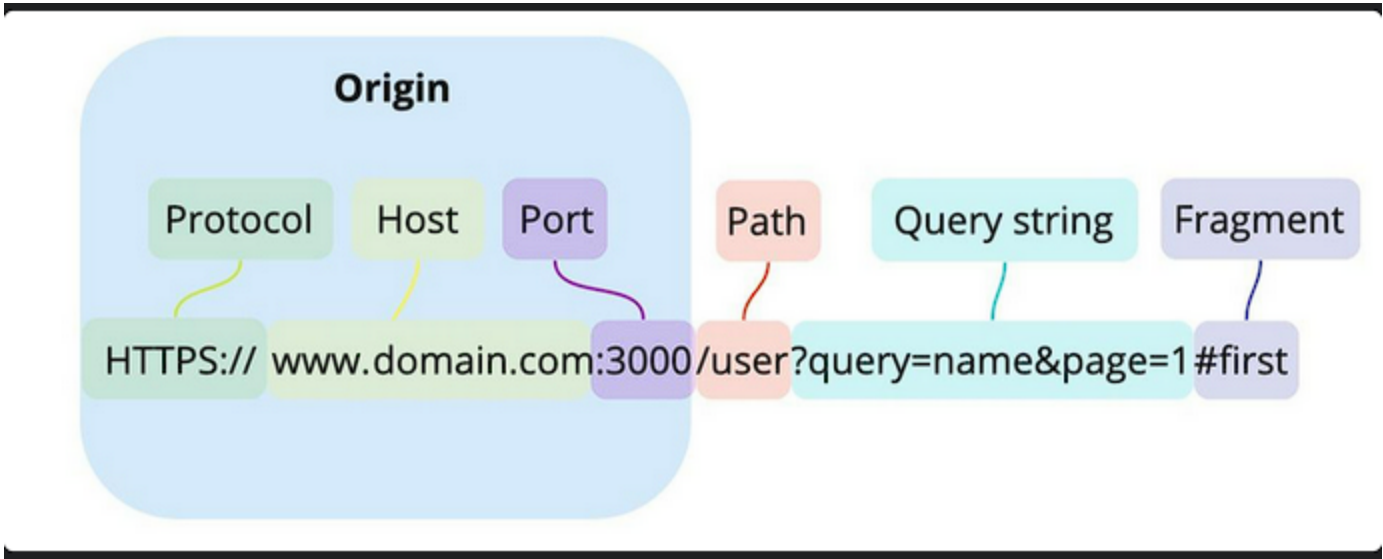


🕒 Created	@2022년 4월 25일 오후 10:07
🏷️ Tags	Personal Study
📄 Property	

CORS(Cross-origin resource sharing)

- CORS는 Cross-Origin Resource Sharing 의 줄임말로, 한국어로 직역하면 교차 출처 리소스 공유라고 해석할 수 있다. 여기서 “교차 출처”라고 하는 것은 “다른 출처”를 의미

출처(Origin)



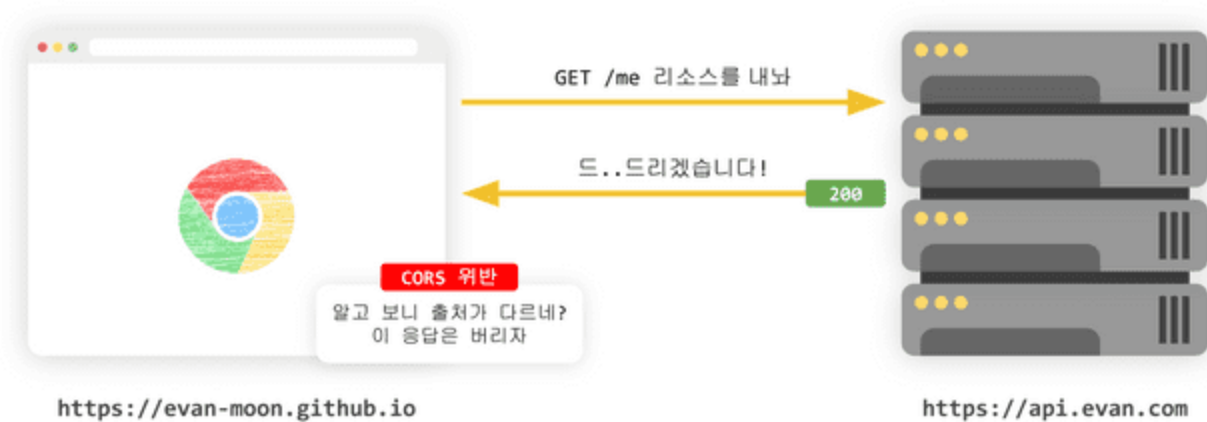
- 서버의 위치를 의미하는 URL들은 여러 개의 구성 요소로 이루어져 있다.
- 출처는 Protocol과 Host, 포트 번호를 합친 것을 의미한다. 즉, 서버의 위치를 찾아가기 위해 필요한 가장 기본적인 것들을 합쳐 놓은 것
- 기본적으로 HTTP, HTTPS 프로토콜의 기본 포트 번호가 있어 포트 번호는 생략이 가능하지만, 명시적으로 포함될 경우 포트 번호까지 모두 일치해야 같은 출처로 인정된다.

SOP와 CORS - 다른 출처로의 리소스 요청을 제한

- SOP(Same-Origin Policy): 같은 출처에서만 리소스를 공유할 수 있음
- 그러나 웹에서 다른 출처에 있는 리소스를 가져와서 사용하는 일은 굉장히 흔한 일이라 무작정 막을 수 없음 → 예외 조항을 두고 이 조항에 해당하는 리소스 요청은 출처가 다르더라도 허용 → 그 중 하나가 “CORS 정책을 지킨 리소스 요청”
- 다른 출처로 리소스를 요청하면 SOP 정책을 위반한 것이 되고, SOP 예외 조항인 CORS 정책까지 지키지 않는다면 아예 다른 출처의 리소스를 사용할 수 없음
- 다른 출처의 리소스 사용을 막는 것은 “보안”때문
  - 클라이언트 사이드 앱들의 경우 공격에 매우 취약
  - CSRF, XSS 등
  - 소스 코드 열람이 가능함(개발자 도구 이용)
  - 다른 출처의 앱이 서로 통신하는 것에 대해 아무런 제약도 존재하지 않는다면 정보를 탈취하기 매우 쉬움

출처의 구분

- 프로토콜, 호스트, 포트 3가지가 동일하면 같은 출처
- 인터넷 익스플로러의 경우 포트 번호를 완전히 무시함
- 인터넷 익스플로러의 경우와 같이 출처를 비교하는 로직은 서버에 구현된 스펙이 아니라 브라우저에 구현되어 있는 스펙
- 만약 우리가 CORS 정책을 위반하는 리소스 요청을 하더라도 해당 서버가 같은 출처에서 보낸 요청만 받겠다는 로직을 가지고 있는 경우가 아니라면 서버는 정상적으로 응답을 하고, 이후 브라우저가 이 응답을 분석해서 CORS 정책 위반이라고 판단되면 그 응답을 사용하지 않고 그냥 버리는 순서



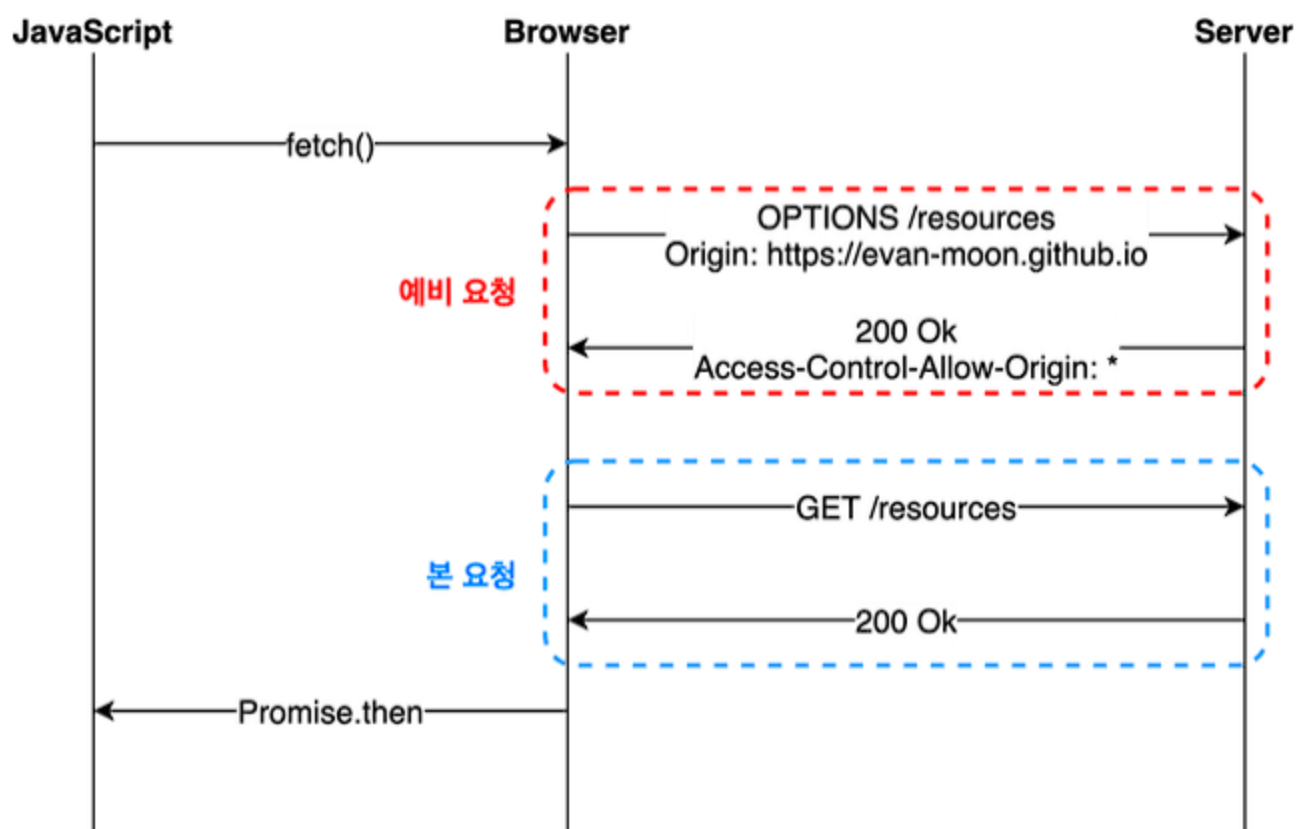
- 즉, CORS는 브라우저의 구현 스펙에 포함되는 정책이기 때문에, 브라우저를 통하지 않고 서버 간 통신을 할 때는 이 정책이 적용되지 않는다. 또한 CORS 정책을 위반하는 리소스 요청 때문에 에러가 발생했다고 해도 서버 쪽 로그에는 정상적으로 응답을 했다는 로그만 남기 때문에, CORS가 돌아가는 방식을 정확히 이해해야한다.

## CORS의 동작

- 기본적으로 웹 클라이언트 어플리케이션이 다른 출처의 리소스를 요청할 때는 HTTP 프로토콜을 사용하여 요청을 보내게 되는데, 이때 브라우저는 요청 헤더에 `Origin` 이라는 필드에 요청을 보내는 출처를 함께 담아보낸다.
- 이후 서버가 이 요청에 대한 응답을 할 때 응답 헤더의 `Access-Control-Allow-Origin` 이라는 값에 “이 리소스를 접근하는 것이 허용된 출처”를 내려주고, 이후 응답을 받은 브라우저는 자신이 보냈던 요청의 `Origin` 과 서버가 보내준 응답의 `Access-Control-Allow-Origin` 을 비교해본 후 이 응답이 유효한 응답인지 아닌지를 결정한다.

## CORS의 3가지 시나리오

- 프리플라이트(Preflight)
  - 브라우저는 요청을 한번에 보내지 않고 예비 요청과 본 요청으로 나누어서 서버로 전송
  - 이때, 예비 요청을 Preflight라고 부르며, 이 예비 요청에는 HTTP 메소드 중 OPTIONS 메소드가 사용된다.
  - 예비 요청의 역할은 본 요청을 보내기 전 브라우저 스스로 이 요청을 보내는 것이 안전한지 확인



- fetch()를 통해 브라우저에게 리소스를 받아오라는 명령을 내리면 브라우저는 서버에게 예비 요청을 먼저 보내고, 서버는 이 예비 요청에 대한 응답으로 현재 자신이 허용하고 있는 것과 그렇지 않은 것을 응답 헤더에 담아서 브라우저에게 다시 보내 준다.
- 브라우저는 자신이 보낸 예비 요청과 서버가 응답에 담아준 허용 정책을 비교한 후, 이 요청을 보내는 것이 안전하다고 판단 되면 본 요청을 보낸다.

```
1 OPTIONS https://evanmoon.tistory.com/rss
2
3 Accept: */*
4 Accept-Encoding: gzip, deflate, br
5 Accept-Language: en-US,en;q=0.9,ko;q=0.8,ja;q=0.7,la;q=0.6
6 Access-Control-Request-Headers: content-type
7 Access-Control-Request-Method: GET
8 Connection: keep-alive
9 Host: evanmoon.tistory.com
10 Origin: https://evan-moon.github.io
11 Referer: https://evan-moon.github.io/2020/05/21/about-cors/
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: cross-site
```

- 예비 요청의 예시
- 서버에게 내가 'content-type' 헤더를 사용할 거고, 'GET'을 쓸 것을 미리 알려줌

```
1 OPTIONS https://evanmoon.tistory.com/rss 200 OK
2
3 Access-Control-Allow-Origin: https://evanmoon.tistory.com
4 Content-Encoding: gzip
5 Content-Length: 699
6 Content-Type: text/xml; charset=utf-8
7 Date: Sun, 24 May 2020 11:52:33 GMT
8 P3P: CP='ALL DSP COR MON LAW OUR LEG DEL '
9 Server: Apache
10 Vary: Accept-Encoding
11 X-UA-Compatible: IE=Edge
```

- 예비 요청에 대한 응답의 예시
- Access-Control-Allow-Origin을 통해 이 리소스에 접근 가능한 출처를 알려준다.
- 이러한 경우 CORS 정책을 위반한 것이다.
- CORS 정책 위반 여부를 판단하는 시점은 예비 요청에 대한 응답을 받은 이후이다. 요청의 성공 여부와 상관없다.
- 200 OK가 나오든 나오지 않든 유효한 값이 있을 경우 정책 위반이 아니다.

• Simple Request

- 예비 요청을 보내지 않고 바로 서버에게 본 요청을 보낸 후, 서버가 이에 대한 응답의 헤더에 Access-Control-Allow-Origin 과 같은 값을 보내주면 브라우저가 CORS 정책 위반 여부를 검사한다. → 특별한 조건을 충족해야 사용할 수 있음


1. 요청의 메소드는 GET, HEAD, POST 중 하나여야 한다.
2. Accept, Accept-Language, Content-Language, Content-Type, DPR, Dnlink, Save-Data, Viewport-Width, Width 를 제외한 헤더를 사용하면 안된다.
3. 만약 Content-Type 를 사용하는 경우에는 application/x-www-form-urlencoded, multipart/form-data, text/plain 만 허용된다.

• Credentialed Request

- 인증된 요청을 사용
- 요청에 인증과 관련된 정보를 담게 해주는 옵션이 credentials 옵션

옵션 값	설명
same-origin (기본값)	같은 출처 간 요청에만 인증 정보를 담을 수 있다
include	모든 요청에 인증 정보를 담을 수 있다
omit	모든 요청에 인증 정보를 담지 않는다

### CORS 에러 해결

- Access-Control-Allow-Origin 세팅
  - 와일드 카드 를 사용하면 모든 출처에서 오는 요청을 다 받는다는 의미로 편할 수 있지만 보안에 취약하다.
  - 적절한 출처를 명시하는 것이 좋다.
- 프론트 개발자의 경우 webpack-dev-server 라이브러리가 제공하는 프록시 기능을 사용해 CORS 정책을 우회할 수 있다.