# **前端跨域方法总结 https://juejin.im/post/5bcc297c51882535074a695e**

## 1.为什么需要跨域

浏览器出于安全的考虑，引入了同源策略。这种策略会对我们页面上执行的js访问资源的时候进行限制，比如我们不能直接通过js访问不同源之下的页面DOM结构，同时在对不同源发送请求时也无法获取到服务器响应内容（服务器会正常处理请求并返回响应内容，但是返回的内容被浏览器拦截掉了）。

同源策略下能做什么：

1. 通过js重定向我们的页面（修改location.href）
2. 表单提交
3. 通过嵌入一些html标签来加载我们需要的资源

Script标签引入一段脚本

Img标签插入一张图片

Link标签加载样式文件

Iframe嵌入不同源的页面

同源策略下不能做什么：

1. 不能直接通过js访问不同源之下的页面DOM结构
2. 在对不同源发送请求时也无法获取到服务器响应内容
3. js中无法访问不属于同个源的cookie、LocalStorage中存储的内容。

## 2.常用的跨域方法

### 客户端与服务端通信

#### 一、CORS（Cross-Origin Resource Sharing）跨域资源共享

基本思想就是引入一些自定义的HTTP Header来完成客户端与服务端的通信

对于一些简单请求，浏览器在发送请求时会带上Origin请求头，指示当前的源，服务器端在处理请求时不会去检查当前请求来源是否合法，依然会正常处理请求并响应，最终浏览器在拿到响应之后会检查服务端响应的Access-Control-Allow-Origin列表中是否存在当前页面所在的源，如果不存在会直接block掉当前请求。

1.简单请求：（在浏览器看来，同时满足以下条件的请求都认为是简单请求）

1. 请求方法为GET或者POST；
2. 只包含Accept、Accept-Language、Content-Language或者Content-Type（取值为application/x-www-form-urlencoded,multipart/form-data, 或者text/plain）

2.对于非简单请求

对于非简单请求，浏览器会先向服务器发送一个Preflight请求，该请求使用Option方法，并包含以下Header：

（1）Origin

（2）Access-Control-Request-Method：询问服务器是否支持某方法；

（3）Access-Control-Request-Headers：询问服务器是否支持请求中包含的非简单Header

其中后两个Header只会出现在Preflight请求中。然后浏览器收到包含以下Header的服务器响应：

（1）Access-Control-Allow-Origin

（2）Access-Control-Allow-Methods：对客户端回应服务器支持的请求方法列表；

（3）Access-Control-Allow-Headers：对客户端回应服务器支持的Header；

Preflight请求至此也算是告一段落，之后浏览器会检查当前请求发出的源是否在服务端响应的Access-Control-Allow-Origin列出的源的列表中，如果是才会发送真正的请求。在实验过程中，浏览器并不一定要在服务器支持Preflight请求查询的请求方法和Header时才发送真正的请求，只要发出请求的源是合法的就会在Preflight请求之后把请求发出去。

1. JSONP( JSON with Padding )



JSONP缺陷:

1. 它只能用于GET请求
2. 后端应用程序在处理过程可能会出现4xx、5xx错误或者遇到其他意外情况，导致无法返回正确的js函数调用格式的字符串的情况，所以还需要监听script标签的onerror事件来处理可能出现的意外情况。
3. Cookie跨域共享

cookie作为客户端存储的一种方案，在客户端设置cookie也有以下几种方法：

1. 配置服务端返回Set-Cookie响应头
2. 在页面上的JavaScript代码中通过document.cookie，直接设置document.cookie为我们需要存储的内容并不会覆盖现有的cookie，举例：



cookie是有过期时间的，如果像上面的代码一样没有显式地设置cookie的过期时间，则在浏览器退出之后相应的cookie也会被清除。

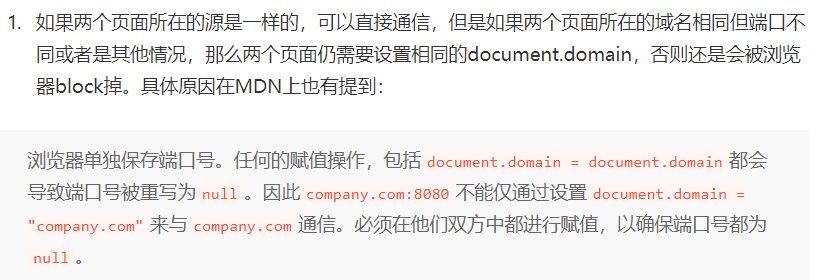
1. 跨页面通信

除了和不同源的服务器进行通信的需求以外，我们还会遇到跨页面通信问题，需要访问其他页面上的一些信息，或者将一些数据持久化，以供其他页面取用。具体方式如下：

1、document.domain

通过这种方式跨域的两个源需要满足一定的条件的，即两个源的域名需要是父子域的关系或者是相同的域。

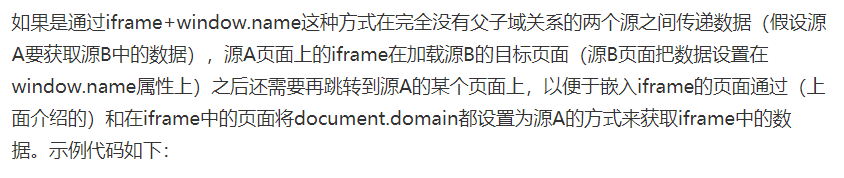
因为页面设置document.domain的值只能是当前域本身，或者是父域，而不能是其他不相关的域名。只有两个页面的document.domain都设置成相同的值，嵌入iframe的页面和iframe加载的页面才能相互获取到彼此的页面信息（包括DOM结构、window对象等）。



1. window对象name属性

浏览器具有这样一个特性：

同一个标签页或者同一个iframe框架加载过的页面，共享相同的window.name属性值，意味着只要是在同一个标签页里面打开过的页面（不管是否同源），这些页面上window.name属性值都是相同的。利用这个特性，就可以将这个属性作为在不同页面之间传递数据的介质。





1. HTML5 cross-document message

HTML5中引入的一种跨页面通信的方式，称为跨文档消息传送。

功能：可以实现主页面和嵌入的iframe子页面（或者由当前页面打开的页面）之间完成数据的传递； 可以用于当前JavaScript引擎线程和其他worker线程之间完成数据交换。

如果是与通过iframe加载的子页面进行通信，则需要先获取到接收数据的目标页面的window对象（具体通过前面提到的方法来获取），通过该对象的postMessage方法可以向目标页面发送数据。



1. LocalStorage

localStorage是HTML5引入的客户端存储方案，通过localStorage存储的内容会一直保存在客户端，除非调用removeItem方法显式移除，否则内容将永久保留。

localStorage存储的内容是以源为单位进行管理的，这意味着即使域名相同，端口不同的页面也无法通过localStorage进行通信的。

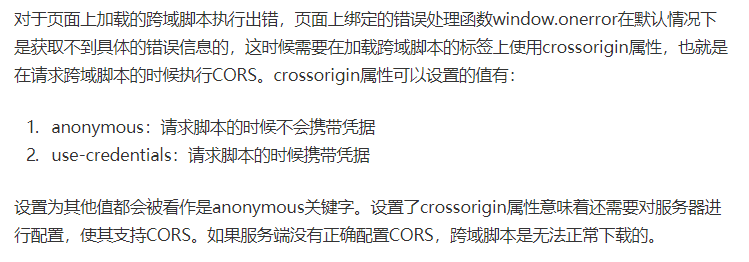
在浏览器的多个标签页中分别打开多个同源页面，这些页面中的window对象可以通过监听storage事件，当其他标签页的页面在设置localStorage中的内容时会触发该事件来进行通知，通过这种方式也可以实现跨页面通信。

### 其他跨域问题

1. 字体文件加载

CSS中引用的字体文件加载也存在跨域问题，需要设置CORS才能加载其他域下的字体文件。默认情况下定义新的字体不会立即去下载对应的字体文件，只有当页面上的元素使用了这种字体才会去下载对应的字体文件。

1. 跨域脚本错误处理



1. canvas绘制内容转化为文件对象

canvas中动态加载的图片可以直接画到canvas中，但是在将canvas转化成文件对象进行操作时也存在跨域问题，会遇到“Tainted canvases may not be exported”错误。这时候需要对动态加载的图片对象设置crossOrigin属性，同时也需要配置服务器使其支持CORS。

