**reference :** <https://brownbears.tistory.com/336>

**CORS :** Cross Origin Resource Sharing

한 출처에서 실행중한 웹 어플리케이션이 다른 출처의 자원에 접근할 수 있는 구너한을 부여하도록

브라우저에 알려주는 체제

**탄생 배경**

: HTTP 요청은 기본적으로 Cross Site HTTP Requests 가 가능함. 다시 말하면 img 태그로 다른 도메인의 이미지 파일을 가져오거나 link 태그로 다른 도메인의 CSS를 가져오거나, script 태그로 다른 도메인의 JavsScript 라이브러리를 가져오는 것이 모두 가능함

하지만 script 로 둘러싸여있는 스크립트에서 생성된 Cross Site HTTP Request는 Same Origin Policy를 적용받기 때문에 Cross Site HTTP Requests가 불가능함. -> 즉 프로토콜, 호스트명, 포트가 같아야만 요청이 가능함

AJAX가 널리 사용되면서 script로 둘러싸여있는 스크립트에서 생성되는 XMLHttpRequest에 대해서도 Cross Site HTTP Request가 가능해야한다는 요구가 늘어나자 W3C에서 CORS라는 이름의 권고안이 나오게 됨.

**CORS 요청의 종류**

**-**Simple/Preflight, Credential/Non-Credential 의 조합으로 4가지가 존재. 브라우저가 요청 내용을

분석하여 4가지 방식 중 해당하는 방식으로 서버에 요청을 날리므로, 프로그래머가 목적에 맞는 방식을

택해야함

Simple Request : 아래의 3가지 조건을 만족하면 됨. 서버에 1번 요청하고, 서버도 1번 회신하면 종료

1. GET, HEAD, POST 중의 한 방식을 사용함
2. POST 방식일 경우 Content-type이 아래 셋 중의 하나여야함
   1. application/x-www-form-urlencoded
   2. multipart/form-data
   3. text/plain
3. 커스텀 헤더를 전송하지 말아야 함

Preflight Request : Simple 이 아니면, 이 방식으로 요청함. GET, HEAD, POST 외 다른 방식으로도

요청을 보낼 수 있고, application/xml 처럼 다른 Content-type으로 요청을 보낼 수 있고,

커스텀헤더도 사용가능하다.

예비요청과 본 요청으로 구성됨. 먼저 서버에 예비요청을 하고, 서버가 응답하면 그때 본요청을 보내고, 서버도 본요청에 응답함

Request with Credential : HTTP Cookie와 HTTP Authentication 정보를 인식할 수 있게 해주는 요청

요청 시 xhr.withcredentials = true 를 지정해서 Credential 요청을 보낼 수 있고,

서버는 Response Header에 반드시 Access-Control-Allow-Credentials: true를 포함해야하며,

Access-Control-Allow-Origin 헤더의 값에는 \*가 오면 안되고 <http://foo.origin> 과 같은 구체적인 도메인이 와야함.

Non Credential : 디폴트 요청. xhr.withCredentials = true 를 지정하지 않으면 Non-Credential임.