

关于跨域， 以及几种实现方式

—— 刘怡瑄 ——

— 目录 —

1-基础概念

2-跨域的几种方式

3-跨域资源共享

4-Jsonp跨域

5-代理跨域

6-postMessage跨域



分享目的

希望大家通过本次分享，能有以下收获

- 1、了解域的概念和浏览器同源策略
- 2、了解JSONP本质
- 3、了解CORS原理和实现
- 4、了解代理如何解决跨域，正向代理和反向代理



01

基础概念



基础概念

域的概念：

域(Domain)是Windows网络中独立运行的单位，域之间相互访问需要建立信任关系(即Trust Relation)。信任关系是连接在域与域之间的桥梁。当一个域与其他域建立了信任关系后，2个域之间不但可以按需要相互进行管理，还可以跨网分配文件和打印机等设备资源，使不同的域之间实现网络资源的共享与管理。

--百度百科



基础概念

一个域名的组成



当协议、子域名、主域名、端口号中任意一个不相同，都算作不同域。不同域之间相互请求资源，就算作“跨域”。



基础概念

跨域常见场景

URL	说明	是否允许通信
http://sunlands.com/a.js http://sunlands.com/b.js http://sunlands.com/c/b.js	同一域名，不同文件或路径	允许
http://sunlands.com:8080/a.js http://sunlands.com/b.js	同一域名，不同端口	不允许
http://sunlands.com/a.js https://sunlands.com/b.js	同一域名，不同协议	不允许
http://sunlands.com/a.js http://42.62.10.155/a.js	域名和域名对应相同ip	不允许
http://www.sunlands.com/a.js http://m.sunlands.com/a.js http://sunlands.com/a.js	主域相同，子域不同	不允许
http://sunlands.com/a.js http://baidu.com/a.js	不同域名	不允许

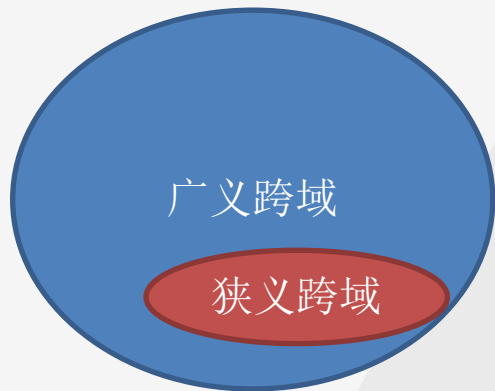


基础概念

跨域的广义与狭义定义：

跨域是指一个域下的文档或脚本试图去请求另一个域下的资源，这里跨域是广义的。

我们通常所说的跨域是狭义的，是由**浏览器同源策略**限制的一类请求场景。



跨域分类：

1. 资源跳转： A链接、重定向、表单提交
2. 资源嵌入： <link>、<script>、、<frame>等dom标签，还有样式background:url()、@font-face()等文件外链
3. 脚本请求：js发起的ajax请求
4. dom和js对象的跨域操作
5. Cookie、LocalStorage 和 IndexDB 的读取



基础概念

****特别说明****

- 1、跨域并不是请求发不出去，请求能发出去，服务端能收到请求并正常返回结果，只是结果被浏览器拦截了。
- 2、如果是协议和端口造成的跨域问题“前台”是无能为力的。
- 3、在跨域问题上，域仅仅是通过“URL的首部”来识别而不会根据域名对应的IP地址是否相同来判断。“URL的首部”可以理解为“协议, 域名和端口必须匹配”。



基础概念

简单请求与非简单请求:

只要满足下面条件就是简单请求

- 请求方式为HEAD、POST 或者 GET
- http头信息不超出以下字段: Accept、Accept-Language、Content-Language、Last-Event-ID、Content-Type(限于三个值: application/x-www-form-urlencoded、multipart/form-data、text/plain)

为什么要分为简单请求和非简单请求, 因为浏览器对这两种请求方式的处理方式是不同的。



02 跨域的几种方式



跨域的几种方式

#	方式
1	跨域资源共享（CORS）
2	proxy代理
3	通过jsonp跨域
4	WebSocket协议跨域
5	postMessage跨域
6	document.domain + iframe
7	location.hash + iframe
8	window.name + iframe



03

跨域资源共享



跨域资源共享-CORS

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。

CORS需要浏览器和服务端同时支持。

- 1、IE浏览器不能低于IE10。IE8+：IE8/9需要使用XDomainRequest对象来支持CORS。
- 2、服务器需要配置response header

跨域资源共享-CORS

Cross-Origin Resource Sharing - LS

Usage % of all users
Global 92.85% + 0.89% = 93.74%

Method of performing XMLHttpRequests across domains

Current aligned	Usage relative	Date relative	Apply filters	Show all	?										
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Blackberry Browser	Opera Mobile *	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsu Interr
6-7		2-3		3.1-3.2											
² 8-9		3.5-60	¹ 4-12	¹ 3 4-5.1	10-11.5	¹ 3 3.2-5.1		¹ 2.1-4.3							
¹ 10	12-17	⁴ 61-63	13-70	³ 6-11.1	12.1-56	³ 6-11.4		4.4-4.4.4	¹ 7	12-12.1			¹ 10		4-6
11	18	⁴ 64	71	³ 12	57	³ 12.1	all	67	10	46	70	63	11	11.8	7.2
		⁴ 65-66	72-74	³ TP											

Notes

Known issues (4)

Resources (6)

Feedback

- ¹ Does not support CORS for images in `<canvas>`
- ² Supported somewhat in IE8 and IE9 using the XDomainRequest object (but has *limitations*)
- ³ Does not support CORS for `<video>` in `<canvas>`: https://bugs.webkit.org/show_bug.cgi?id=135379
- ⁴ Does not support CORS for resources which redirect: https://bugzilla.mozilla.org/show_bug.cgi?id=1346749

跨域资源共享-CORS

Fetch - LS

Usage % of all users
Global 87.79% + 0.05% = 87.84%

A modern replacement for XMLHttpRequest.

Current aligned

Usage relative

Date relative

Apply filters

Show all

?

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Blackberry Browser	Opera Mobile *	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsi Interr
		2-33	4-39		10-26										
		^{1 4} 34-38	² 40		² 27										
	12-13	⁴ 39	^{2 3} 41	3.1-10	^{2 3} 28	3.2-10.2									
6-10	14-17	40-63	42-70	10.1-11.1	29-56	10.3-11.4		2.1-4.4.4	7	12-12.1			10		4-6
11	18	64	71	12	57	12.1	all	67	10	46	70	63	11	11.8	7.2
		65-66	72-74	TP											

Notes

Sub-features (1)

Known issues (0)

Resources (7)

Feedback

¹ Partial support can be enabled in Firefox with the `dom.fetch.enabled` flag.

² Only available in Chrome and Opera within ServiceWorkers.

³ Available in Chrome and Opera within Window and Workers by enabling the "Experimental Web Platform Features" flag in `chrome://flags`

⁴ Firefox <40 is not completely conforming to the specs and does not respect the `<base>` tag for relative URIs in fetch requests. https://bugzilla.mozilla.org/show_bug.cgi?id=1161625

XMLHttpRequest advanced features - LS

Usage

% of all users

Global

89.73% + 3.66% = 93.39%

Adds more functionality to XHR (aka AJAX) requests like file uploads, transfer progress information and the ability to send form data. Previously known as **XMLHttpRequest Level 2**, these features now appear simply in the XMLHttpRequest spec.

Current aligned	Usage relative	Date relative	Apply filters	Show all	?										
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Blackberry Browser	Opera Mobile *	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsu Interr
		2-3													
		1 2 3 3.5-5	4-6	3.1-4	10-11.5	3.2-4.3		2.1-2.3							
		1 2 6-9	1 2 7-28	1 2 5-6	12.1	5-6.1		1 2 3 3-4.3							
6-9		2 10-11	1 29-30	1 6.1-7	1 15-17	1 7.1		1 2 4.4							
1 1 10	12-17	12-63	31-70	7.1-11.1	18-56	8-11.4		4.4.4	1 2 7	12-12.1			1 10		4-6
1 1 11	18	64	71	12	57	12.1	all	67	1 1 10	46	70	63	1 1 11	11.8	7.2
		65-66	72-74	TP											

Notes

Known issues (5)

Resources (4)

Feedback



跨域资源共享-CORS

```
/*
 * 该字段是必须的。它的值要么是请求时origin字段的值，要么是一个*，表示接受任意域名的请求
 * */
res.header("Access-Control-Allow-Origin", "*");

/*
 * 可选。允许带cookie;
 * Access-Control-Allow-Origin指定域请求时，该参数可用;
 * 前端开发者必须在AJAX请求中打开withCredentials属性。
 * */
res.header("Access-Control-Allow-Credentials", "true");

/*
 * 可选。CORS请求时，XMLHttpRequest对象的getResponseHeader()方法只能拿到6个基本字段：
 * Cache-Control、Content-Language、Content-Type、Expires、Last-Modified、Pragma。
 * 如果想拿到其他字段，就必须在Access-Control-Expose-Headers里面指定。
 * */
res.header("Access-Control-Expose-Headers", "token");

/*
 * 非简单请求的CORS请求，会在正式通信之前，增加一次HTTP查询请求，称为"预检"请求 (preflight)
 * 这种情况下除了设置origin，还需要设置Access-Control-Request-Method和Access-Control-Request-Headers
 * */
// 该字段必需，它的值是逗号分隔的一个字符串，表明服务器支持的所有跨域请求的方法。
res.header("Access-Control-Allow-Methods", "*");
// 如果浏览器请求包括Access-Control-Request-Headers字段，则Access-Control-Allow-Headers字段是必需的。
// 它也是一个逗号分隔的字符串，表明服务器支持的所有头信息字段
res.header("Access-Control-Allow-Headers", "Content-Type");

/*
 * 该字段可选，用来指定本次预检请求的有效期，在此期间，不用发出另一条预检请求。
 * */
res.header("Access-Control-Max-Age", "1728000");
```



04

Jsonp跨域



Jsonp跨域

1. JSONP原理

利用 `<script>` 元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 数据。JSONP 请求一定需要对方的服务器做支持才可以。

2. JSONP和AJAX对比

JSONP和AJAX相同，都是客户端向服务器端发送请求，从服务器端获取数据的方式。但AJAX属于同源策略，JSONP属于非同源策略（跨域请求）。

3. JSONP优缺点

JSONP优点是兼容性好。缺点是仅支持**get**方法具有局限性。



JSONP的实现流程

- 声明一个回调函数，其函数名(如fn)当做参数值，要传递给跨域请求数据的服务器，函数形参为要获取目标数据(服务器返回的data)。
- 创建一个 `<script>` 标签，把那个跨域的API数据接口地址，赋值给script的src，还要在这个地址中向服务器传递该函数名（可以通过问号传参 `:?callback=fn`）。
服务器接收到请求后，需要进行特殊的处理：把传递进来的函数名和它需要给你的数据拼接成一个字符串，例如：传递进去的函数名是fn，它准备好的数据是 `fn([{"name":"jianshu"}])`。
- 最后服务器把准备的数据通过HTTP协议返回给客户端，客户端再调用执行之前声明的回调函数（fn），对返回的数据进行操作。



Jsonp跨域

```
<script type="text/javascript">  
  function fn(data) {  
    alert(data.msg);  
  }  
</script>
```

```
<script type="text/javascript" src="http://crossdomain.com/jsonServer?callback=fn"></script>
```

```
app.route('/jsonp')  
  .get(function (req, res) {  
    console.log('有客户端连接: get - /user ');  
    const str = JSON.stringify({code: '200', msg: '成功'});  
    res.send(`fn(${str})`); // 必须返回 fn({"code":200,"msg":"成功"}) 格式  
  });
```



Jsonp跨域

动态jsonp实现

```
function getJsonp() {  
  
    const script = document.createElement('script');  
  
    script.src = 'http://127.0.0.1:9000/jsonp?callback=fn';  
  
    document.body.appendChild(script);  
  
    function fn(data) {  
        console.log(data);  
        alert(data.msg);  
    }  
}
```



Jsonp跨域

jQuery的jsonp形式

```
$.ajax({  
  url:"http://crossdomain.com/jsonServerResponse",  
  dataType:"jsonp",  
  type:"get", //可以省略  
  jsonpCallback:"fn", //->自定义传递给服务器的函数名，而不是使用jQuery自动生成的，可省略  
  jsonp:"jsonp", //->把传递函数名的那个形参callback变为jsonp，可省略  
  success:function (data){  
    console.log(data);  
  }  
});
```




05

代理跨域



代理跨域

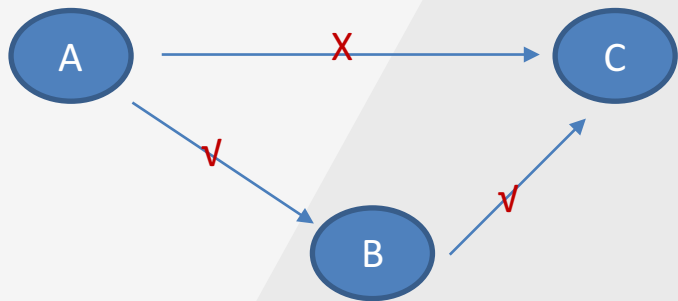
Webpack配置代理

```
"proxy": {  
  "/api": {  
    "target": "http://localhost:9000/",  
    "changeOrigin": true  
  }  
}
```

代理跨域

正向代理：

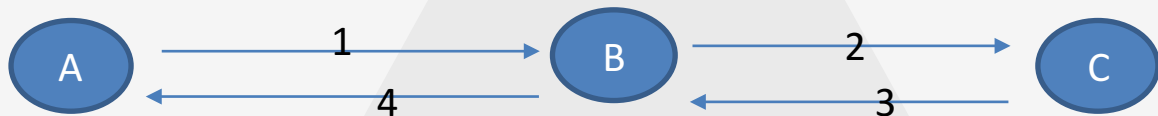
正向代理（**forward proxy**），一个位于客户端和原始服务器之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并制定目标（原始服务器），然后代理向原始服务器转发请求并将获得的内容返回给客户端，客户端才能使用正向代理。我们平时说的代理就是指正向代理。



代理跨域

反向代理：


反向代理（Reverse Proxy），以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求的客户端，此时代理服务器对外表现为一个反向代理服务器。





06

postMessage跨域



postMessage跨域

语法:

```
otherWindow.postMessage(message, targetOrigin, [transfer]);
```

otherWindow:

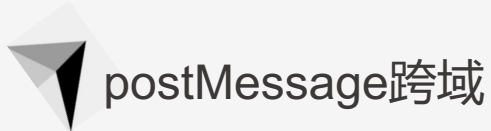
其他窗口的一个引用，比如iframe的contentWindow属性、执行[window.open](#)返回的窗口对象、或者是命名过或数值索引的[window.frames](#)。

message:

将要发送到其他 window 的数据。它将会被[结构化克隆算法](#)序列化。这意味着你可以不受什么限制的将数据对象安全的传送给目标窗口而无需自己序列化

targetOrigin:

通过窗口的origin属性来指定哪些窗口能接收到消息事件，其值可以是字符串"*"（表示无限制）或者一个URI。



postMessage跨域

message事件监听:

```
window.addEventListener("message", receiveMessage, false);
```

data:

其他窗口的一个引用，比如iframe的contentWindow属性、执行[window.open](#)返回的窗口对象、或者是命名过或数值索引的[window.frames](#)。

origin:

调用 postMessage 时消息发送方窗口的 [origin](#) .

source:

对发送消息的[窗口](#)对象的引用; 您可以使用此来在具有不同origin的两个窗口之间建立双向通信。



谢谢观看

—— 刘怡瑄 ——