COTATO

https://www.cotato.com

CS EDUCATION

CORS

23.12.22
8기 교육팀장 신유승

COTATO

https://www.cotato.com

API 요청 예시

COTATO

https://www.cotato.com

Access to XMLHttpRequest at 'http://localhos login:1
t:8080/login' from origin 'http://localhost:3000' has
been blocked by CORS policy: Response to preflight
request doesn't pass access control check: No 'Access-
Control-Allow-Origin' header is present on the
requested resource.

COTATO

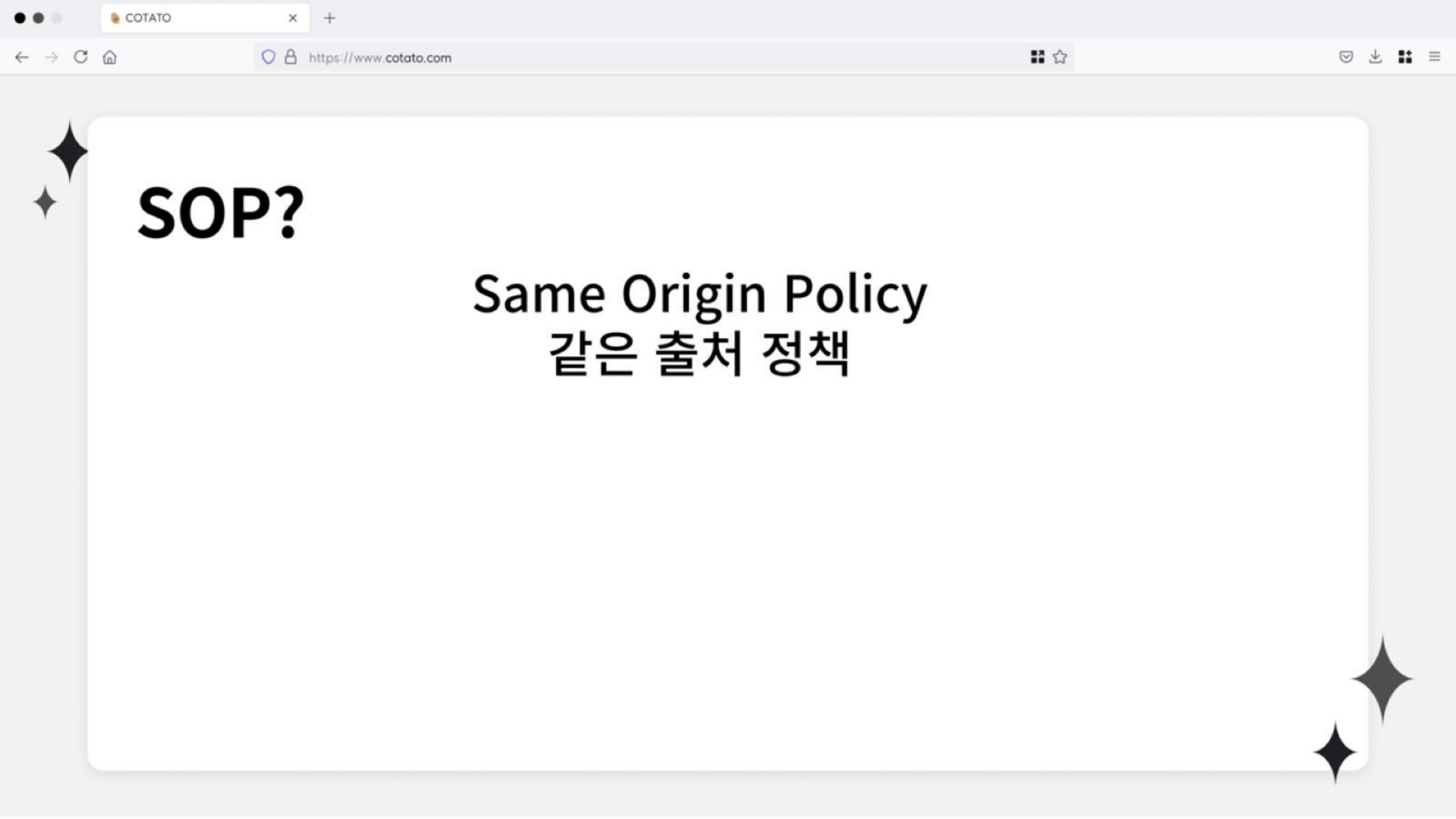
https://www.cotato.com

```
Access to XMLHttpRequest at 'http://localhos  login:1
t:8080/login' from origin 'http://localhost:3000' has
been blocked by CORS policy: Response to preflight
request doesn't pass access control check: No 'Access-
Control-Allow-Origin' header is present on the
requested resource.
```

CORS 정책에 의해 요청이 차단됨!

교육 목표

1. CORS를 이해하고 관련 설정을 하자!
2. '출처'가 무엇인지 이해하자



SOP?

Same Origin Policy
같은 출처 정책


COTATO
https://www.cotato.com
SOP? CORS?
Same Origin Policy
같은 출처 정책
Cross Origin Resource Sharing
다른 출처 자원 공유

SOP? CORS?

Same Origin Policy
같은 출처 정책

Cross Origin Resource Sharing
다른 출처 자원 공유

COTATO

https://www.cotato.com

출처(Origin?)

프로토콜 + 도메인 + 포트

출처(Origin?)

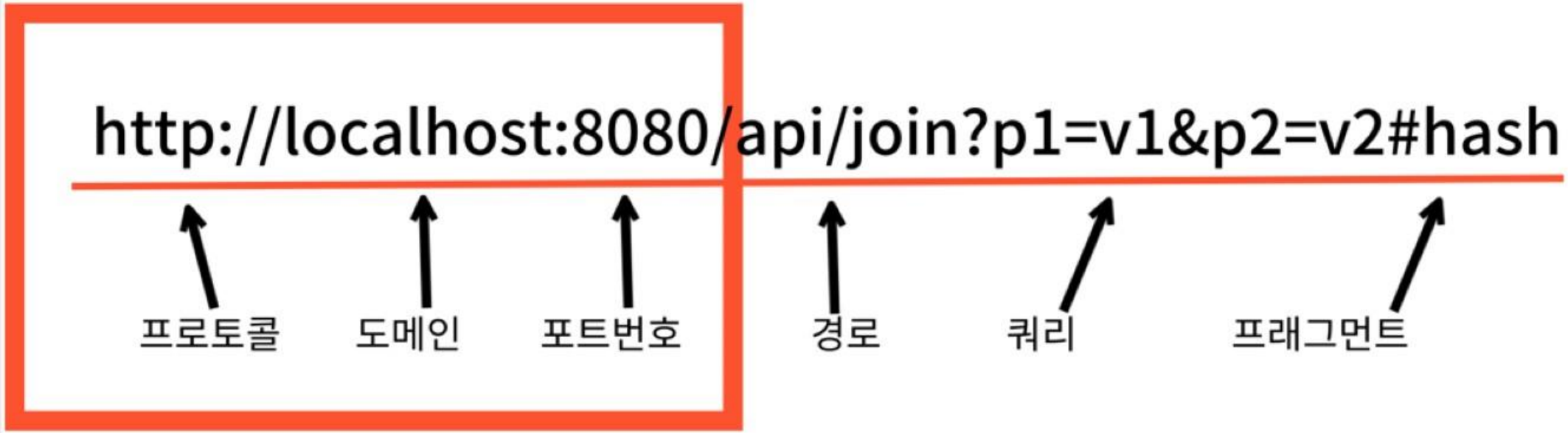
http://localhost:8080/api/join?p1=v1&p2=v2#hash

출처(Origin?)

COTATO

https://www.cotato.com

출처(Origin?)



COTATO

https://www.cotato.com

SOP란?

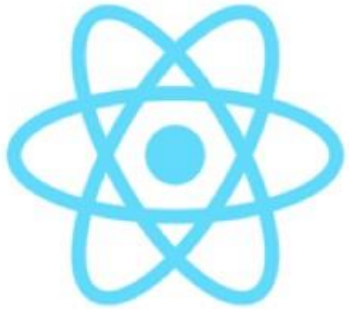
Same Origin Policy
같은 출처 정책

→ 같은 출처끼리만 자원을 공유한다!

CSRF,XSS 공격으로부터 자원 보호

CORS 정책이란?

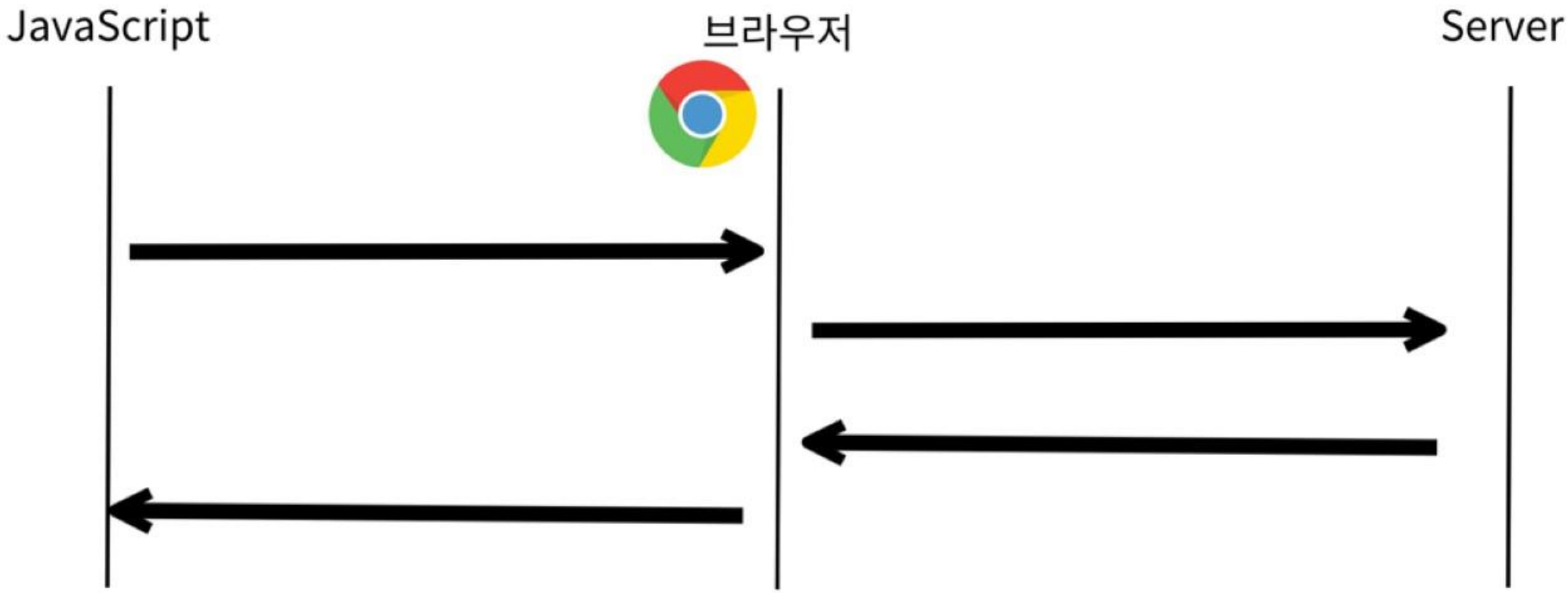
Cross Origin
Resource Sharing
다른 출처 자원 공유



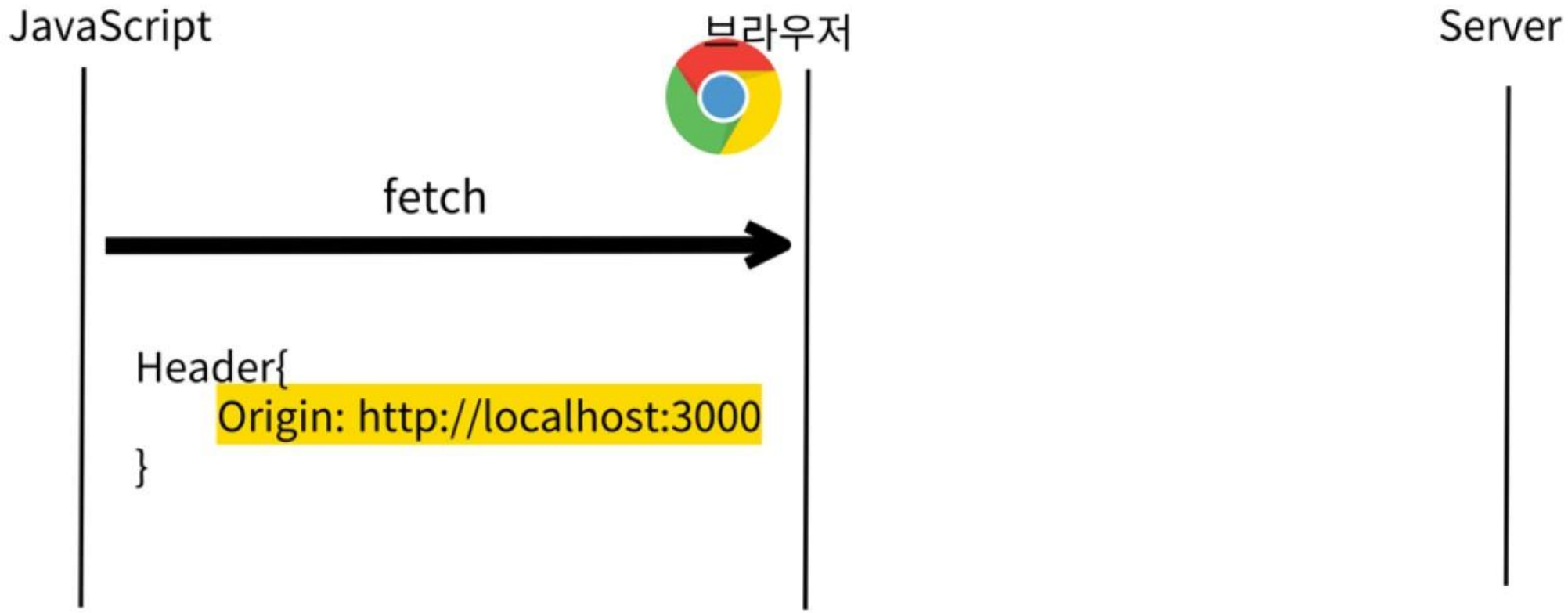
→

다른 출처의 자원 공유를
허용하는 브라우저 정책

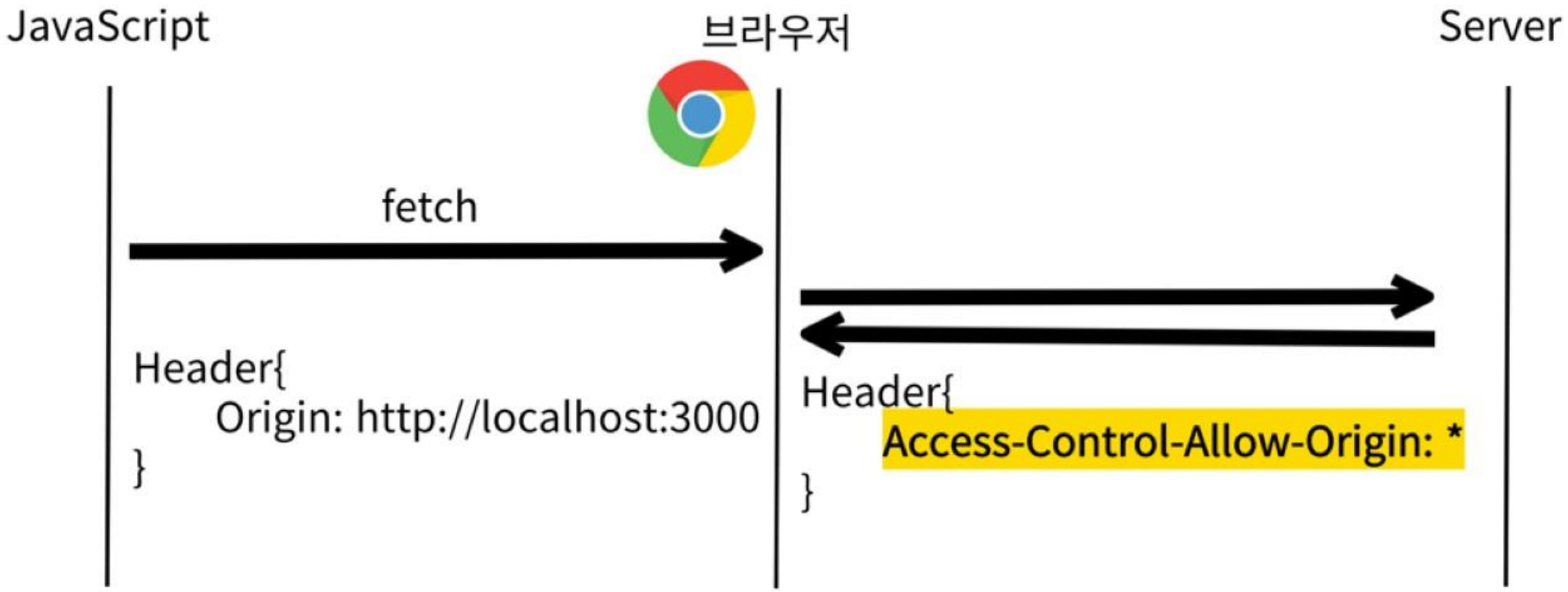
CORS 동작원리



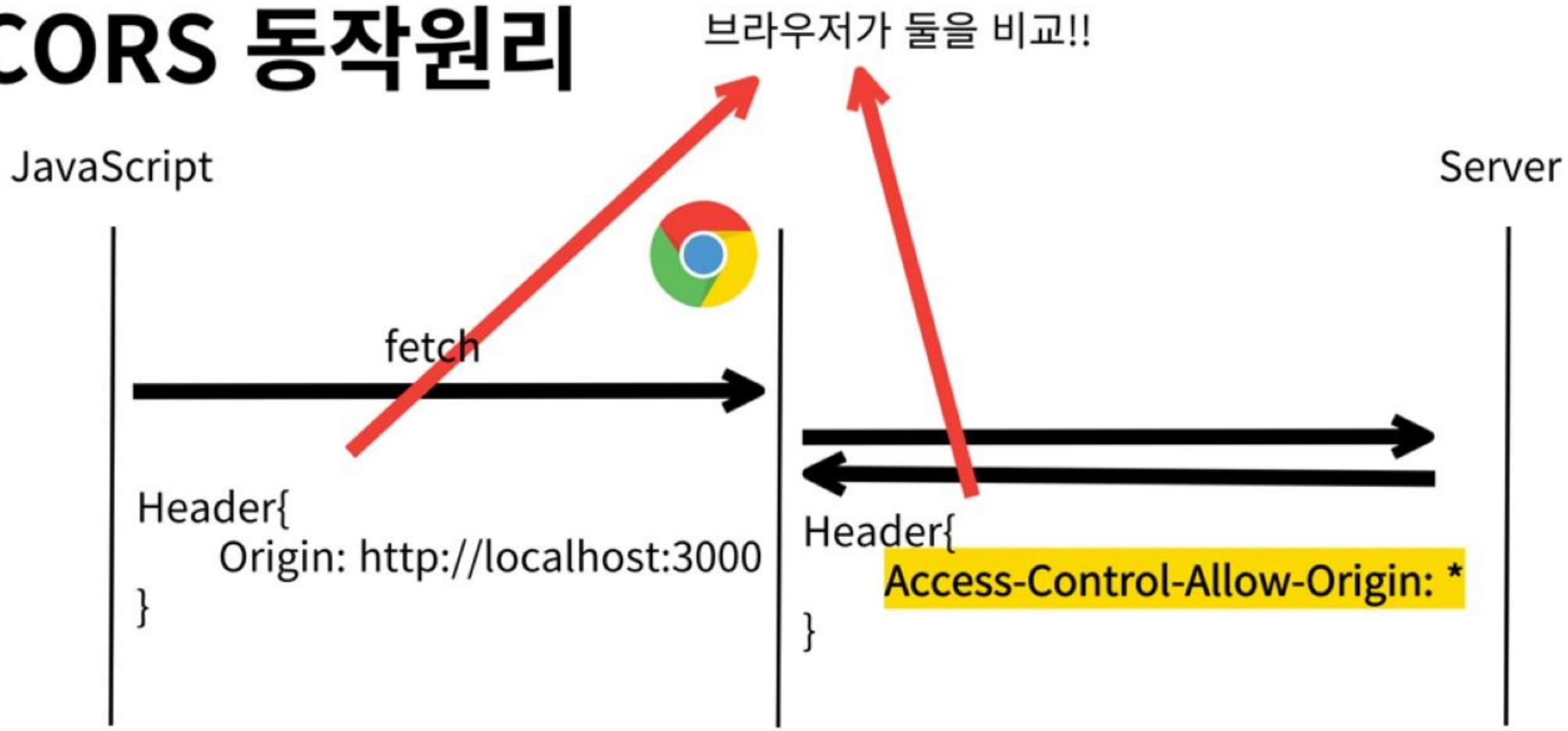
CORS 동작원리



CORS 동작원리



CORS 동작원리


브라우저가 둘을 비교!!
JavaScript
Server
fetch
Header{
Origin: http://localhost:3000
}
Header{
Access-Control-Allow-Origin: *
}

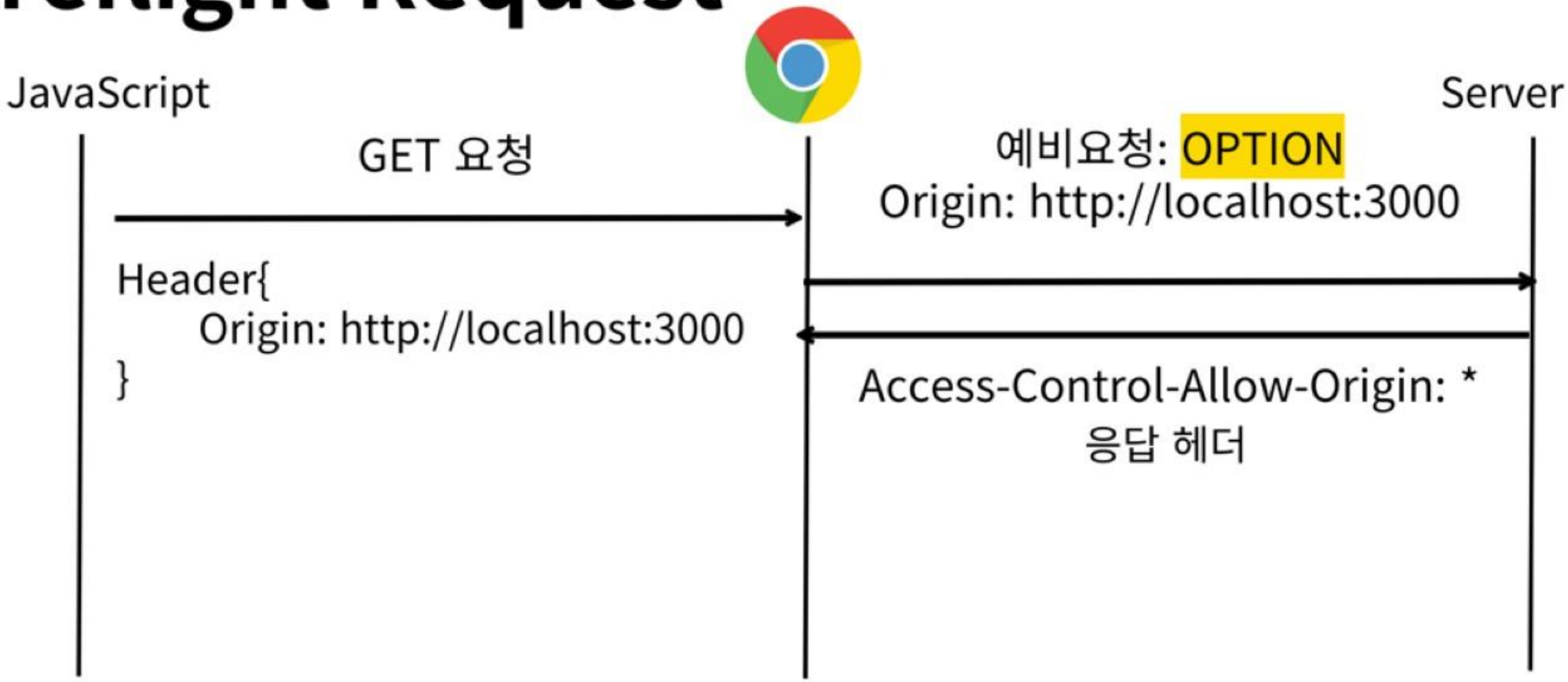
COTATO

https://www.cotato.com

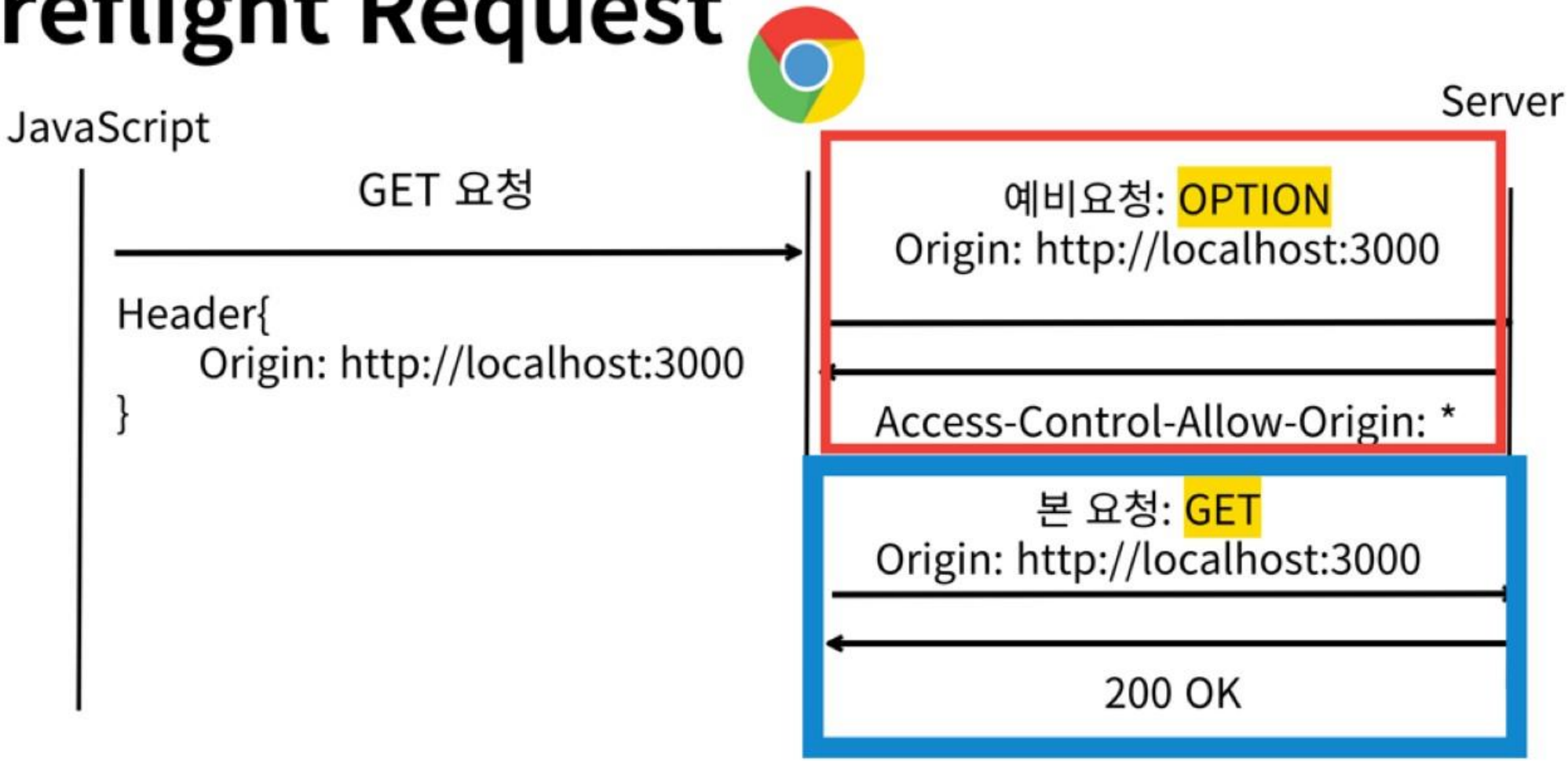
CORS 시나리오

1. Simple Request
2. Preflight Request
3. Credentialed Request

Preflight Request



Preflight Request



Preflight Request

예비 요청을 보내는 이유?

불필요한 리소스를 낭비하는 일을 방지하기 위함!!!

Credentialed Request

Preflight Request와 동일하게 예비요청을 보냄.
`인증 정보`가 필요한 요청

COTATO

https://www.cotato.com

Credentialed Request

Server

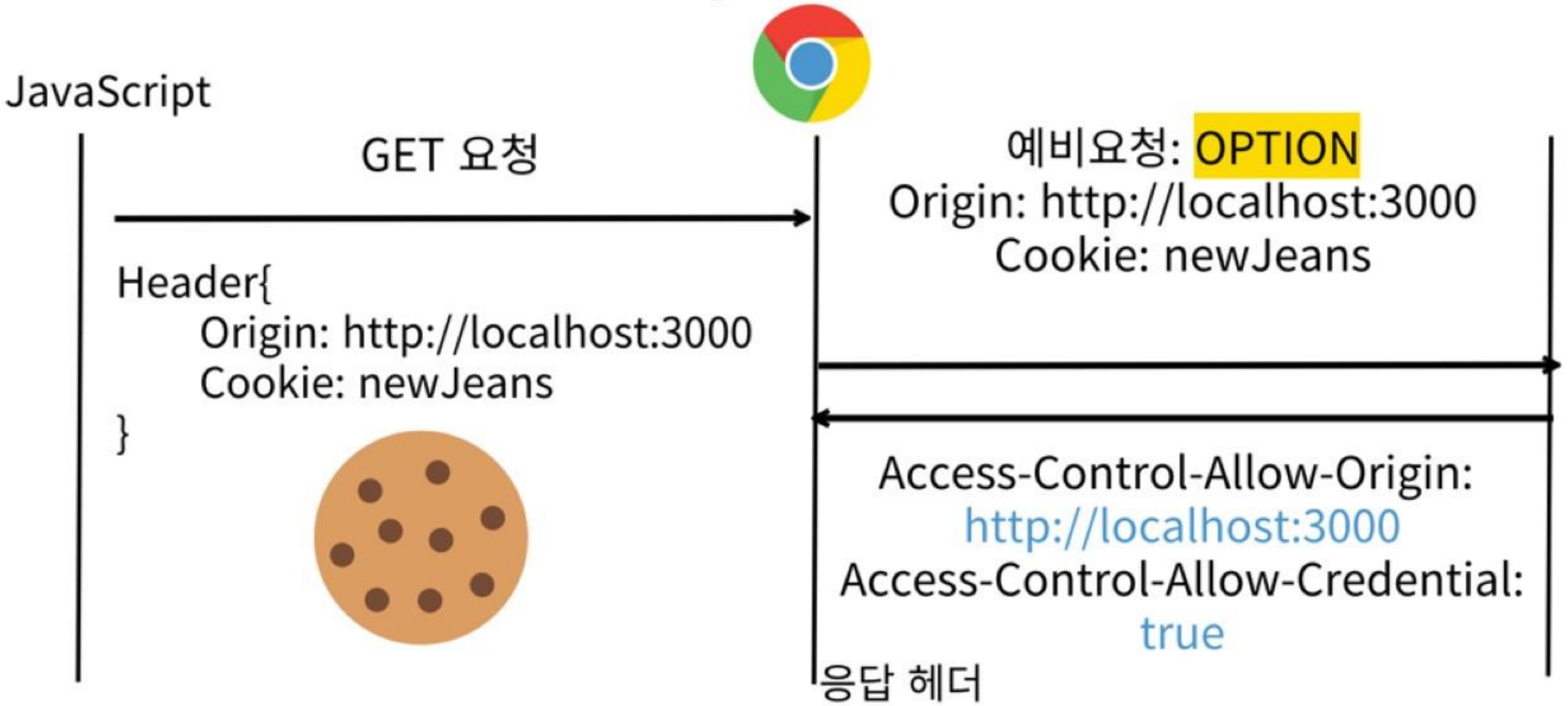
Credentialed Request

Server

Access-Control-Allow-Credential을
true로 설정하자!

Credentialed Request

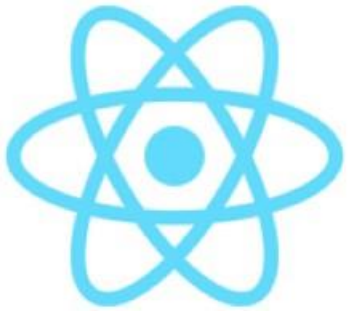
Credentialed Request



Credentialed Request: 주의할 점

Access-Control-Allow-Origin에는 *을 사용할 수 없다.
명시적인 url을 지정해야한다!!!
응답헤더에는 Access-Control-Allow-Credential: true여야한다.

Credentialed Request: 주의할 점



요청에 Credential 설정이 필요함!

```
fetch("https://example.com:1000/projects", {
        credentials: "include",
});

axios.get("https://example.com/items", {
        withCredentials: true,
});
```

시나리오 비교

	Preflight Request	Credentialed Request
예비요청	존재함	존재함
Access-Control-Allow-Origin	와일드 카드 사용 가능	명시적인 url
자격증명	쿠키를 전달하지 않음	쿠키 전달

COTATO

https://www.cotato.com

CS QUIZ

CORS 정책을 사용하는 이유로 옳은 것을 고르시오.

1. CORS는 SOP를 따르기 위한 정책이다.
2. CORS는 네트워크 요청 속도를 빠르게 한다.
3. CORS는 같은 출처끼리 자원 공유를 허용하는 브라우저 정책이다.
4. CORS는 Cross - Origin Resource Sharing의 약자이다.

A1

CORS 정책을 사용하는 이유로 옳은 것을 고르시오.

1. CORS는 SOP를 따르기 위한 정책이다.
2. CORS는 네트워크 요청 속도를 빠르게 한다.
3. CORS는 같은 출처끼리 자원 공유를 허용하는 브라우저 정책이다.
4. CORS는 Cross - Origin Resource Sharing의 약자이다.

COTATO

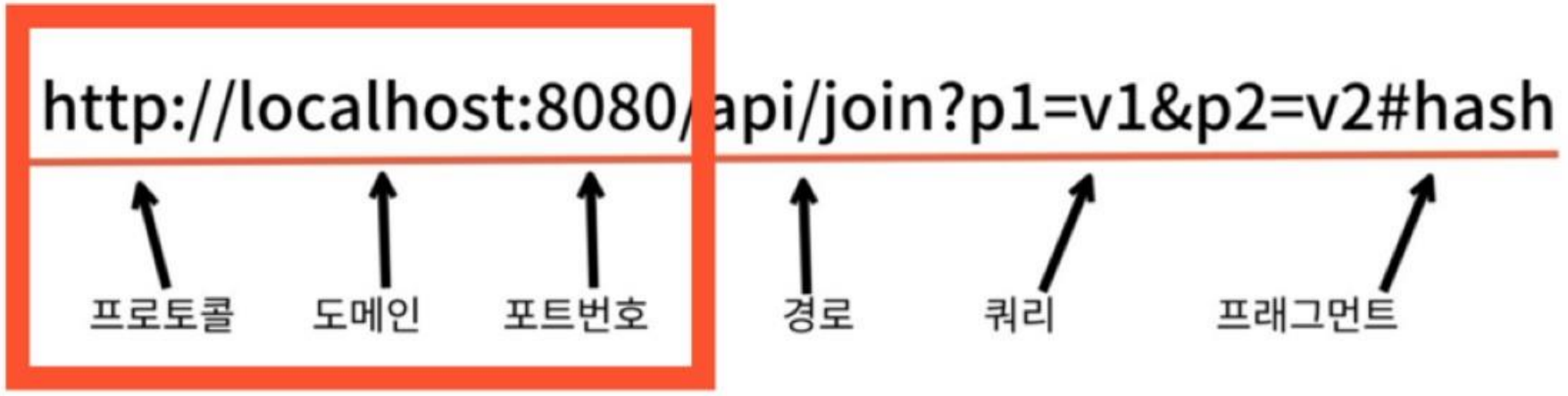
https://www.cotato.com

Q2

그림을 보고 웹 상에서 출처, Origin, 자원의 소유자를 구분할 때 필요한 요소를 모두 고르시오.

A2

그림을 보고 웹 상에서 출처, Origin, 자원의 소유자를 구분할 때 필요한 요소를 모두 고르시오.



프로토콜, 도메인, 포트번호

Q3

Credentialed Request에서 서버에서 와일드카드(*)를 사용하면 발생할 수 있는 문제는?

a) 인증 정보가 누출될 수 있는 보안 문제 발생

b) 예비 요청이 전달되지 않음

c) CORS 정책에 영향을 미치지 않음

A3

Credentialed Request에서 서버에서 와일드카드(*)를 사용하면 발생할 수 있는 문제는?

a) 인증 정보가 누출될 수 있는 보안 문제 발생

b) 예비 요청이 전달되지 않음

c) CORS 정책에 영향을 미치지 않음

Q4 SOP 및 CORS 관련 개념에 대해 바르게 설명하는 친구를 고르시오.

송강: SOP는 다른 출처의 자원 공유를 허용하는 브라우저 정책이고 CORS는 같은 출처끼리만 자원을 공유하는 정책이야.

도현: SOP는 Same Only Policy의 줄임말이고 CORS는 Cross Origin Resource Strategy의 줄임말이야.

진욱: SOP와 CORS에서 등장하는 개념인 Origin, 출처는 url의 프로토콜, 도메인, 그리고 포트까지 3 가지를 합친 것을 뜻해.

A4 SOP 및 CORS 관련 개념에 대해 바르게 설명하는 친구를 고르시오.

송강: SOP는 다른 출처의 자원 공유를 허용하는 브라우저 정책이고 CORS는 같은 출처끼리만 자원을 공유하는 정책이야.

도현: SOP는 Same Only Policy의 줄임말이고 CORS는 Cross Origin Resource Strategy의 줄임말이야.

진욱: SOP와 CORS에서 등장하는 개념인 Origin, 출처는 url의 프로토콜, 도메인, 그리고 포트까지 3 가지를 합친 것을 뜻해.

Q5 다음 중 옳은 것끼리 짝지은 보기를 고르시오.

ㄱ. Preflight Request - 예비요청 존재함, Credentialed Request - 예비요청 존재함
ㄴ. Preflight Request - 예비요청 존재하지 않음, Credentialed Request - 예비요청 존재함
ㄷ. Preflight Request - 쿠키를 전달함, Credentialed Request - 쿠키를 전달하지 않음
ㄹ. Preflight Request - 쿠키를 전달하지 않음, Credentialed Request - 쿠키를 전달함

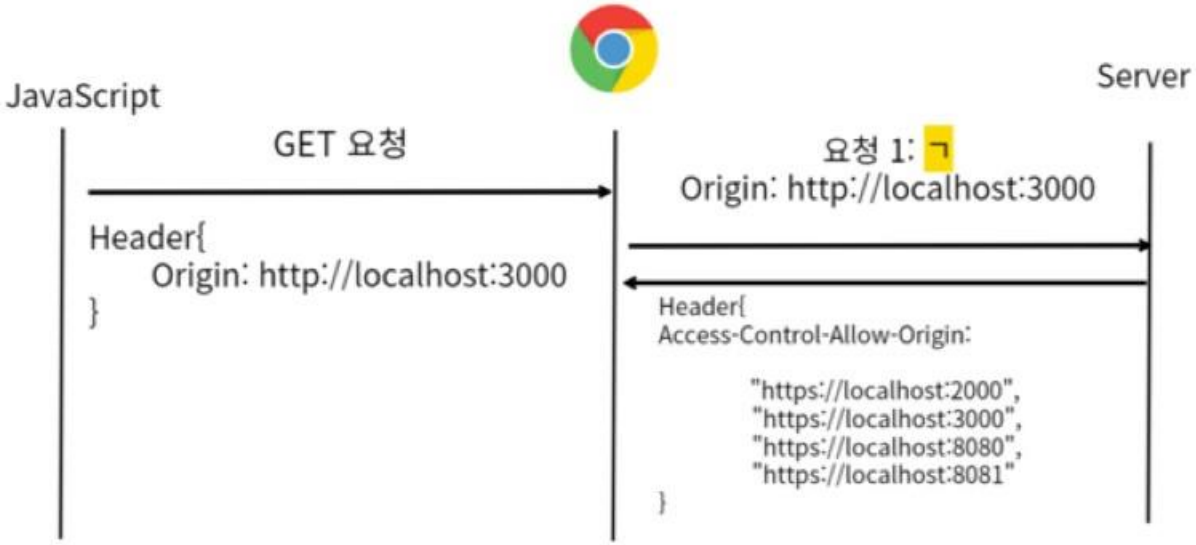
1. ㄱ ㄴ
2. ㄴ ㄷ
3. ㄴ ㄹ
4. ㄱ ㄹ

A5 다음 중 옳은 것끼리 짝지은 보기를 고르시오.

ㄱ. Preflight Request - 예비요청 존재함, Credentialed Request - 예비요청 존재함
ㄴ. Preflight Request - 예비요청 존재하지 않음, Credentialed Request - 예비요청 존재함
ㄷ. Preflight Request - 쿠키를 전달함, Credentialed Request - 쿠키를 전달하지 않음
ㄹ. Preflight Request - 쿠키를 전달하지 않음, Credentialed Request - 쿠키를 전달함

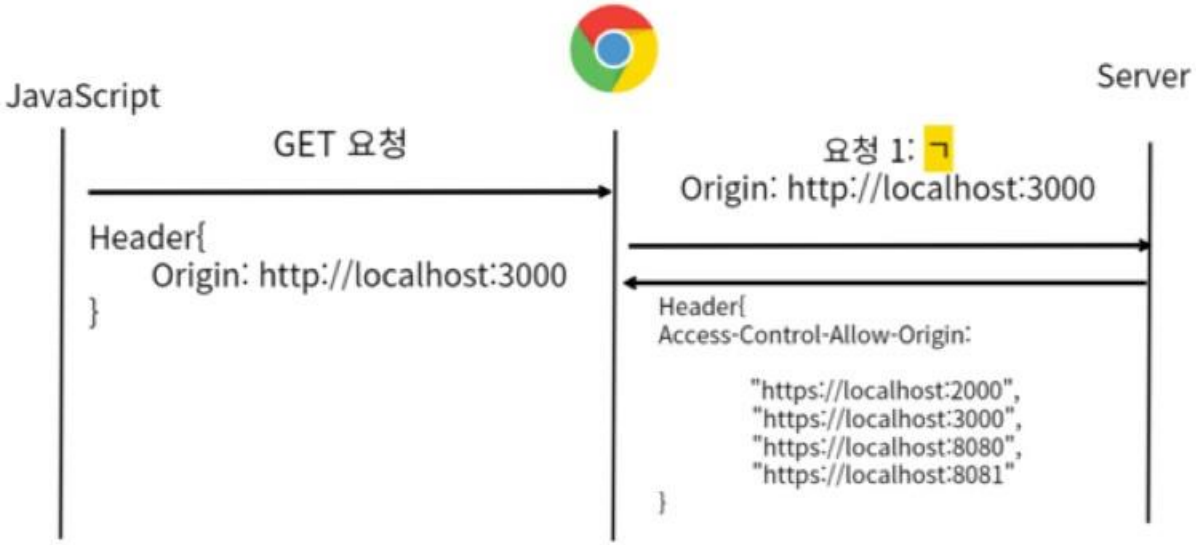
1. ㄱ ㄴ
2. ㄴ ㄷ
3. ㄴ ㄹ
4. ㄱ ㄹ

Q6 다음은 Preflight Request의 Flow다. 옳게 말하는 친구를 고르시오



산타클로스: ㄱ에 들어갈 메서드는 GET 메서드이다.
루돌프: 서버는 요청의 Origin과 Access-Control-Allow-Origin을 비교 후 일치하지 않으면 CORS에러를 발생시킨다.
신순록: 위 요청의 경우 CORS에러가 발생하지 않는다.
썰매: FE에서 요청을 보낼 때 Credential 설정을 true 또는 include로 설정해줘야해
막대사탕: 요청1을 마치고 에러가 발생하지 않으면 진짜 요청을 다시 보내겠네

다음은 Preflight Request의 Flow다. 옳게 말하는 친구를 고르시오



산타클로스: ㄱ에 들어갈 메서드는 GET 메서드이다.
루돌프: 서버는 요청의 Origin과 Access-Control-Allow-Origin을 비교 후 일치하지 않으면 CORS에러를 발생시킨다.
신순록: 위 요청의 경우 CORS에러가 발생하지 않는다.
썰매: FE에서 요청을 보낼 때 Credential 설정을 true 또는 include로 설정해줘야해
막대사탕: 요청1을 마치고 에러가 발생하지 않으면 진짜 요청을 다시 보내겠네

CORS의 정책은 어떤 목적으로 만들어진 브라우저 정책인가요?

a) CSRF 방지
b) XSS 방지
c) 다른 출처 간 자원 공유 허용

CORS의 정책은 어떤 목적으로 만들어진 브라우저 정책인가요?

a) CSRF 방지
b) XSS 방지
c) 다른 출처 간 자원 공유 허용

Q8 CORS 동작 방식을 모두 고르시오

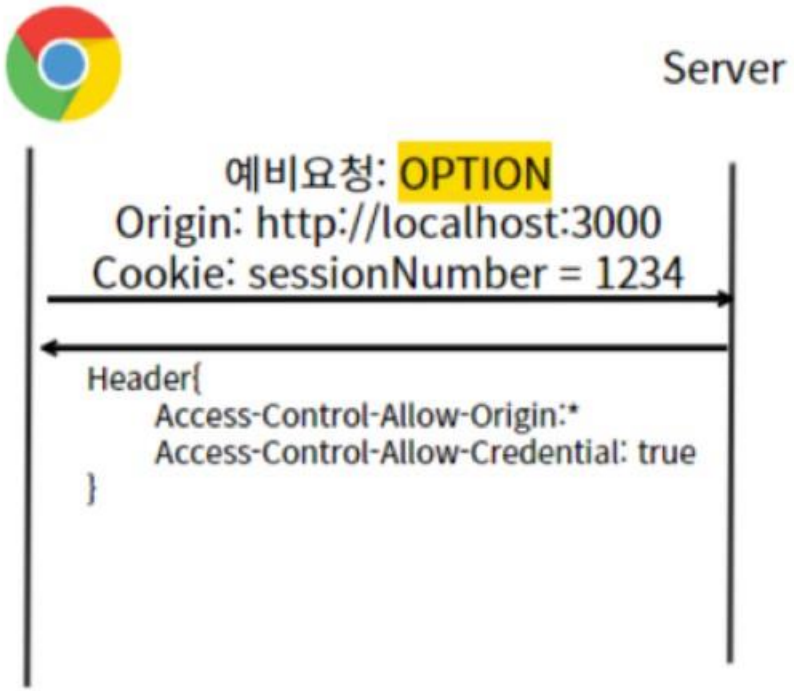
1. Fetch Request
2. Simple Request
3. Preflight Request
4. API Request
5. Credentialed Request

A8

CORS 동작 방식을 모두 고르시오

1. Fetch Request
2. Simple Request
3. Preflight Request
4. API Request
5. Credentialed Request

Q9 브라우저가 서버에 요청을 한 이후 에러가 발생하지 않기 위해 무엇을 어떻게 바꿔야할까?



브라우저가 서버에 요청을 한 이후 에러가 발생하지 않기 위해 무엇을 어떻게 바꿔야할까?

Access-Control-Allow-Origin 헤더를 와일드카드로 설정하지말고 명시적인 url을 설정해야한다.

Q10 CORS 시나리오에 관한 설명으로 옳지 않은 것을 고르시오.

1. Credentials Request Access-Control-Allow-Origin에 fetch 요청을 허용하는 출처를 표시한다.
2. CORS 에러에 관해 클라이언트 쪽 해결책으론 프록시라는 방법이 있다.
3. 서버가 Access-Control-Allow-Credential을 True로 설정한 경우
   클라이언트는 요청을 보낼 때 Credential 설정을 해야 오류를 방지할 수 있다.
4. 브라우저는 예비 요청과 본 요청에 동일한 http 메서드를 사용한다.

A10

CORS 시나리오에 관한 설명으로 옳지 않은 것을 고르시오.

1. Credentials Request Access-Control-Allow-Origin에 fetch 요청을 허용하는 출처를 표시한다.
2. CORS 에러에 관해 클라이언트 쪽 해결책으론 프록시라는 방법이 있다.
3. 서버가 Access-Control-Allow-Credential을 True로 설정한 경우
   클라이언트는 요청을 보낼 때 Credential 설정을 해야 오류를 방지할 수 있다.
4. 브라우저는 예비 요청과 본 요청에 동일한 http 메서드를 사용한다.

Q11 빈칸에 알맞은 선택을 하고 단어를 넣으시오

Credentialed Request는 Preflight Request의 예비 요청을 (보내고 / 보내지 않고)
자격 증명을 위한 OOOO가 필요한 요청이다.

A11 빈칸에 알맞은 선택을 하고 단어를 넣으시오

Credentialed Request는 Preflight Request의 예비 요청을 (보내고 / 보내지 않고)
자격 증명을 위한 인증정보가 필요한 요청이다.

Q12 다음 빈칸에 알맞은 용어를 순서대로 적고 알맞은 답을 선택하시오.

같은 출처 정책의 약자인 ( )는 CSRF, XSS 공격등으로부터 자원을 보호하기 위해 설정한 브라우저 정책이다.
초기 웹 개발엔 해당 정책에 문제 없이 개발을 진행할 수 있었지만, 프론트엔드와 백엔드가 구분되어 다른 출처간의 자원 공유 (허용하기 , 차단하기)위해 등장한 ( )라는 브라우저 정책이 등장했다.

다음 빈칸에 알맞은 용어를 순서대로 적고 알맞은 답을 선택하시오.

같은 출처 정책의 약자인 ( )는 CSRF, XSS 공격등으로부터 자원을 보호하기 위해 설정한 브라우저 정책이다.
초기 웹 개발엔 해당 정책에 문제 없이 개발을 진행할 수 있었지만, 프론트엔드와 백엔드가 구분되어 다른 출처간의 자원 공유 (허용하기 , 차단하기)위해 등장한 ( )라는 브라우저 정책이 등장했다.

SOP, 허용하기, CORS

다음 빈칸에 알맞은 용어를 순서대로 적고 알맞은 답을 선택하시오.

같은 출처 정책의 약자인 ( )는 CSRF, XSS 공격등으로부터 자원을 보호하기 위해 설정한 브라우저 정책이다.
초기 웹 개발엔 해당 정책에 문제 없이 개발을 진행할 수 있었지만, 프론트엔드와 백엔드가 구분되어 다른 출처간의 자원 공유 (허용하기 , 차단하기)위해 등장한 ( )라는 브라우저 정책이 등장했다.

SOP, 허용하기, CORS