

CORS详细解析

Henry-baobao

2021年8月13日



目录

1、CORS是什么

2、怎么解决CORS

3、CORS Header详细解析

为什么报CORS错误

安全性



前言

✖ Access to fetch at 'http://localhost:5001/list/menu' from origin 'http://localhost:3000' has been blocked by CORS policy: localhost/:1
No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

✖ ▶GET http://localhost:5001/list/menu net::ERR_FAILED

App.js:8

大部分情形下，CORS 都不是前端的问题，**纯前端是解决不了的**

1、什么是跨来源？

2、为什么不能跨来源请求API？

前言

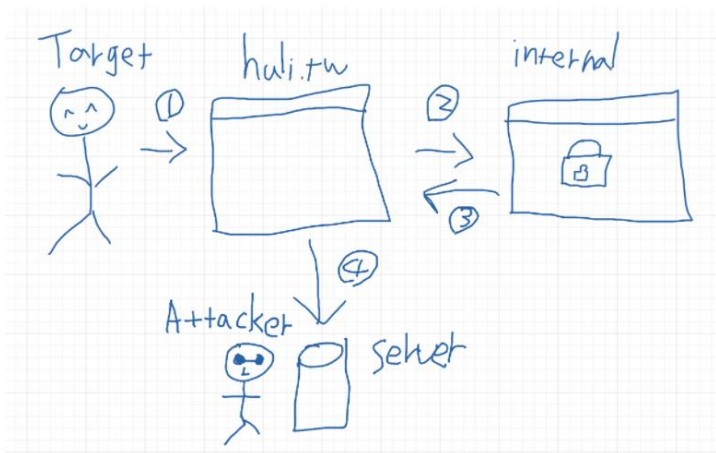
CORS（Cross-Origin Resource Sharing，跨域资源共享）是一个系统，它由一系列传输的[HTTP头](#)组成，这些HTTP头决定[浏览器](#)是否阻止[前端 JavaScript 代码](#)获取跨域请求的[响应](#)。

如果两个 URL 的 protocol、port (如果有指定的话)和 host 都相同的话，则这两个 URL 是同源。

1. `https://huli.tw` 跟 `https://huli.tw/api` 同源，因为 scheme + host + port 都一样（`/api` 是 path 的部分，不是 host）
2. `https://huli.tw` 跟 `http://huli.tw` 不同源，因为 scheme 不一样
3. `http://huli.tw` 跟 `http://huli.tw:3000` 不同源，因为 port 不一样
4. `https://api.huli.tw` 跟 `https://data.huli.tw` 不同源，因为 host 不一样
5. `https://huli.tw` 跟 `https://api.huli.tw` 不同源，因为 host 不一样

前言

- 为什么不允许跨来源请求？
- 确切的说，是不允许ajax（fetch、XMLHttpRequest）获取跨来源的资源
- 跨来源获取资源：img、script、link，有问题没？？？



如果跨来源的AJAX请求没有限制，网站的创建者可透过使用者的浏览器，拿到【任意网站】的内容，包含可能有各种敏感资源的网站。

前言

第一題

小明正負責寫一個專案，網址是：<https://best-landing-page.tw>。這網站會需要用到公司其他網站的某個檔案，裡面是一些使用者資料，網址是：<https://lidemy.com/users.json>。小明直接點開這個網址，發現用瀏覽器可以看到檔案的內容，於是就說：

既然我用瀏覽器可以看得到內容，就表示瀏覽器打得開，那用 AJAX 的時候也一定可以拿得到資料！我們來用 AJAX 拿資料吧！

請問小明的說法是正確的嗎？如果錯誤，請指出錯誤的地方。

前言

第二題

小明正在做的專案需要串接 API，而公司內部有一個 API 是拿來刪除文章的，只要把文章 id 用 POST 以 `application/x-www-form-urlencoded` 的 content type 帶過去即可刪除。

舉例來說：`POST https://lidy.com/deletePost` 並帶上 `id=13`，就會刪除 id 是 13 的文章（後端沒有做任何權限檢查）。

公司前後端的網域是不同的，而且後端並沒有加上 CORS 的 header，因此小明認為前端用 AJAX 會受到同源政策的限制，request 根本發不出去。

而實際上呼叫以後，果然 console 也出現：「request has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource」的錯誤。

所以小明認為前端沒辦法利用 AJAX 呼叫這個 API 刪除文章，文章是刪不掉的。

請問小明的說法是正確的嗎？如果錯誤，請指出錯誤的地方。

前言



如何解决CORS

- 关闭浏览器安全设置
- 把fetch mode设置成no-cors
- 不通过AJAX获取资源



治标不治本

- 设置CORS的HTTP Header




解决方案



关闭浏览器设置

- 在自己电脑的浏览器不报CORS错误，可拿到请求response
- 但是其他访问者电脑如果不设置，还会报错
- 没法还原生产环境，保证项目交付效果
- Chrome关闭配置（**禁止**）：<https://alfilatov.com/posts/run-chrome-without-cors/>

no-cors mode

no-cors request response: `▼ Response {type: "opaque", url: "", redirected: false, status: 0, ok: false, ...}`  [App.js:41](#)

```
body: null
bodyUsed: false
▶ headers: Headers {}
ok: false
redirected: false
status: 0
statusText: ""
type: "opaque"
url: ""
▶ [[Prototype]]: Response
```

- 当设置mode: no-cors后，浏览器不报跨域错误
- 但是，请求一定拿不到response，不管后端是否设置CORS header

No AJAX->JSONP

- 利用script标签请求资源不受CORS限制的特性
- JSONP: JSON with Padding (padding即callback的名称)
- 在CORS规范未成熟时经常使用
- 缺点: 只能使用GET方法; 需要后端的配合使用。

设置CORS Header

- 后端在response添加Access-Control-Allow-Origin
- *代表所有均能访问;

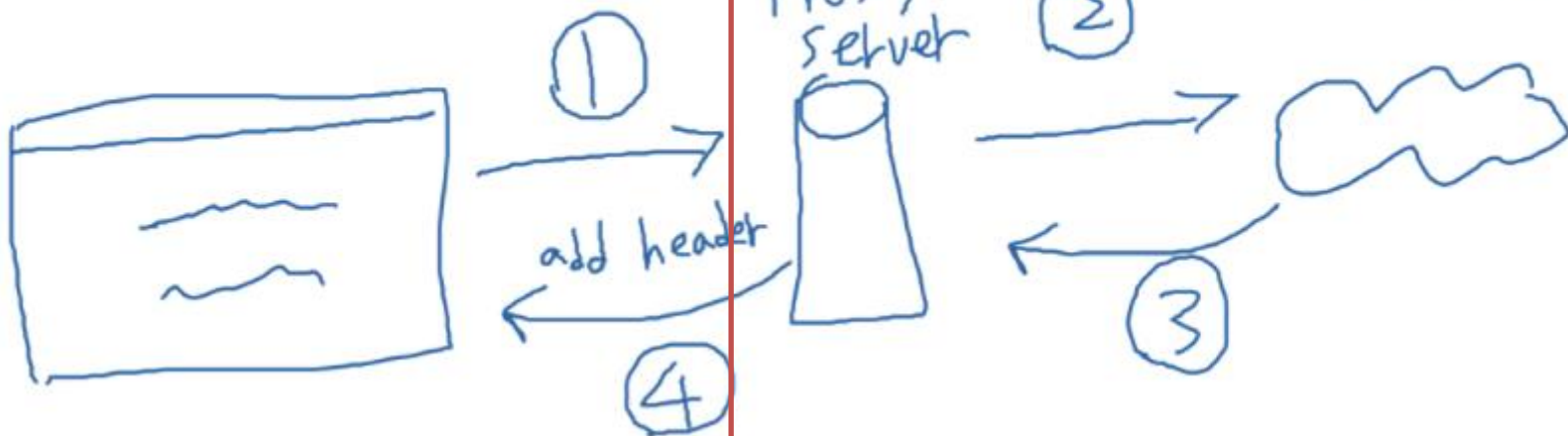
```
1 app.get('/', function (req, res) {
2   res.set('Access-Control-Allow-Origin', 'http://localhost:8081');
3   res.end('hello world');
4 });
```

Copy

- 问题：如果后端不受控制，怎么办？？？

Proxy Server

- 使用受自己控制的Server



Created with WhiteboardFox.com

CORS 详解

- 非简单请求如何设置
- 如何在跨来源请求中传送cookie



简单的CORS问题

- mode: 'no-cors', 没办法拿到请求结果, 解决不了cors问题
- 碰到CORS问题, 首先确认后端是否添加header: Access-Control-Allow-Origin; 其次确认返回的header是否包含请求origin, Access-Control-Allow-Origin可以使* (wildcard, 通配符), 也可以是具体的地址, 如 <http://localhost:3000>.

复杂的CORS问题

- POST方法, Content-Type: application/x-www-form-urlencoded

✖ Access to fetch at 'http://localhost:5001/list/menu' from origin 'http://localhost:3000' has been blocked by CORS policy: localhost/:1
No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

✖ ▶ GET http://localhost:5001/list/menu net::ERR_FAILED

App.js:8

- 请求设置header: Access-Control-Allow-Origin

- POST方法, Content-Type: application/json

✖ Access to fetch at 'http://localhost:5003/menu/complex' 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. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled. localhost/:1

✖ Access to fetch at 'http://localhost:5003/menu/complex' from origin 'http://localhost:3000' has been blocked by CORS policy: Request header field content-type is not allowed by Access-Control-Allow-Headers in preflight response. localhost/:1

✖ ▶ POST http://localhost:5003/menu/complex net::ERR_FAILED

App.js:85

- 请求设置header: Access-Control-Allow-Origin

- 预检请求设置header: Access-Control-Allow-Origin、Access-Control-Allow-Headers

复杂的CORS问题

简单请求

请求方法为GET、POST和HEAD;

不带自定义header;

Content-Type为: application/x-www-form-urlencoded、multipart/form-data、text/plain.

对于简单请求, 只用设置response header:
[Access-Control-Allow-Origin](#)

VS

非简单请求

非简单请求会发出preflight request (预检请求), 方法为**OPTIONS**;

预检请求会在request header中加上: [Access-Control-Request-Headers](#)、[Access-Control-Request-Method](#)

对于非简单请求, 需要设置:

- 请求设置response header: [Access-Control-Allow-Origin](#)
- 预检请求设置response header: [Access-Control-Allow-Origin](#)、[Access-Control-Allow-Headers](#)、[Access-Control-Allow-Methods](#)

CORS携带cookie

- 跨来源请求，预设不带cookie
 - 在请求中，配置credentials: 'include'
 - 后端设置响应头：Access-Control-Allow-Origin（不能为*，要明确指定来源）、Access-Control-Allow-Credentials
 - 预检请求设置响应头：Access-Control-Allow-Origin（不能为*，要明确指定来源）、Access-Control-Allow-Credentials

存取自定义header

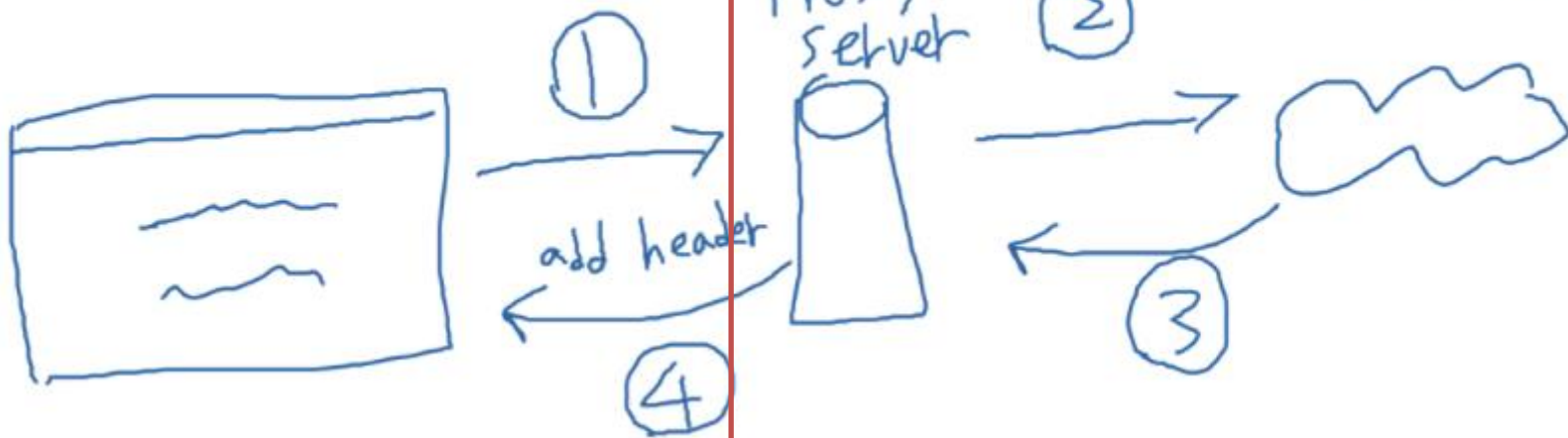
- 前端如何获取后端自定义header
 - 后端设置自定义header: [X-Service-Version](#)
 - 后端请求设置响应header: [Access-Control-Expose-Headers](#)
 - 前端通过response的headers取数据

缓存预检请求

- 每次复杂请求都要提前进行预检请求，就算同一个使用者同一个请求预检过了，每次还是要进行预检，需求：同一浏览器重复发送request，不用预检。
- 后端请求设置响应header: [Access-Control-Max-Age](#)

延伸Proxy Server

- 使用受自己控制的Server



Created with WhiteboardFox.com

延伸Proxy Server



- 1、自己写后台，配置CORS headers，将请求转发至目标服务对应请求；
- 2、NGINX 代理，添加CORS headers。
- 3、[cors-anywhere](#)、[webpack-dev-server](#)（[http-proxy-middleware](#)）等，本质都是通过Proxy Server添加CORS headers。

参考内容:

- Huili个人博客《CORS完全手册系列》：<https://blog.huli.tw/2021/02/19/cors-guide-1>
- CORS MDN: <https://developer.mozilla.org/zh-CN/docs/Glossary/CORS>、https://developer.mozilla.org/zh-CN/docs/Web/Security/Same-origin_policy
- Fetch MDN: https://developer.mozilla.org/zh-CN/docs/Web/API/Fetch_API/Using_Fetch
- Github repository: <https://github.com/Henry-baobao/explain-cors-in-detail>

