Null4U 전공탐색동아리

(2차시)

디스코드 링크

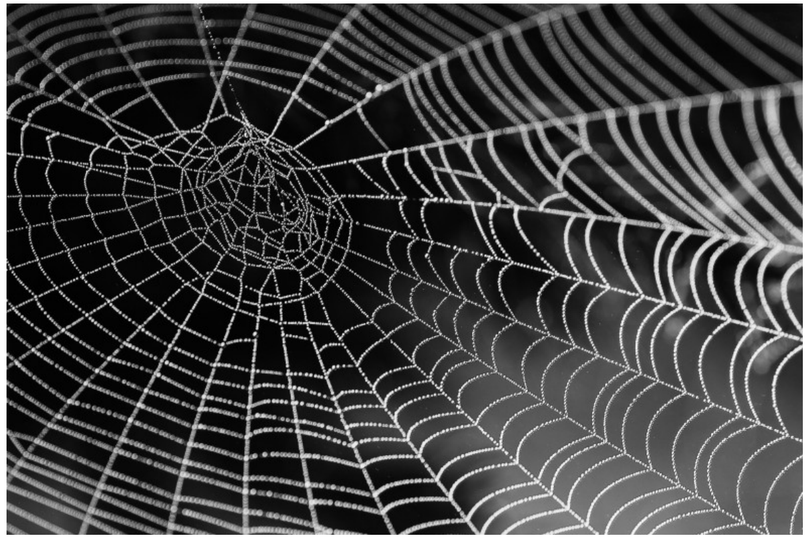
https://discord.gg/RSmXz5dUjG

강의자

부장 **김준영**

전 시간 복습 10분만

웹(Web) 이란?



원래 "거미줄"을 의미함

사실 풀네임은 World Wide Web (WWW), 흔히 웹이라 부름

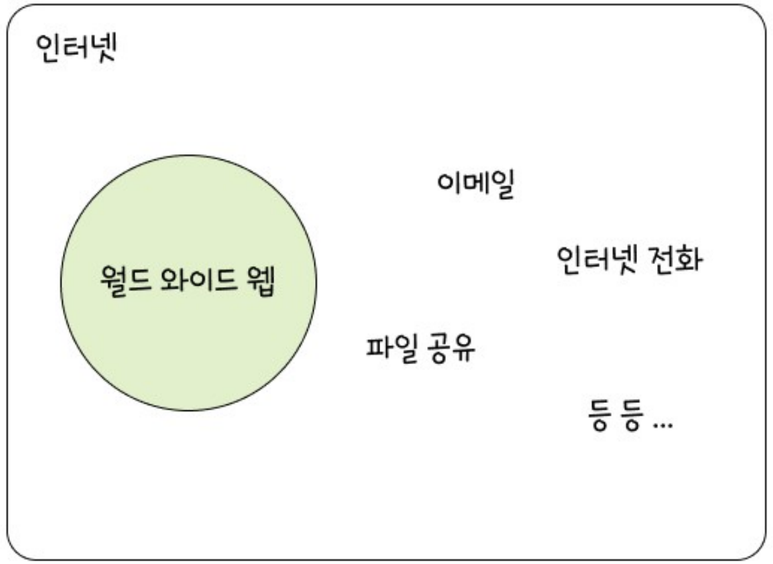
인터넷 등에 연결된 전세계 사용자들이 서로의 정보를 공유할 수 있는 공간

컴퓨터들이 연결되는 것을 네트워크라고 함. (그물망 처럼 연결됐다고 해서)

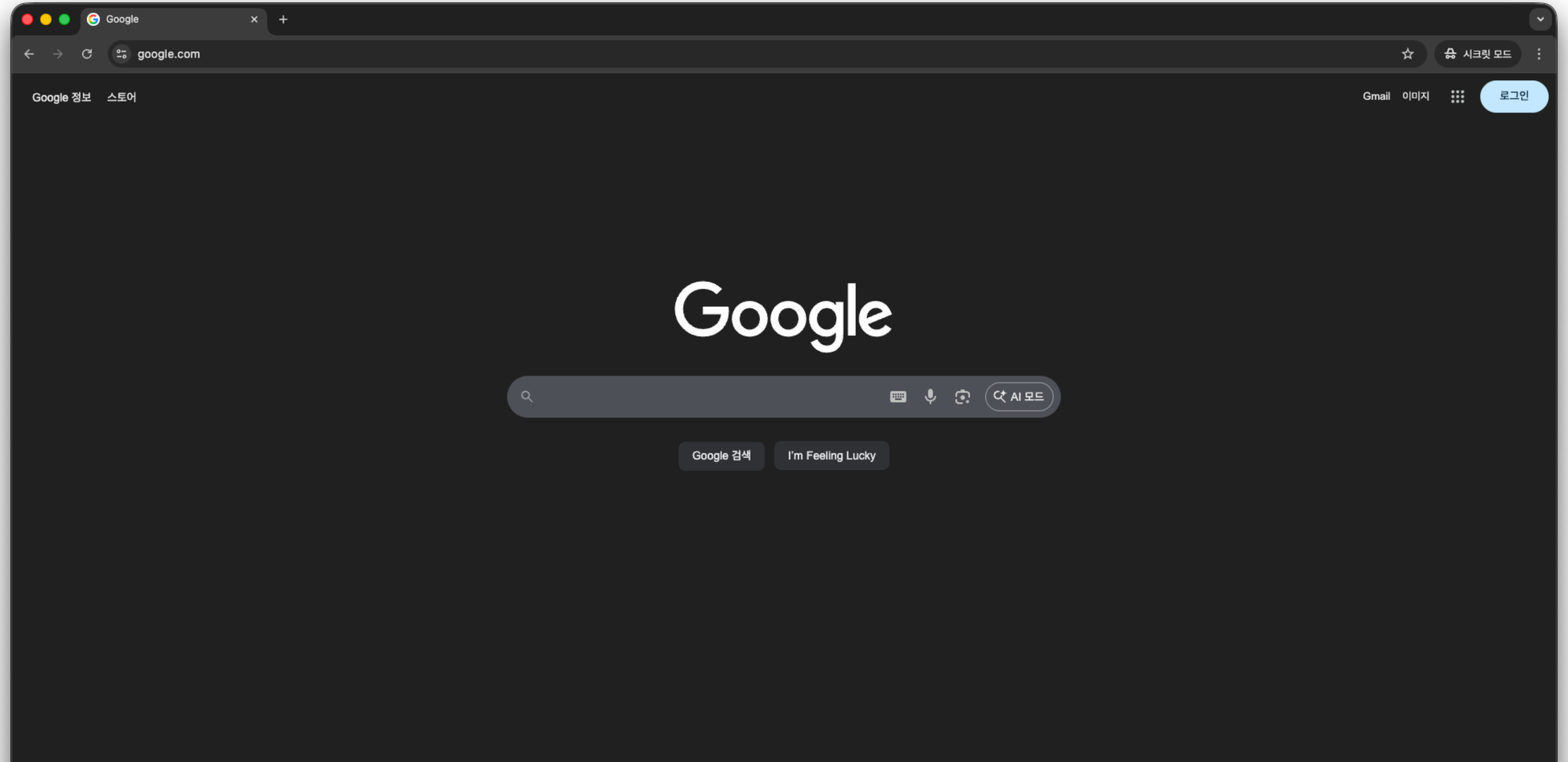
컴퓨터들이 연결되는 것을 네트워크라고 함. (그물망 처럼 연결됐다고 해서)

인터넷 = 전 세계 컴퓨터들을 하나로 연결하는 거대한 컴퓨터 통신망을 의미함

(인터넷 != 네트워크 != 웹)



구글에 접속하면 무슨 일이 벌어질까?



구글에 접속하면 무슨 일이 벌어질까?

검색 결과 주세요.

네. "연신내역 근처 맛집"에 대한 검색 결과입니다.

유저

Google

클라이언트
(받는 쪽)

서버
(주는 쪽)

너무나도 중요한 개념!!!!!

부록) 이름만 들어본 URL, 뜯어보자.

검색 결과 주세요.

https://www.google.com/search?q=연신내역+맛집

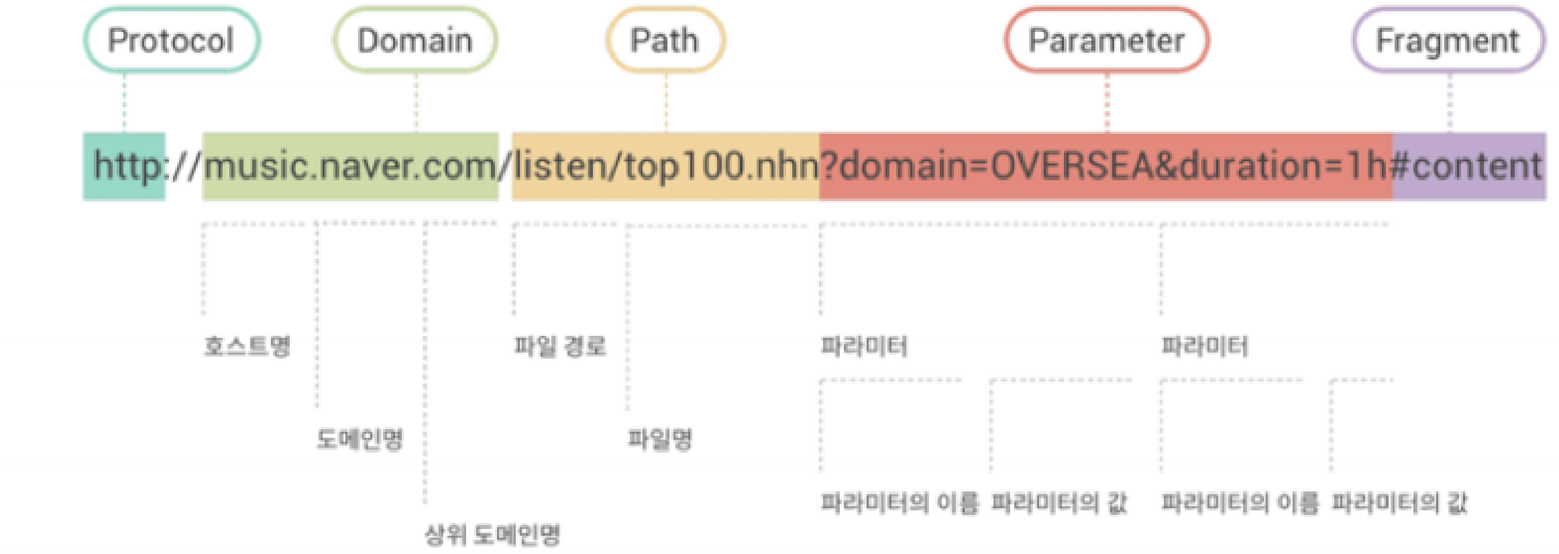
유저 (클라이언트)

Google (서버)

흔히 사이트 주소라고 부르는게 URL이라고 함.

URL 말고도 URI, URN이 있는데 이건 추후 시간나면 설명해주겠음. + URL 인코딩

부록) 이름만 들어본 URL, 뜯어보자.



부록) 이름만 들어본 URL, 뜯어보자.

프로토콜은 브라우저에선 대부분 http/https, file로 많이 봤을거임.

프로토콜은 네트워크 설명할때 설명하겠지만, 일단 통신 규약/규격과 같다고 판단하면 됨.

웹은 대부분 HTTP/HTTPS 기반임.

부록) 이름만 들어본 URL, 뜯어보자.

도메인은 google.com, naver.com 같이 IP 주소를 유저가 쉽게 기억할 수 있도록 문자열로 바꾸는 것

이것도 네트워크 설명할때 DNS와 함께 설명함. 네트워크/웹 개발, 운영에서 매우 중요한 개념임.

서브 도메인 개념도 나중에 설명

부록) 이름만 들어본 URL, 뜯어보자.

경로(Path)는 파일 디렉토리의 경로와 동일하게 생각하면 됨. 특정 리소스(페이지, 이미지) 등의 위치를 가리키는 주소임.

경우에 따라 진짜 파일 디렉토리를 쓰는 경우도 있지만 직접 쓰는건 보안상 권장되지 않고,

이후에 다룰 API 백엔드 서버는 경로(이 경우엔 엔드포인트)를 개발자가 직접 지정해서 만듬.

부록) 이름만 들어본 URL, 뜯어보자.

Parameter와 Fragment는 URL에 부가적인 정보를 줄때 사용할 수 있음.

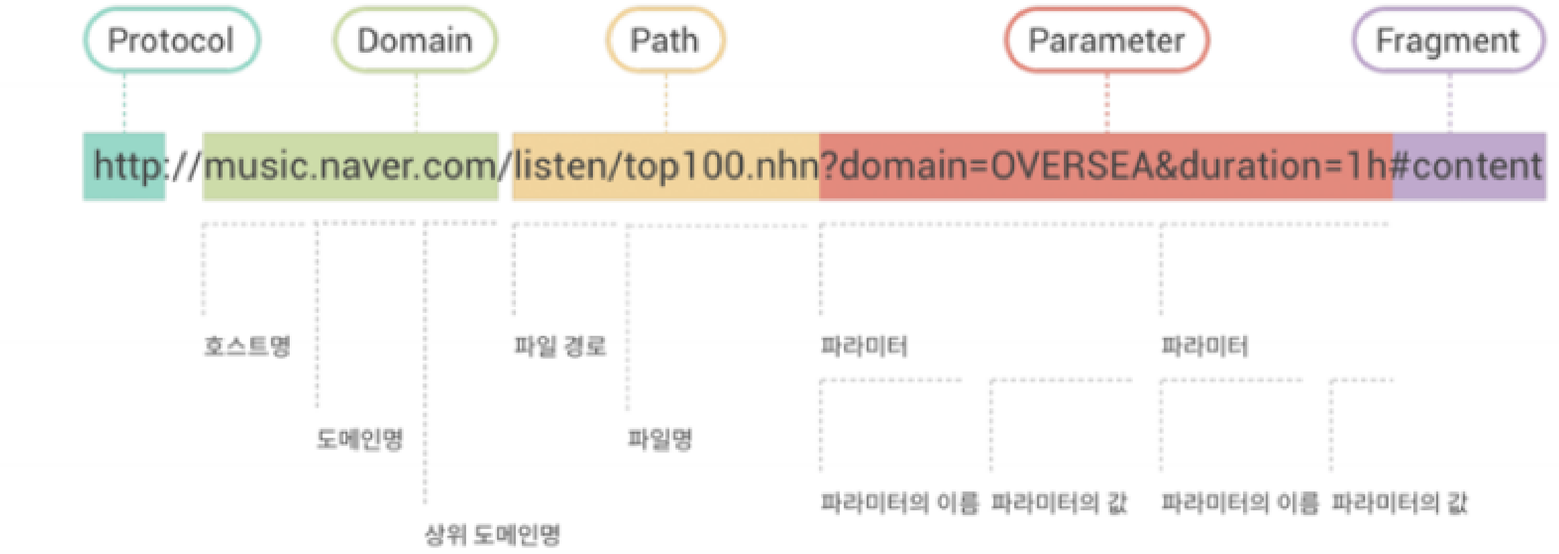
일반적으로 GET Method에서만 사용되며, 이는 나중에 HTTP 메서드를 다룰때 다시 언급하겠음.

부록) 이름만 들어본 URL, 뜯어보자.

한국어로://은평롯데점.빕스.레스토랑/세트메뉴/파티세트.html&인원=3명#메뉴판-섹션

영어로://백엔드.서버:8080/api/고객조회/김준영/즐겨찾기-목록

부록) 이름만 들어본 URL, 뜯어보자.



https://www.ivips.co.kr/menu?categoryIdx=6

https://api.ivips.co.kr/v1/users/3/favorites

웹은 레스토랑에 비유할 수 있다.

내용: 참깨빵, 순쇠고기, 패티두장, 특별한소스, 양상추, 치즈, 피클, 양파

주문서 = 요청 본문

(요청)

햄부기 주세요

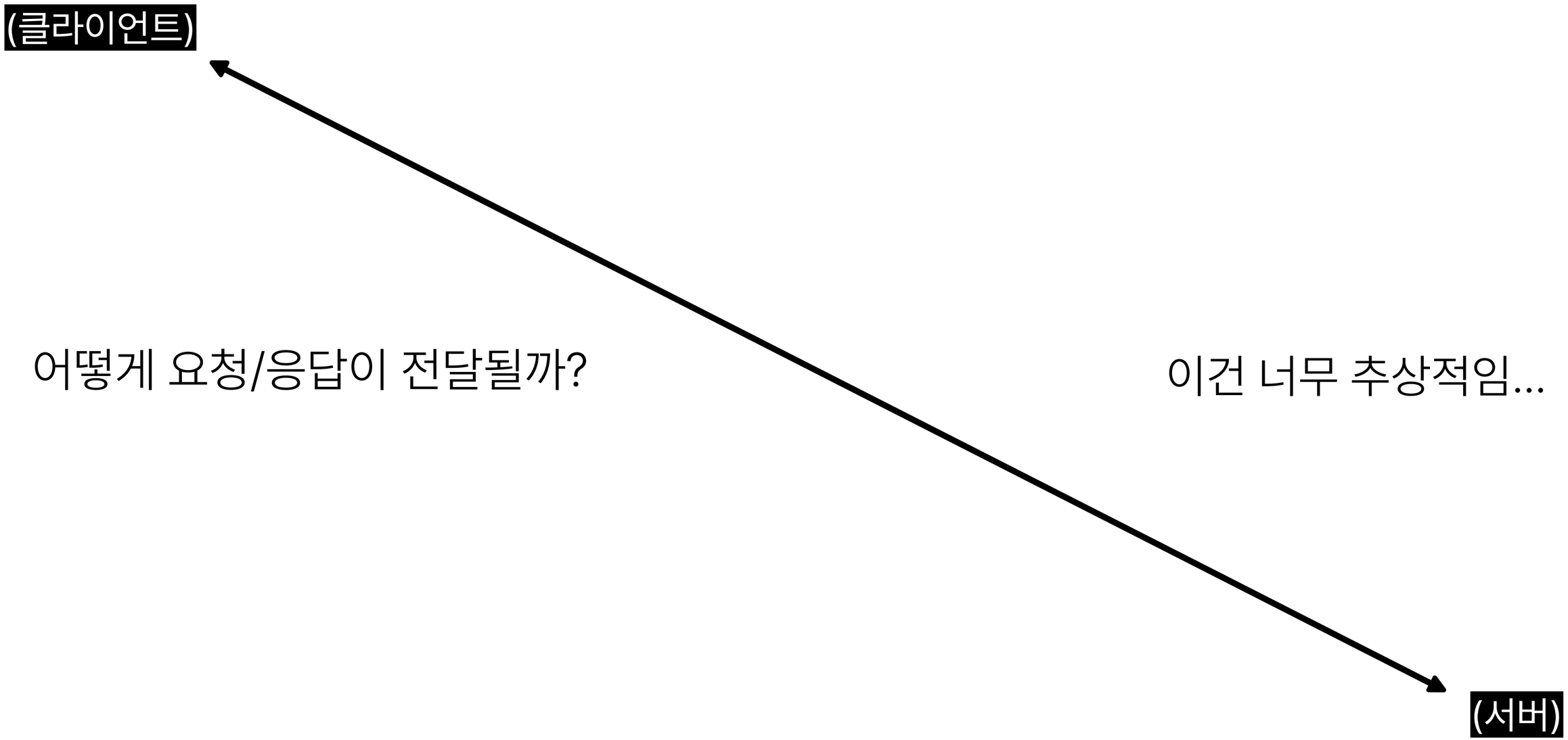
네 ^^

주문하신 음식 나왔습니다. 맛있게 드세요 ^^

(응답 본문)

(응답)

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초)



어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 도메인, DNS

(클라이언트)

" 도메인 "

https://google.com

→ 172.217.209.101 ...

```
/Users/user [ky0422@MacBookPro] [10:34]
> nslookup google.com
Server:         1.0.0.1
Address:        1.0.0.1#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.209.101
Name:   google.com
Address: 172.217.209.102
Name:   google.com
Address: 172.217.209.100
Name:   google.com
Address: 172.217.209.138
Name:   google.com
Address: 172.217.209.139
Name:   google.com
Address: 172.217.209.113
```

" DNS "

사람이 기억하기 쉬운 도메인 주소를 네트워크에서 사용하는 IP 주소로 바꾸는 과정

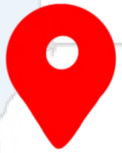
010-2980-1336 (기억하기 어려움)

(연락처 등록) " 김준영 " : 010-2980-1336 (기억하기 쉬움)

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 라우팅, 라우터

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 라우팅, 라우터

(서버)



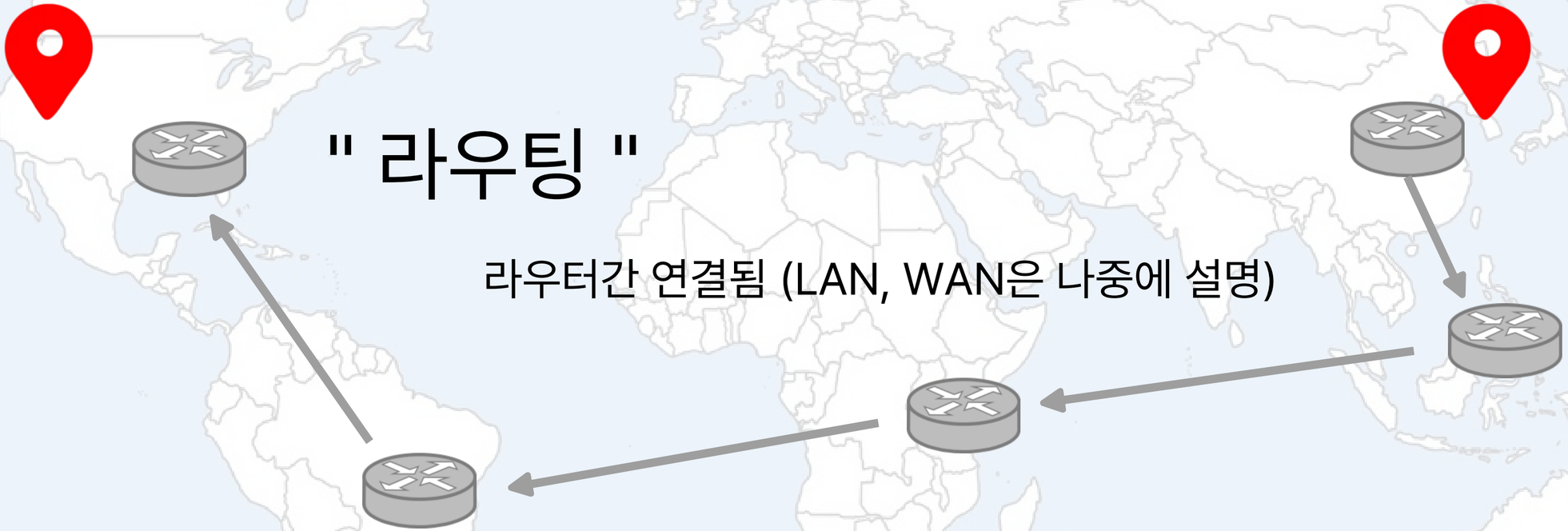
(클라이언트)

멀리 있는 서버에 어떻게 가지?

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 라우팅, 라우터

(서버)

(클라이언트)



" 라우팅 "

라우터간 연결됨 (LAN, WAN은 나중에 설명)

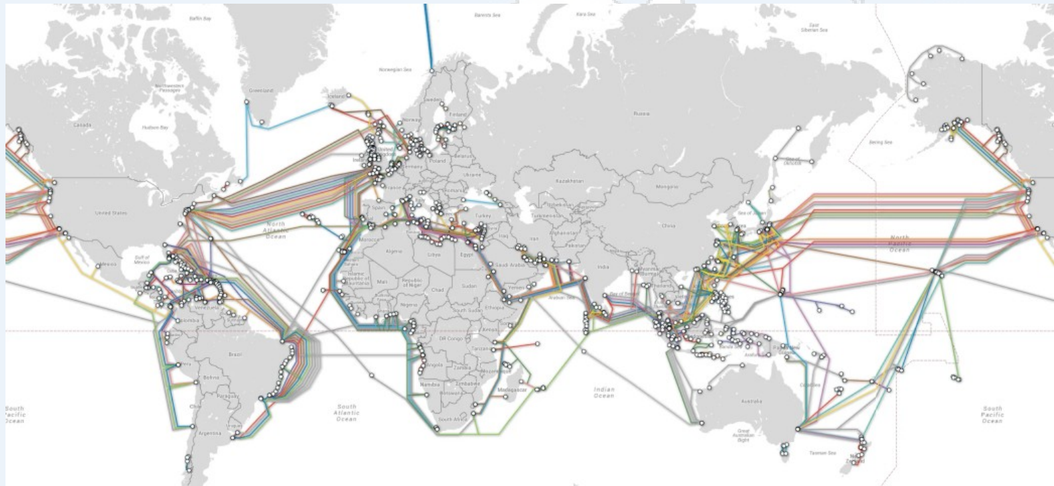
라우터 = 둘 이상의 서로 다른 네트워크를 연결하고, 목적지까지의 최적의 경로로 전송해주는 네트워크 장비

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 라우팅, 라우터

(서버)

(클라이언트)

(해저 케이블)



어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 라우팅, 라우터

물론 이때 LAN/WAN 공유기, 허브/스위치, 통신사 ISP 등등 매우 많은 개념이 있지만 지금은 생략..

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리

(앞서 최종 라우터에서 트래픽을 받음)

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리

(앞서 최종 라우터에서 트래픽을 받음)

HTML, CSS, JS, 이미지 등 정적인 콘텐츠를 제공하는 서버

Web Server (빠름)

Web Application Server (WAS)

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리

(앞서 최종 라우터에서 트래픽을 받음)

Web Server

동적 로직 처리, 데이터베이스 접근 등 동적 콘텐츠를 생성하는 애플리케이션

Web Application Server (WAS)

개발자가 서버를 개발한다는건 보통 이걸 다룸

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리

(앞서 최종 라우터에서 트래픽을 받음)

Web Server

Nginx, Apache 등등

Web Application Server (WAS)

Tomcat, JBoss 등등 + 수많은 개발 라이브러리/프레임워크 등등..

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리

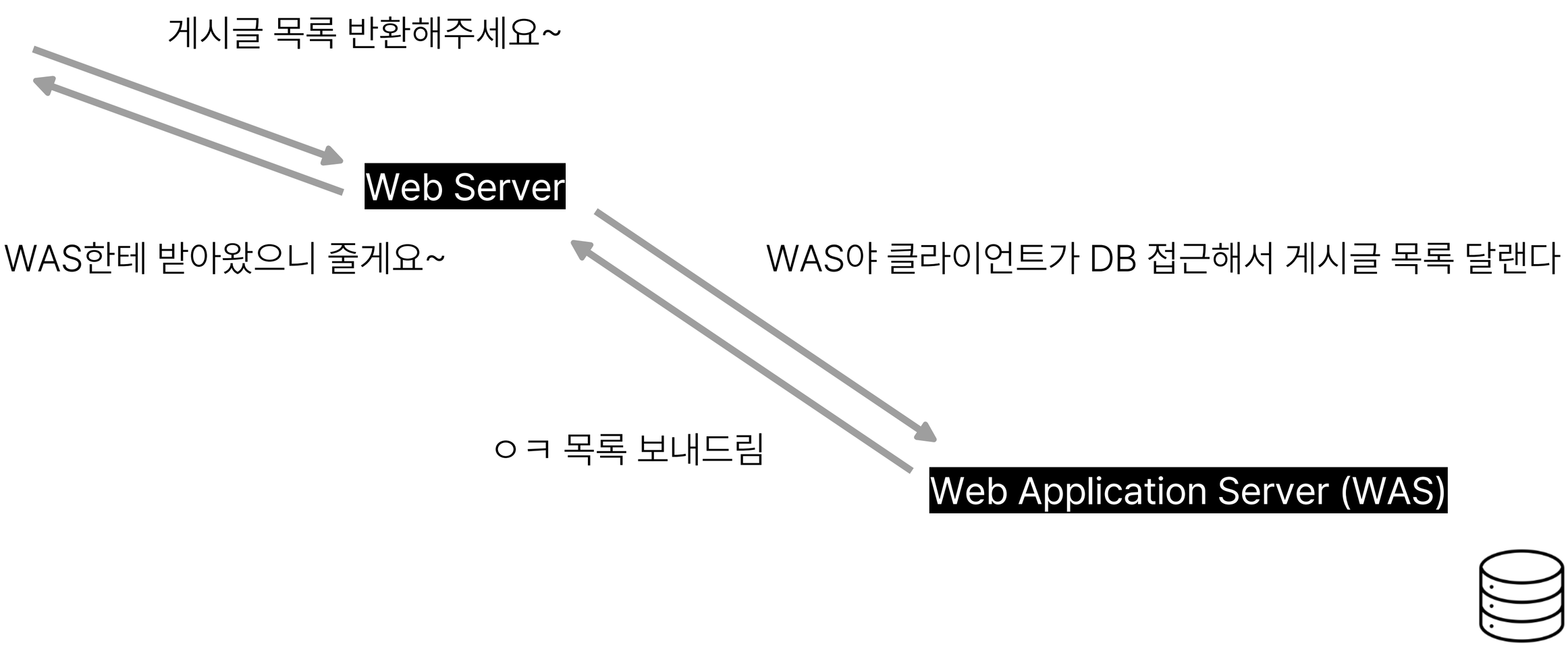
이미지 파일 주세요~

Web Server

내 선에서 해줄 수 있으니깐 바로 주겠음 ㅇㅇ

Web Application Server (WAS)

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리



어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 라우팅, 라우터

프론트엔드와 백엔드

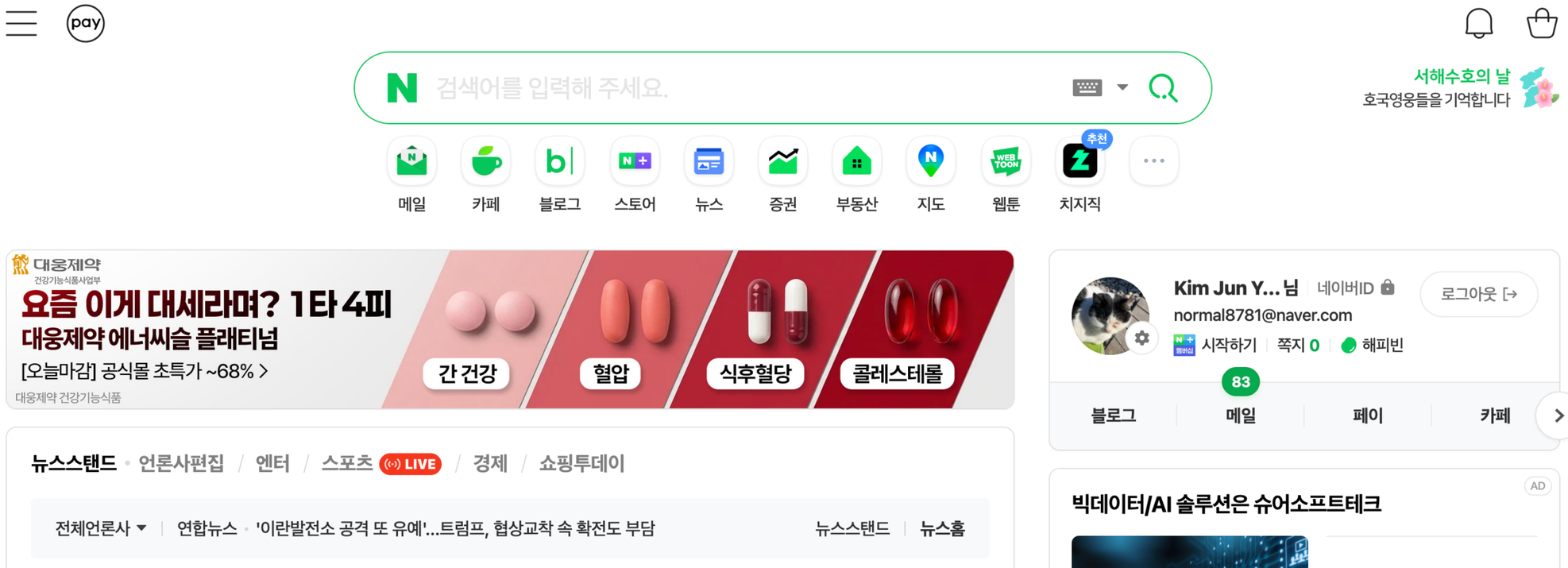
지금까지 웹에서의 통신 과정, 흐름을 살펴보았다면 이제 개발 분야로 넘어감

(프론트엔드/백엔드 개념은 IT 전반적으로 다양하게 사용되나, 여기선 웹 기준임)

프론트엔드와 백엔드 - Frontend

웹사이트나 앱에서 사용자가 눈으로 보고 상호작용하는 화면 (UI/UX)

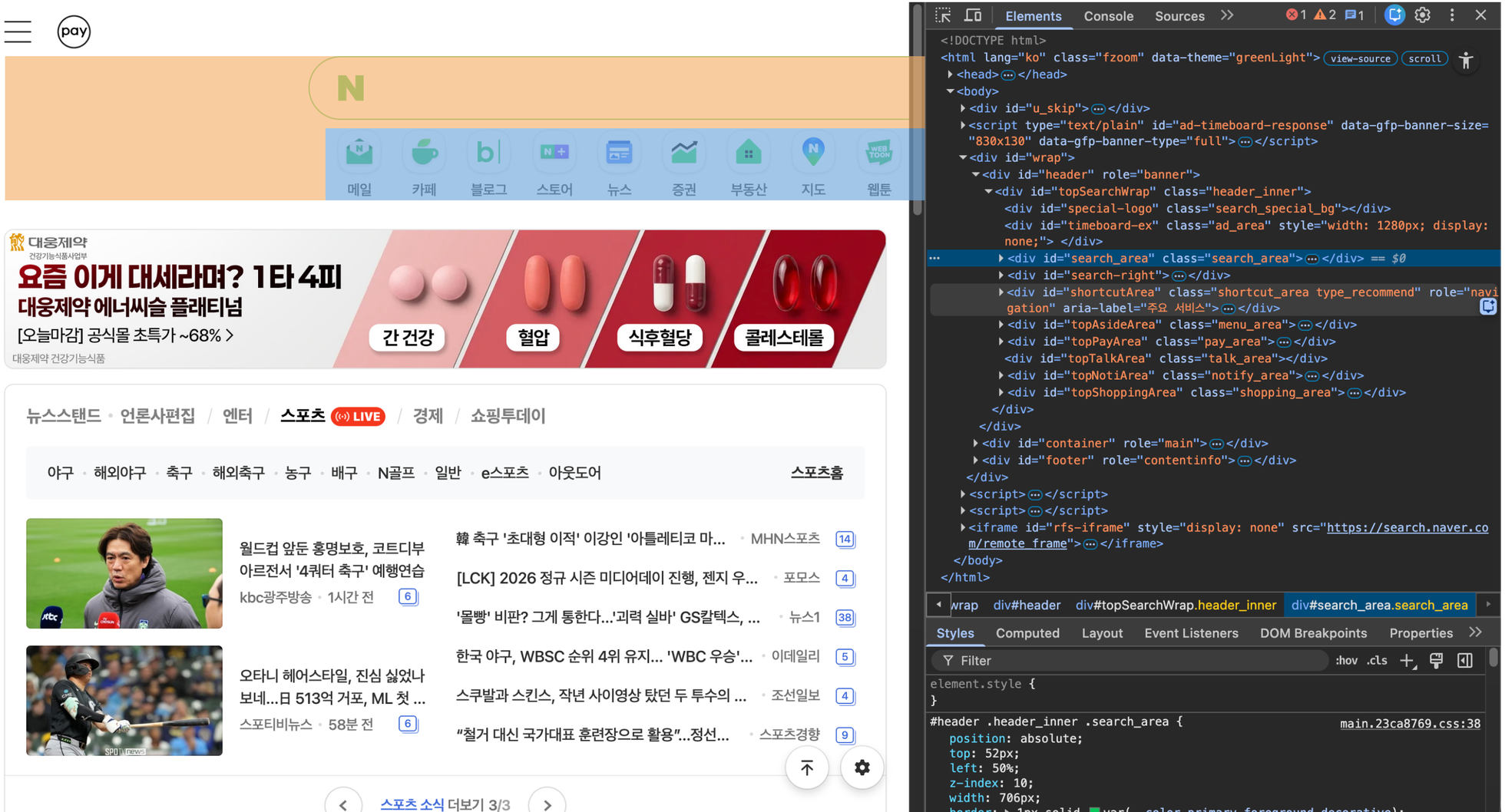
기본적으로 HTML, CSS, JS로 구성됨. (React, VueJS 등도 있으나 결국엔 HTML 코드 덩어리)



프론트엔드 - 웹 페이지는 사실 코드 덩어리다.

아무 웹 페이지에 접속하여 개발자 도구를 열어봅시다.

HTML(Hyper Text Markup Language)



프론트엔드 - 웹 페이지는 사실 코드 덩어리다.

HTML(Hyper Text Markup Language)

```html
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Planet data table</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Planet data table</h1>
  </body>
</html>
```

Ps, Ai 처럼 디자인을 마우스로 만드는게 아닌 코드를 작성해야함

실습) 자신만의 HTML 페이지 만들어보기

1. VSCode 실행하기.
    a. 반드시 VSCode만 사용해야 할까? (Feat. VSCode도 메모장일 뿐이다)
2. VSCode Live Server 익스텐션 설치하기
    a. VSCode가 메모장보다 훨씬 강력한 도구가 된 이유.
3. 디렉토리 열기 및 HTML 문서 작성하기 (자기 소개 페이지 만들어보기)
4. !를 입력후 탭을 쳐서 나오는 내용은 무슨 의미일까?
    a. 메타데이터의 중요성 (Feat. 검색 엔진 노출, SEO 등)
5. 주요 태그 설명 (h1...h6, p, span, img, input 등)
6. Live Server로 실행해보기
    a. 퀴즈) Live Server로 실행하면 서버가 실행되는걸까?
    b. 스스로 찾아보기) 다른 컴퓨터에서 내 페이지에 접속되게 해보기

실습) 자신만의 HTML 페이지 만들어보기

1. VSCode 실행하기.
    a. 반드시 VSCode만 사용해야 할까? (Feat. VSCode도 메모장일 뿐이다)
2. VSCode Live Server 익스텐션 설치하기
    a. VSCode가 메모장보다 훨씬 강력한 도구가 된 이유.
3. 디렉토리 열기 및 HTML 문서 작성하기 (자기 소개 페이지 만들어보기)
4. !를 입력후 탭을 쳐서 나오는 내용은 무슨 의미일까?
    a. 메타데이터
5. 주요 태그 설명
6. Live Server로 실행해보기
    a. 퀴즈) Live Server로 실행하면 서버가 실행되는걸까?
    b. 스스로 찾아보기) 다른 컴퓨터에서 내 페이지에 접속되게 해보기

시간 관계상 넘어갔었는데, Live Server도 로컬(내 컴퓨터)에서 간단하게 백엔드 서버를 만들어주는 확장임.

실습) 자신만의 HTML 페이지 만들어보기

1. VSCode 실행하기.
    a. 반드시 VSCode만 사용해야 할까? (Feat. VSCode도 메모장일 뿐이다)
2. VSCode Live Server 익스텐션 설치하기
    a. VSCode가 메모장보다 훨씬 강력한 도구가 된 이유.
3. 디렉토리 열기 및 HTML 문서 작성하기 (자기 소개 페이지 만들어보기)
4. !를 입력후 탭을 쳐서 나오는 내용은 무슨 의미일까?
    a. 메타데이터의 중요성 (Feat. 검색 엔진 노출, SEO 등)
5. 주요 태그 설명 (h1...h6, p, span, img, input 등)
6. Live Server로 실행해보기
    a. 퀴즈) Live Server로 실행하면 서버가 실행되는걸까?
    b. 스스로 찾아보기) 다른 컴퓨터에서 내 페이지에 접속되게 해보기

localhost:8080 이러한 주소가 내 컴퓨터 내에서 돌아가는 서버를 의미.

실습) 자신만의 HTML 페이지 만들어보기

1. VSCode 실행하기.
    a. 반드시 VSCode만 사용해야 할까? (Feat. VSCode도 메모장일 뿐이다)
2. VSCode Live Server 익스텐션 설치하기
    a. VSCode가 메모장보다 훨씬 강력한 도구가 된 이유.
3. 디렉토리 열기 및 HTML 문서 작성하기 (자기 소개 페이지 만들어보기)
4. !를 입력후 탭을 쳐서 나오는 내용은 무슨 의미일까?
    a. 메타데이터의 중요성 (Feat. 검색 엔진 노출, SEO 등)
5. 주요 태그 설명 (h1...h6, p, span, img, input 등)
6. Live Server로 실행해보기
    a. 퀴즈) Live Server로 실행하면 서버가 실행되는걸까?
    b. 스스로 찾아보기) 다른 컴퓨터에서 내 페이지에 접속되게 해보기

Live Server는 HTML, CSS, 이미지 등을 서빙하는 정적 서버인 것.

프론트엔드만 있으면 무슨 문제가 생길까?

로그인하고 싶습니다. 프론트엔드에 로그인 메뉴가 있네?

프론트엔드만 있으면 무슨 문제가 생길까?

로그인하고 싶습니다. 프론트엔드에 로그인 메뉴가 있네?

로그인 처리는 데이터베이스 접근, 세션/토큰 생성 등
내부 비즈니스 로직 실행이 필요함

프론트엔드만 있으면 무슨 문제가 생길까?

로그인하고 싶습니다. 프론트엔드에 로그인 메뉴가 있네?

로그인 처리는 데이터베이스 접근, 세션/토큰 생성 등
내부 비즈니스 로직 실행이 필요함

" 프론트엔드는 그러한 작업을 처리하지 못함. 단순히 화면에 표시해주는 것. "

백엔드는 눈에 보이지 않는다.

Backend의 본질적인 뜻은 "눈에 보이지 않는" 이라는 의미를 가짐.

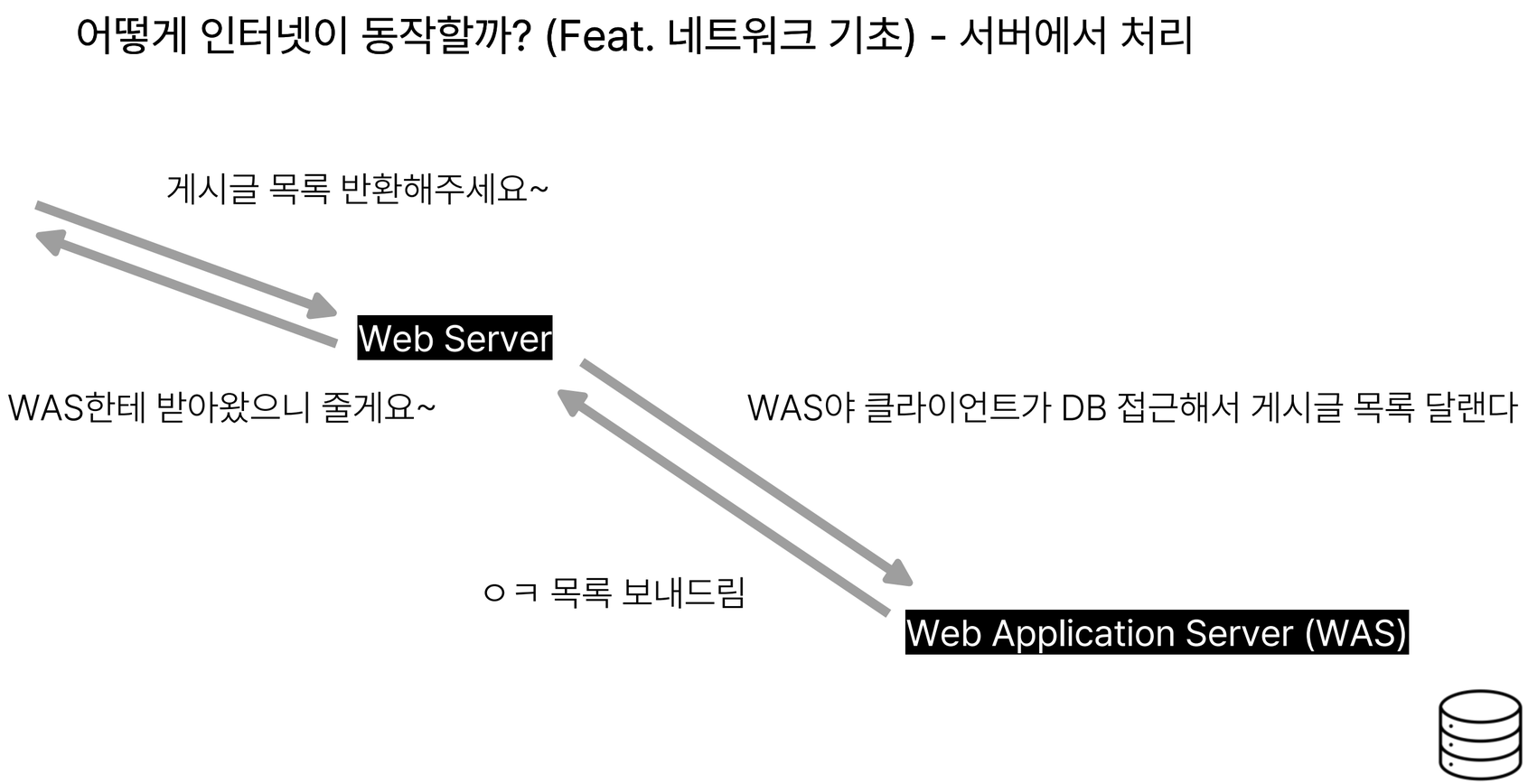
로그인 처리는 데이터베이스 접근, 세션/토큰 생성 등
내부 비즈니스 로직 실행이 필요함

위와 같은 작업을 처리하는 것은 눈에 보이지 않는 백엔드의 영역임

백엔드 애플리케이션 = 그러한 작업을 처리하는 애플리케이션.

백엔드는 눈에 보이지 않는다.

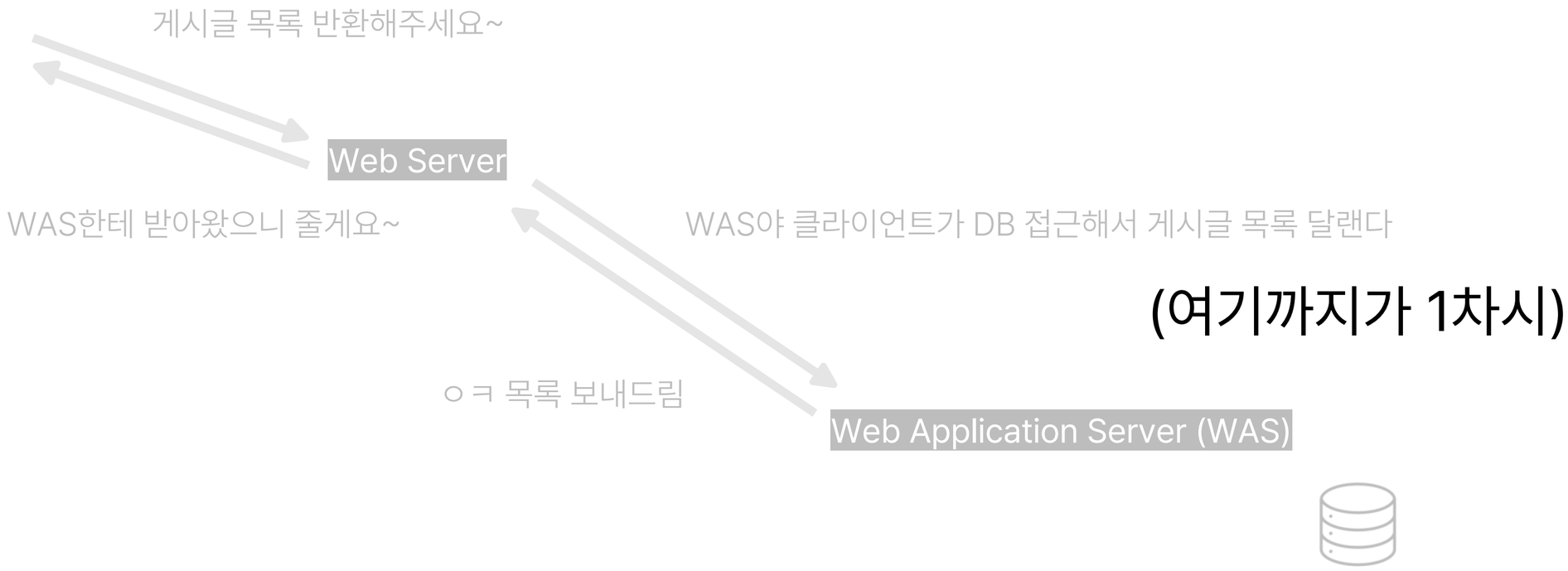
아까 다뤘던 "서버"는 백엔드 애플리케이션을 실행해주는 공간임.
개념적으로는 다른 개념이지만, 혼용해도 큰 문제는 없음.



백엔드는 눈에 보이지 않는다.

아까 다뤘던 "서버"는 백엔드 애플리케이션을 실행해주는 공간임.
개념적으로는 다른 개념이지만, 혼용해도 큰 문제는 없음.

어떻게 인터넷이 동작할까? (Feat. 네트워크 기초) - 서버에서 처리



백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

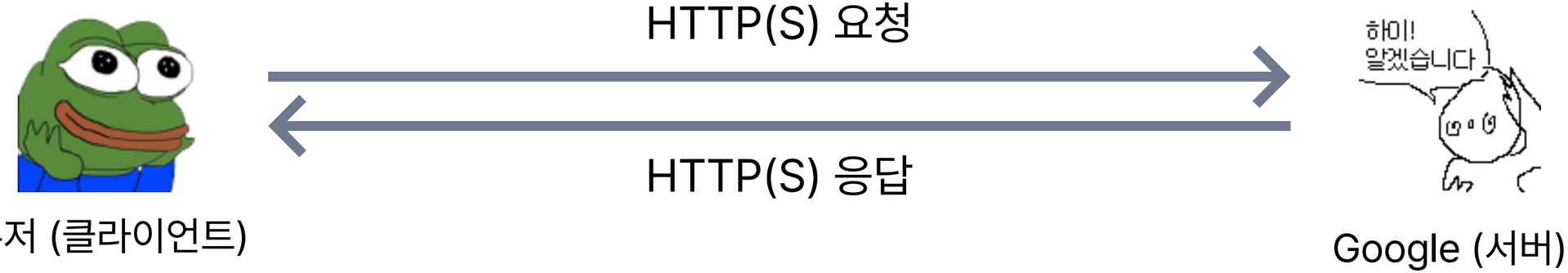
https://www.google.com/search?q=연신내역+맛집

유저 (클라이언트)

Google (서버)

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

https://www.google.com/search?q=연신내역+맛집



유저 (클라이언트)

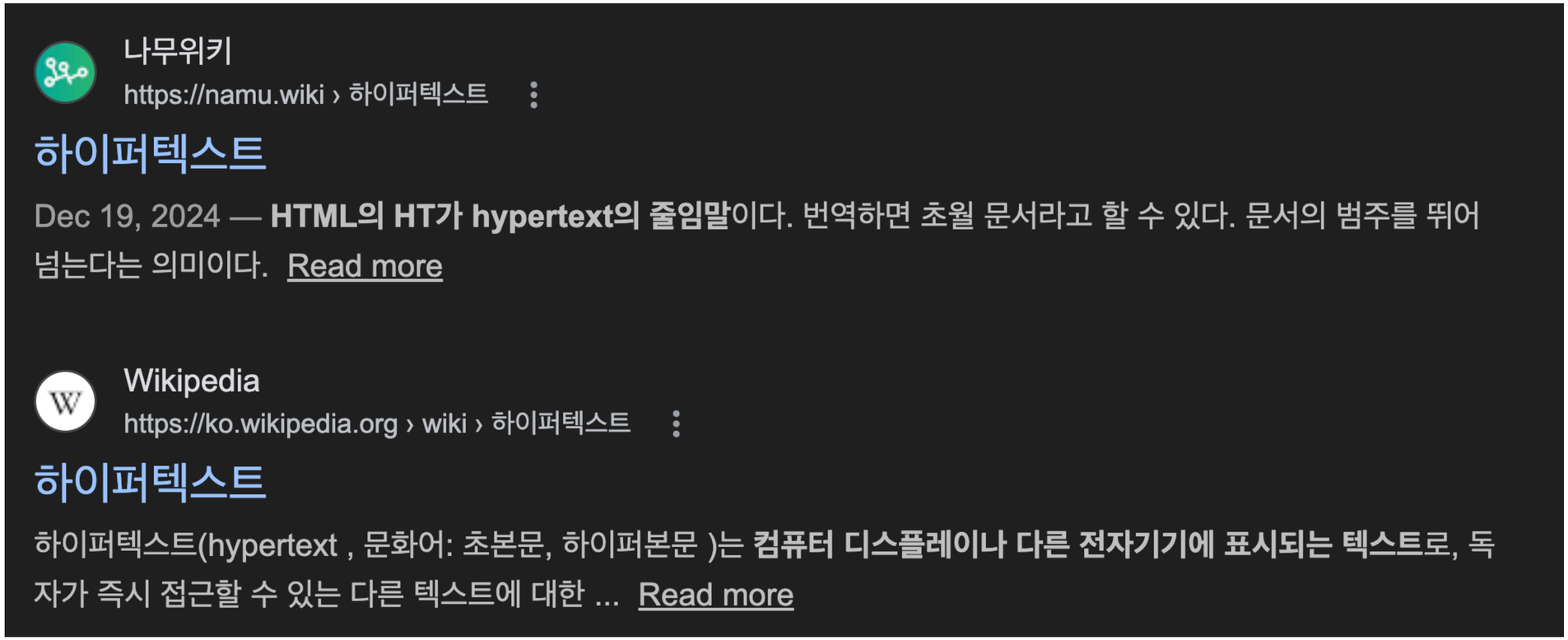
Google (서버)

HTTP = Hyper Text Transfer Protocol

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

HTTP = Hyper Text Transfer Protocol

Hyper Text =



백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

HTTP = Hyper Text Transfer Protocol

Hyper Text = 대충 HTML 문서라고 생각하면 됨

대신 이 프로토콜을 통해서 HTML 문서 뿐만 아니라 CSS, JS, 이미지 등 여러 리소스를 주고 받는 것

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

HTTP = Hyper Text Transfer Protocol

전송

Protocol = 통신 규약

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

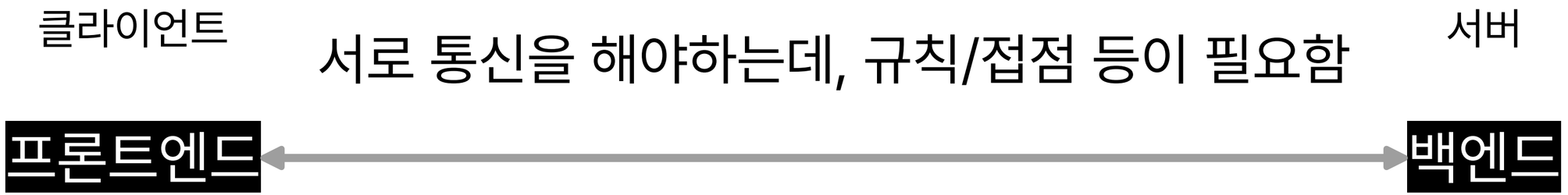
HTTP = Hyper Text Transfer Protocol

HTML 문서 및 웹(WWW)과 관련된 모든 것을 전송하는 통신 규약

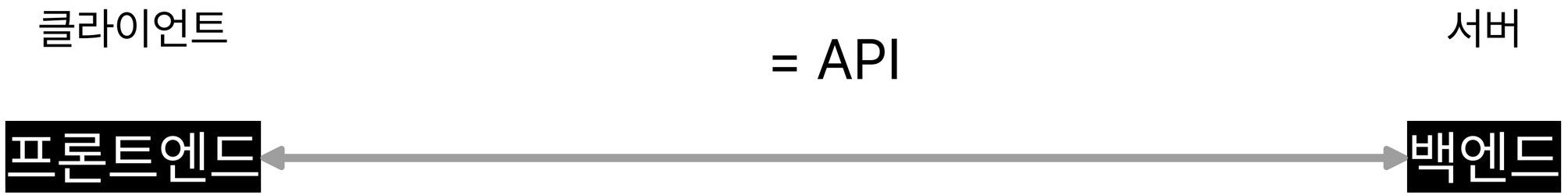
= 웹 통신시 필수적인 통신 규약(프로토콜)

굳이 HTML 문서 뿐만 아니라 거의 모든 웹 리소스, API 호출 등에서 모두 사용됨

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

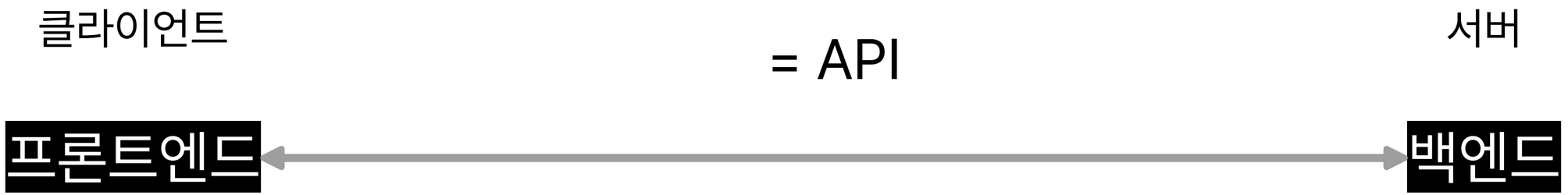


백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)



API? Application Programming Interface

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)



API? Application Programming Interface

두 소프트웨어가 서로 데이터를 주고받을 수 있게 하기 위한 규칙 및 접점의 정의 (인터페이스)

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

https://www.ivips.co.kr/menu?categoryIdx=6

https://api.ivips.co.kr/v1/users/3/favorites

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

프론트엔드 URL

https://www.ivips.co.kr/menu?categoryIdx=6

https://api.ivips.co.kr/v1/users/3/favorites

백엔드 API

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

프론트엔드 URL

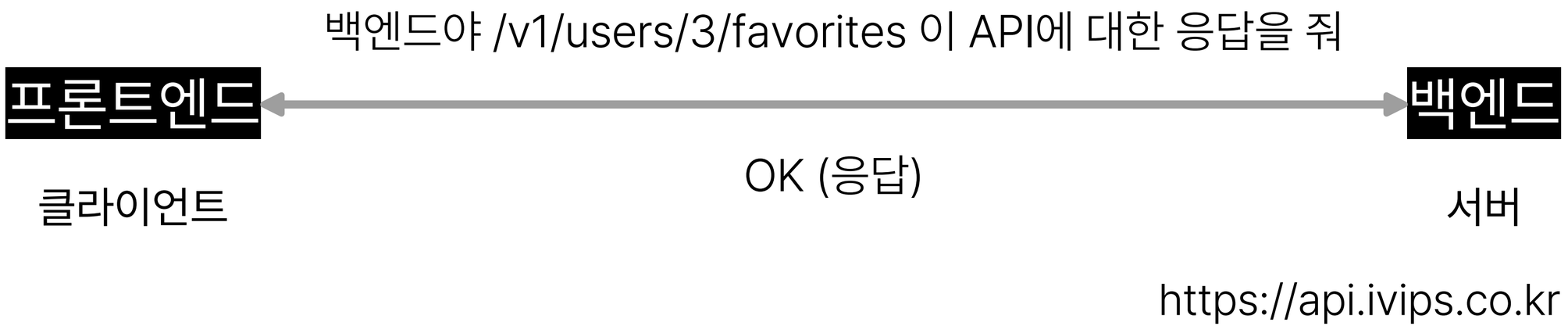
https://www.ivips.co.kr/menu?categoryIdx=6

https://api.ivips.co.kr/v1/users/3/favorites

백엔드 API

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)



백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

GET /v1/users/3/favorites　　POST /v1/login　　PUT /v1/posts/34

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

GET /v1/users/3/favorites　　POST /v1/login　　PUT /v1/posts/34

HTTP 프로토콜을 기반으로, URI와 메서드(GET, POST, PUT, DELETE 등)를 통해 자원의 상태를 주고받는 API 유형 → **REST API**

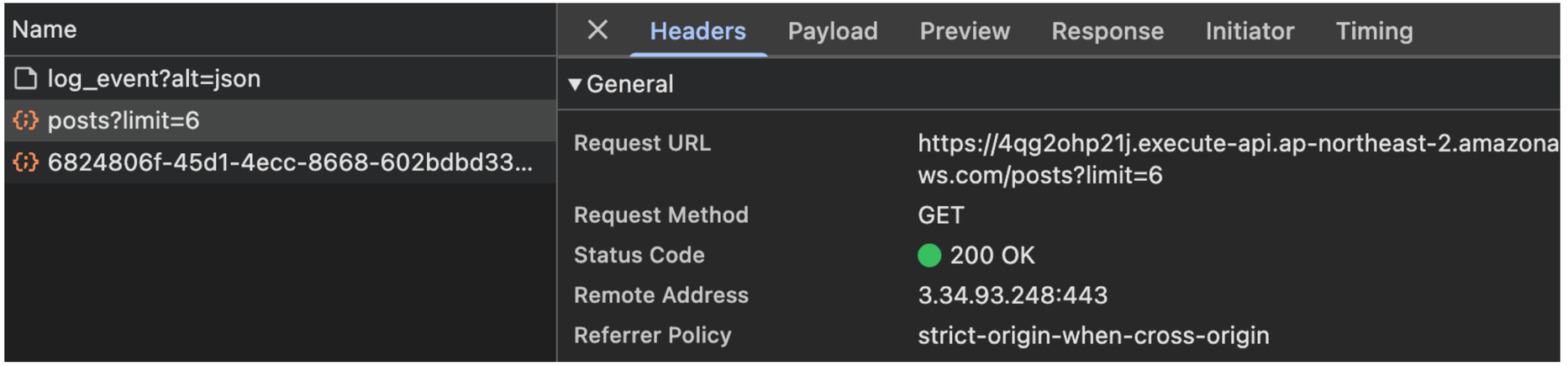
보통 JSON이나 XML을 통해 요청/응답을 주고받고, 일반적으로 Stateless함.

백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

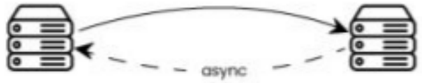
GET /v1/users/3/favorites　　POST /v1/login　　PUT /v1/posts/34

HTTP 프로토콜을 기반으로, URI와 메서드(GET, POST, PUT, DELETE 등)를 통해 자원의 상태를 주고받는 API 유형 → **REST API** (또는 RESTful)

보통 JSON이나 XML을 통해 요청/응답을 주고받고, 일반적으로 Stateless함.



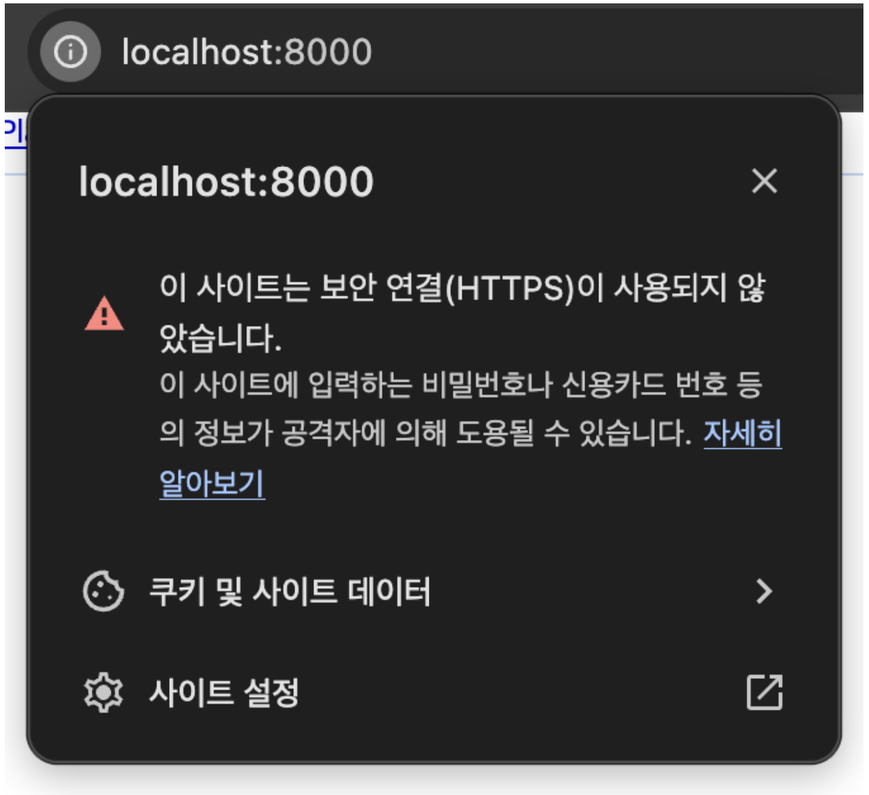
백엔드 - 웹은 소통할때 "HTTP API"를 사용한다. (Feat. REST API는 뭘까?)

Style	Illustration	Use Cases
SOAP		XML-based for enterprise applications
RESTful		Resource-based for web servers
GraphQL		Query language reduce network load
gRPC		High performance for microservices
WebSocket		Bi-directional for low-latency data exchange
Webhook		Asynchronous for event-driven application

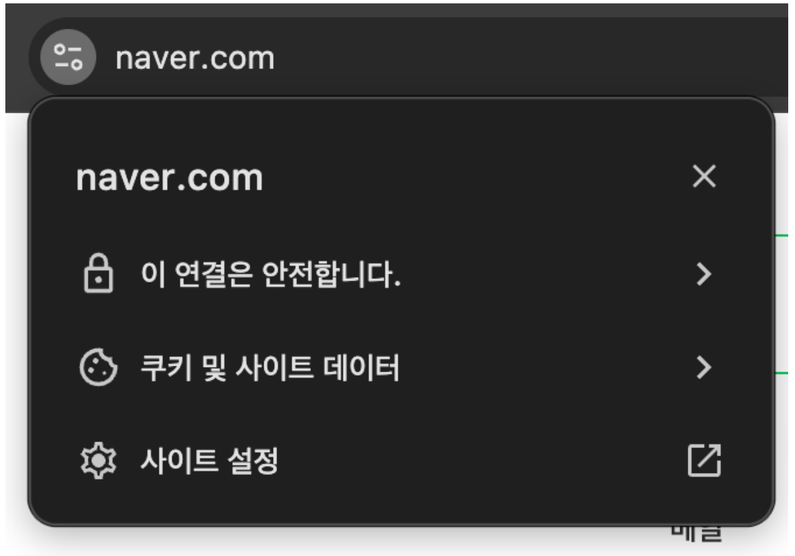
REST API는 인터넷/웹 통신에 있어 가장 많이 사용되지만, 이 밖에도 용도나 장단점에 따라 다양한 HTTP 기반 스타일/프로토콜이 있음.

특히 gRPC, WebSocket, WebHook 등은 실무에서 많이 사용되는 개념이니 공부해두면 굳

퀴즈) HTTP와 HTTPS는 뭔 차이일까?



HTTP

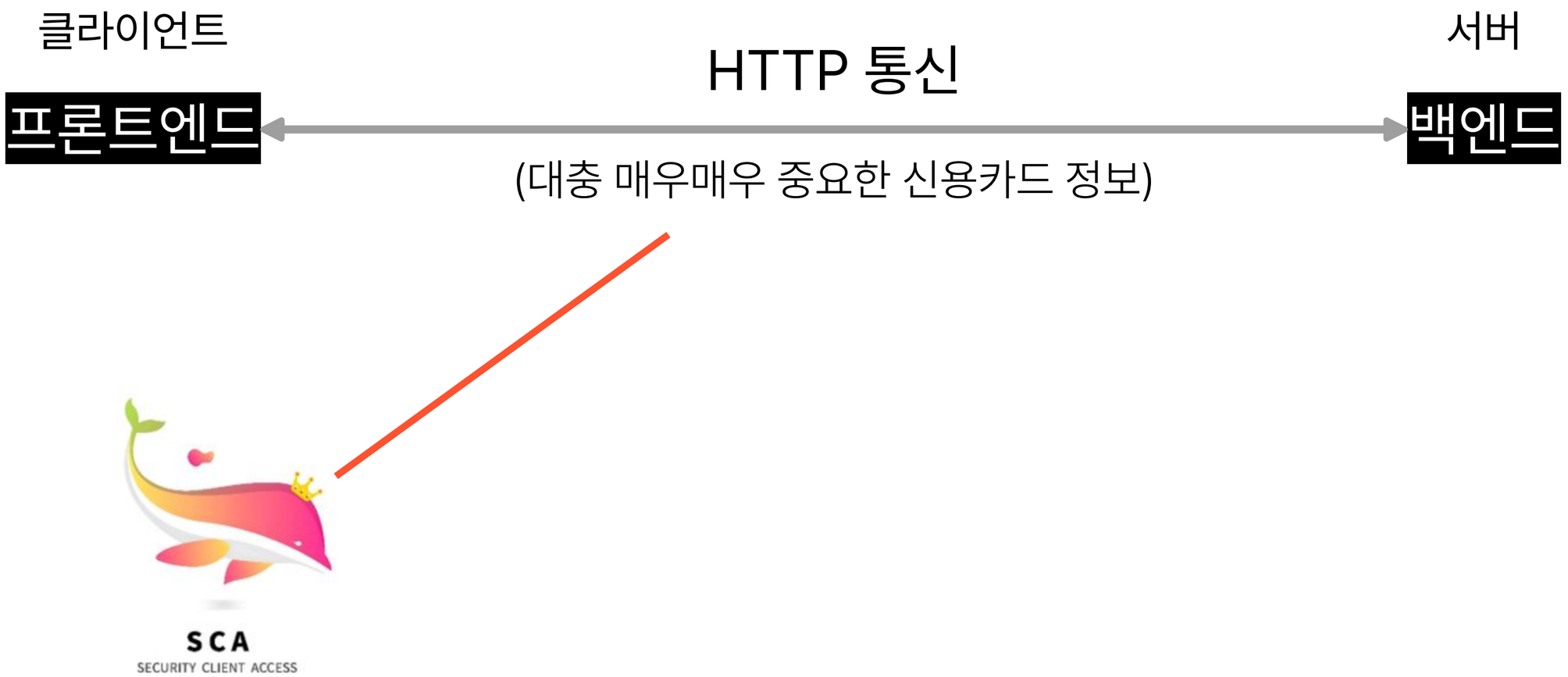


HTTPS

퀴즈) HTTP와 HTTPS는 뭔 차이일까?

퀴즈) HTTP와 HTTPS는 뭔 차이일까?

퀴즈) HTTP와 HTTPS는 뭔 차이일까?



퀴즈) HTTP와 HTTPS는 뭔 차이일까?

클라이언트

서버

HTTP 통신

프론트엔드 ⟷ 백엔드

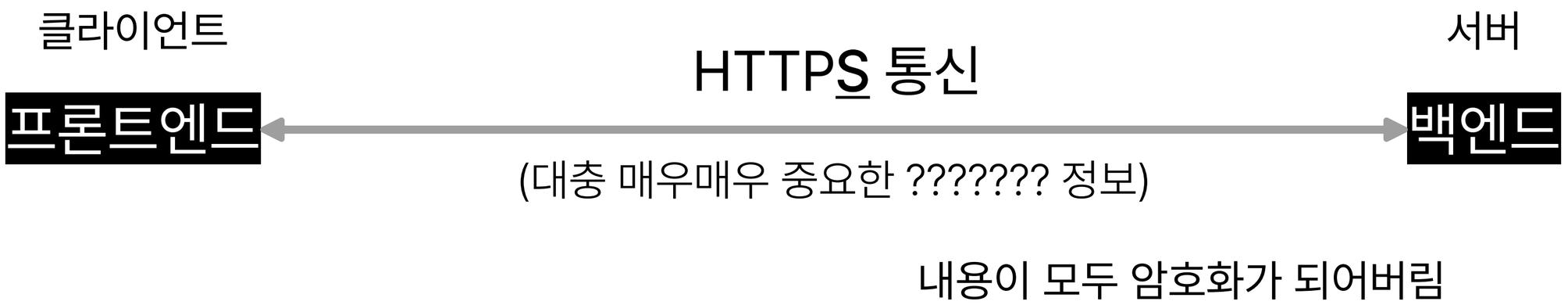
(대충 매우매우 중요한 신용카드 정보)

SCA
SECURITY CLIENT ACCESS

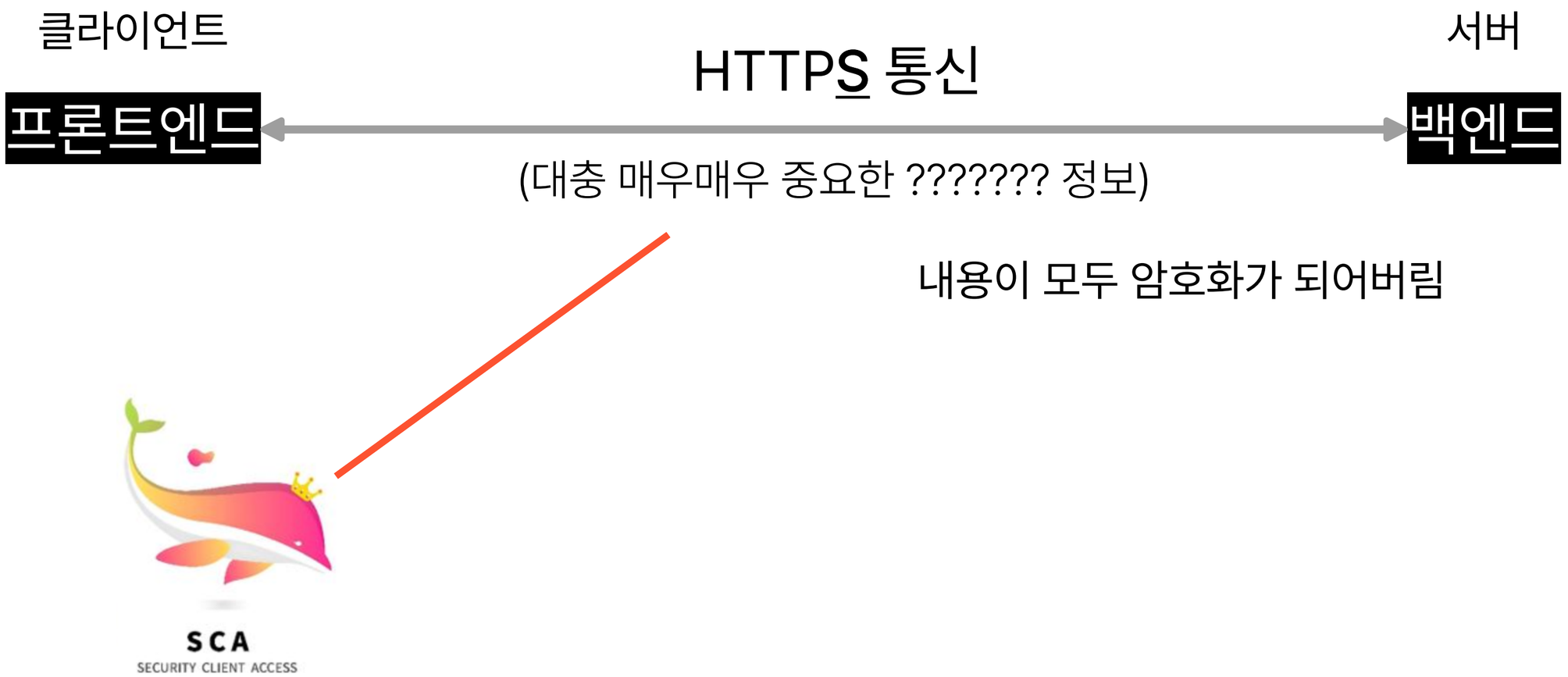
요청/응답 내용(페이로드)를 해커나 공격자가 탈취할 수 있음.

퀴즈) HTTP와 HTTPS는 뭔 차이일까?

퀴즈) HTTP와 HTTPS는 뭔 차이일까?



퀴즈) HTTP와 HTTPS는 뭔 차이일까?



퀴즈) HTTP와 HTTPS는 뭔 차이일까?

클라이언트

서버

HTTPS 통신

프론트엔드 ⟷ 백엔드

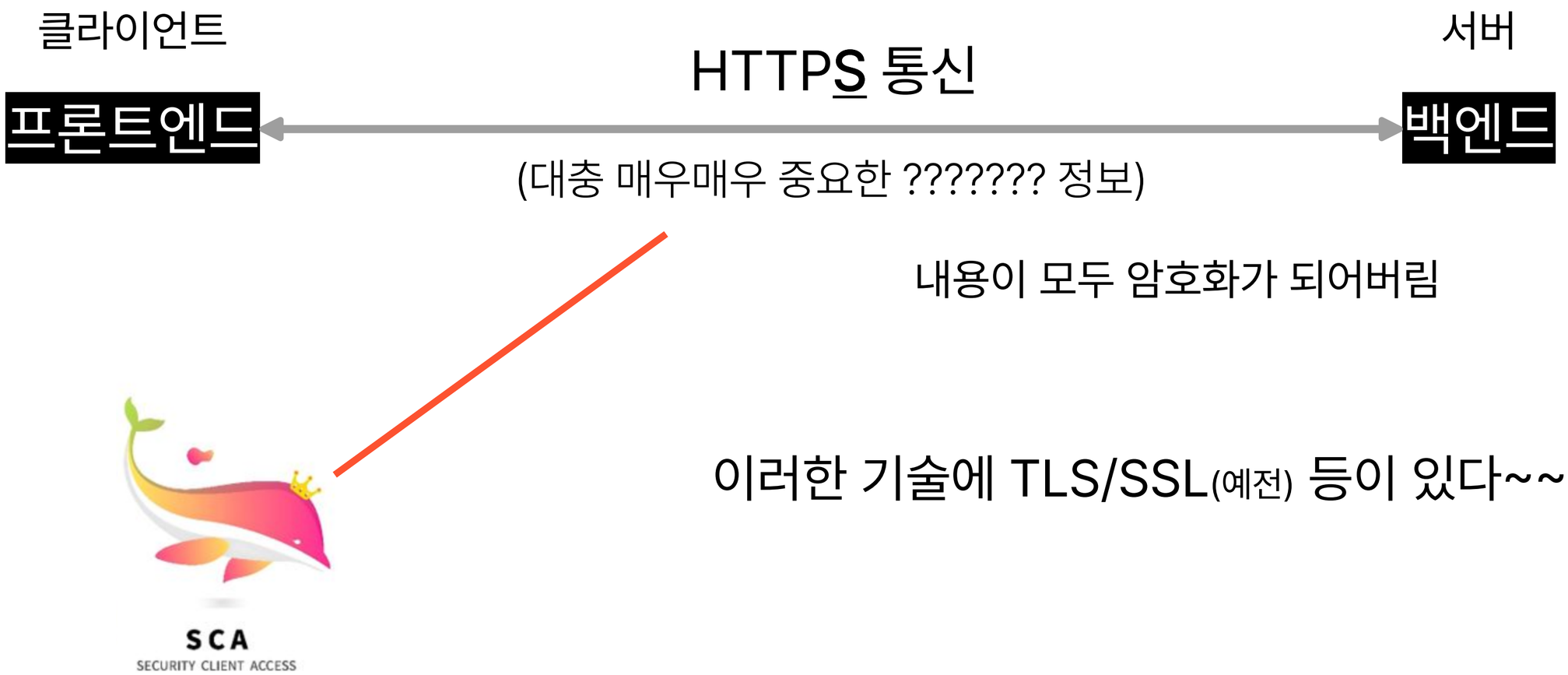
(대충 매우매우 중요한 ??????? 정보)

내용이 모두 암호화가 되어버림

SCA
SECURITY CLIENT ACCESS

내용 탈취 자체는 가능할 수 있지만, 탈취를 해도 그 내용을 알 수 없음

퀴즈) HTTP와 HTTPS는 뭔 차이일까?



퀴즈) HTTP와 HTTPS는 뭔 차이일까?

내용이 모두 암호화가 되어버림

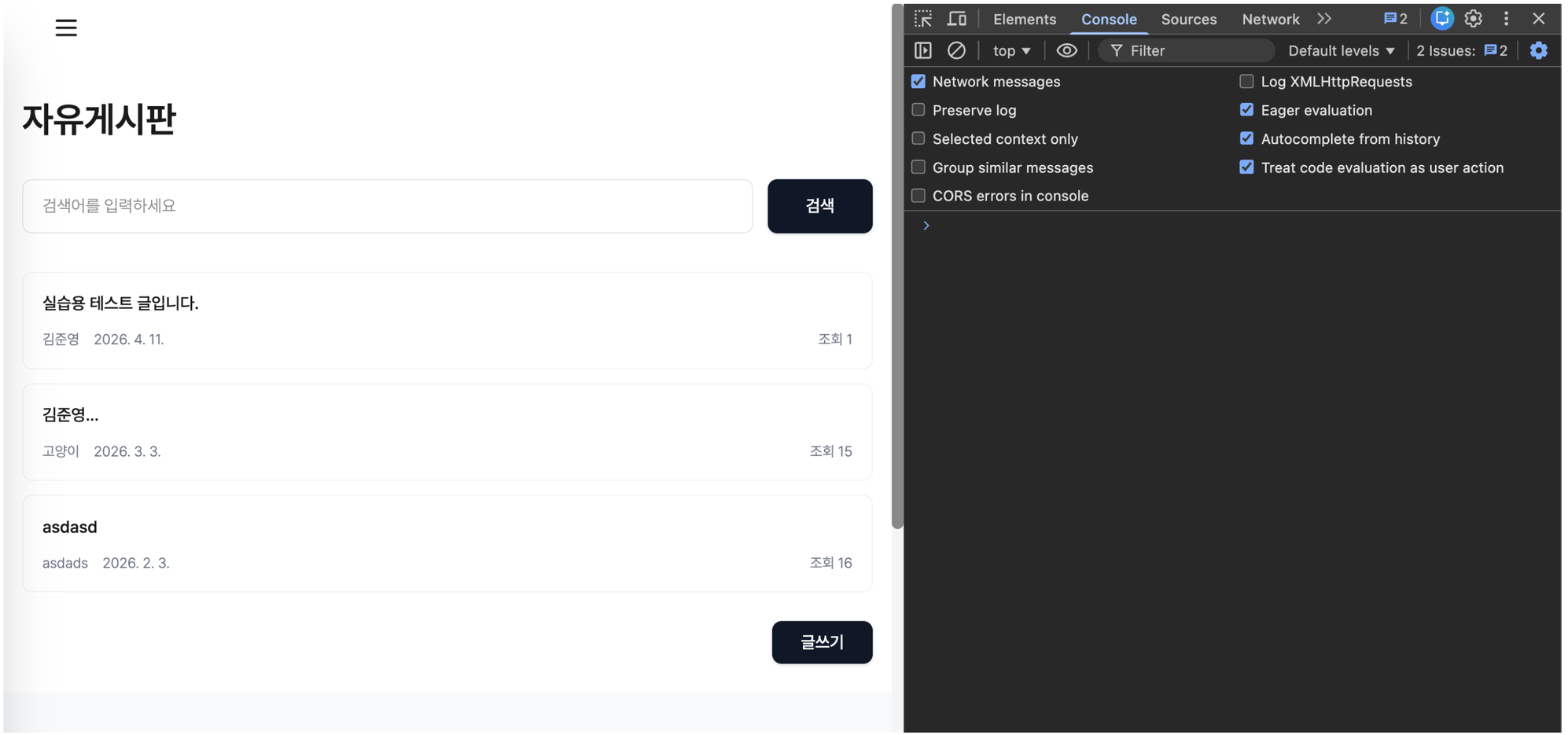
세부적인 내용은 언젠가 따로 다뤄볼 예정임

부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)

접속해볼 웹 페이지 : https://null4u.cloud/board

접속 후 개발자 도구(F12) 실행

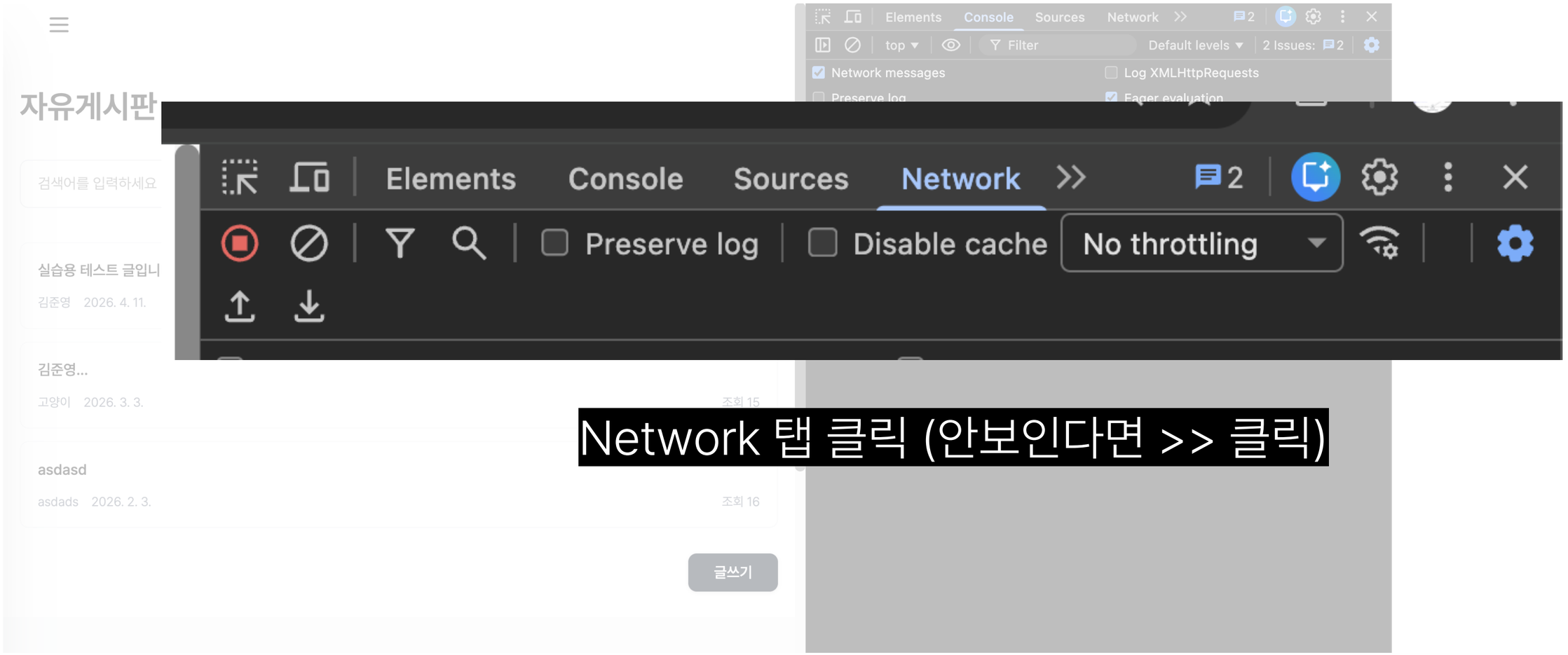


부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)

접속해볼 웹 페이지 : https://null4u.cloud/board

접속 후 개발자 도구(F12) 실행

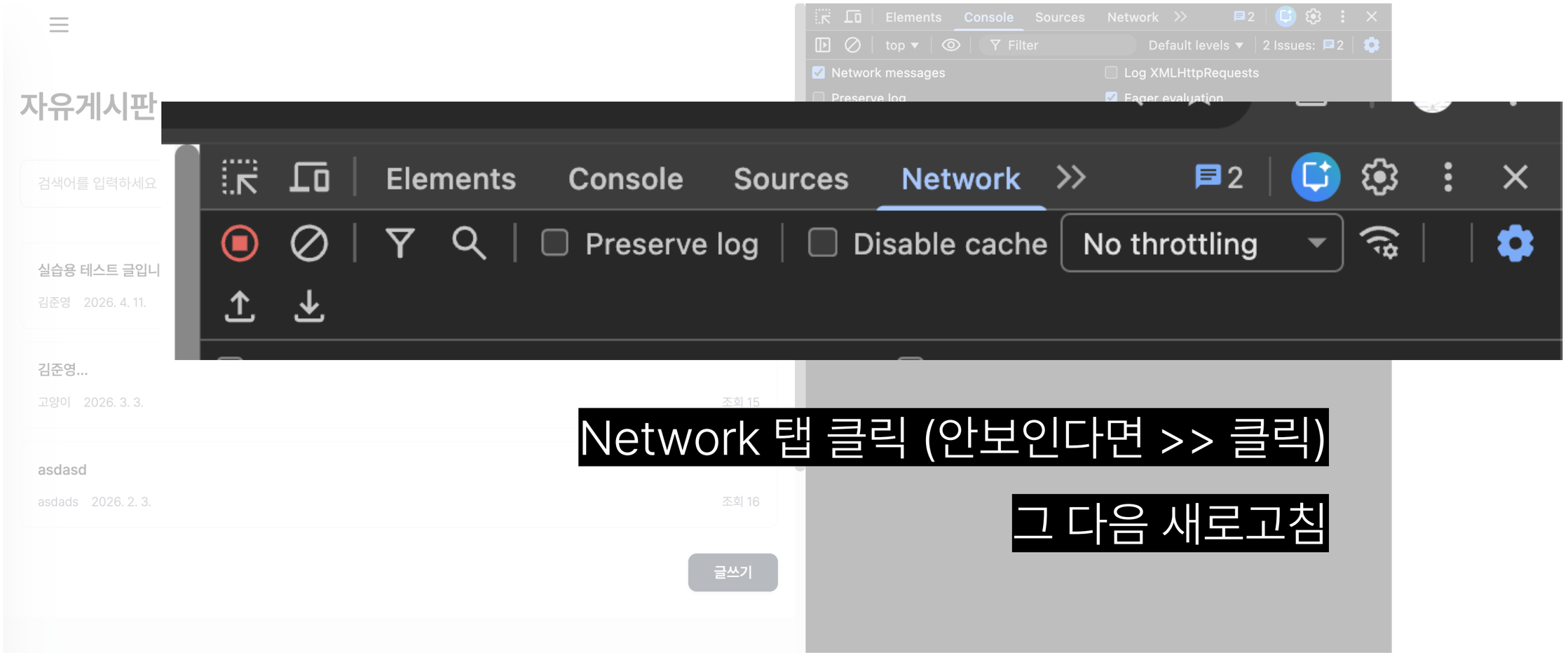


부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)

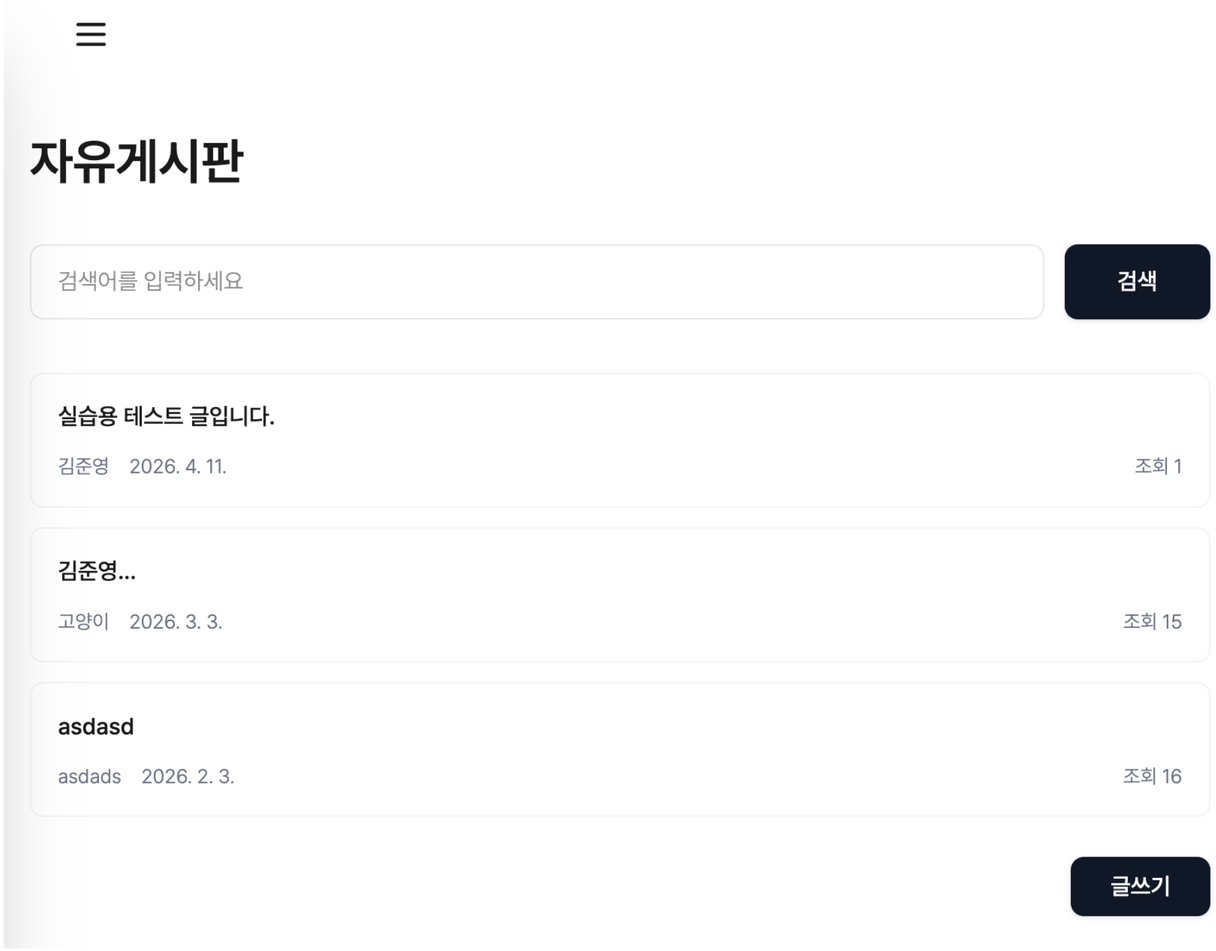
접속해볼 웹 페이지 : https://null4u.cloud/board

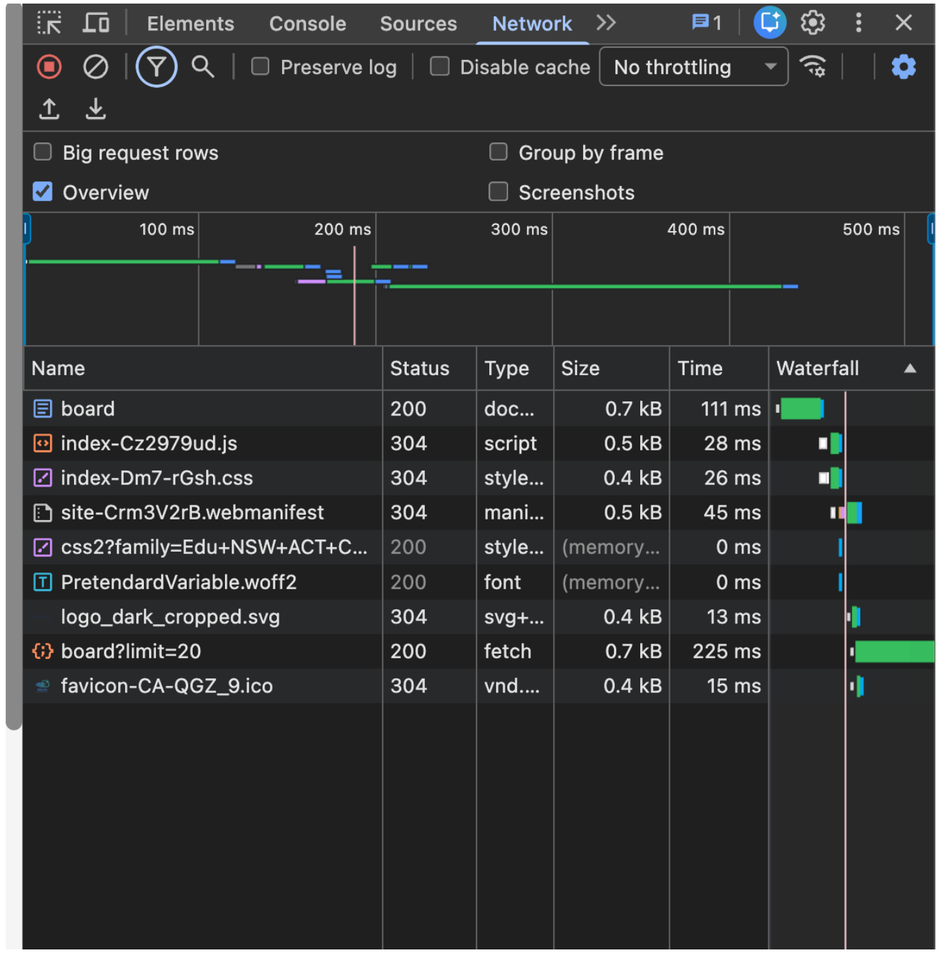
접속 후 개발자 도구(F12) 실행



부록/실습) HTTP 요청/응답을 뜯어보자!

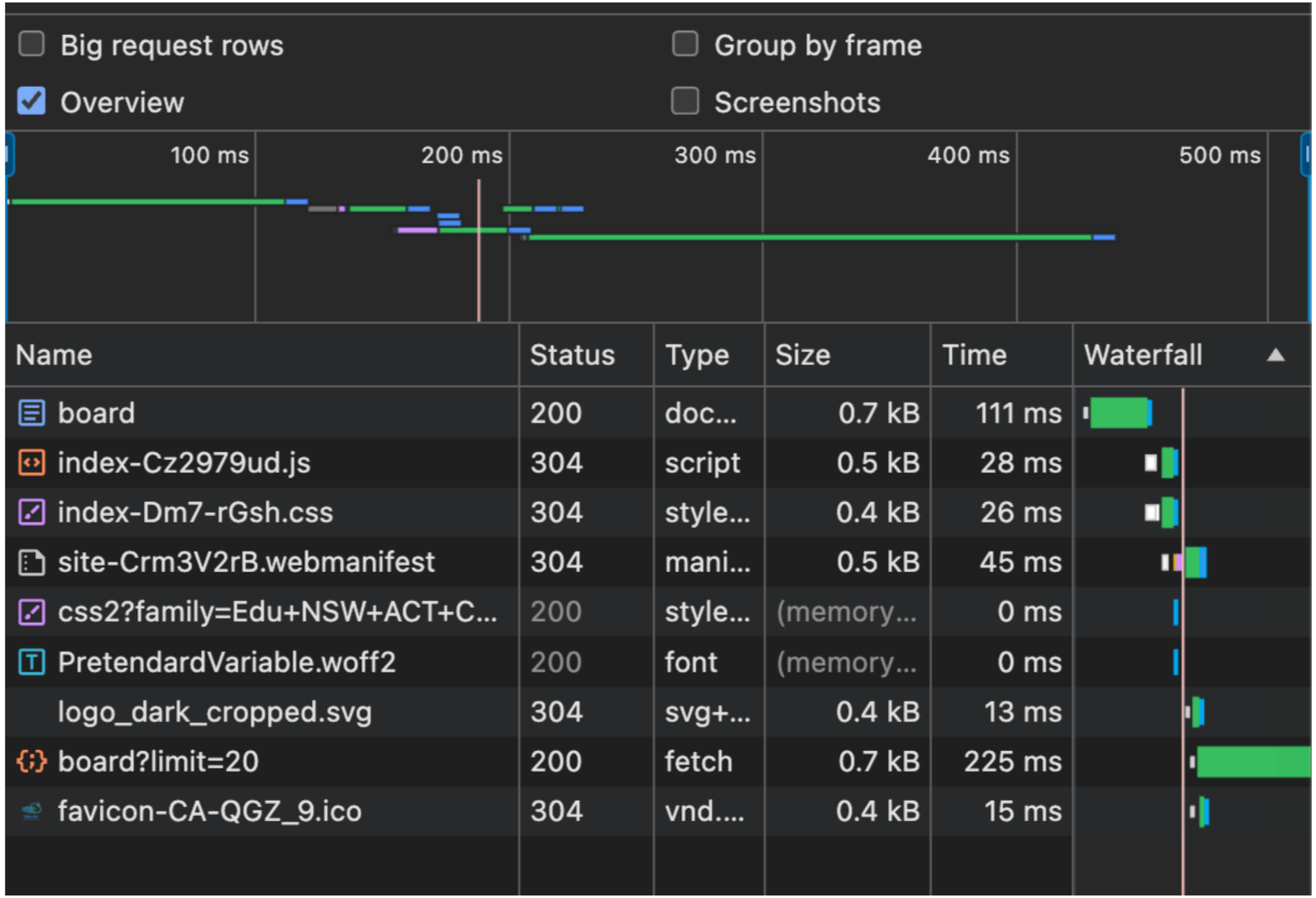
(따라하면 좋겠지만 그냥 보기만 해도 됨)





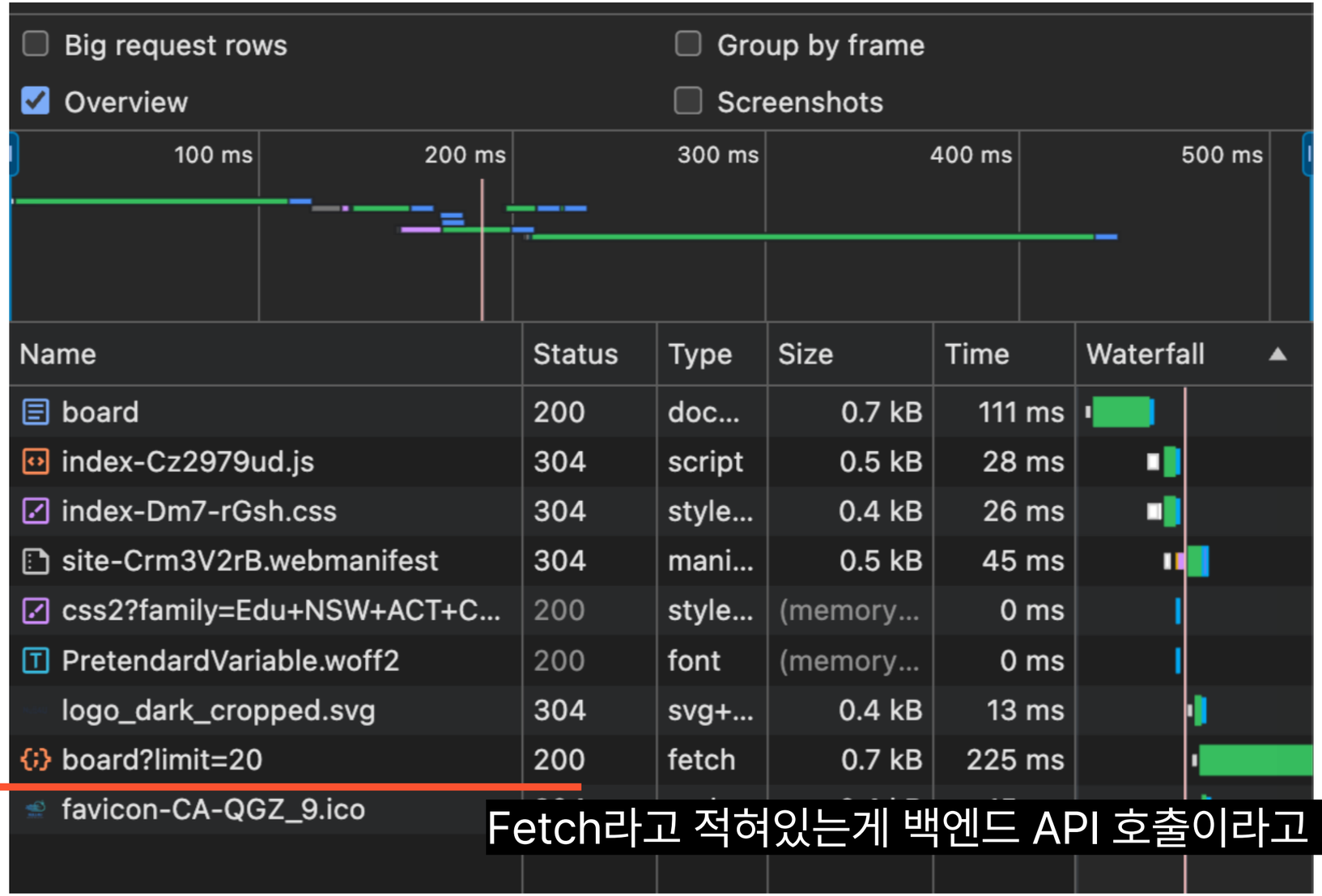
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



부록/실습) HTTP 요청/응답을 뜯어보자!

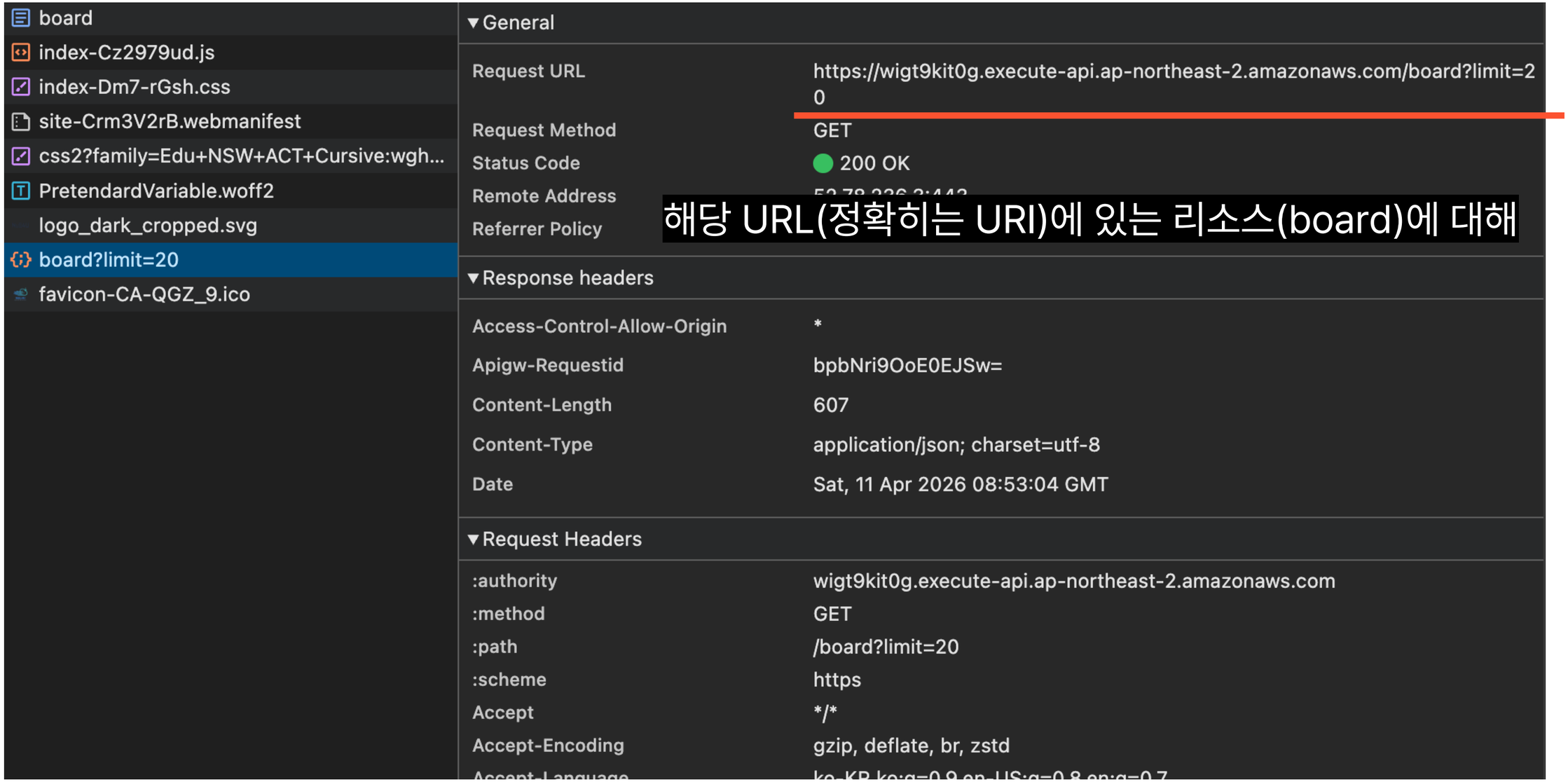
(따라하면 좋겠지만 그냥 보기만 해도 됨)



Fetch라고 적혀있는게 백엔드 API 호출이라고 보면 됨.

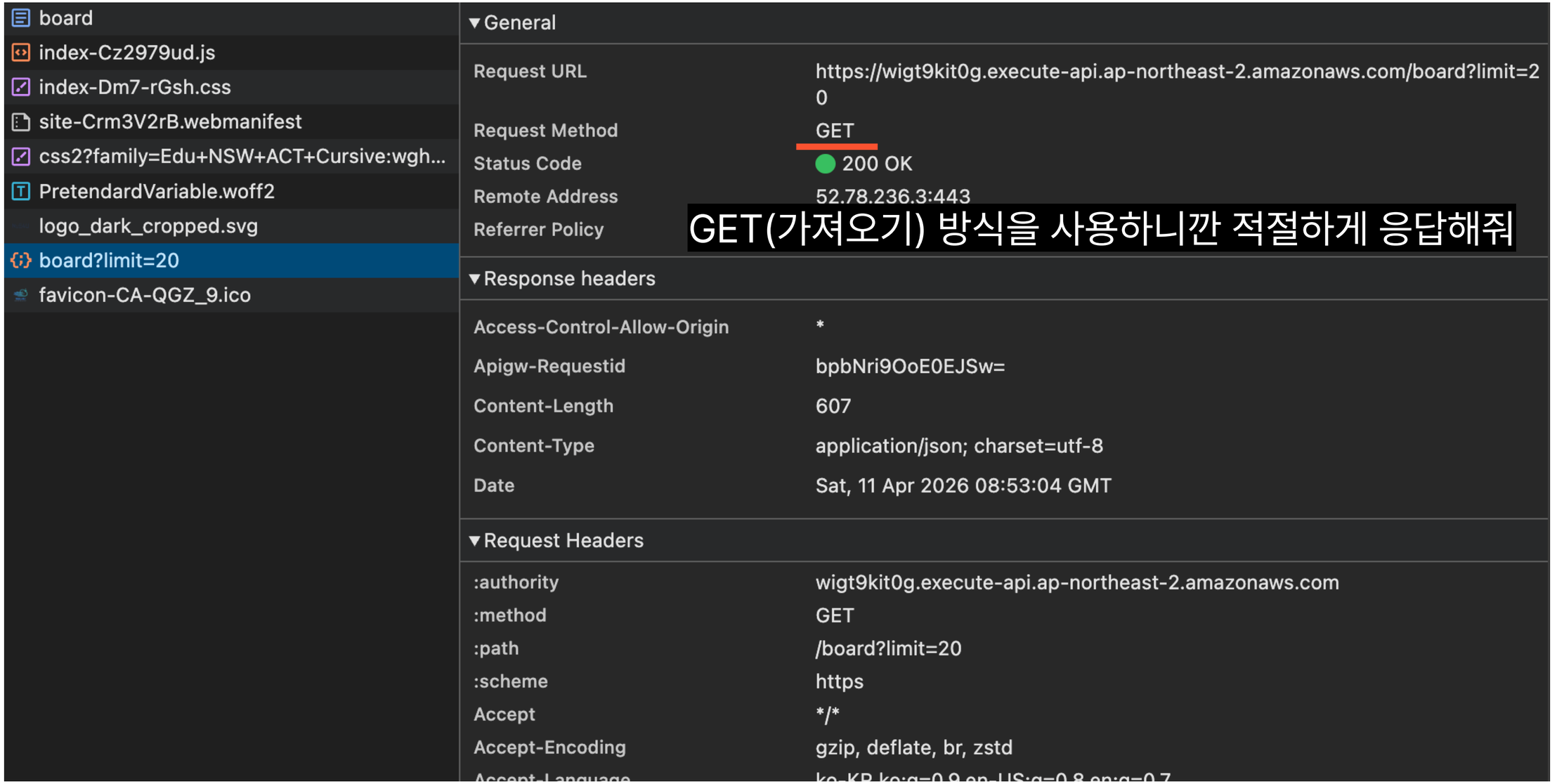
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



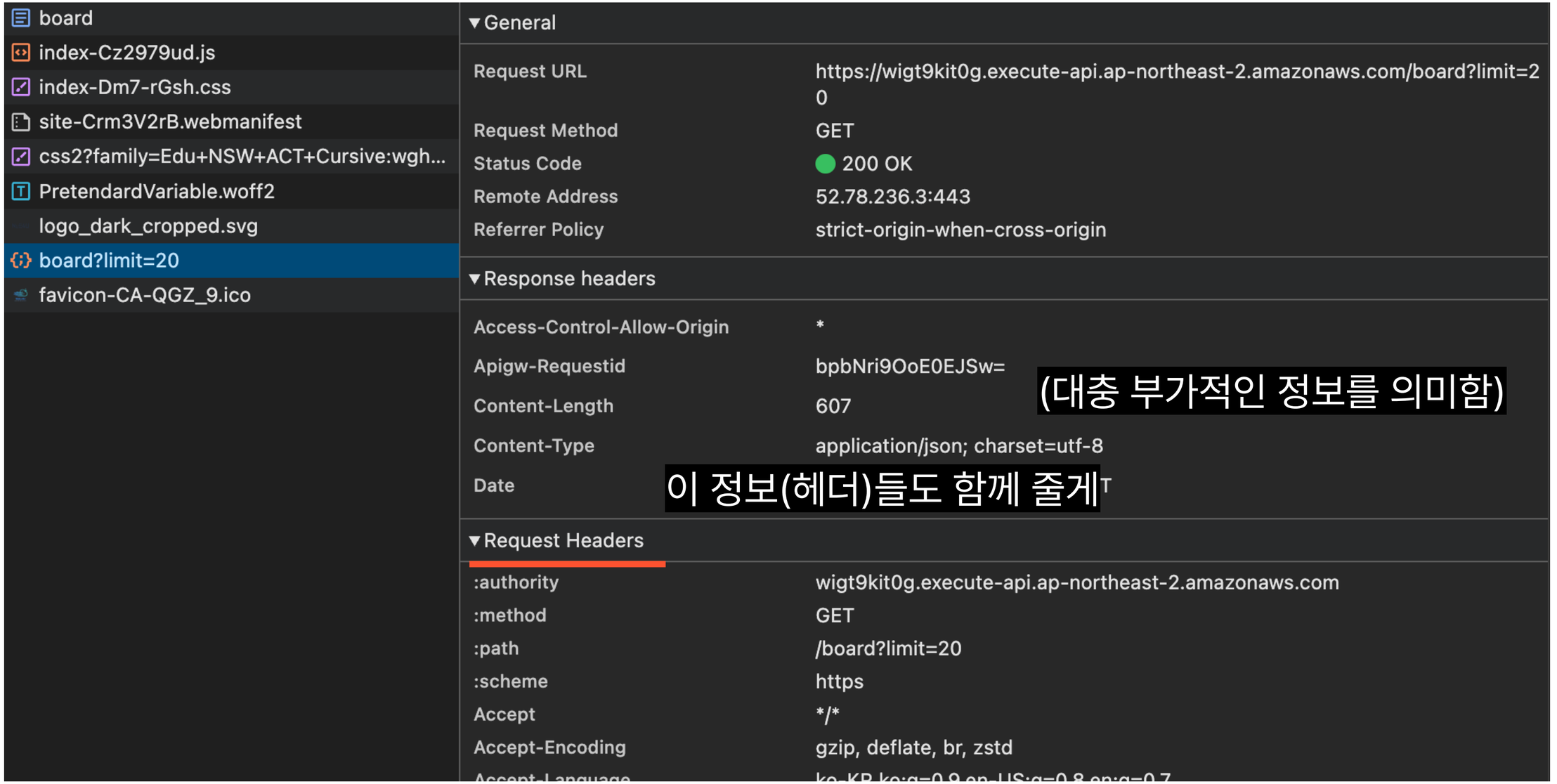
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



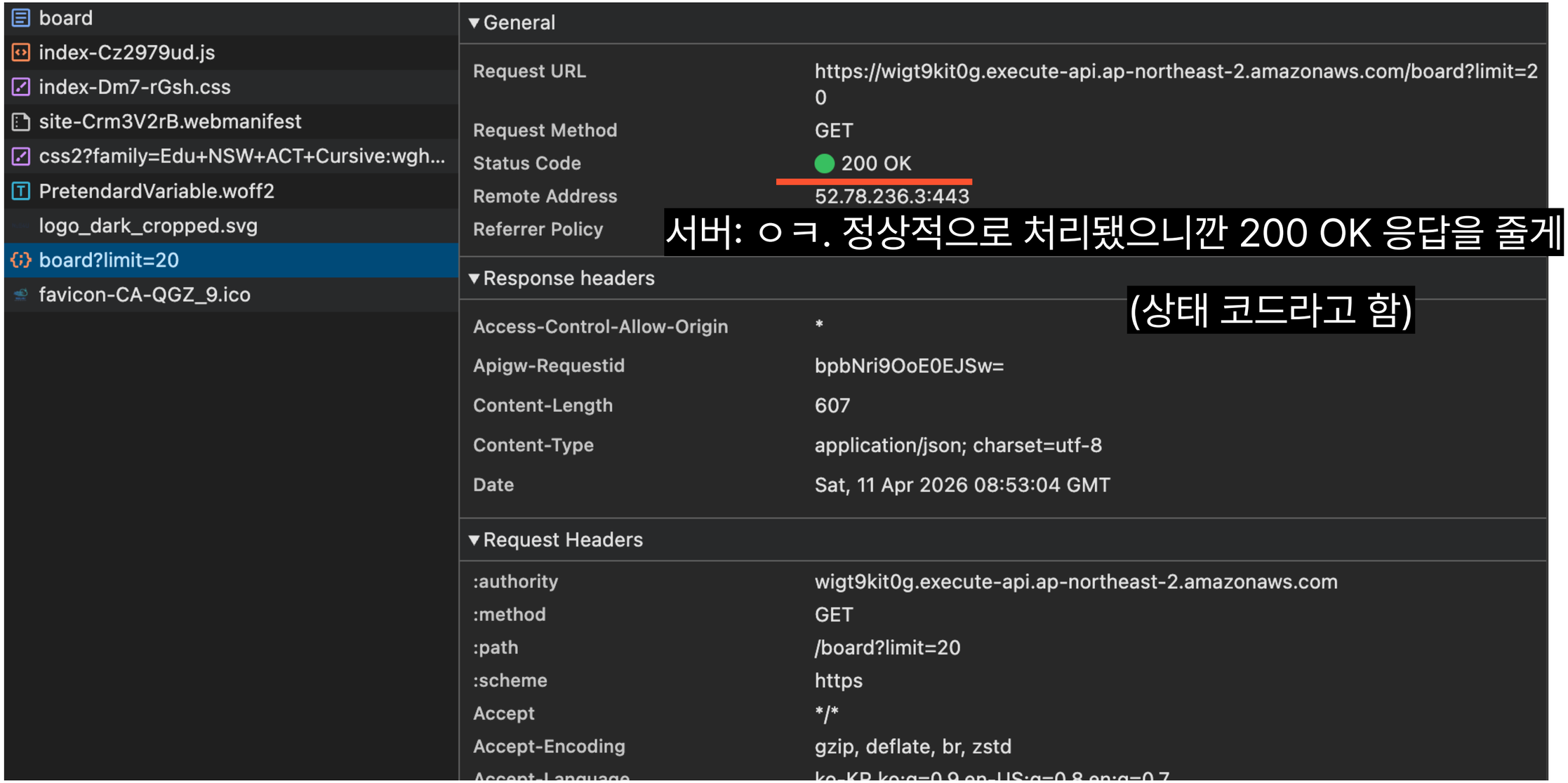
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



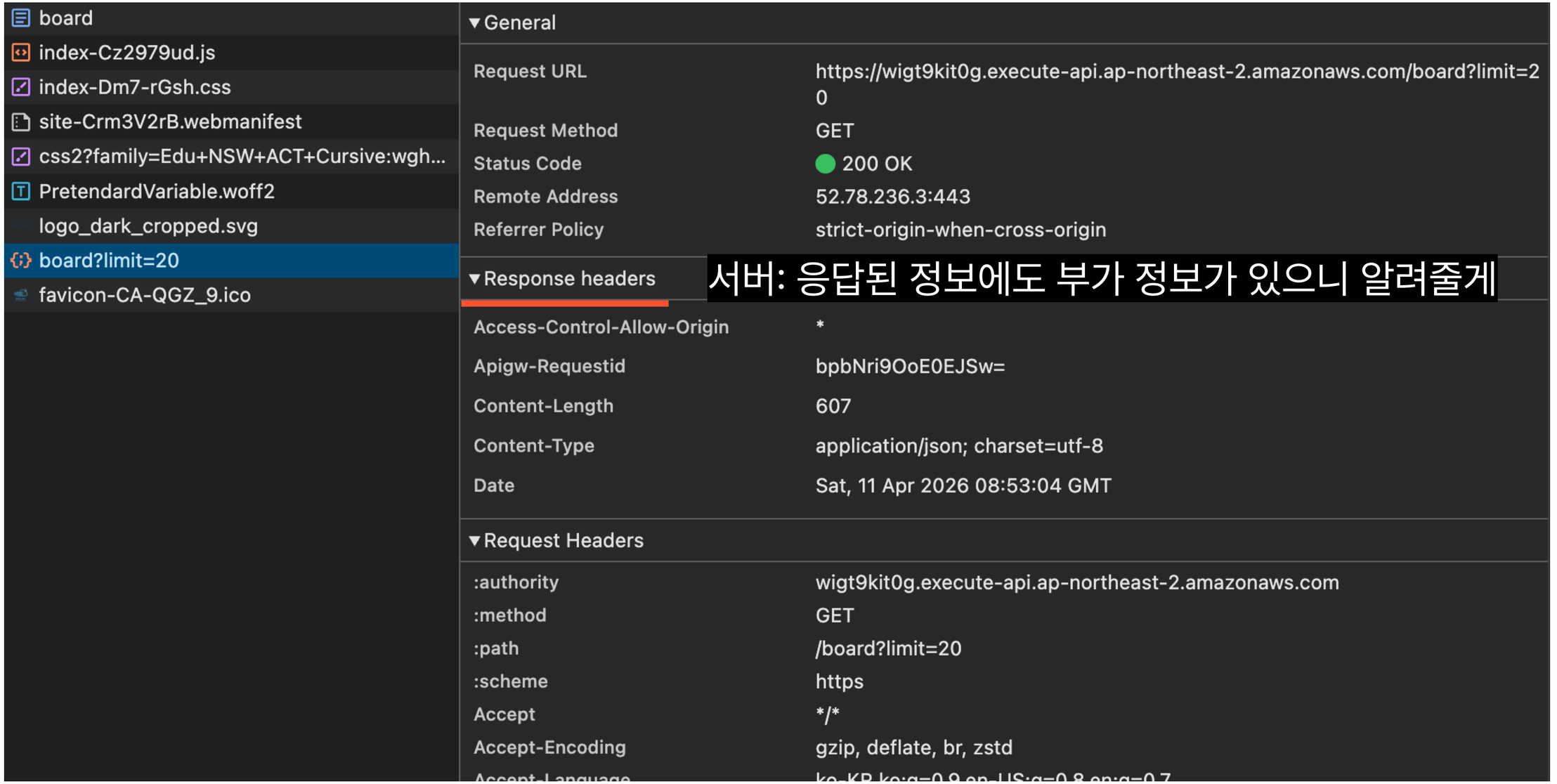
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



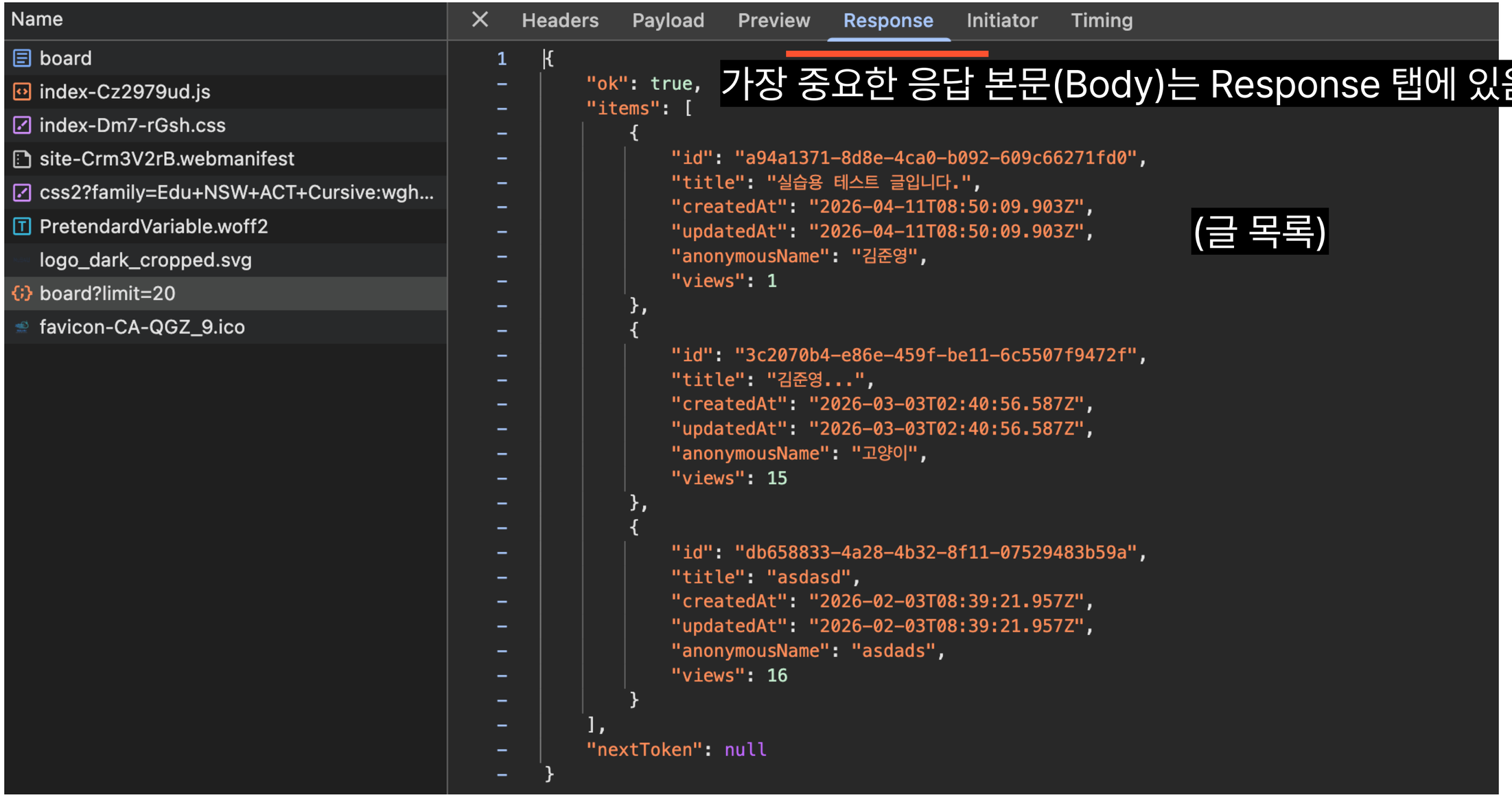
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



부록/실습) HTTP 요청/응답을 뜯어보자!

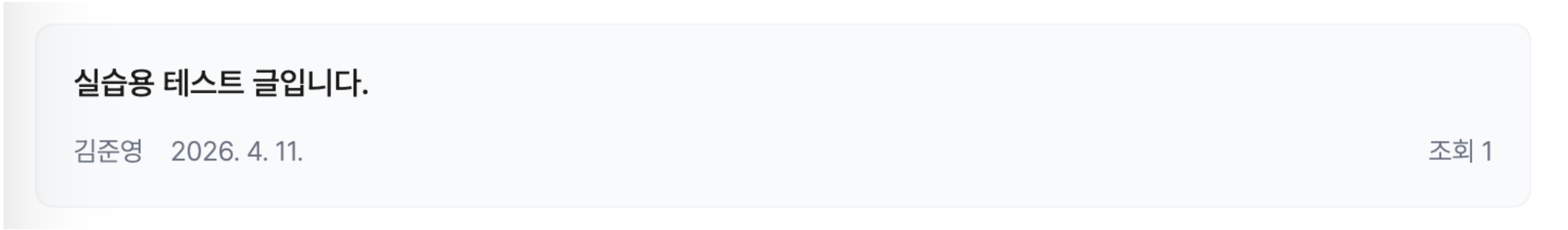
(따라하면 좋겠지만 그냥 보기만 해도 됨)




```
{
    "ok": true,
    "items": [
        {
            "id": "a94a1371-8d8e-4ca0-b092-609c66271fd0",
            "title": "실습용 테스트 글입니다.",
            "createdAt": "2026-04-11T08:50:09.903Z",
            "updatedAt": "2026-04-11T08:50:09.903Z",
            "anonymousName": "김준영",
            "views": 1
        },
        {
            "id": "3c2070b4-e86e-459f-be11-6c5507f9472f",
            "title": "김준영...",
            "createdAt": "2026-03-03T02:40:56.587Z",
            "updatedAt": "2026-03-03T02:40:56.587Z",
            "anonymousName": "고양이",
            "views": 15
        },
        {
            "id": "db658833-4a28-4b32-8f11-07529483b59a",
            "title": "asdasd",
            "createdAt": "2026-02-03T08:39:21.957Z",
            "updatedAt": "2026-02-03T08:39:21.957Z",
            "anonymousName": "asdads",
            "views": 16
        }
    ],
    "nextToken": null
}
```

부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



개발자 도구에서 네트워크 탭 유지한 상태에서 글 클릭해보기

부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)

실습용 테스트 글입니다.

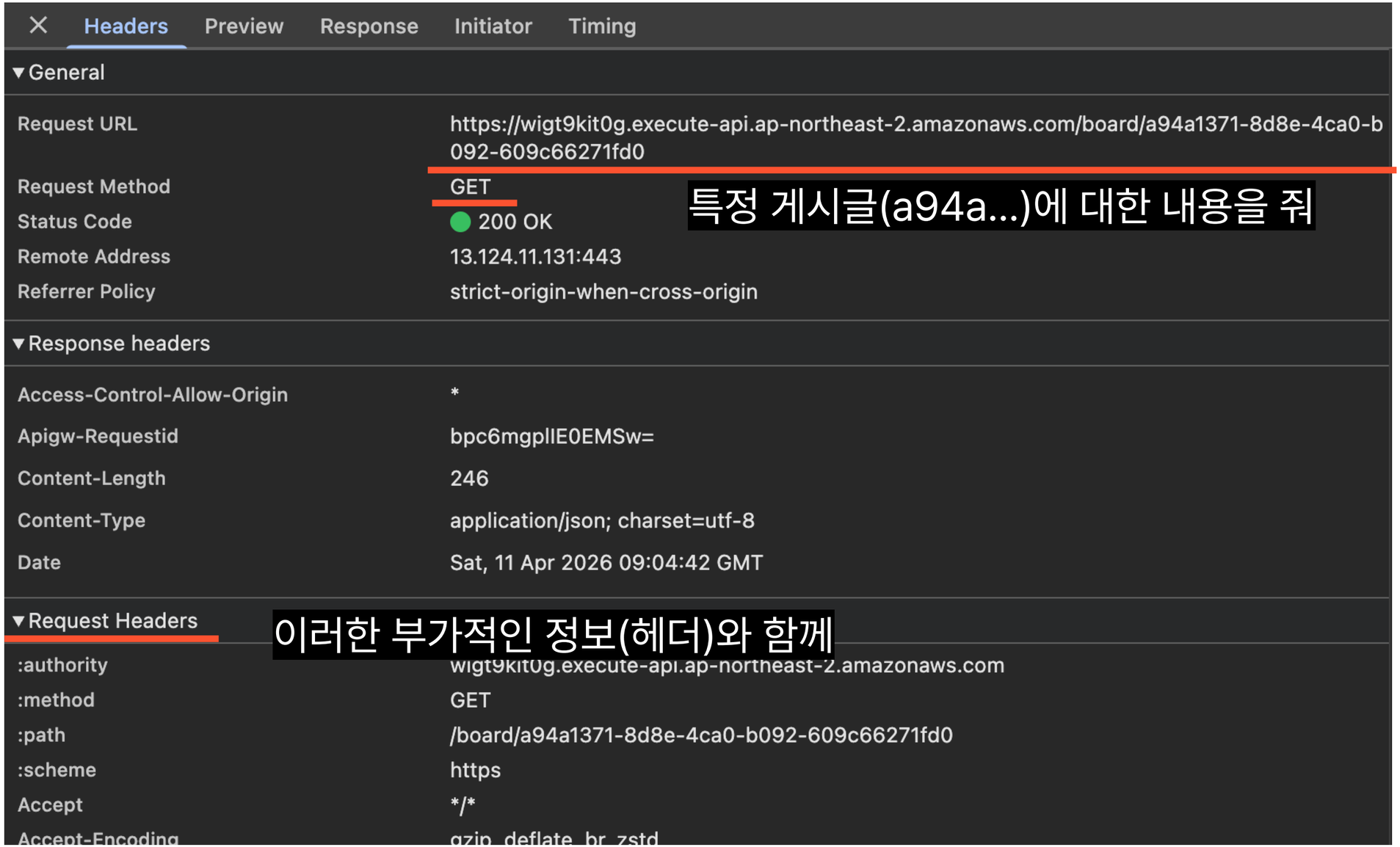
김준영 2026. 4. 11. 조회 1

개발자 도구에서 네트워크 탭 유지한 상태에서 글 클릭해보기

Name	Status	Type	Size	Time	W
{;} board?limit=20	200	fetch	0.7 kB	225 ms	
{;} a94a1371-8d8e-4ca0-b092-609c66271fd0	200	fetch	0.4 kB	1.08 s	

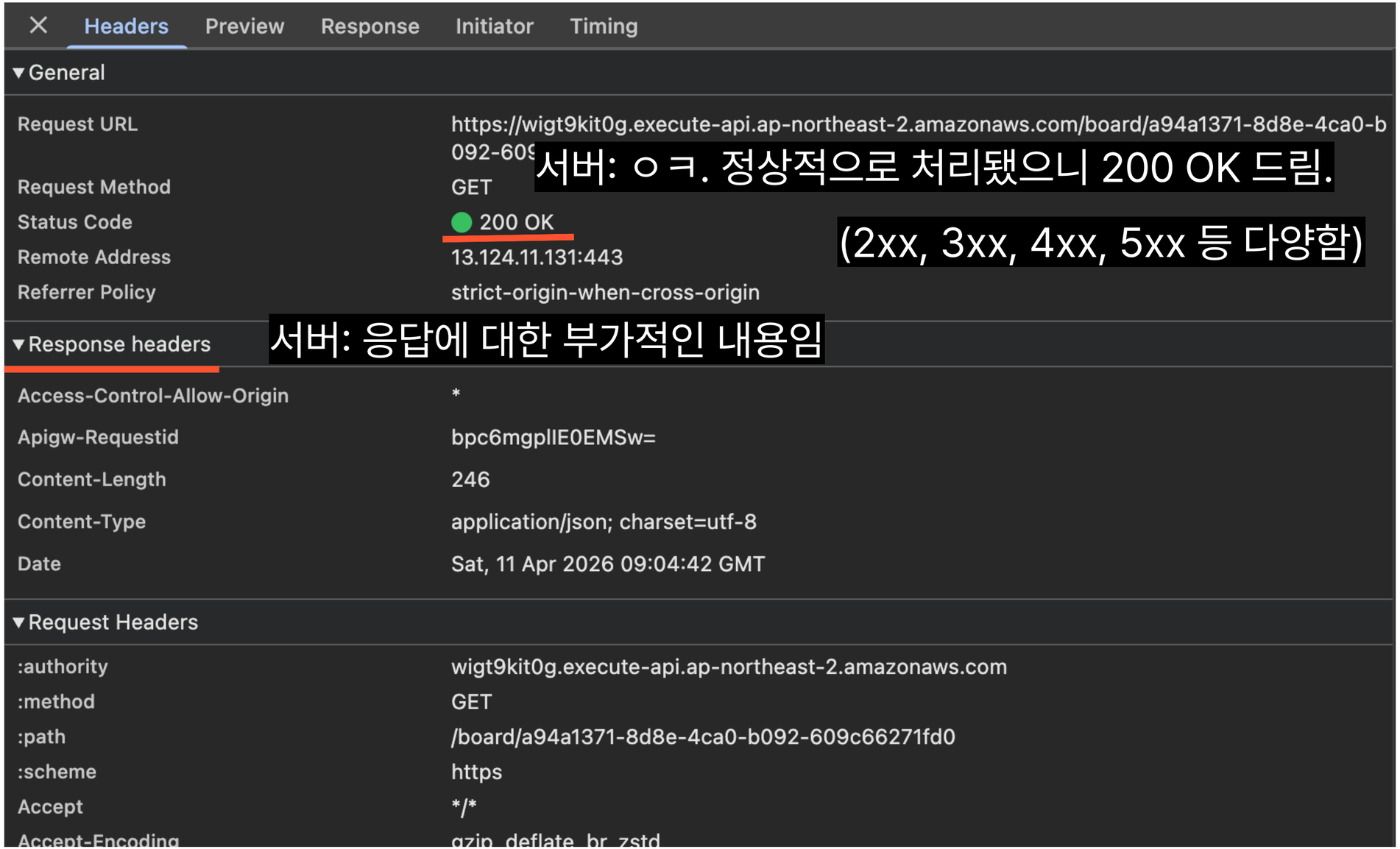
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)

글쓰기

아무 내용의 글 작성해보기

제목

테스트 글

작성자

ㅇㅇ

비밀번호 (수정/삭제 시 필요)

••••

내용

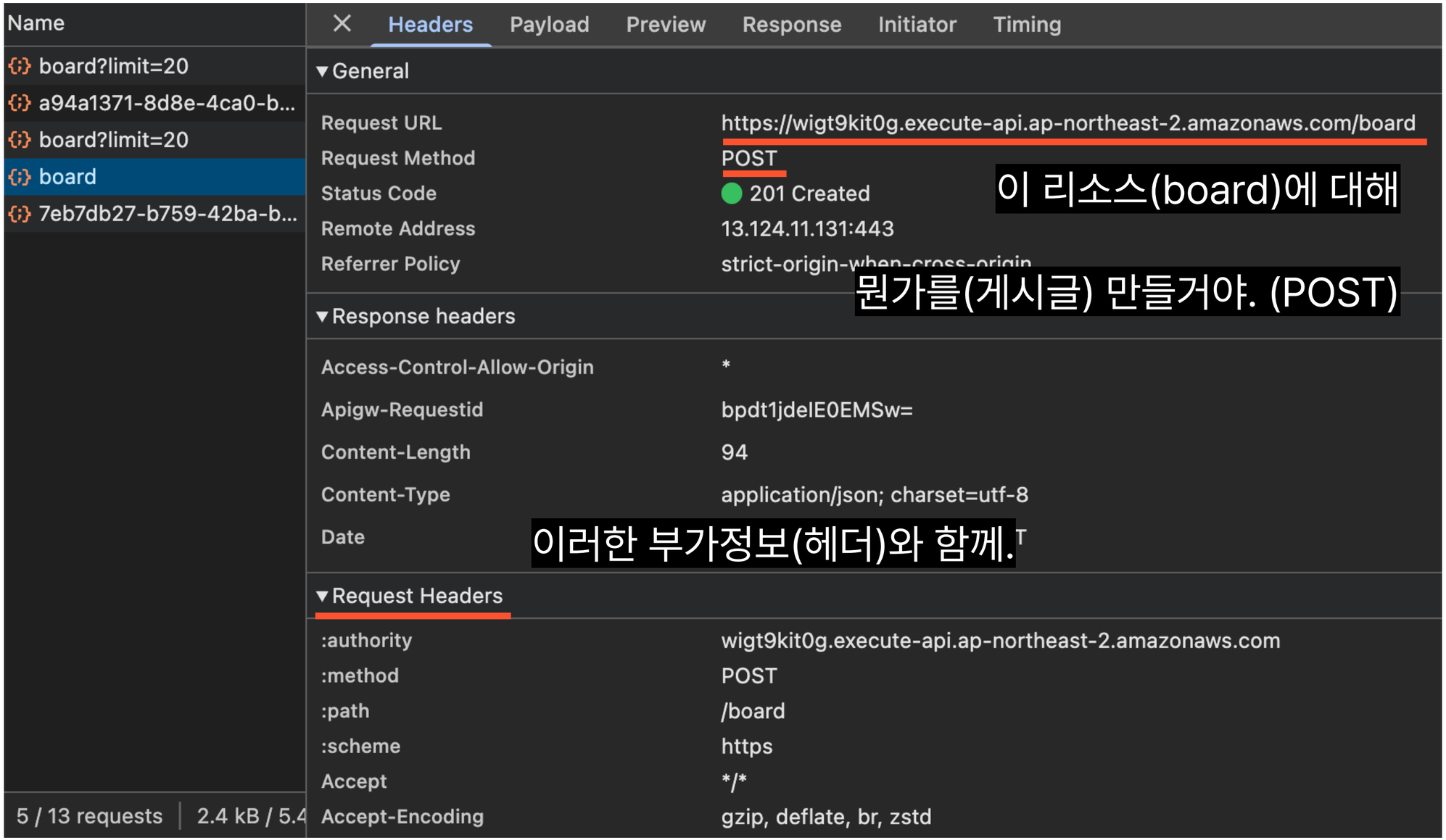
안녕하세요?

취소

작성 완료

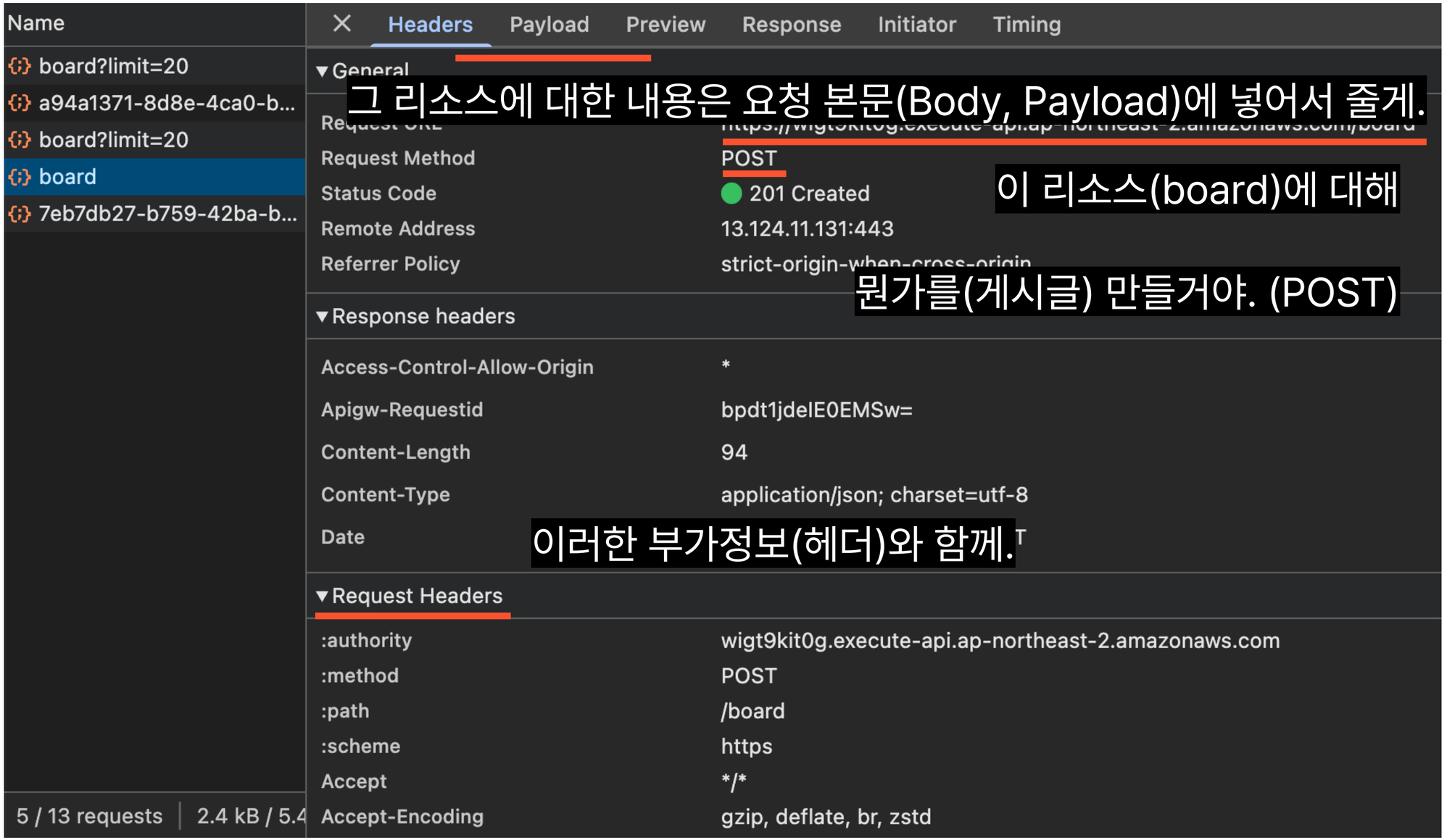
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



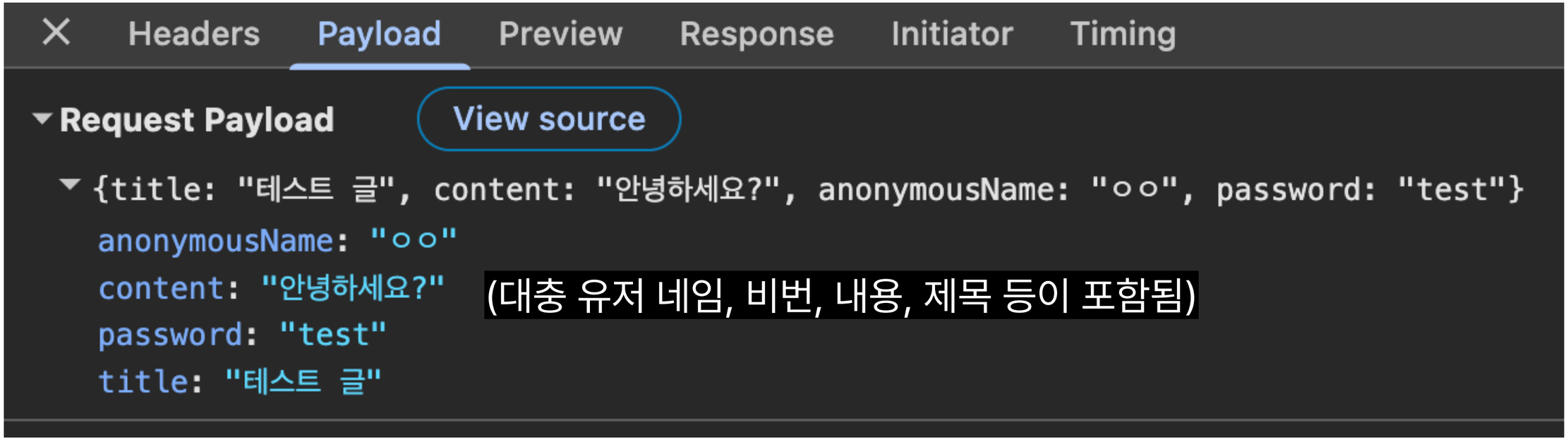
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



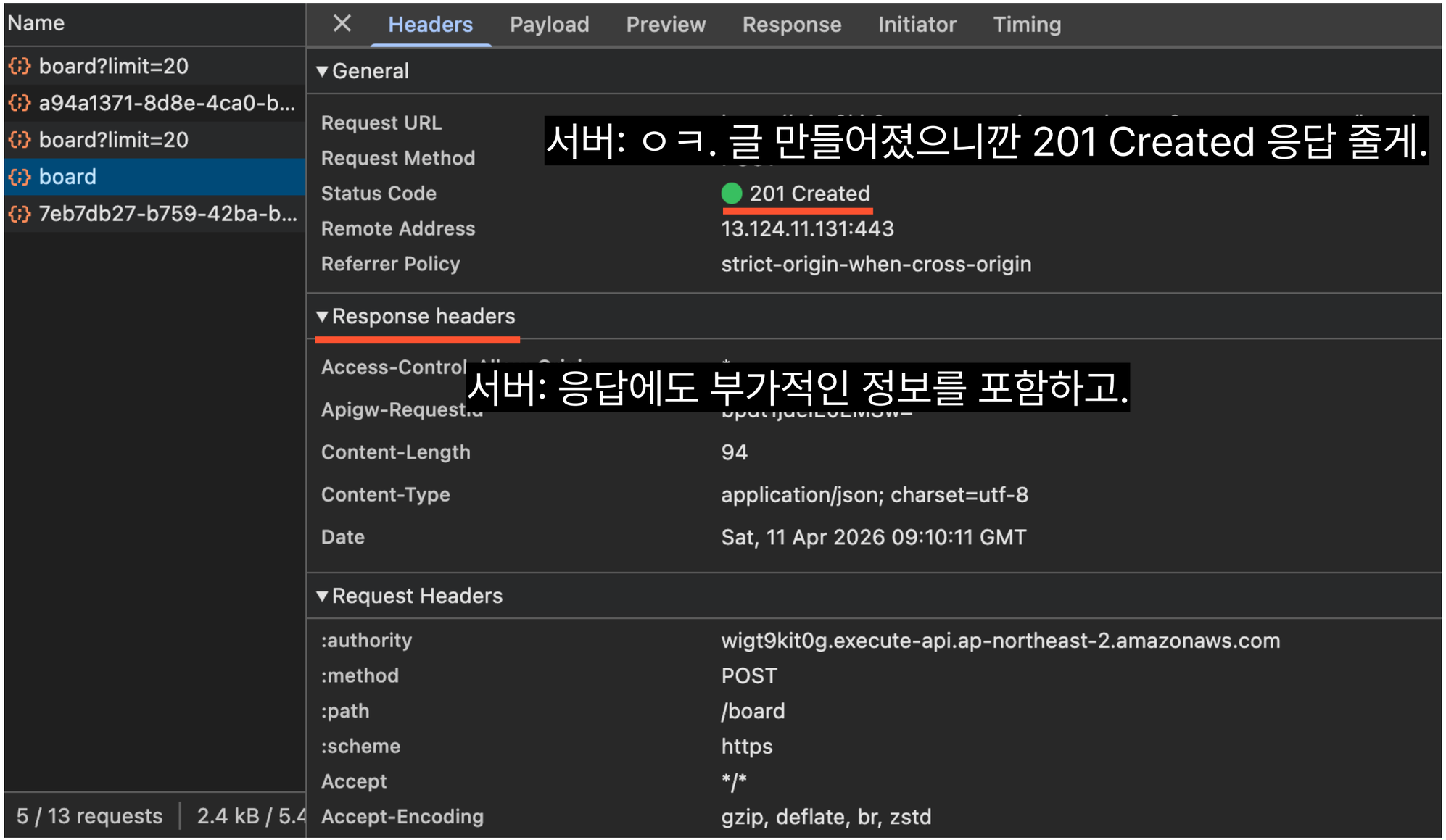
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



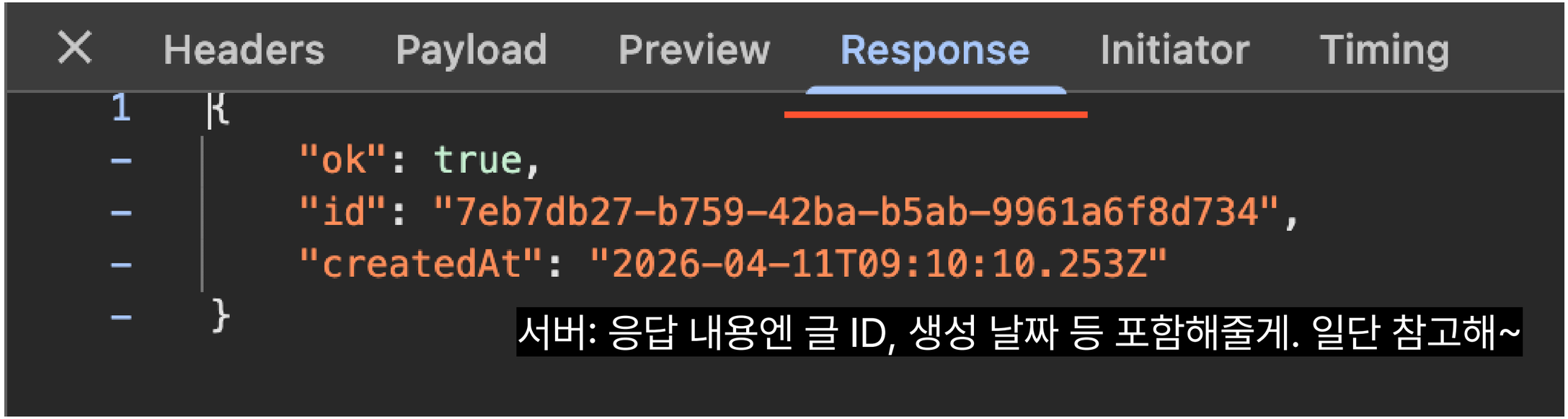
부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



부록/실습) HTTP 요청/응답을 뜯어보자!

(따라하면 좋겠지만 그냥 보기만 해도 됨)



부록/실습) HTTP 요청/응답을 뜯어보자!

```
POST /api/users HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "username": "user1",
    "email": "user1@example.com"
}
```

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain

Hello World! My payload is a raw plain text string.
```

가공되지 않은 Raw HTTP 요청/응답 데이터는 이러함.

실습) HTTP 요청을 직접 보내보자.

HTTP 요청을 보낼 수 있는 프로그램(클라이언트 도구)가 필요함.

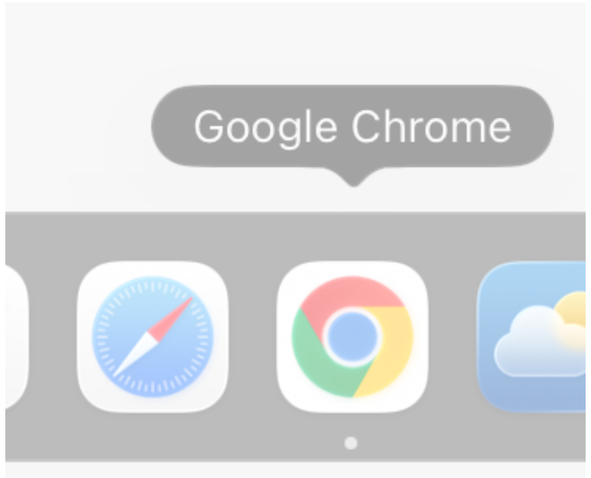

```
> curl -I example.com
HTTP/1.1 200 OK
Date: Sat, 11 Apr 2026 09:24:09 GMT
Content-Type: text/html
Connection: keep-alive
Server: cloudflare
Last-Modified: Fri, 10 Apr 2026 01:29:17 GMT
Allow: GET, HEAD
Accept-Ranges: bytes
Age: 13468
cf-cache-status: HIT
CF-RAY: 9ea8efc8990c8142-HKG
```

POSTMAN

사실 네트워크 요청을 보낼 수 있는 모든 주체들이 HTTP 클라이언트가 될 수 있음.

실습) HTTP 요청을 직접 보내보자.

HTTP 요청을 보낼 수 있는 프로그램(클라이언트 도구)가 필요함.



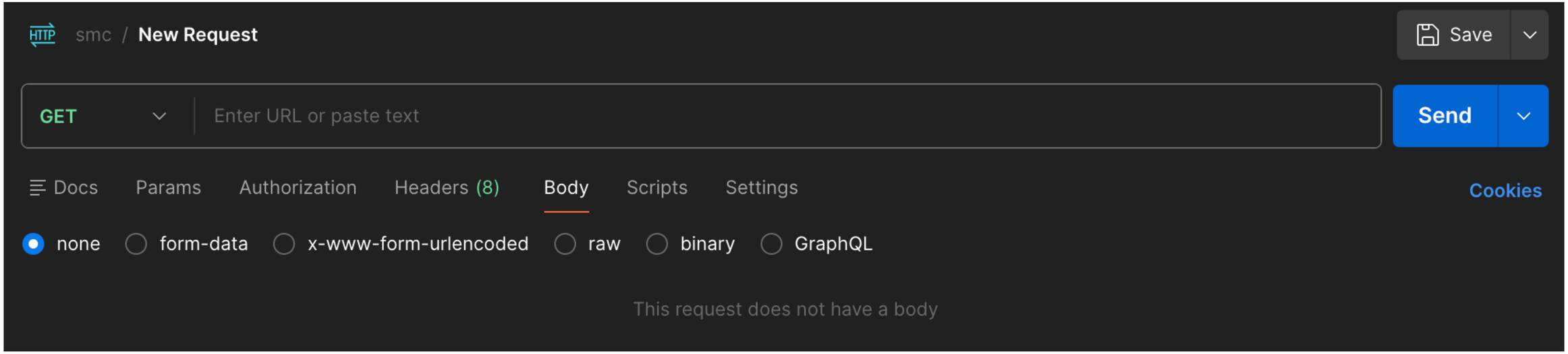

```
> curl -I example.com
HTTP/1.1 200 OK
Date: Sat, 11 Apr 2026 09:24:09 GMT
Content-Type: text/html
Connection: keep-alive
Server: cloudflare
Last-Modified: Fri, 10 Apr 2026 01:29:17 GMT
Allow: GET, HEAD
Accept-Ranges: bytes
Age: 13468
cf-cache-status: HIT
CF-RAY: 9ea8efc8990c8142-HKG
```

POSTMAN

예시로 Postman을 써보겠음.

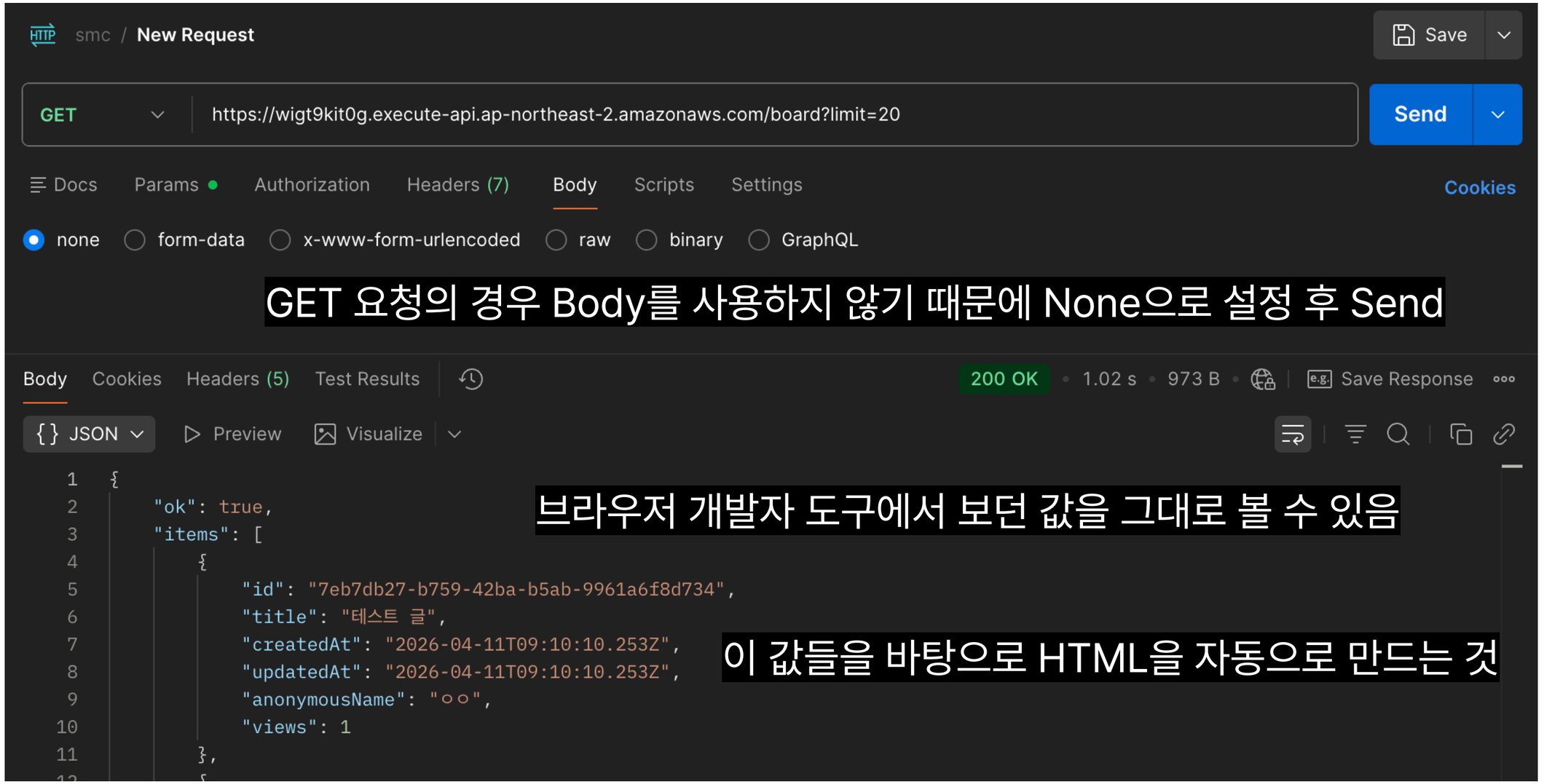
(일부 유료 기능이 포함되며 Apidog, Insomnia, Bruno 등의 대안 프로그램들이 많음.)

실습) HTTP 요청을 직접 보내보자.

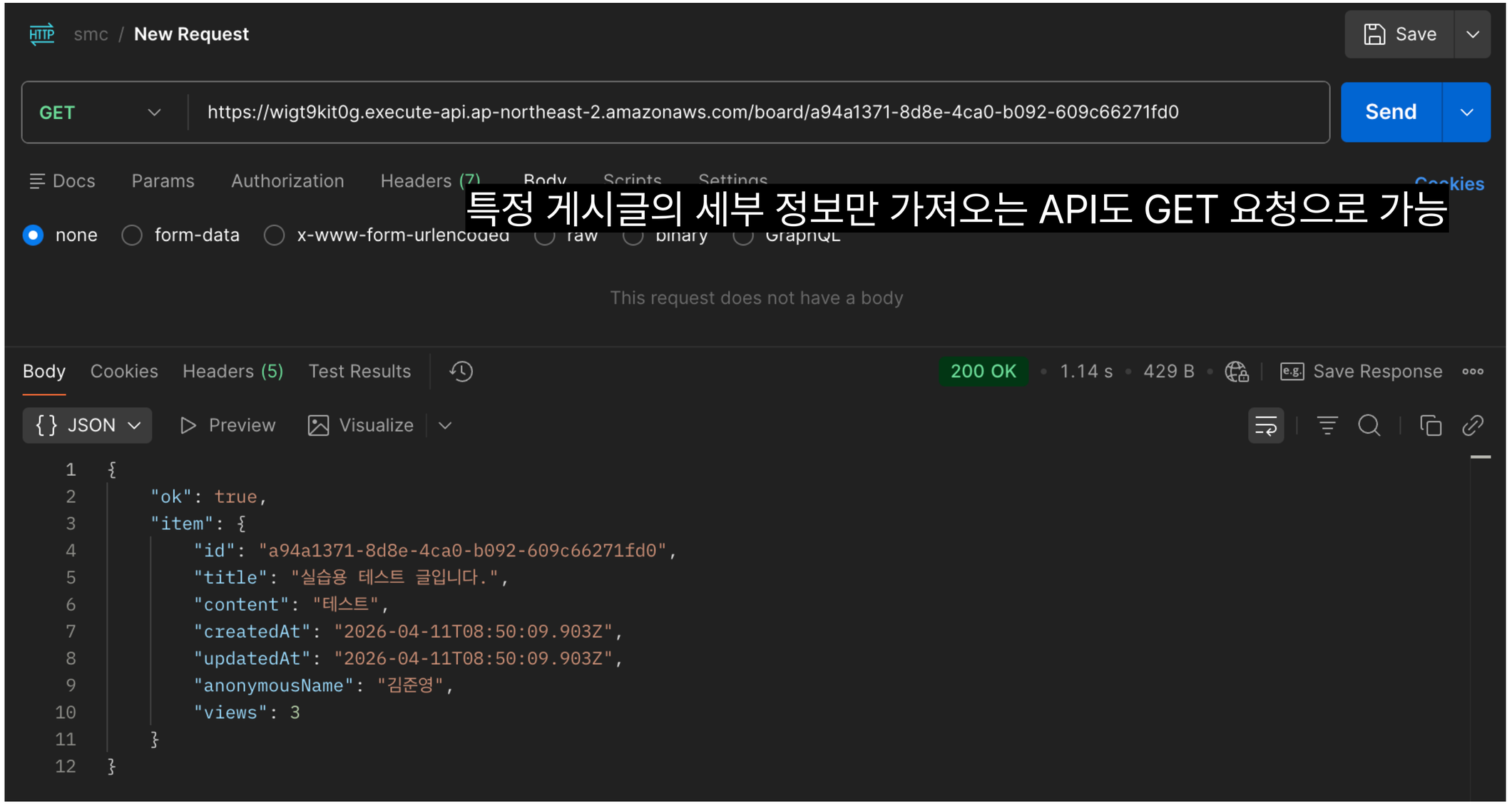


필요한 값은 URI(여기선 URL), HTTP 메서드, Body(페이로드)임.

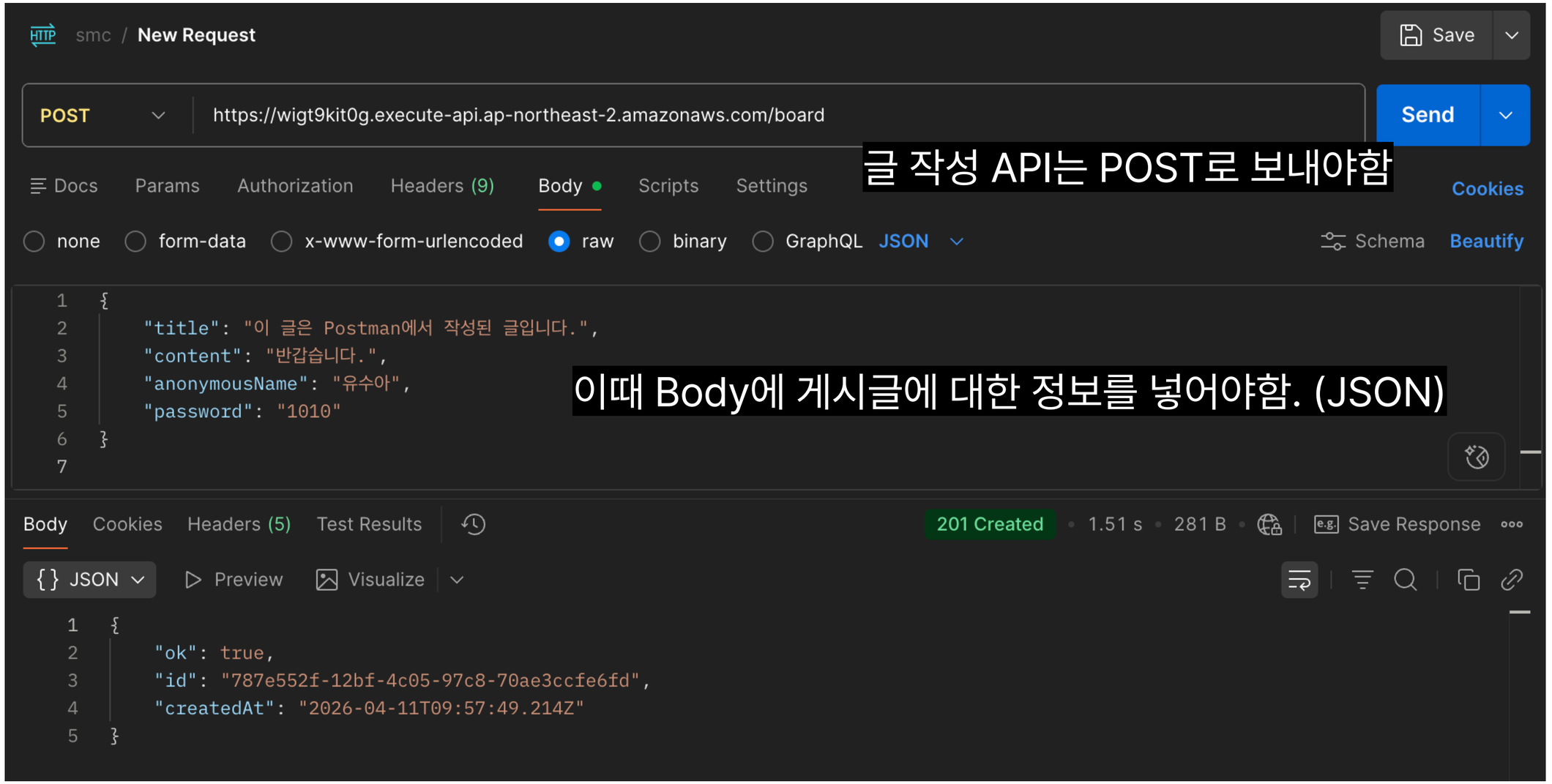
실습) HTTP 요청을 직접 보내보자.



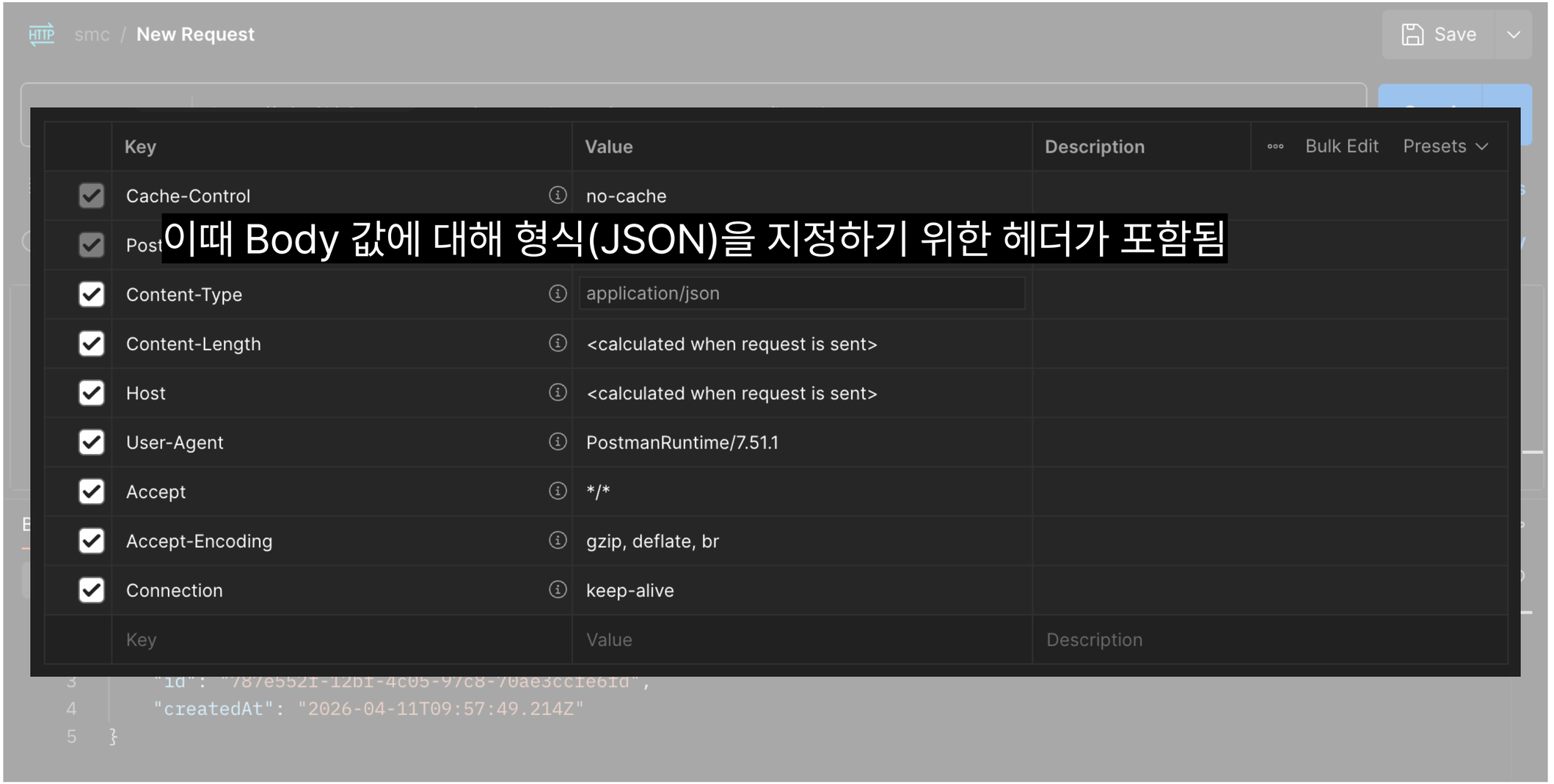
실습) HTTP 요청을 직접 보내보자.



실습) HTTP 요청을 직접 보내보자.



실습) HTTP 요청을 직접 보내보자.

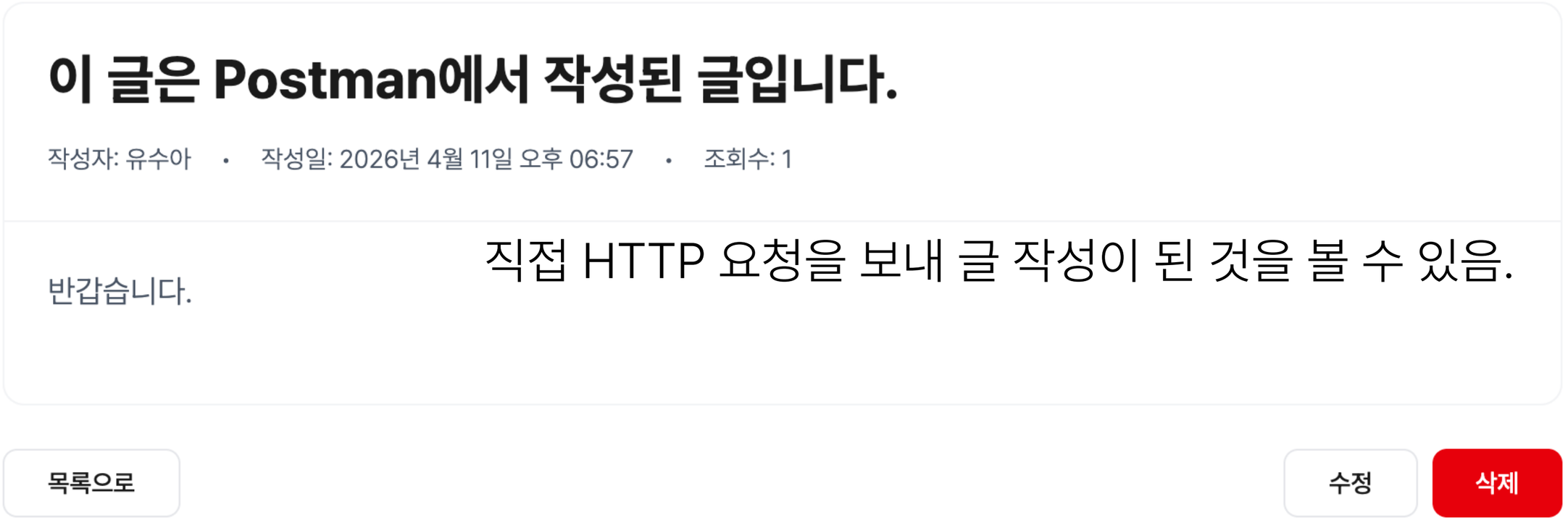


이때 Body 값에 대해 형식(JSON)을 지정하기 위한 헤더가 포함됨

실습) HTTP 요청을 직접 보내보자.

제목	작성자	작성일	조회
이 글은 Postman에서 작성된 글입니다.	유수아	2026. 4. 11.	0
테스트 글	ㅇㅇ	2026. 4. 11.	1
실습용 테스트 글입니다.	김준영	2026. 4. 11.	3
김준영...	고양이	2026. 3. 3.	15
asdasd	asdads	2026. 2. 3.	16

실습) HTTP 요청을 직접 보내보자.



실습) Hello, World를 출력하는 백엔드 서버 만들어보기

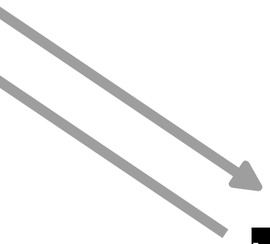
... 는 시간 관계상 생략함. 추후 시간이 남는다면 Python Flask 또는 NodeJS Express를 통해 웹 서버를 구현해보는 실습을 진행해보겠음.

??? - 우리의 정보는 어디에 저장될까?

데이터베이스 - 우리의 정보는 어디에 저장될까?

데이터베이스 = 여러 사람들이 공유하여 사용될 목적으로 통합하여 관리되는 체계적인 데이터들의 집합

회원 정보, 게시글, 온라인 쇼핑몰의 상품 정보, 주문 정보 등등 사실상 대부분이 데이터베이스 저장됨

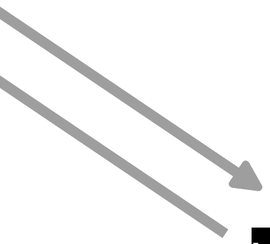


Web Application Server (WAS)

데이터베이스 - 우리의 정보는 어디에 저장될까?

데이터베이스 = 여러 사람들이 공유하여 사용될 목적으로 통합하여 관리되는 체계적인 데이터들의 집합

회원 정보, 게시글, 온라인 쇼핑몰의 상품 정보, 주문 정보 등등 사실상 대부분이 데이터베이스 저장됨



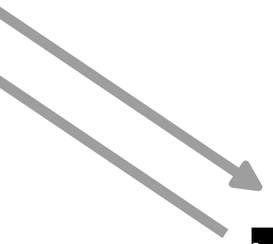
Web Application Server (WAS)

이 데이터베이스는 어디에 있나요?

데이터베이스 - 우리의 정보는 어디에 저장될까?

데이터베이스 = 여러 사람들이 공유하여 사용될 목적으로 통합하여 관리되는 체계적인 데이터들의 집합

회원 정보, 게시글, 온라인 쇼핑몰의 상품 정보, 주문 정보 등등 사실상 대부분이 데이터베이스 저장됨



Web Application Server (WAS)

이 데이터베이스는 어디에 있나요?

정답: 운영 방식, DB 종류 마다 다르다.

데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 A. 백엔드와 같은 서버에 두기

솔루션 A-1. 별도의 DB 애플리케이션 실행 및 사용

솔루션 A-2. 백엔드 애플리케이션 자체에 포함

솔루션 B. DB 전용 서버 운영

솔루션 B-1. 단일 서버로 DB 운영

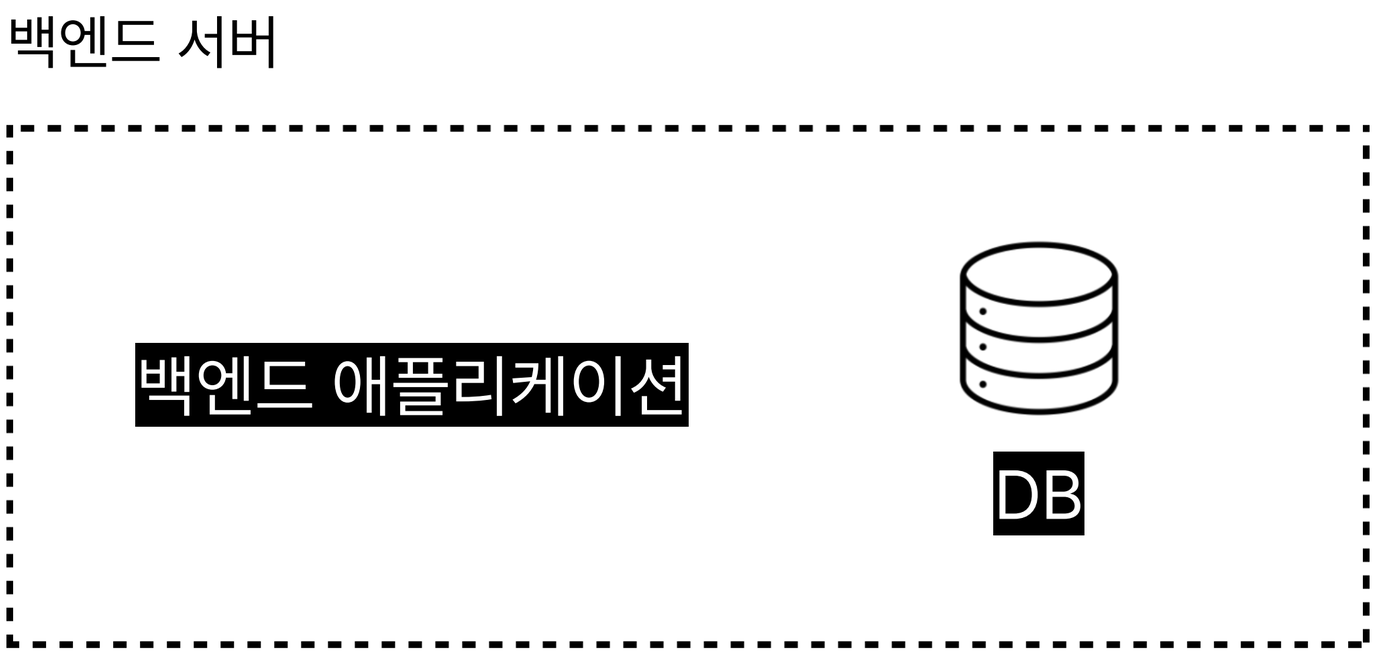
솔루션 B-2. 분산 서버로 DB 운영 (Primary/Replica/Standby/Writer-ReadOnly 구조 등..)

솔루션 B-3. 관리형(Managed) DB 서비스 사용 (AWS RDS, DynamoDB, ElastiCache ...)

솔루션 C. DB를 사용하지 않기 (feat. 왜 서버에서 직접 파일을 만들어서 저장하지 않을까?)

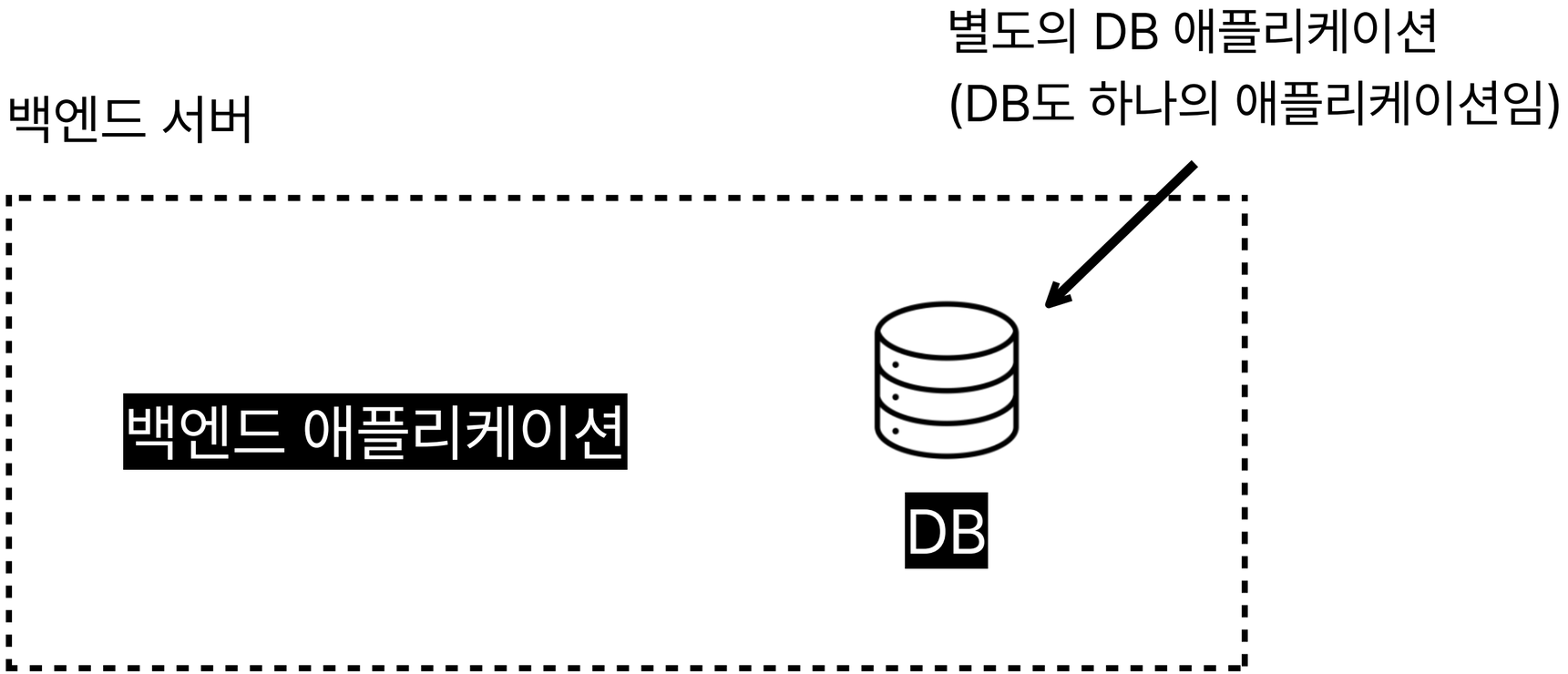
데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 A. 백엔드와 같은 서버에 두기



데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 A-1. 별도의 DB 애플리케이션 실행 및 사용



데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 A-2. 백엔드 애플리케이션 자체에 포함

ex. SQLite, DuckDB ...

특수한 경우가 아니라면 백엔드 애플리케이션에서 잘 안쓰임.

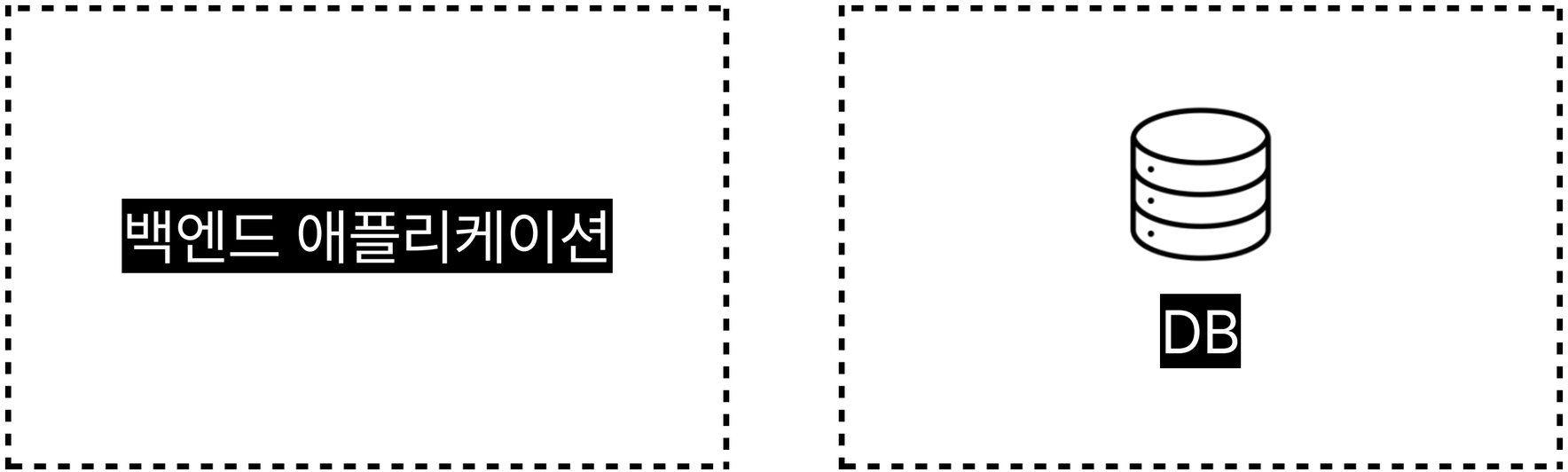
데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 B. DB 전용 서버 운영

보통은 서로 다른 네트워크(서버)에 배치됨.

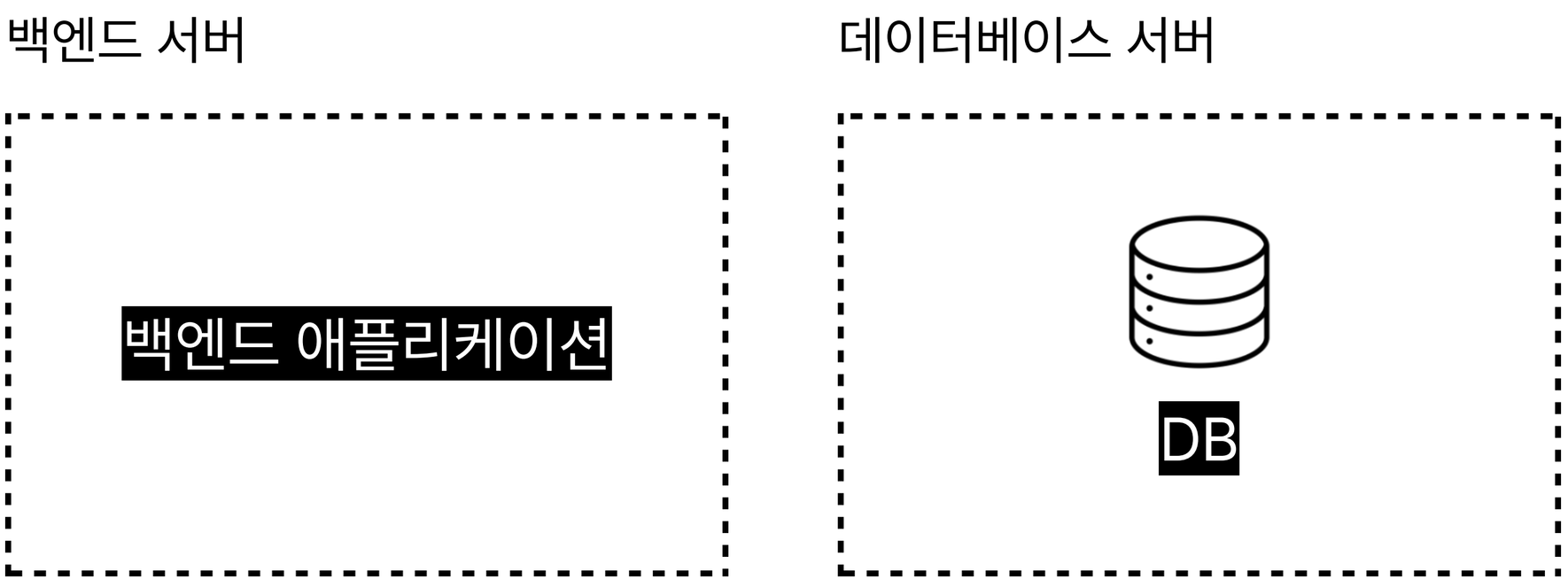
백엔드 서버

데이터베이스 서버



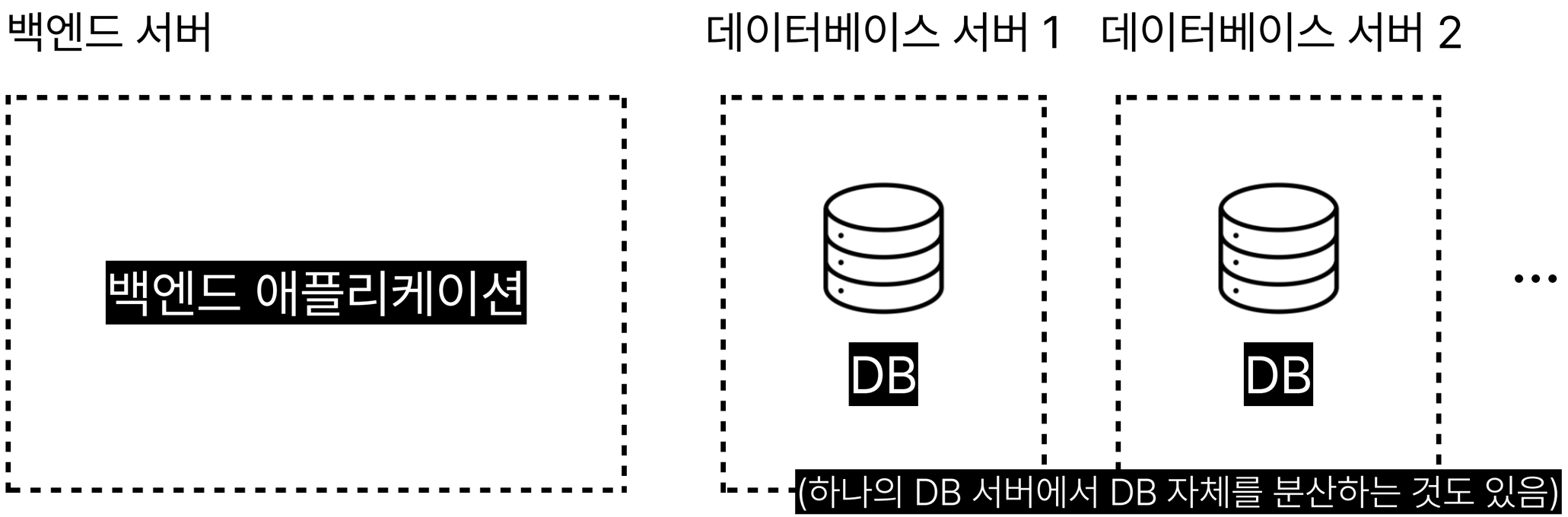
데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 B-1. 단일 서버로 DB 운영



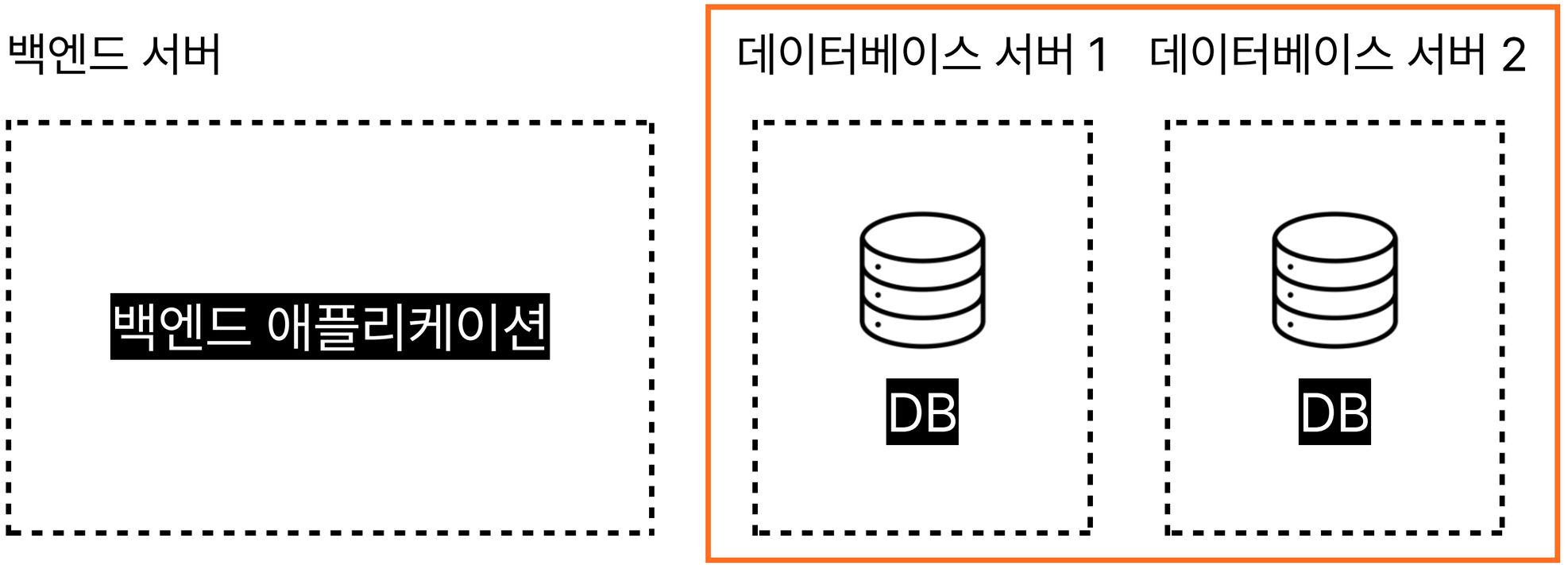
데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 B-2. 분산 서버로 DB 운영 (Primary/Replica/Standby/Writer-ReadOnly 구조 등..)



데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 B-3. 관리형(Managed) DB 서비스 사용 (AWS RDS, DynamoDB, ElastiCache ...)

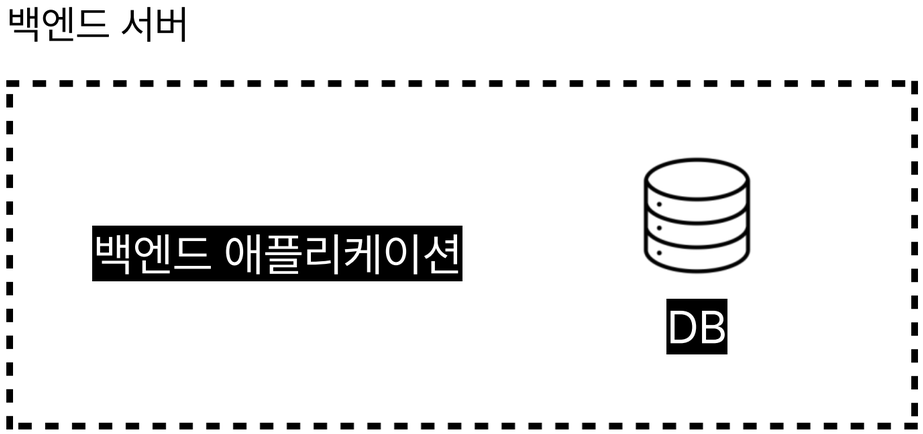


데이터베이스를 직접 구축하고 관리할 필요가 거의 없음

데이터베이스 - 어떠한 운영 방식이 더 좋나요?

데이터베이스 - 어떠한 운영 방식이 더 좋나요? 상황에 따라 다름.

데이터베이스 - 어떠한 운영 방식이 더 좋나요?



솔루션 A. 백엔드와 같은 서버에 두기

장점

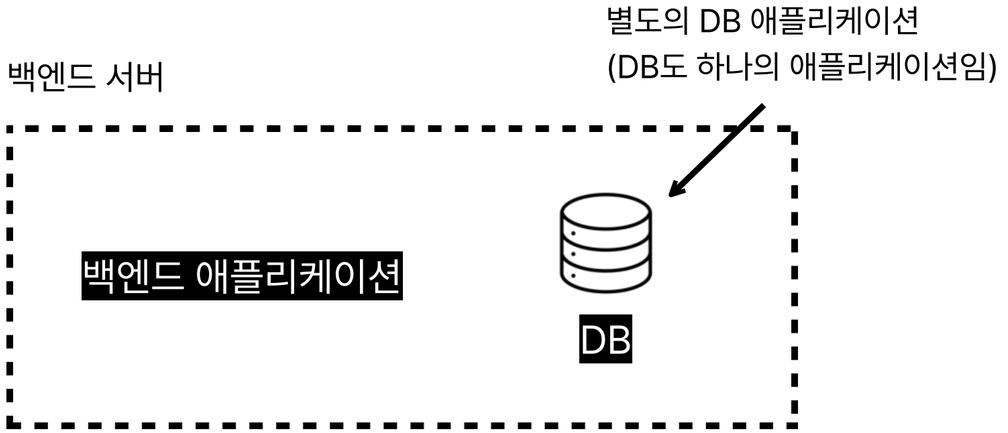
- 네트워크 구성이 간단해짐
- DB 운영을 위한 비용이 비교적 적음

단점

- 백엔드 서버가 죽을 경우 DB도 죽음
- DB 분산/Failover가 어려움
- 로컬에 파일로 저장되기 때문에 주의 필요

데이터베이스 - 어떠한 운영 방식이 더 좋나요?

솔루션 A-1. 별도의 DB 애플리케이션 실행 및 사용



장점

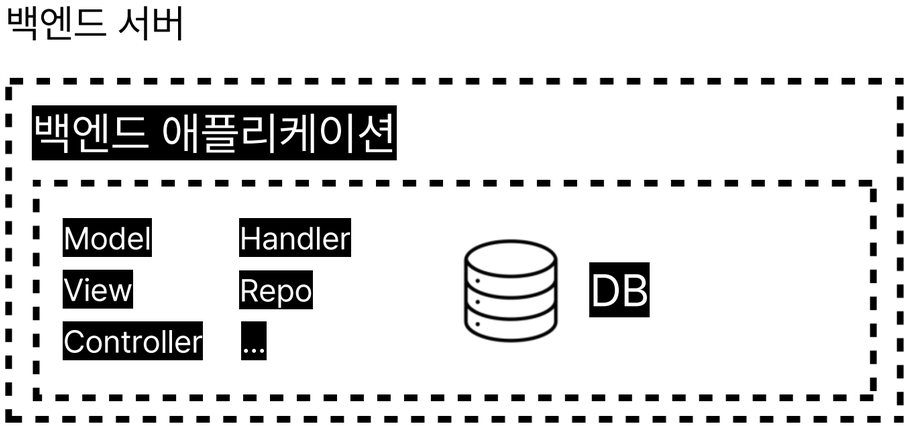
- DB 구축하기 쉬움
- 네트워크 구성이 간단해짐
- DB 운영을 위한 비용이 거의 줄어듬

단점

- 백엔드 서버가 죽을 경우 DB도 죽음
- DB 분산/Failover가 어려움

데이터베이스 - 어떠한 운영 방식이 더 좋나요?

솔루션 A-2. 백엔드 애플리케이션 자체에 포함



장점

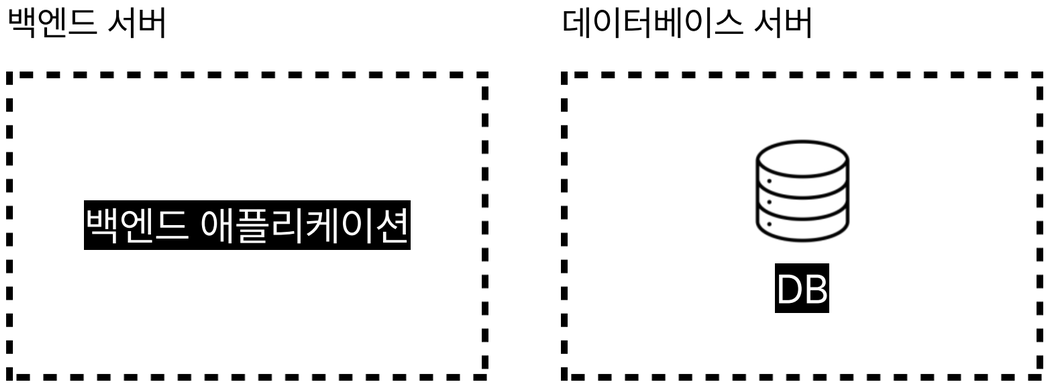
- DB 구축하기 너무 쉬움
- 네트워크 구성은 사실상 불필요
- DB 운영을 위한 비용이 사실상 없음

단점

- 분산은 사실상 불가능
- 백엔드 애플리케이션에 종속적임
- 사용 가능한 DBMS(DB 종류)가 매우 제한적

데이터베이스 - 우리의 정보는 어디에 저장될까?

솔루션 B. DB 전용 서버 운영



장점

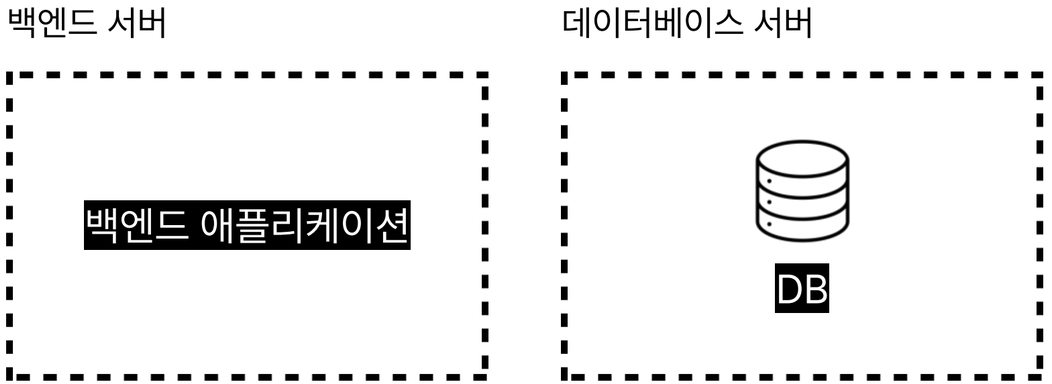
- 백엔드 서버가 죽어도 살아남음
- 분산 및 확장이 쉬워짐
- 역할 분리 가능(App / DB)

단점

- 네트워크 구성이 복잡해질 수 있음
- DB 운영을 위한 추가적인 비용이 필요할 수 있음

데이터베이스 - 어떠한 운영 방식이 더 좋나요?

솔루션 B-1. 단일 서버로 DB 운영



장점

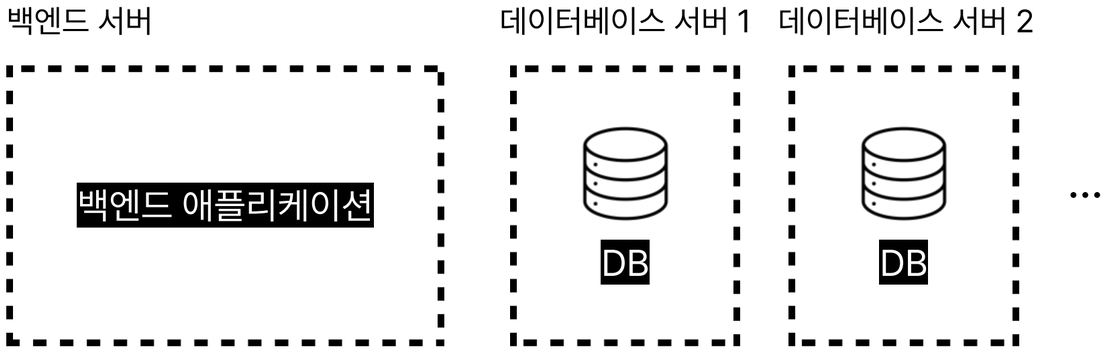
- 솔루션 B의 장점을 살리면서 비교적 적은 비용으로 운영 가능

단점

- 하나의 DB 서버가 죽을 경우 말짱도루묵
- Failover가 여전히 어려움

데이터베이스 - 어떠한 운영 방식이 더 좋나요?

솔루션 B-2. 분산 서버로 DB 운영



장점

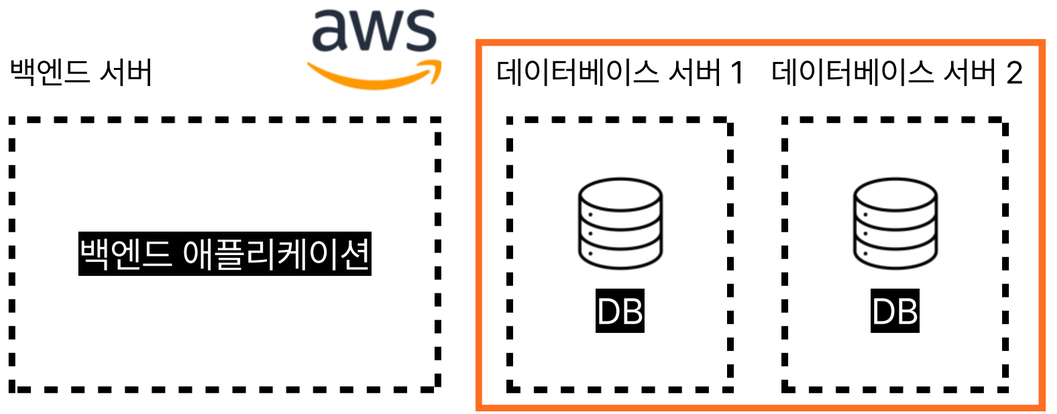
- 하나의 DB 서버가 죽어도 다른 DB 서버로 Failover하거나 대체 가능

단점

- DB, 네트워크 구성이 복잡해질 수 있음
- 운영 비용이 추가적으로 더 발생함
- 2개 이상의 DB 동기화가 필요해짐

데이터베이스 - 어떠한 운영 방식이 더 좋나요?

솔루션 B-3. 관리형(Managed) DB 서비스 사용



장점

- 직접 DB를 구축하고 관리할 필요가 사실상 없어짐
- 업체의 문제로 인한 다운타임/데이터 소실은 보상받을 수 있음
- 연동 가능한 서비스가 다양함
- DB 분산/Failover 구성이 간단해짐
- 데이터 백업/스냅샷 구성이 간단해짐

단점

- 업체에서 책정한 요금대로 약간의 비용이 더 발생 가능
- 클라우드 인프라 지식을 별도로 알아야함

사실 다루고 싶은 내용은 많지만...

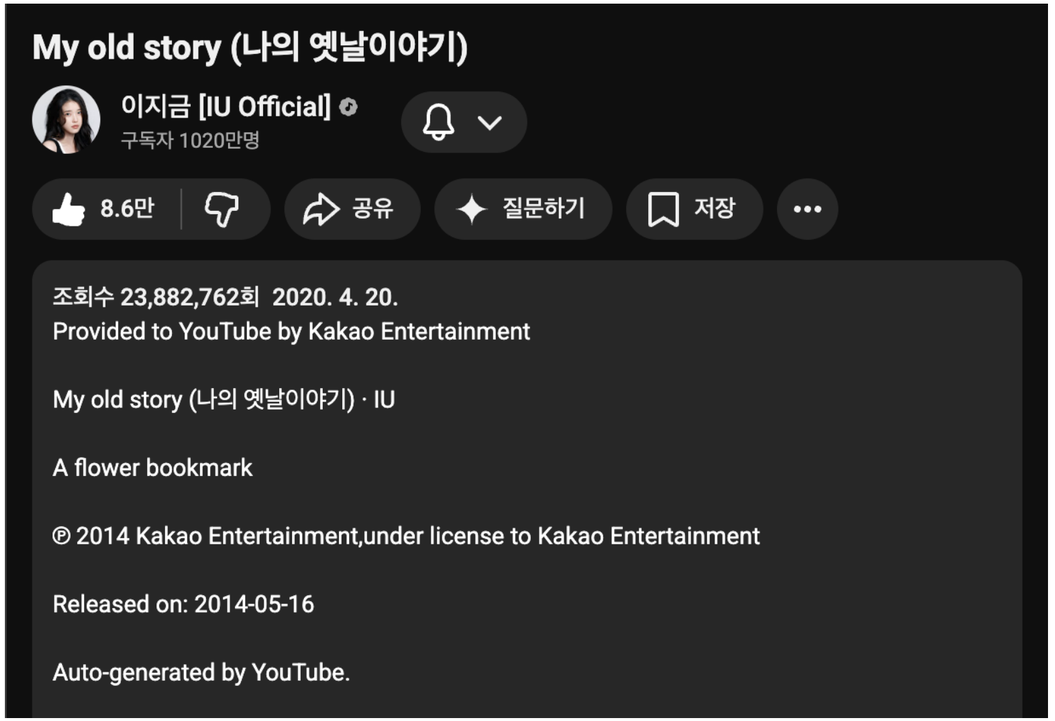
데이터베이스 기술이 쉬운 개념은 아니고 사전 지식도 필요하여 더이상의 설명은 시간 관계상 생략하겠음.

퀴즈) 유튜브 동영상도 데이터베이스에 저장될까?

퀴즈) 유튜브 동영상도 데이터베이스에 저장될까? 정답: 반은 맞고 반은 틀리다.

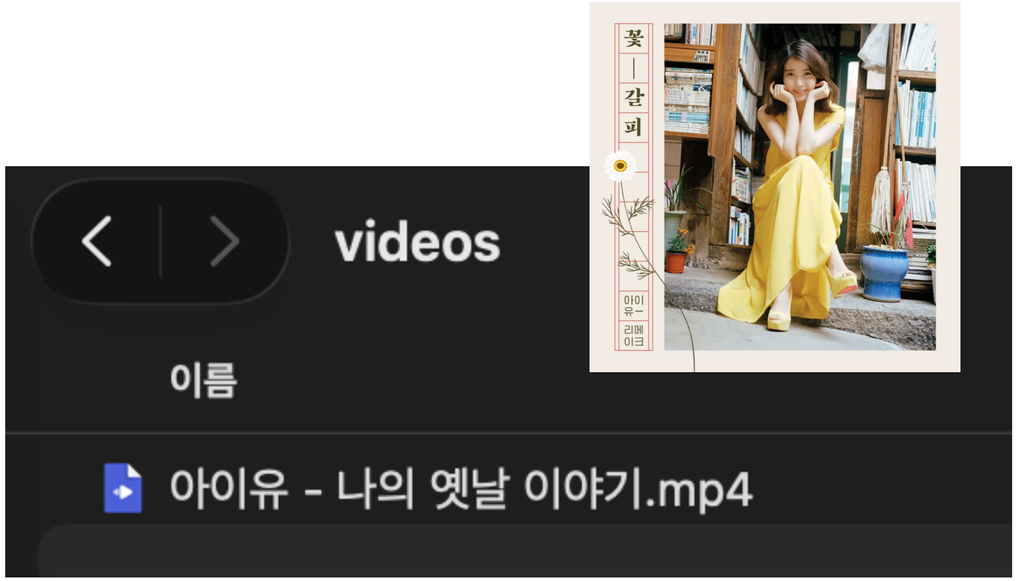
퀴즈) 유튜브 동영상도 데이터베이스에 저장될까?

정답: 반은 맞고 반은 틀리다.



영상에 대한 부가적인 정보 (메타데이터)

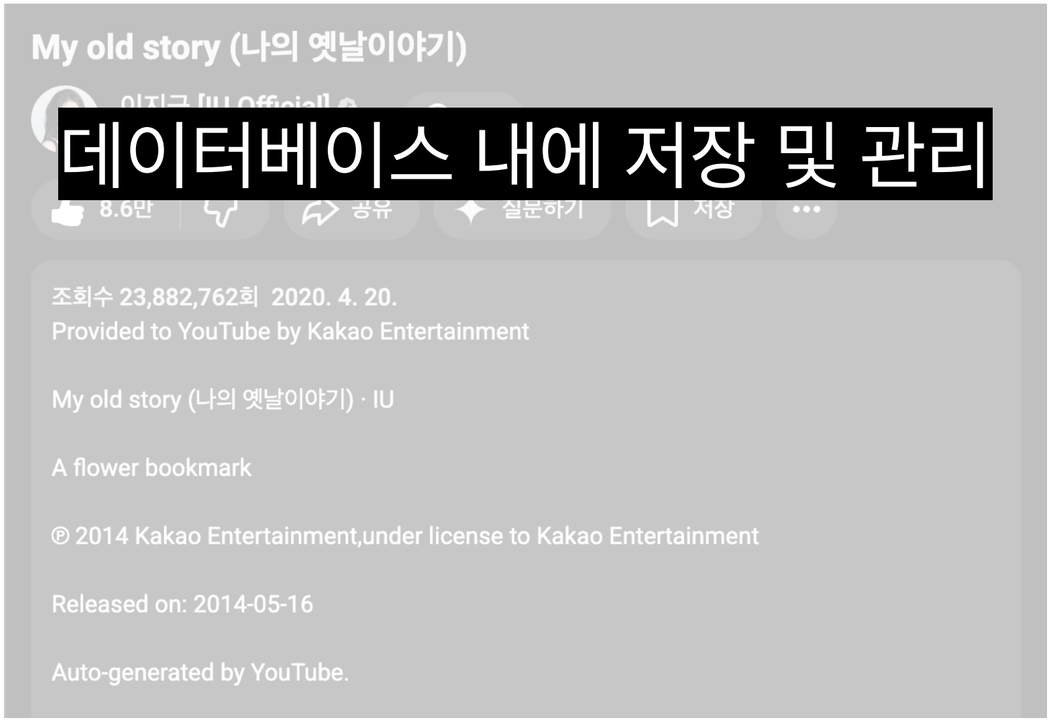
ex. 영상 제목/설명, 조회수, 좋아요 수, 업로드 날짜 등등..

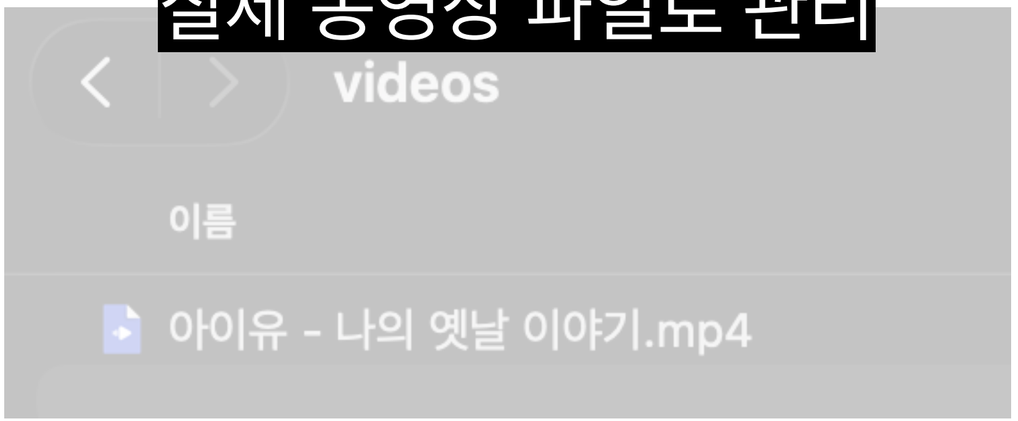


실제 영상 (동영상 파일)

(물론 실제로 저렇게 저장되진 않음)

퀴즈) 유튜브 동영상도 데이터베이스에 저장될까? 정답: 반은 맞고 반은 틀리다.





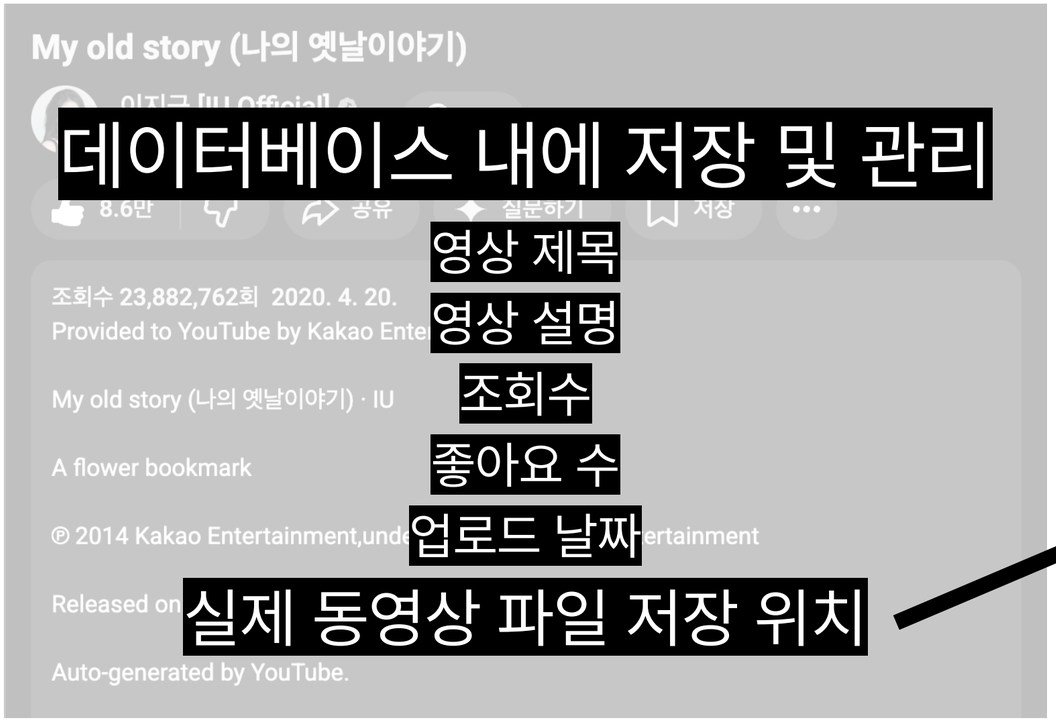
영상에 대한 부가적인 정보 (메타데이터)

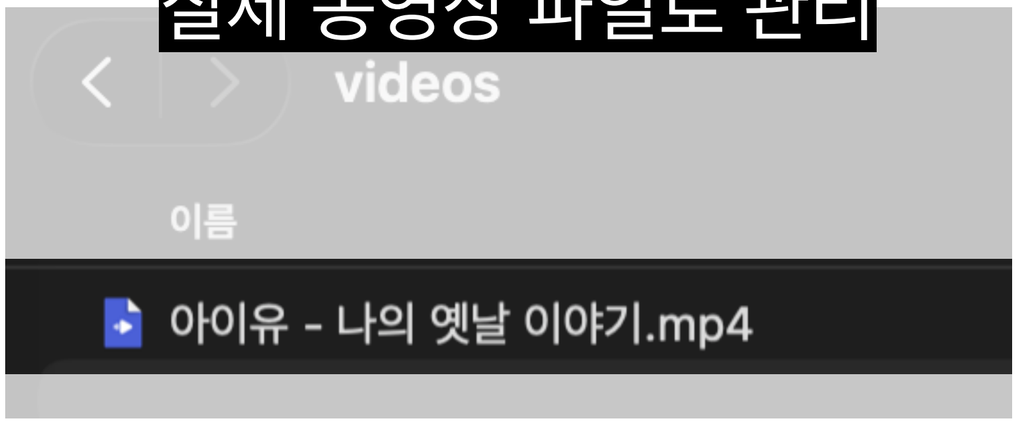
ex. 영상 제목/설명, 조회수, 좋아요 수, 업로드 날짜 등등..

실제 영상 (동영상 파일)

(물론 실제로 저렇게 저장되진 않음)

퀴즈) 유튜브 동영상도 데이터베이스에 저장될까? 정답: 반은 맞고 반은 틀리다.





영상에 대한 부가적인 정보 (메타데이터)

ex. 영상 제목/설명, 조회수, 좋아요 수, 업로드 날짜 등등..

실제 영상 (동영상 파일)

(물론 실제로 저렇게 저장되진 않음)

전공탐색동아리 널포유 2번째 시간은 여기서 끝.

그럼 웹 개발자는 뭘 할까?

TODO/WIP. 널포유 강습 마지막 시간에 다루는 편이 더 좋을 듯.

오늘날 웹 개발은 절대 건너뛸 수 없다.

개발자는 를 알아야한다.

개발자는 프로그래밍 언어 을(를) 알아야한다.

개발자는 컴퓨터 과학 을(를) 알아야한다.

개발자는 알고리즘 을(를) 알아야한다.

개발자는 네트워크 을(를) 알아야한다.

개발자는 보안 을(를) 알아야한다.

개발자는 버전 관리 을(를) 알아야한다.

개발자는 운영체제 을(를) 알아야한다.

(웹) 개발자는 서버 을(를) 알아야한다.

(웹) 개발자는 클라우드와 인프라 을(를) 알아야한다.

개발자는 협업/소통 능력 을(를) 알아야한다.

지금 시대 개발자는 AI/LLM 사용 능력 을(를) 알아야한다.