



◆ CS EDUCATION ◆

Q

REST API vs GraphQL

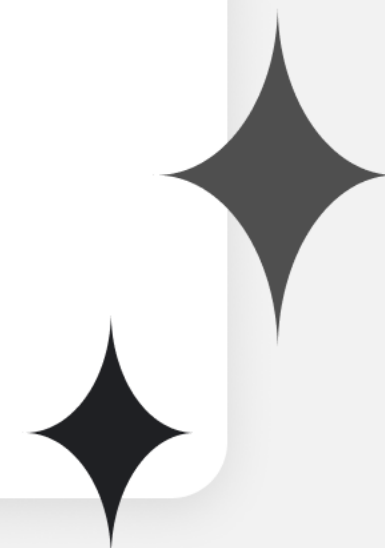


2024.11.22
10기 교육팀장 강다형



교육 목표

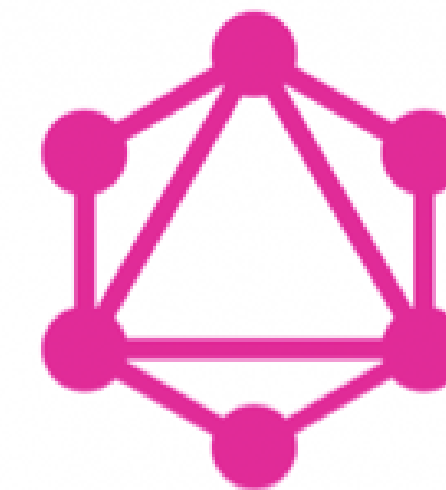
1. REST API와 GraphQL에 대한 이해
2. REST API와 GraphQL 차이점



API architecture style

{ REST API }

Representational State Transfer API



GraphQL

1. 클라이언트가 서버의 엔드포인트 또는 여러 엔드포인트에 API 요청을 전송
2. 서버가 데이터, 데이터 상태 또는 오류 코드가 포함된 응답을 제공

REST란?

{ REST }

Representational State Transfer

자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 모든 것을 의미

REST의 구성요소

자원

소프트웨어가
관리하는 모든 것

행위

자원에 대한 작업

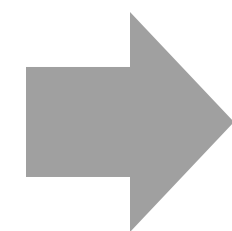
표현

자원의 상태를
나타내는 데이터

REST의 구성요소

자원

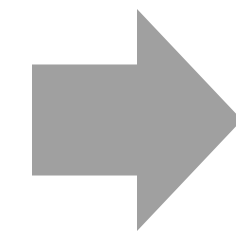
고유한 URI로 표현



/products/{id}
/products/101

행위

HTTP Method로 표현



GET, POST, PUT, PATCH, DELETE

표현

JSON, XML 등으로
표현

```
{  
  "id": 101,  
  "name": "Wireless Mouse",  
  "price": 25.99,  
  "description": "A high-precision wireless mouse with ergonomic  
design."  
}
```

REST의 구성요소

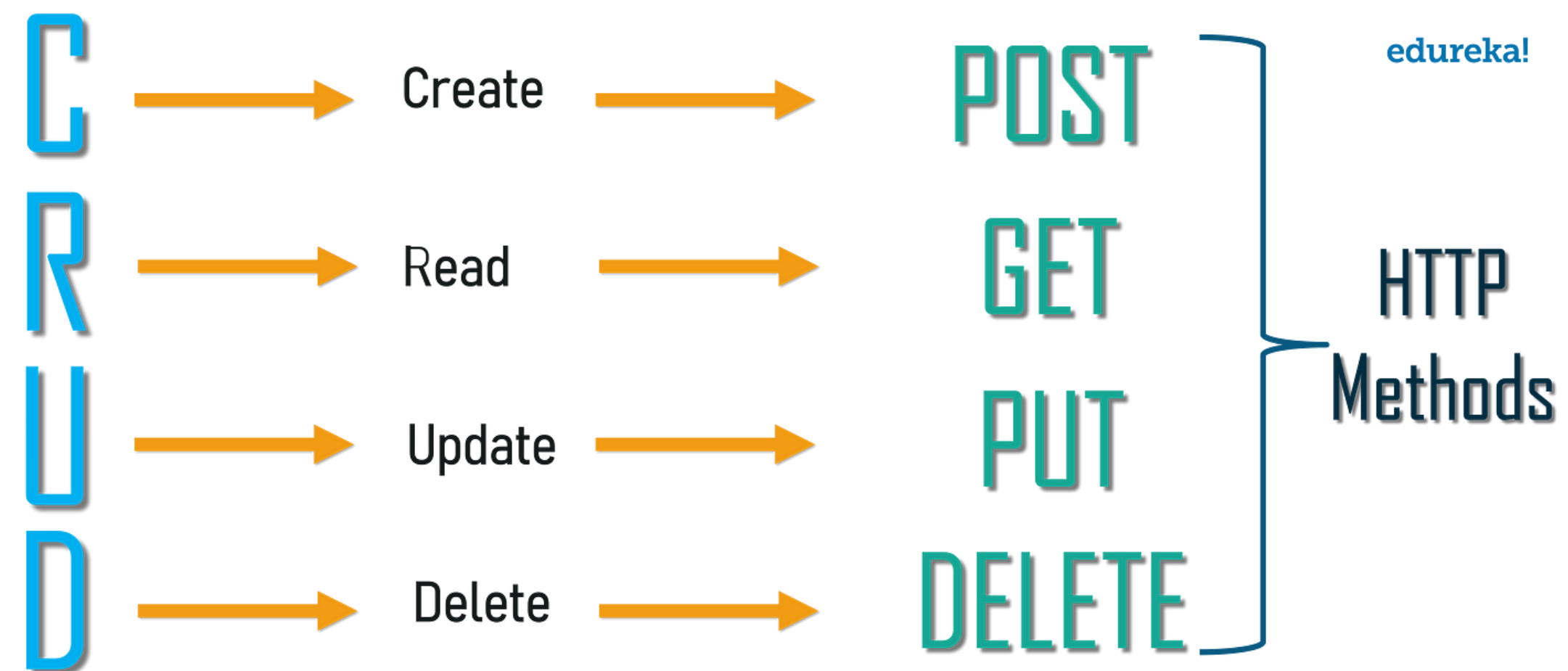
✕	Headers	Payload	Preview	Response
▼	[{_id: 13, toName: "용가리", fromName: "용가리", fromId: 0, message: "빨리 종강했으면 좋겠다~", type: "뭐를 써야되지...?", ...}]			
▶	0: {_id: 13, toName: "용가리", fromName: "용가리", fromId: 0, message: "빨리 종강했으면 좋겠다~", type: "뭐를 써야되지...?", ...}			
▼	1: {_id: 14, toName: "용가리", fromName: "김철수", fromId: 0, message: "빨리 종강했으면 좋겠다~", type: "뭐를 써야되지...?", ...}			
	date: "2024-11-12T00:58:23.000Z"			
	fromId: 0			
	fromName: "김철수"			
	message: "빨리 종강했으면 좋겠다~"			
	toName: "용가리"			
	type: "뭐를 써야되지...?"			
	_id: 14			
▶	2: {_id: 15, toName: "용가리", fromName: "피크닉", fromId: 0, message: "피크닉을 가고싶당~", type: "뭐를 써야되지...?", ...}			
▶	3: {_id: 16, toName: "용가리", fromName: "익명", fromId: 0, message: "종강까지 한달 남음~~~", type: "뭐를 써야되지...?", ...}			
▶	4: {_id: 17, toName: "용가리", fromName: "익명", fromId: 0, message: "오늘은 11/13", type: "뭐를 써야되지...?", ...}			

```
async function getMessageData() {
  try {
    const response = await instance.get(`/messages?
name=${userName}`);
    if (response.status === 200) {
      const fetchedMsg = response.data;
      setMessages(fetchedMsg);
      console.log("편지 가져오기 성공");
    }
  } catch (error) {
    console.error("메세지를 가져오지 못했습니다.");
  }
}
```


REST란?

REST는 HTTP Method를 사용해 자원을 생성, 읽기, 수정, 삭제하는 방식

자원을 중심으로 설계



REST의 특징

01 | Server-Client

02 | Stateless

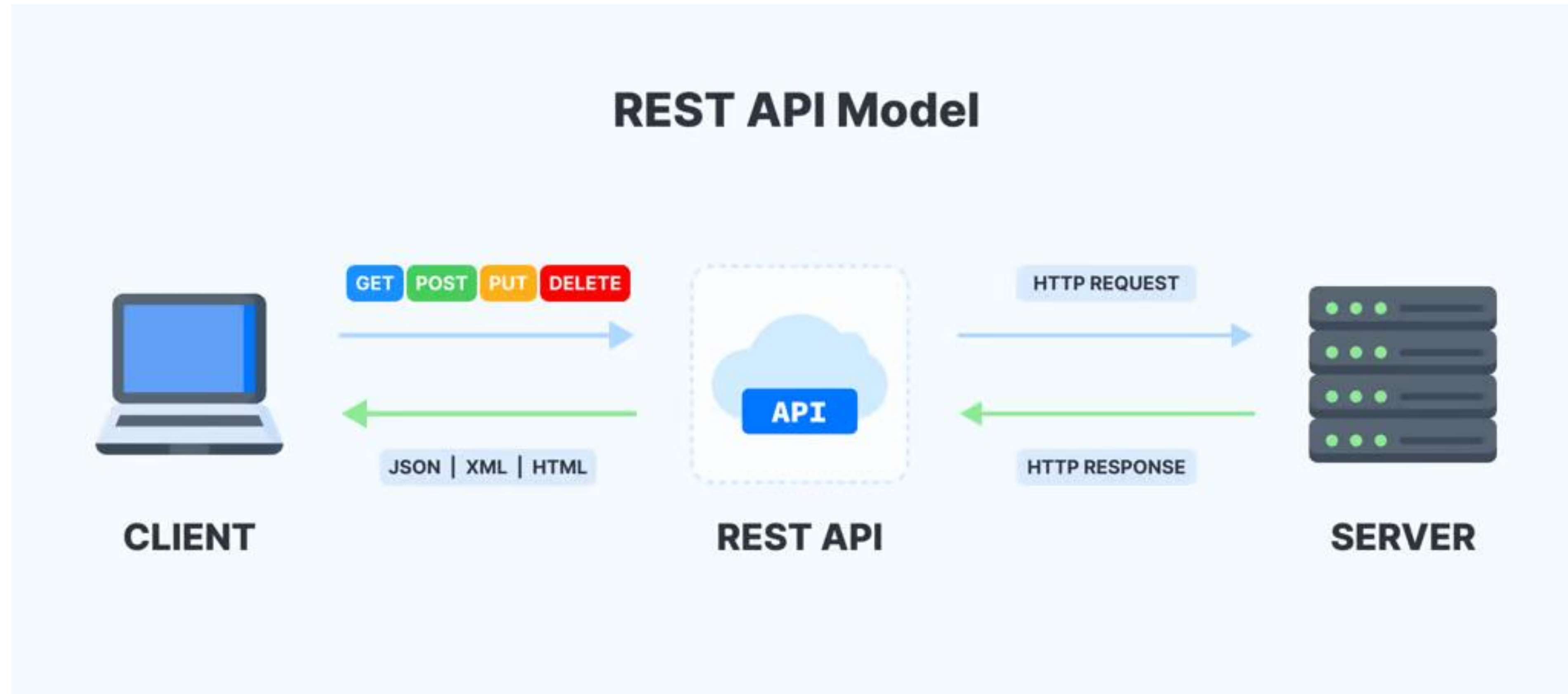
03 | Cacheable

04 | Layered System

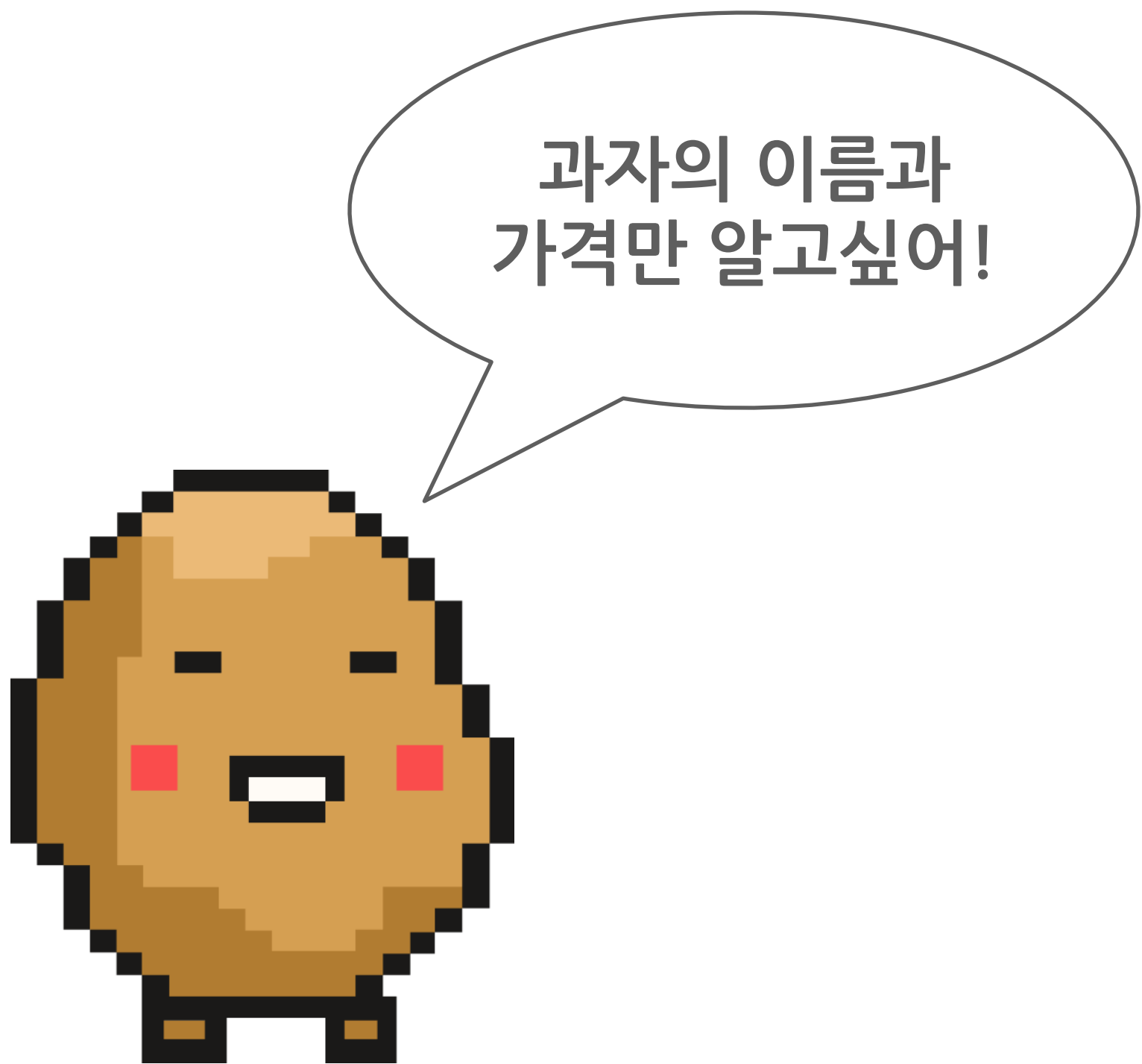
05 | Uniform Interface

06 | Self-Descriptiveness

REST API? RESTful API?

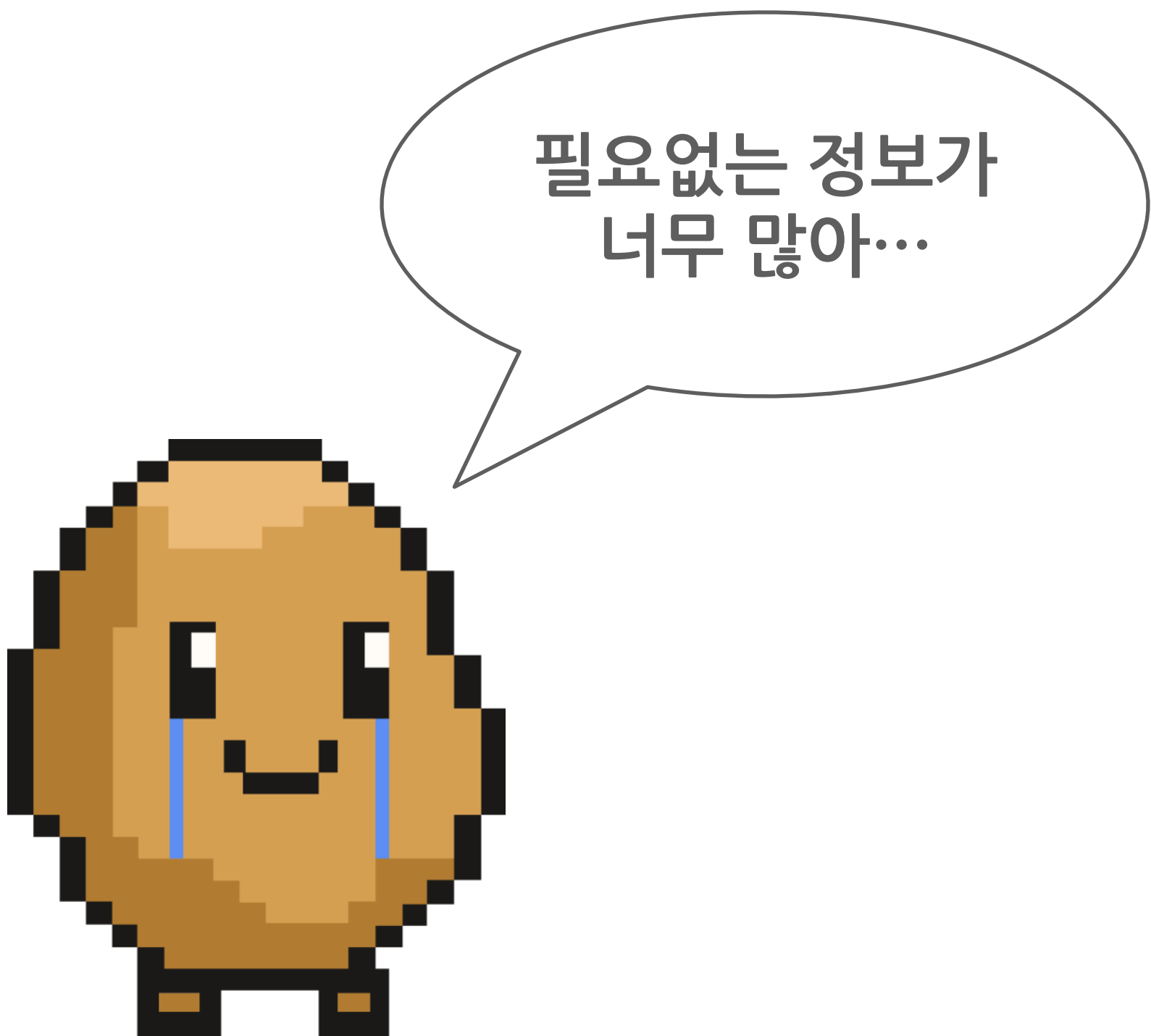


REST의 단점: Over-fetching



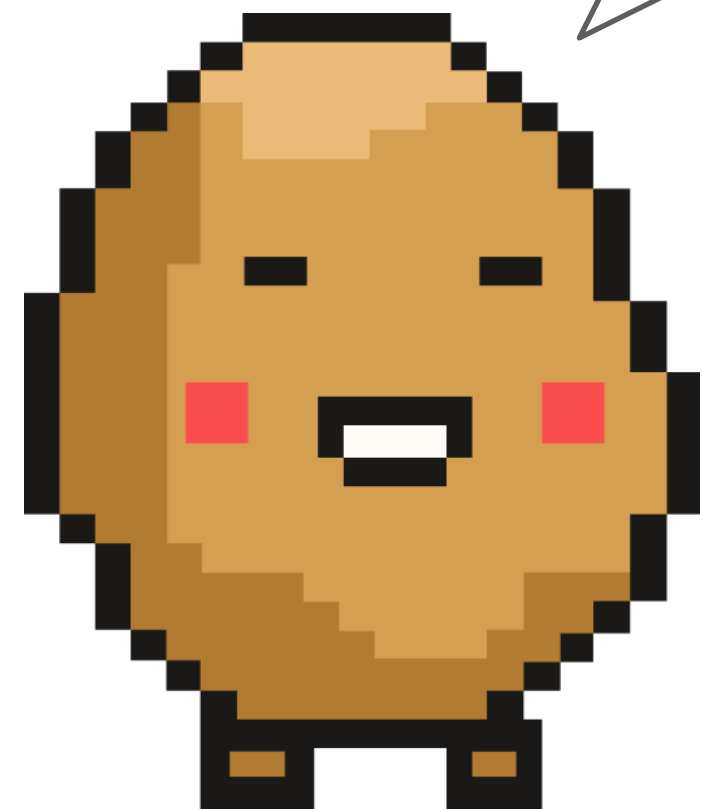
```
fetch('http://example.com/api/snacks')
  .then(response => response.json())
  .then(data => {
    // 이름과 가격만 사용
    data.forEach(snack => {
      console.log(`Name: ${snack.name}, Price:
${snack.price}`);
    });
  });
```

REST의 단점: Over-fetching



```
[
  {
    "id": 1,
    "name": "포카칩",
    "price": 1800,
    "calories": 200,
    "ingredients": ["감자", "소금", "설탕"],
    "manufacturer": {
      "name": "오리온",
      "address": "서울특별시 용산구 백범로90다길
13", "contact": "02-710-6000"
    }
  },
  {
    "id": 2,
    "name": "눈을 감자",
    "price": 2000,
    "calories": 250,
    "ingredients": ["감자", "설탕", "우유"],
    "manufacturer": {
      "name": "오리온",
      "address": "서울특별시 용산구 백범로90다길
13", "contact": "02-710-6000"
    }
  }
]
```

REST의 단점: under-fetching



과자 정보와
제조사 정보를
모두 알고싶어!

```
[
  {
    "id": 1,
    "name": "눈을 감자",
    "price": 2000,
    "manufacturerId": 101
  },
  {
    "id": 2,
    "name": "무뚝뚝 감자칩",
    "price": 1900,
    "manufacturerId": 102
  }
]
```

```
[
  {
    "id": 101,
    "name": "오리온"
  },
  {
    "id": 102,
    "name": "농심",
  }
]
```

REST의 단점: under-fetching

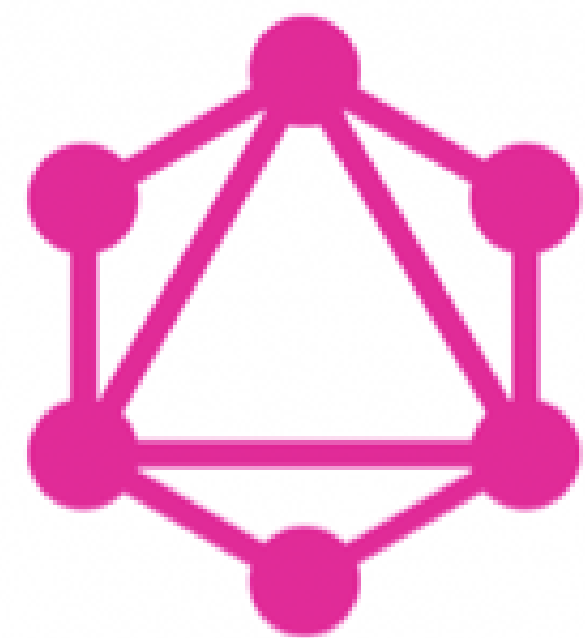
요청을 여러번
해야해.....



```
fetch('http://example.com/api/snacks')
  .then(response => response.json())
  .then(snacks => {
    // 제조사 정보를 위해 추가 요청
    snacks.forEach(snack => {
      fetch(`http://example.com/api/manufacturers/${snack.manufacturerId}`)
        .then(response => response.json())
        .then(manufacturer => {
          console.log(`Name: ${snack.name}, Price: ${snack.price}, Manufacturer:
${manufacturer.name}`);
        });
    });
  });
```

Name: 눈을 감자, Price: \$2000, Manufacturer: 오리온
Name: 무뚝뚝 감자칩, Price: \$1900, Manufacturer: 농심

GraphQL의 등장



GraphQL

API를 위한 쿼리 및 조작 언어

GraphQL의 특징

```
// REST API
→ example.com/class
→ example.com/class/{반 index}
→ example.com/class/{반 index}/students
→ example.com/class/{반 index}/students/{학생 index}

// GraphQL
→ example.com/graphql
```

하나의 엔드포인트를 가짐 => POST만 존재

GraphQL의 특징

Query

데이터를 요청

Mutation

CRUD 작업에 필요

Subscription

실시간 작업에 필요

Resolver

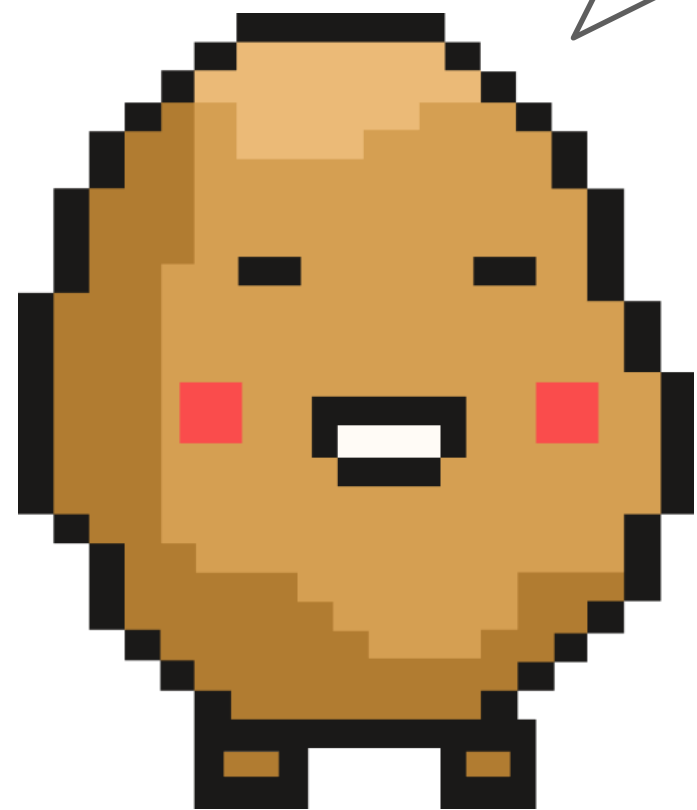
스키마의 각 필드와
데이터 소스(DB, API)를
연결

백엔드에서 스키마를 반드시 정의해주어야함

<https://graphql-kr.github.io/learn/schema/>

GraphQL의 특징: over-fetching 해결

영화의 제목과
개봉날짜만
알고싶어!



```
1 ▼ query {  
2   allFilms(first: 3) {  
3     edges {  
4       node {  
5         title  
6         releaseDate  
7       }  
8     }  
9   }  
10 }  
11
```

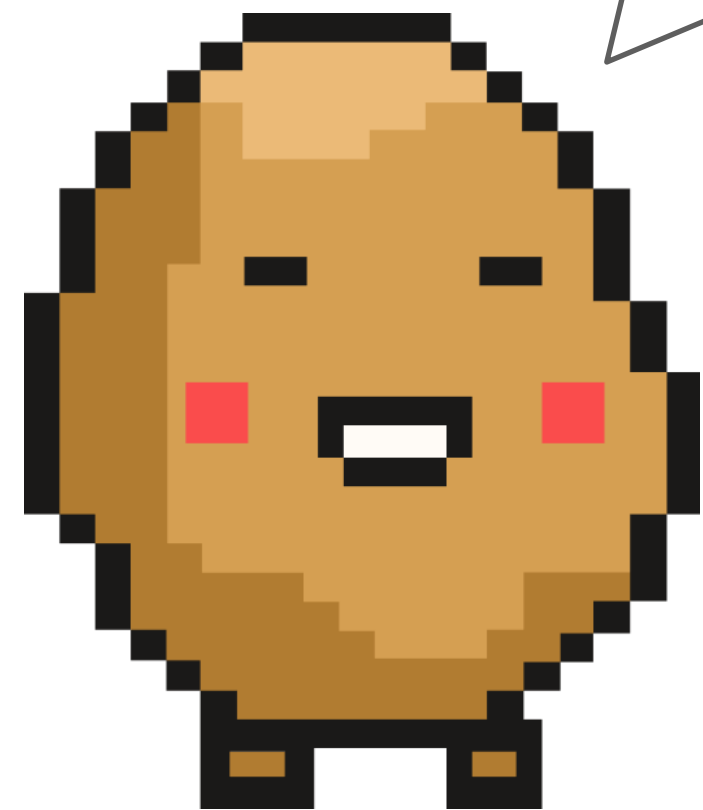
GraphQL의 특징: over-fetching 해결

내가 필요한
정보만 오는군!



```
{
  "data": {
    "allFilms": {
      "edges": [
        {
          "node": {
            "title": "A New Hope",
            "releaseDate": "1977-05-25"
          }
        },
        {
          "node": {
            "title": "The Empire Strikes Back",
            "releaseDate": "1980-05-17"
          }
        },
        {
          "node": {
            "title": "Return of the Jedi",
            "releaseDate": "1983-05-25"
          }
        }
      ]
    }
  }
}
```

GraphQL의 특징: under-fetching 해결



영화 제목과
사람들에 대한
정보를 모두 알고
싶어!

```
1 ▼ {  
2   allFilms {  
3     totalCount  
4     films {  
5       title  
6     }  
7   }  
8   allPeople {  
9     people {  
10      name  
11      hairColor  
12    }  
13  }  
14 }  
15
```

GraphQL의 특징: under-fetching 해결

REST API는 /api/films,
/api/people으로 두번의
요청을 보내야했는데,
한번의 요청으로 두 개의
정보를 받아왔군!



```
    },  
    {  
      "title": "Attack of the Clones"  
    },  
    {  
      "title": "Revenge of the Sith"  
    },  
    {  
      "title": "The Force Awakens"  
    }  
  ],  
  "allPeople": {  
    "people": [  
      {  
        "name": "Luke Skywalker",  
        "hairColor": "blond"  
      },  
      {  
        "name": "C-3PO",  
        "hairColor": "n/a"  
      }  
    ]  
  }  
}
```

GraphQL의 장점

1. HTTP 요청 횟수를 줄일 수 있음
2. HTTP 응답 사이즈를 감소시킬 수 있음
3. 개발자의 부담을 덜 수 있음

GraphQL의 단점

1. 고정된 요청과 응답만 필요할 때에는 query로 인해 요청의 크기가 Rest보다 커질 수 있음.
2. 요청에 대한 필터링이 어려움.
3. 각 요청에 대한 Http 캐싱을 사용할 수 없음.

차이점 정리

	REST	GraphQL
정의	클라이언트-서버간의 데이터 교환을 정의하는 규칙	API를 생성하고 조작하기 위한 쿼리 언어
데이터 요청방식	여러 엔드포인트와 HTTP 메소드를 사용해 데이터를 요청	하나의 엔드포인트에서 클라이언트가 원하는 데이터를 쿼리 언어로 요청
반환된 데이터	고정된 구조로 데이터를 반환	클라이언트가 요청한 데이터만 반환
데이터 구조	서버가 정해놓은 고정된 데이터 구조로 응답.	JSON 형태로 클라이언트가 요청한 데이터 구조에 맞춰 응답.
오류 검사	HTTP 상태 코드로 오류를 전달 (예: 404, 500).	오류가 errors 필드에 포함되어 상세한 오류 정보를 제공.



◆ CS QUIZ ◆



Q.1

REST의 구성요소에 대한 설명으로 틀린 것을 고르시오.

1. 구성요소 중 행위는 HTTP 메서드를 통해 정의된다.
2. 자원은 서버가 관리하는 데이터나 개체를 의미하며, 고유한 URI로 식별된다.
3. 자원은 클라이언트가 생성하며, 서버는 이를 단순히 저장만 한다.
4. 자원은 자원, 행위, 표현으로 구성된다.



A.1

REST의 구성요소에 대한 설명으로 틀린 것을 고르시오.

1. 구성요소 중 행위는 HTTP 메서드를 통해 정의된다.
2. 자원은 서버가 관리하는 데이터나 개체를 의미하며, 고유한 URI로 식별된다.
3. 자원은 클라이언트가 생성하며, 서버는 이를 단순히 저장만 한다.
4. 자원은 자원, 행위, 표현으로 구성된다.

3



Q.2

REST의 특징에 대한 설명으로 틀린 것을 고르시오.

1. REST는 클라이언트-서버 구조를 기반으로 설계되며, 클라이언트와 서버는 서로 독립적으로 동작한다.
2. REST는 클라이언트와 서버 간 상태 정보를 공유하여 세션을 유지한다.
3. REST는 Self-Descriptiveness를 통해 클라이언트가 요청에 필요한 정보를 쉽게 이해할 수 있도록 한다.
4. REST는 Layered System을 통해 클라이언트가 중간 서버의 존재를 알 필요없이 요청을 처리할 수 있다.



A.2

REST의 특징에 대한 설명으로 틀린 것을 고르시오.

1. REST는 클라이언트-서버 구조를 기반으로 설계되며, 클라이언트와 서버는 서로 독립적으로 동작한다.
2. REST는 클라이언트와 서버 간 상태 정보를 공유하여 세션을 유지한다.
3. REST는 Self-Descriptiveness를 통해 클라이언트가 요청에 필요한 정보를 쉽게 이해할 수 있도록 한다.
4. REST는 Layered System을 통해 클라이언트가 중간 서버의 존재를 알 필요없이 요청을 처리할 수 있다.



Q.3

GraphQL에 대한 설명으로 틀린 것을 모두 고르시오.

1. 클라이언트가 원하는 정보를 하나의 쿼리에 모두 담아 요청할 수 있다.
2. 쿼리 언어 자체에서 필터링의 명확한 표준을 제공한다.
3. 오류가 발생했을 때 HTTP 상태 코드로 오류를 전달한다.
4. Over-fetching과 Under-fetching 문제를 해결할 수 있다.



A.3

GraphQL에 대한 설명으로 틀린 것을 모두 고르시오.

2,3

1. 클라이언트가 원하는 정보를 하나의 쿼리에 모두 담아 요청할 수 있다.
2. 쿼리 언어 자체에서 필터링의 명확한 표준을 제공한다.
3. 오류가 발생했을 때 HTTP 상태 코드로 오류를 전달한다.
4. Over-fetching과 Under-fetching 문제를 해결할 수 있다.

←

→

↺

🏠

🔒

🔗

https://www.cotato.com

🖨

☆

🛡

📄

⛶

☰

Q.4

다음 코드와 관련된 설명 중 올바른 것을 고르시오.

```
import requests

# 첫 번째 요청: 사용자 데이터를 가져옴
response_user = requests.get("https://api.example.com/users/1")
user_data = response_user.json()

# 두 번째 요청: 해당 사용자의 게시물 목록을 가져옴
response_posts = requests.get(f"https://api.example.com/users/{user_data['id']}/posts")
posts_data = response_posts.json()

# 세 번째 요청: 게시물마다 작성자 정보 확인
for post in posts_data:
    response_author = requests.get(f"https://api.example.com/posts/{post['id']}/author")
    author_data = response_author.json()
    print(f"Post: {post['title']}, Author: {author_data['name']}")
```

Q.4

다음 코드와 관련된 설명 중 올바른 것을 고르시오.

- 1.API 설계가 잘못되어 데이터 일관성을 유지할 수 없다.
- 2.Over-fetching이 발생하여 불필요한 데이터를 한 번에 가져온다.
- 3.Under-fetching이 발생하여 필요한 데이터를 가져오기 위해 여러 번의 추가 요청이 필요하다.
- 4.클라이언트-서버 구조의 문제로 인해 데이터 전송 속도가 느려진다.



A.4

다음 코드와 관련된 설명 중 올바른 것을 고르시오.

3

- 1.API 설계가 잘못되어 데이터 일관성을 유지할 수 없다.
- 2.Over-fetching이 발생하여 불필요한 데이터를 한 번에 가져온다.
- 3.Under-fetching이 발생하여 필요한 데이터를 가져오기 위해 여러 번의 추가 요청이 필요하다.
- 4.클라이언트-서버 구조의 문제로 인해 데이터 전송 속도가 느려진다.

Q.5

REST API에서 클라이언트가 자원을 조회하기 위해 사용하는 HTTP 메서드는?



A.5

REST API에서 클라이언트가 자원을 조회하기 위해 사용하는 HTTP 메서드는?

GET

Q.6

Stateless에 대한 설명으로 올바른 것을 고르시오.

1. 서버가 이전 요청의 상태를 기억하고 다음 요청에 반영한다.
2. 클라이언트와 서버 간의 모든 요청이 독립적으로 처리된다.
3. 모든 요청은 동일한 데이터 상태를 유지해야 한다.
4. 서버는 클라이언트의 세션 정보를 저장해야 한다.



A.6

Stateless에 대한 설명으로 올바른 것을 고르시오.

2

- 1. 서버가 이전 요청의 상태를 기억하고 다음 요청에 반영한다.
- 2. 클라이언트와 서버 간의 모든 요청이 독립적으로 처리된다.
- 3. 모든 요청은 동일한 데이터 상태를 유지해야 한다.
- 4. 서버는 클라이언트의 세션 정보를 저장해야 한다.

←

→

↺

🏠

🔒

🔒

https://www.cotato.com

🖼️

☆

🛡️

📄

⛶

☰

□

Q.7

□

GraphQL은 모든 요청을 동일한 _____로 보내며, 요청 방식도 _____로 통일되어 있다.



A.7

다음 빈칸에 들어갈 말로 적절한 것을 순서대로 쓰시오.

GraphQL은 모든 요청을 동일한 **엔드포인트**로 보내며, 요청 방식도 **POST**로 통일되어 있다.

Q.8

API는 클라이언트와 서버 간의 데이터를 교환할 때 어떤 방식으로 작동하는지
고르시오.

1. 서버가 클라이언트의 요청 없이 데이터를 자동으로 전송한다.
2. 클라이언트는 데이터를 요청할 수 없고, 서버만 데이터 전송을 관리한다.
3. 클라이언트가 요청을 보내면 서버가 데이터, 상태, 또는 오류 코드를 응답으로 반환한다.
4. 클라이언트와 서버는 항상 동일한 데이터 포맷을 사용해야한다.



A.8

API는 클라이언트와 서버 간의 데이터를 교환할 때 어떤 방식으로 작동하는지
고르시오.

3

1. 서버가 클라이언트의 요청 없이 데이터를 자동으로 전송한다.
2. 클라이언트는 데이터를 요청할 수 없고, 서버만 데이터 전송을 관리한다.
3. 클라이언트가 요청을 보내면 서버가 데이터, 상태, 또는 오류 코드를 응답으로 반환한다.
4. 클라이언트와 서버는 항상 동일한 데이터 포맷을 사용해야한다.

Q.9

다음 중 over-fetching의 예로 적절한 것을 고르시오.

1. 클라이언트가 사용자의 이름과 이메일을 요청했지만 서버가 이름만 반환한다.
2. 클라이언트가 사용자 이름만 필요하지만 서버가 이름, 이메일, 주소 등 모든 데이터를 반환한다.
3. 클라이언트가 주문 내역을 요청했는데 주문 내역을 받기 위해 추가적인 서버요청을 해야 한다.
4. 클라이언트가 요청한 데이터와 서버에서 반환된 데이터가 정확히 일치한다.



A.9


다음 중 over-fetching의 예로 적절한 것을 고르시오. **2**

1. 클라이언트가 사용자의 이름과 이메일을 요청했지만 서버가 이름만 반환한다.
2. 클라이언트가 사용자 이름만 필요하지만 서버가 이름, 이메일, 주소 등 모든 데이터를 반환한다.
3. 클라이언트가 주문 내역을 요청했는데 주문 내역을 받기 위해 추가적인 서버요청을 해야 한다.
4. 클라이언트가 요청한 데이터와 서버에서 반환된 데이터가 정확히 일치한다.



Q.10

다음 중 REST와 GraphQL의 차이점으로 적절한 것을 고르시오.

1. REST는 하나의 엔드포인트를 사용하고, GraphQL은 여러 엔드포인트를 사용한다.
 2. REST는 고정된 데이터 형식을 반환하고, GraphQL 요청에 따라 유연하게 데이터를 반환한다.
 3. REST는 POST 요청만 지원하고, GraphQL은 GET 요청만 지원한다.
 4. REST와 GraphQL은 데이터 요청 방식에서 차이가 없다.
- 

A.10

다음 중 REST와 GraphQL의 차이점으로 적절한 것을 고르시오.

2

1. REST는 하나의 엔드포인트를 사용하고, GraphQL은 여러 엔드포인트를 사용한다.
2. REST는 고정된 데이터 형식을 반환하고, GraphQL 요청에 따라 유연하게 데이터를 반환한다.
3. REST는 POST 요청만 지원하고, GraphQL은 GET 요청만 지원한다.
4. REST와 GraphQL은 데이터 요청 방식에서 차이가 없다.