⚠️

CORS ERROR

SOP?

자바스크립트 엔진 표준 스펙의 보안 규칙 중,

하나의 출처(Origin)에서 로드된 자원(문서나 스크립트)이 일치하지 않는 자원과 상호작용하지 못하도록 요청 발생을 제한하는 정책이 있습니다.

그것이 바로 `SOP` (Same Origin Policy, `동일 출처 정책`)입니다.

http://localhost:8000와

http://localhost:8000/posts는

같은 출처라서 상호작용이 가능한데,

http://google.com에서

http://localhost:8000를 호출하면 SOP에 위배됩니다.

그렇다면 동일한 출처의 기준은 무엇일까요?

동일 출처의 기준

```
 http://example.com:8042/over/there?name=ferret&page=1#nose
 \__/  \_________/\___/ \_______/ \_________________/\___/
   |         |       |       |                |         |
protocol    host    port    path       query string  Fragment
```

`protocol, host, port`가 같아야 동일한 출처 입니다.

그렇다면 왜 SOP를 지켜야 할까요?

이런 경우를 생각해봅시다.

사용자가 스팸메일을 열었습니다.

그 메일을 오픈하면 은행에 돈을 출금하는 request를 전송합니다.

요청은 **http://hacker.com**라는 출처로 전송이 됩니다.

그렇다면, request 출처는 **http://hacker.com**이고,

은행의 출처인 response는 **https://bank.com**이기 때문에

서로 출처가 다르기에 브라우저는 CORS 에러를 뿜고, 돈이 빠져나가지 않습니다.

그렇다면, 다른 출처의 리소스가 필요하다면 어떻게 할까요?

`→ CORS를 사용하면 됩니다.`

CORS

Cross Origin Resource Sharing, 교차 출처 리소스 공유

CORS는 다른 출처의 자원의 공유를 가능하게 만듭니다.

또한, 추가 HTTP 헤더를 사용하여,

한 출처에서 실행 중인 웹 에플리케이션이 다른 출처의 선택한 자원에 접근할 수 있는 권한을 부여하도록 `브라우저`에 알려주는 체제입니다.

CORS 에러는 브라우저가 뿜어내는 것입니다. Server↔Server는 CORS 에러가 나지 않습니다.

자 이제 CORS를 어떻게 해결해야 할까요?

CORS 해결하기

CORS 에러는 언제 나타날까요?

클라이언트에서 Server(api)에 접근하여 리소스를 가져올때,

출처가 같지 않으면 브라우저는 CORS 에러를 뿜습니다.

CORS Error는 Server에서 해결할 수도 있고, Client에서 해결할 수도 있습니다.

Server에서 해결하는 방법은 CORS 미들웨어를 사용하거나, Server에서 Access-access-control-allow-origin 헤더를 세팅해주면 해결됩니다.

그런데 만약, `Server를 수정할 수 없거나 Open API를 사용하는 경우`에는 클라이언트에서 처리를 해주어야 합니다. 클라이언트에서 CORS 에러를 해결하는 방법은 `프록시 서버를 이용`하는 것입니다.

프록시 서버란, 다른 네트워크 서비스에 간접적으로 접속할 수 있게 해주는 서버를 가리킵니다.

즉, Front에서 CORS 에러를 핸들링 하는 것은 서버로 가기 전에 프록시 서버를 거쳐서 출처를 response와 같게 수정하고, Server에 접근하도록 하는 것입니다.

React에서 Proxy Server를 구축하는 방법 은 2가지가 있습니다.

1. Webpack DevServer Proxy
2. http-proxy-middleware

첫번째 방법인 Webpack DevServer Proxy보다 http-proxy-middleware가 더 세세하게 세팅이 가능하다.

그러나 이러한 방법들 모두 `로컬서버 즉, 개발할때만 사용`이 가능합니다.