Ajax 보충

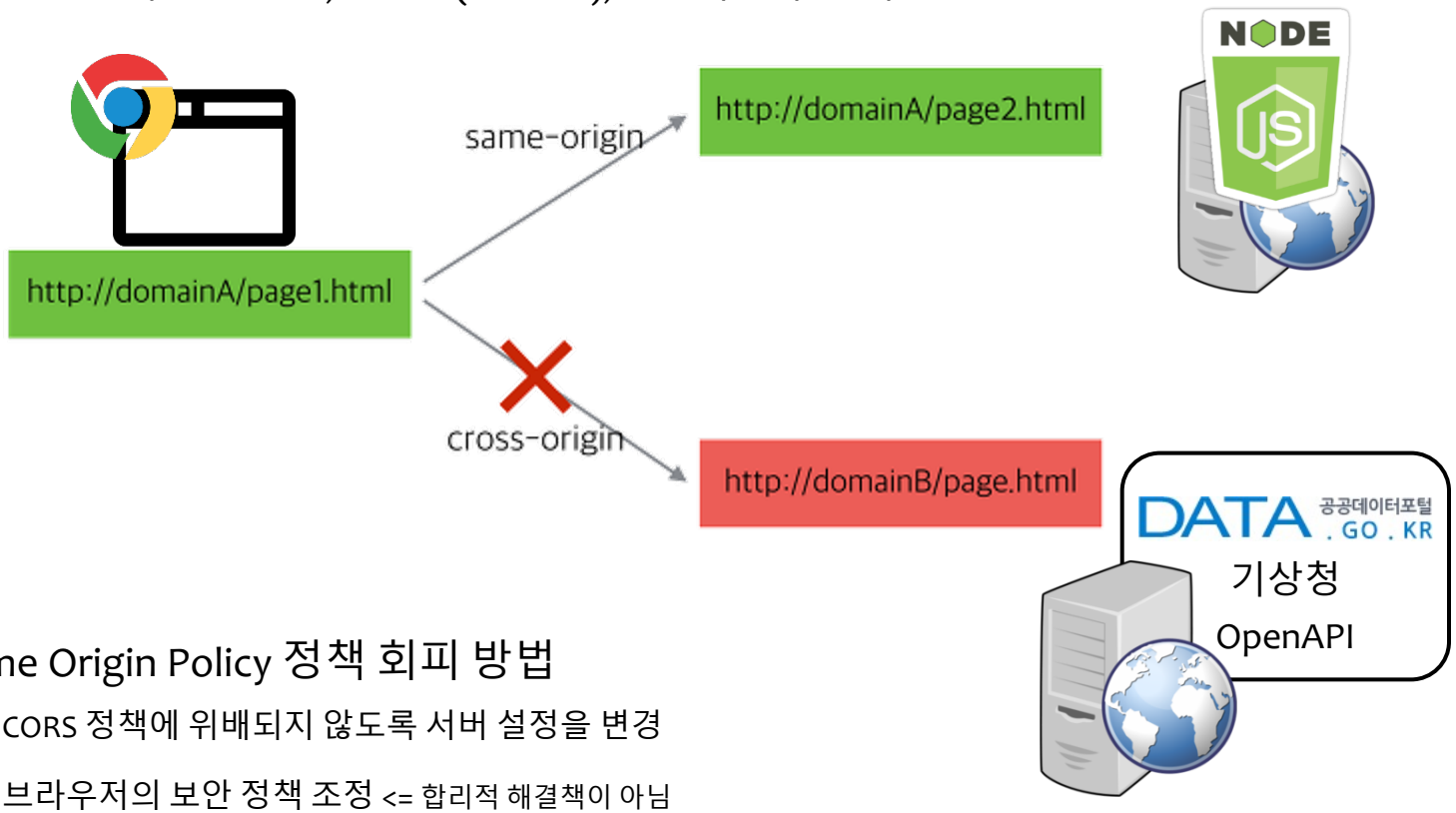
SOP, CORS, fetch API

부산대학교 정보·의생명 공학대학
정보컴퓨터공학부

SOP(Same Origin Policy)

❖ 웹 어플리케이션에서 보안을 위해 사용하는 정책 중 하나

- 자바스크립트에서 다른 웹페이지에 요청을 보낼 때(XMLHttpRequest)는 같은 출처(Same Origin)의 페이지에만 접근이 가능
  - 같은 출처: 프로토콜, 호스트(Domain), 포트가 모두 같다



- Same Origin Policy 정책 회피 방법
  - CORS 정책에 위배되지 않도록 서버 설정을 변경
  - 브라우저의 보안 정책 조정 <= 합리적 해결책이 아님
    - Chrome일 경우 Allow-Control-Allow-Origin 플러그인 설치

CORS(Cross-Origin Resource Sharing)

❖ SOP 정책의 예외 조항

- 자바스크립트에서 다른 웹페이지에 요청을 보낼 때(XMLHttpRequest) 다른 출처(Cross-Origin)라도 CORS 정책을 지키면 접근을 허용
- 요청 보낼 때 HTTP Header의 Origin 정보와 서버의 응답 HTTP Header에 있는 Access-Control-Allow-Origin 정보가 같으면 통신을 허용
  - Origin 정보는 요청을 보낼 때 자동으로 입력됨.

```
OPTIONS https://evanmoon.tistory.com/rss

Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ko;q=0.8,ja;q=0.7,la;q=0.6
Access-Control-Request-Headers: content-type
Access-Control-Request-Method: GET
Connection: keep-alive
Host: evanmoon.tistory.com
Origin: https://evan-moon.github.io
Referer: https://evan-moon.github.io/2020/05/21/about-cors/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: cross-site
```

요청 Header

```
OPTIONS https://evanmoon.tistory.com/rss 200 OK

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

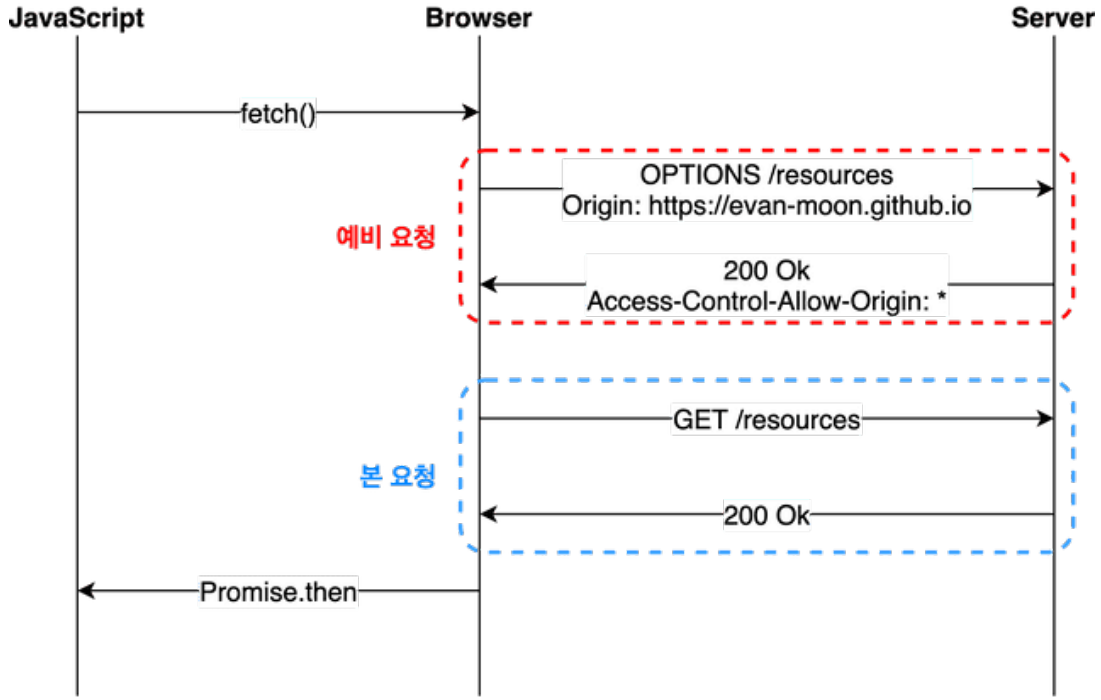
응답 Header

CORS 정책 위반!

Preflight Request

❖ 웹 어플리케이션 개발할 때 가장 일반적인 요청 시나리오

- 브라우저가 요청을 보낼 때 예비 요청, 본 요청 2번의 요청으로 나누어서 서버로 전송하는 시나리오
- 이때 예비 요청을 Preflight라고 함
- 예비 요청에서는 페이지 전체를 요청하는 게 아닌 요청을 보내도 되는 안전한 출처인지만 확인
  - OPTION 메소드 사용
  - SOP, CORS 정책 위반 여부 확인



CORS 문제 해결법

❖ 서버에서 Access-Control-Allow-Origin 세팅

- Apache나 Nginx의 서버 설정 파일 수정
  - Apache 설정법: https://ubiq.co/tech-blog/set-access-control-allow-origin-cors-headers-apache/

- 소스코드에서 HTTP 헤더값 설정하는 API 사용
  - PHP: `header("Access-Control-Allow-Origin: *");`
  - `*`는 모든 Origin 허용. 개발용으로만 사용 권장

Fetch API

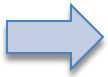
❖ ES6(ECMAScript 6) 표준에서 새로 도입된 API

- JS 표준이기때문에 외부 라이브러리 import가 필요없음
- XMLHttpRequest 방식보다 훨씬 간결한 문법

```
let xhr = new XMLHttpRequest();
xhr.open('get', 'http://localhost:3000/messages');

//요청의 상태 변화를 추적한다.
xhr.onreadystatechange = function() {
 if(xhr.readyState !== 4) return;
  //readyState 4: 완료

  if(xhr.status === 200) {
    //status 200: 성공
    console.log(xhr.responseText); //서버로부터 온 응답
  } else {
        console.log('에러: ' + xhr.status);  //요청 도중 에러 발생
  }
}
xhr.send(); //요청 전송
```



```
        //요청할 서버 url을 입력
fetch('http://localhost:3000/messages')
.then(function(response) {
        return response.json();
})
.then(function(json) {
        //json 형태로 전달받은 서버로부터의 응답
})
```

- 리턴 객체인 respons는 promise 객체 타입(비동기 호출 가능 객체)
  - Reponse의 값을 보는 5가지 메소드: text(), arrayBuffer(), blob(), json(), formData()
- 기본 문법: fetch(URL, [options])
  - options에 아무것도 안 넘기면 GET Method로 요청

Fetch API

❖ POST로 요청 보내는 방법

```
let input = document.getElementById('input');
let target = document.getElementById('title');
fetch('/title', {
  method: 'post',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({ 'title': input.value }),
})
  .then(res => res.json())
  .then(json => target.innerHTML = json.title)
  .catch(error => console.error('Error: ', error));
```

```
let formData = new FormData();
formData.append('name', 'choi');
fetch('/title', {
   method: 'post',
   body: formData,
})
.then(res => res.json())
.then(json => target.innerHTML = json.title)
.catch(error => console.error('Error: ', error));
```

❖ PHP에서 fetch API로 보낸 JSON 데이터를 가져오는 방법

```
$post_data = json_decode(file_get_contents('php://input'));
if($post_data->title == '')
    die('false');
```

❖ PHP에서 JSON 형식의 응답을 보내는 방법

```
header("Content-Type: application/json");
echo(json_encode(array("name" => $name, "email" => $email)));
```