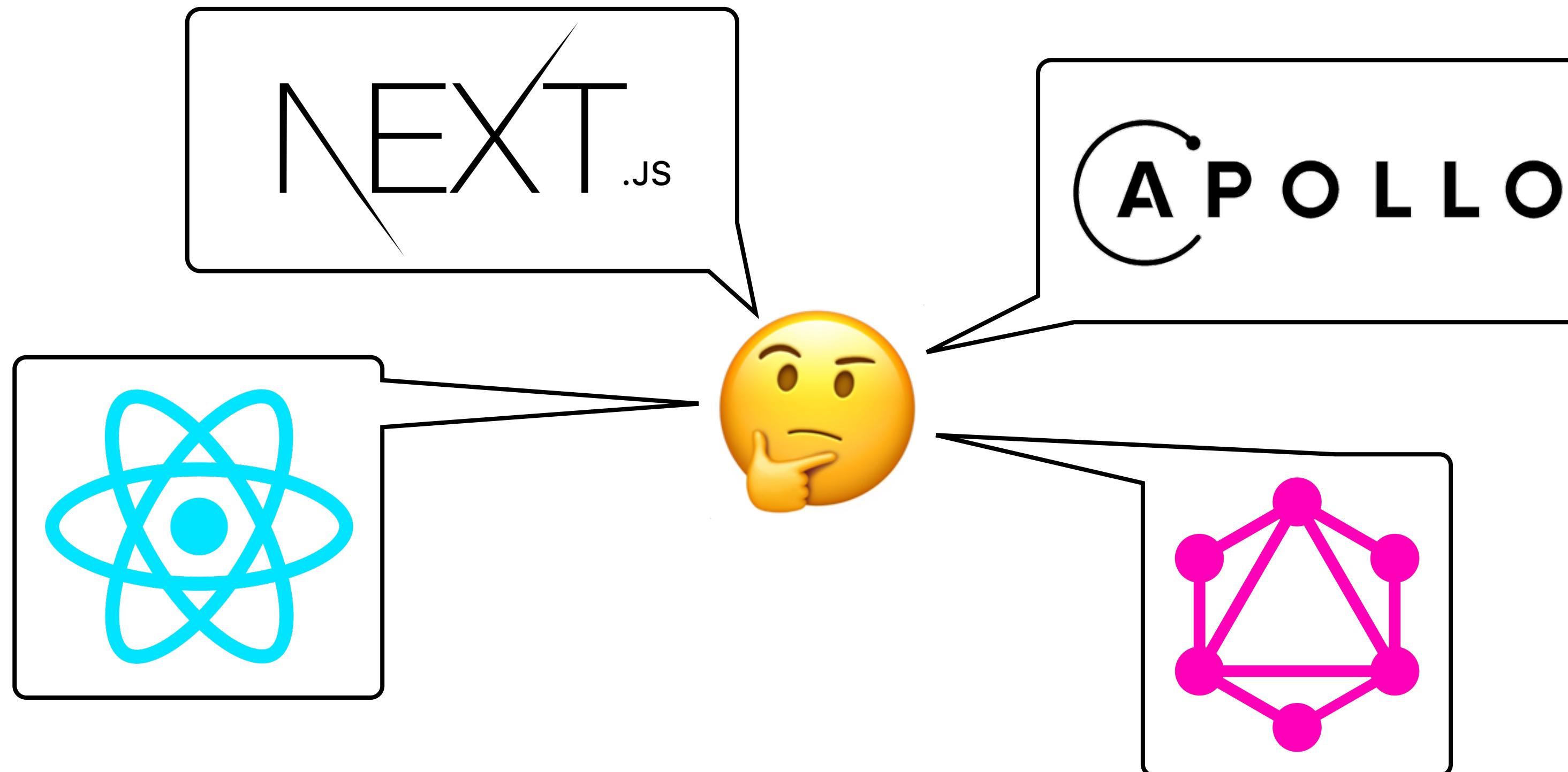
codegen.yml

```
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



# Thank You

