链滴

深入理解前端跨域方法和原理

作者：gentoo666

原文链接：https://ld246.com/article/1496984279650

来源网站：链滴

前言

受浏览器同源策略的限制，本域的js不能操作其他域的页面对象（比如DOM）。但在安全限制的同时给注入iframe或是ajax应用上带来了不少麻烦。所以我们要通过一些方法使本域的js能够操作其他域页面对象或者使其他域的js能操作本域的页面对象（iframe之间）。

这里需要明确的一点是：所谓的域跟js的存放服务器没有关系，比如baidu.com的页面加载了google.om的js，那么此js的所在域是baidu.com而不是google.com。也就是说，此时该js能操作baidu.co的页面对象，而不能操作google.com的页面对象。

跨域的方法总结

单向跨域（一般用于获取数据）

一、使用JSONP跨域

原理：因为通过script标签引入的js是不受同源策略的限制的（正如前文提到的baidu.com的页面加了google.com的js）。所以我们可以通过script标签引入一个js或者是一个其他后缀形式（如PHP，js等）的文件，此文件返回一个js函数的调用，如返回JSONP_getUsers(["paco","john","lili"])，也就说此文件返回的结果调用了JSONP_getUsers函数，并且把["paco","john","lili"]传进去，这个["paco",john","lili"]是一个用户列表。那么如果此时我们的页面中有一个JSONP_getUsers函数，那么JSONP_etUsers就被调用到，并且传入了用户列表。此时就实现了在本域获取其他域数据的功能，也就是跨。

实现例子如下：

前端引入远程js并定义好JSONP_getUsers函数，注意需要先定义好JSONP_getUsers函数，避免在程js加载完成并调用JSONP_getUsers时，此函数不存在：

[html] view plain copy

print?

```
1. //本域为baidu.com
2.
<script>
3.

 function JSONP_getUsers(users){
4.

    console.dir(users);
5.

 }
6. script>
7. //加载google.com的getUsers.php
```

8.

<script src="http://www.google.com/getUsers.php">script>

需要google.com提供支持，getUsers.php代码如下：

[html] view plain copy

print?

1. php>
2.

echo 'JSONP_getUsers(["paco","john","lili"])';//返回一个js函数的调用
3. ?>

为什么script标签引入的文件不受同源策略的限制？因为script标签引入的文件内容是不能够被客户端js获取到的，不会影响到被引用文件的安全，所以没必要使script标签引入的文件遵循浏览器的同源策。而通过ajax加载的文件内容是能够被客户端js获取到的，所以ajax必须遵循同源策略，否则被引入件的内容会泄漏或者存在其他风险。

JSONP的缺点则是：它只支持GET请求而不支持POST等其它类型的HTTP请求（虽然采用post+动态成iframe是可以达到post跨域的目的，但这样做是一个比较极端的方式，不建议采用）。一般get请能完成所有功能。比如如果需要给其他域服务器传送参数可以在请求后挂参数（注意不要挂隐私数据，即

[html] view plain copy

print?

1.

<script src="http://www.google.com/getUsers.php?flag=do&time=1">script>。

JSONP易于实现，但是也会存在一些安全隐患，如果第三方的脚本随意地执行，那么它就可以篡改页内容，截获敏感数据。但是在受信任的双方传递数据，JSONP是非常合适的选择。可以看出来JSON跨域一般用于获取其他域的数据。

一般能够用JSONP实现跨域就用JSONP实现，这也是前端用的最多的跨域方法。

二、动态创建script标签

这种方法其实是JSONP跨域的简化版，JSONP只是在此基础上加入了回调函数。

比如上例中的getUsers.php返回的如果不是一个js函数的调用，而是一个js变量，如：

[html] view plain copy

print?

1. php>
2.

```
 echo 'var users=["paco","john","lili"]';//返回一个js变量users
3. ?>
```

那么在本域下就可以取到data变量，这里需要注意判断script节点是否加载完毕，如：

[html] view plain copy

print?

```
1. js.onload = js.onreadystatechange = function() {
2.
 if (!this.readyState || this.readyState === 'loaded' || this.readyState === 'complete') {
3.
    console.log(users);//此处取出其他域的数据
4.
    js.onload = js.onreadystatechange = null;
5.
 }
6. };
```

三、flash URLLoader

flash有自己的一套安全策略，服务器可以通过crossdomain.xml文件来声明能被哪些域的SWF文件问，SWF也可以通过API来确定自身能被哪些域的SWF加载。当跨域访问资源时，例如从域baidu.co请求域google.com上的数据，我们可以借助flash来发送HTTP请求。首先，修改域google.com上的cossdomain.xml(一般存放在根目录，如果没有需要手动创建)，把baidu.com加入到白名单。其次，过Flash URLLoader发送HTTP请求，最后，通过Flash API把响应结果传递给JavaScript。Flash URLLader是一种很普遍的跨域解决方案，不过需要支持iOS的话，这个方案就不可行了。

四、Access Control

此跨域方法目前只在很少的浏览器中得以支持，这些浏览器可以发送一个跨域的HTTP请求（Firefox, oogle Chrome等通过XMLHTTPRequest实现，IE8下通过XDomainRequest实现），请求的响应必包含一个Access- Control-Allow-Origin的HTTP响应头，该响应头声明了请求域的可访问权限。例如aidu.com对google.com下的getUsers.php发送了一个跨域的HTTP请求（通过ajax），那么getUsersphp必须加入如下的响应头：

[html] view plain copy

print?

```
1. header("Access-Control-Allow-Origin: http://www.baidu.com");//表示允许baidu.com跨域请本文件
```

五、window.name

window 对象的name属性是一个很特别的属性，当该window的location变化，然后重新加载，它的ame属性可以依然保持不变。那么我们可以在页面 A中用iframe加载其他域的页面B，而页面B中用JaaScript把需要传递的数据赋值给window.name，iframe加载完成之后（iframe.onload），页面A修iframe的地址，将其变成同域的一个地址，然后就可以读出iframe的window.name的值了（因为A的window.name和iframe中的window.name互相独立的，所以不能直接在A中获取window.name而要通过iframe获取其window.name）。这个方式非常适合单向的数据请求，而且协议简单、安全不会像JSONP那样不做限制地执行外部脚本。

六、服务器代理

在数据提供方没有提供对JSONP协议或者 window.name协议的支持，也没有对其它域开放访问权限，我们可以通过server proxy的方式来抓取数据。例如当baidu.com域下的页面需要请求google.co下的资源文件getUsers.php时，直接发送一个指向 google.com/getUsers.php的Ajax请求肯定是会浏览器阻止。这时，我们在baidu.com下配一个代理，然后把Ajax请求绑定到这个代理路径下，例如baidu.com/proxy/, 然后这个代理发送HTTP请求访问google.com下的getUsers.php，跨域的HTTP请是在服务器端进行的（服务器端没有同源策略限制），客户端并没有产生跨域的Ajax请求。这个跨域式不需要和目标资源签订协议，带有侵略性。

双向跨域（两个iframe之间或者两个页面之间，一般用于获取对方数据，document.domain方式可以直接操作对方DOM）

七、document.domain（两个iframe之间）

通过修改document的domain属性，我们可以在域和子域或者不同的子域之间通信。同域策略认为和子域隶属于不同的域，比如baidu.com和 youxi.baidu.com是不同的域，这时，我们无法在baidu.cm下的页面中调用youxi.baidu.com中定义的JavaScript方法。但是当我们把它们document的domai属性都修改为baidu.com，浏览器就会认为它们处于同一个域下，那么我们就可以互相获取对方数据者操作对方DOM了。

问题：

1、安全性，当一个站点被攻击后，另一个站点会引起安全漏洞。

2、如果一个页面中引入多个iframe，要想能够操作所有iframe，必须都得设置相同domain。

八、location.hash（两个iframe之间），又称FIM，Fragment Identitier Messaging的简写

因为父窗口可以对iframe进行URL读写，iframe也可以读写父窗口的URL，URL有一部分被称为hash就是#号及其后面的字符，它一般用于浏览器锚点定位，Server端并不关心这部分，应该说HTTP请求程中不会携带hash，所以这部分的修改不会产生HTTP请求，但是会产生浏览器历史记录。此方法的理就是改变URL的hash部分来进行双向通信。每个window通过改变其他 window的location来发送息（由于两个页面不在同一个域下IE、Chrome不允许修改parent.location.hash的值，所以要借助于窗口域名下的一个代理iframe），并通过监听自己的URL的变化来接收消息。这个方式的通信会造成些不必要的浏览器历史记录，而且有些浏览器不支持onhashchange事件，需要轮询来获知URL的改，最后，这样做也存在缺点，诸如数据直接暴露在了url中，数据容量和类型都有限等。下面举例说明：

假如父页面是baidu.com/a.html，iframe嵌入的页面为google.com/b.html（此处省略了域名等url性），要实现此两个页面间的通信可以通过以下方法。

1、a.html传送数据到b.html

（1） a.html下修改iframe的src为google.com/b.html#paco

（2） b.html监听到url发生变化，触发相应操作

2、b.html传送数据到a.html，由于两个页面不在同一个域下IE、Chrome不允许修改parent.location.ash的值，所以要借助于父窗口域名下的一个代理iframe

（1） b.html下创建一个隐藏的iframe，此iframe的src是baidu.com域下的，并挂上要传送的hash据，如src="http://www.baidu.com/proxy.html#data"

（2） proxy.html监听到url发生变化，修改a.html的url（因为a.html和proxy.html同域，所以proxy.tml可修改a.html的url hash）

（3） a.html监听到url发生变化，触发相应操作

b.html页面的关键代码如下

[html] view plain copy

print?

```
1. try {
2.
 parent.location.hash = 'data';
3. } catch (e) {
4.
 // ie、chrome的安全机制无法修改parent.location.hash，
5.
 var ifrproxy = document.createElement('iframe');
6.
 ifrproxy.style.display = 'none';
7.
 ifrproxy.src = "http://www.baidu.com/proxy.html#data";
8.
 document.body.appendChild(ifrproxy);
9. }
```

proxy.html页面的关键代码如下

[html] view plain copy

print?

1. //因为parent.parent（即baidu.com/a.html）和baidu.com/proxy.html属于同一个域，所以可改变其location.hash的值

2. parent.parent.location.hash = self.location.hash.substring(1);

九、使用HTML5的postMessage方法（两个iframe之间或者两个页面之间）

高级浏览器Internet Explorer 8+, chrome，Firefox , Opera 和 Safari 都将支持这个功能。这个功主要包括接受信息的"message"事件和发送消息的"postMessage"方法。比如baidu.com域的A页面过iframe嵌入了一个google.com域的B页面，可以通过以下方法实现A和B的通信

A页面通过postMessage方法发送消息：

[html] view plain copy

print?

```
1. window.onload = function() {
2.
 var ifr = document.getElementById('ifr');
3.
 var targetOrigin = "http://www.google.com";
4.
 ifr.contentWindow.postMessage('hello world!', targetOrigin);
5. };
```

postMessage的使用方法：

otherWindow.postMessage(message, targetOrigin);

otherWindow: 指目标窗口，也就是给哪个window发消息，是 window.frames 属性的成员或者由indow.open 方法创建的窗口

message: 是要发送的消息，类型为 String、Object (IE8、9 不支持)

targetOrigin: 是限定消息接收范围，不限制请使用 '*'

B页面通过message事件监听并接受消息:

[html] view plain copy

print?

```
1. var onmessage = function (event) {
2. var data = event.data;//消息
3. var origin = event.origin;//消息来源地址
```

```
4. var source = event.source;//源Window对象
5. if(origin=="http://www.baidu.com"){
6. console.log(data);//hello world!
7. }
8. };
9. if (typeof window.addEventListener != 'undefined') {
10. window.addEventListener('message', onmessage, false);
11. } else if (typeof window.attachEvent != 'undefined') {
12. //for ie
13. window.attachEvent('onmessage', onmessage);
14. }
```

同理，也可以B页面发送消息，然后A页面监听并接受消息。

总结

跨域的方法很多，不同的应用场景我们都可以找到一个最合适的解决方案。比如单向的数据请求，我应该优先选择JSONP或者window.name，双向通信优先采取location.hash，在未与数据提供方达成信协议的情况下我们也可以用server proxy的方式来抓取数据。