无双
博学善思，自强不息

导航
博客园
首页
新随笔
联系
订阅 XML
管理

< 2016年8月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

js中几种实用的跨域方法原理详解

这里说的js跨域是指通过js在不同的域之间进行数据传输或通信，比如用ajax向一个不同的域请求数据，或者通过js获取页面中不同域的框架中(iframe)的数据。只要协议、域名、端口有任何一个不同，都被当作是不同的域。

下表给出了相对http://store.company.com/dir/page.html同源检测的结果：

URL	结果	原因
http://store.company.com/dir2/other.html	成功	
http://store.company.com/dir/inner/another.html	成功	
https://store.company.com/secure.html	失败	协议不同
http://store.company.com:81/dir/etc.html	失败	端口不同
http://news.company.com/dir/other.html	失败	主机名不同

要解决跨域的问题，我们可以使用以下几种方法：

一、通过**jsonp**跨域

在js中，我们直接用XMLHttpRequest请求不同域上的数据时，是不可以的。但是，在页面上引入不同域上的js脚本文件却是可以的，jsonp正是利用这个特性来实现的。

比如，有个a.html页面，它里面的代码需要利用ajax获取一个不同域上的json数据，假设这个json数据地址是http://example.com/data.php,那么a.html中的代码就可以这样：

```
<script>
function dosomething(jsondata){
    //处理获得的json数据
}
</script>
<script src="http://example.com/data.php?callback=dosomething"></script>
```

我们看到获取数据的地址后面还有一个callback参数，按惯例是用这个参数名，但是你用其他的也一样。当然如果获取数据的jsonp地址页面不是你自己能控制的，就得按照提供数据的那一方的规定格式来操作了。

因为是当做一个js文件来引入的，所以http://example.com/data.php返回的必须是一个能执行的js文件，所以这个页面的php代码可能是这样的：

```
<?php
$callback = $_GET['callback']; //得到回调函数名
$data = array('a','b','c'); //要返回的数据
echo $callback.'('.json_encode($data).')'; //输出
?>
```

最终那个页面输出的结果是：

```
dosomething(['a','b','c'])
```

所以通过http://example.com/data.php?callback=dosomething得到的js文件，就是我们之前定义的dosomething函数,并且它的参数就是我们需要的json数据，这样我们就跨域获得了我们需要的数据。

这样jsonp的原理就很清楚了，通过script标签引入一个js文件，这个js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。所以jsonp是需要服务器端的页面进行相应的配合的。

知道jsonp跨域的原理后我们就可以用js动态生成script标签来进行跨域操作了，而不用特意的手动的书写那些script标签。如果你的页面使用jquery，那么通过它封装的方法就能很方便的来进行jsonp操作了。

```
<script>
$.getJSON('http://example.com/data.php?callback=?',function(jsondata){
    //处理获得的json数据
});
</script>
```

原理是一样的，只不过我们不需要手动的插入script标签以及定义回掉函数。jquery会自动生成一个全局函数来替换callback=?中的问号，之后获取到数据后又会自动销毁，实际上就是起一个临时代理函数的作用。$.getJSON方法会自动判断是否跨域，不跨域的话，就调用普通的ajax方法；跨域的话，则会以异步加载js文件的形式来调用jsonp的回调函数。

2、通过修改**document.domain**来跨子域

浏览器都有一个同源策略，其限制之一就是第一种方法中我们说的不能通过ajax的方法去请求不同源中的文档。 它的第二个限制是浏览器中不同域的框架之间是不能进行js的交互操作的。有一点需要说明，不同的框架之间（父子或同辈），是能够获取到彼此的window对象的，但蛋疼的是你却不能使用获取到的window对象的属性和方法(html5中的postMessage方法是一个例外，还有些浏览器比如ie6也可以使用top、parent等少数几个属性)，总之，你可以当做是只能获取到一个几乎无用的window对象。比如，有一个页面，它的地址是http://www.example.com/a.html ， 在这个页面里面有一个iframe，它的src是http://example.com/b.html, 很显然，这个页面与它里面的iframe框架是不同域的，所以我们是无法通过在页面中书写js代码来获取iframe中的东西的：

```
<script>
function onLoad(){
    var iframe = document.getElementById('iframe');
    var win = iframe.contentWindow; //这里是能够获取到 iframe 里的 window对象的，但该window对象的属性和方法几乎是不可用的
    var doc = win.document; //这里是获取不到 iframe 里的document对象的
    var name = win.name; //这里同样是获取不到window对象的name属性的
    ...
}
</script>
<iframe id="iframe" src="http://example.com/b.html" onload="onLoad()"></iframe>
```

这个时候，document.domain就可以派上用场了，我们只要把http://www.example.com/a.html 和http://example.com/b.html这两个页面的document.domain都设成相同的域名就可以了。但要注意的是，document.domain的设置是有限制的，我们只能把document.domain设置成自身或更高一级的父域，且主域必须相同。例如：a.b.example.com 中某个文档的document.domain 可以设成a.b.example.com、b.example.com 、example.com中的任意一个，但是不可以设成c.a.b.example.com,因为这是当前域的子域，也不可以设成baidu.com,因为主域已经不相同了。

在页面 http://www.example.com/a.html 中设置document.domain:

```
<iframe src="http://example.com/b.html" id="iframe" onload="test()"></iframe>
<script>
document.domain = 'example.com'; //设置成主域
function test(){
    alert(document.getElementById('iframe').contentWindow);
}
</script>
```

在页面 http://example.com/b.html 中也设置document.domain，而且这也是必须的，虽然这个文档的domain就是example.com,但是还是必须显示的设置document.domain的值：

```
<script>
document.domain = 'example.com'; //在iframe载入的这个页面也设置document.domain,使之与主页面的document.domain相同
</script>
```

这样我们就可以通过js访问到iframe中的各种属性和对象了。

不过如果你想在http://www.example.com/a.html 页面中通过ajax直接请求http://example.com/b.html 页面，即使你设置了相同的document.domain也还是不行的，所以修改document.domain的方法只适用于不同子域的框架间的交互。如果你想通过ajax的方法去与不同子域的页面交互，除了使用jsonp的方法外，还可以用一个隐藏的iframe来做一个代理。原理就是让这个iframe载入一个与你想要通过ajax获取数据的目标页面处在相同的域的页面，所以这个iframe中的页面是可以正常使用ajax去获取你要的数据的，然后就是通过我们刚刚讲得修改document.domain的方法，让我们能通过js完全控制这个iframe，这样我们就可以让iframe去发送ajax请求，然后收到的数据我们也可以获得了。

3、使用**window.name**来进行跨域

window对象有个name属性，该属性有个特征：即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，window.name是持久存在一个窗口载入过的所有页面中的，并不会因新页面的载入而进行重置。

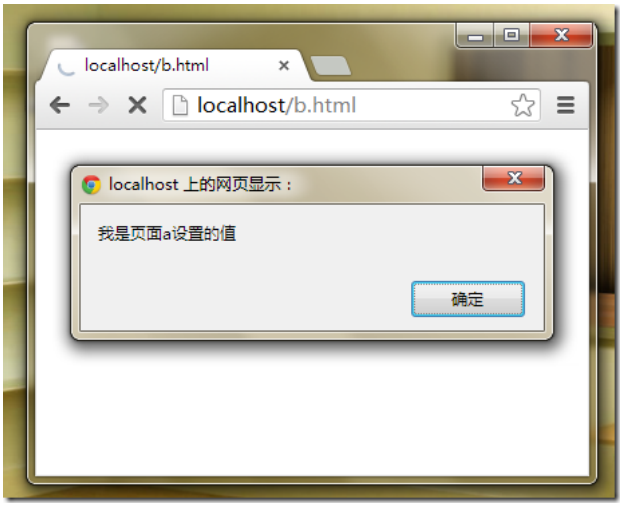
比如：有一个页面a.html,它里面有这样的代码：

```
<script>
window.name = '我是页面a设置的值';  //设置window.name的值
setTimeout(function(){
    window.location = 'b.html'
},3000); //3秒后把一个新页面b.html载入到当前的window
</script>
```

再看看b.html页面的代码：

```
<script>
alert(window.name); //读取window.name的值
</script>
```

a.html页面载入后3秒，跳转到了b.html页面，结果为：



我们看到在b.html页面上成功获取到了它的上一个页面a.html给window.name设置的值。如果在之后所有载入的页面都没对window.name进行修改的话，那么所有这些页面获取到的window.name的值都是a.html页面设置的那个值。当然，如果有需要，其中的任何一个页面都可以对window.name的值进行修改。注意，window.name的值只能是字符串的形式，这个字符串的大小最大能允许2M左右甚至更大的一个容量，具体取决于不同的浏览器，但一般是够用了。

上面的例子中，我们用到的页面a.html和b.html是处于同一个域的，但是即使a.html与b.html处于不同的域中，上述结论同样是适用的，这也正是利用window.name进行跨域的原理。

下面就来看一看具体是怎么样通过window.name来跨域获取数据的。还是举例说明。

比如有一个www.example.com/a.html页面,需要通过a.html页面里的js来获取另一个位于不同域上的页面www.cnblogs.com/data.html里的数据。

data.html页面里的代码很简单，就是给当前的window.name设置一个a.html页面想要得到的数据值。data.html里的代码：

```
<script>
window.name = '我就是页面a.html想要的数据,所有可以转化成字符串来传递的数据都可以在这里使用，比如可以传递一个json数据';
</script>
```

那么在a.html页面中，我们怎么把data.html页面载入进来呢？显然我们不能直接在a.html页面中通过改变window.location来载入data.html页面，因为我们想要即使a.html页面不跳转也能得到data.html里的数据。答案就是在a.html页面中使用一个隐藏的iframe来充当一个中间人角色，由iframe去获取data.html的数据，然后a.html再去得到iframe获取到的数据。

充当中间人的iframe想要获取到data.html的通过window.name设置的数据，只需要把这个iframe的src设为www.cnblogs.com/data.html就行了。然后a.html想要得到iframe所获取到的数据，也就是想要得到iframe的window.name的值，还必须把这个iframe的src设成跟a.html页面同一个域才行，不然根据前面讲的同源策略，a.html是不能访问到iframe里的window.name属性的。这就是整个跨域过程。

看下a.html页面的代码：

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>window.name跨域</title>
    <script>
    function getData(){//iframe载入data.html页面后会执行此函数
        var iframe = document.getElementById('proxy');
        iframe.onload = function(){//这个时候a.html与iframe已经是处于同一源了，可以互相访问
            var data = iframe.contentWindow.name; //获取iframe里的window.name,也就是data.html页面给它设置的数据
            alert(data); //成功获取到了data.html里的数据
        }
        iframe.src = 'b.html'; //这里的b.html为随便的一个页面，只要与a.html同源就行了，目的是让a.html能访问到iframe里的东西，设置成about:blank
    }
    </script>
</head>
<body>
    <iframe id="proxy" src="http://www.cnblogs.com/data.html" style="display:none" onload="getData()"></iframe>
</body>
</html>
```

上面的代码只是最简单的原理演示代码，你可以对使用js封装上面的过程，比如动态的创建iframe,动态的注册各种事件等等，当然为了安全，获取完数据后，还可以销毁作为代理的iframe。网上也有很多类似的现成代码，有兴趣的可以去找一下。

通过window.name来进行跨域，就是这样子的。

4、使用**HTML5**中新引进的**window.postMessage**方法来跨域传送数据

window.postMessage(message,targetOrigin) 方法是html5新引进的特性，可以使用它来向其它的window对象发送消息，无论这个window对象是属于同源或不同源，目前IE8+、FireFox、Chrome、Opera等浏览器都已经支持window.postMessage方法。

调用postMessage方法的window对象是指要接收消息的那一个window对象，该方法的第一个参数message为要发送的消息，类型只能为字符串；第二个参数targetOrigin用来限定接收消息的那个window对象所在的域，如果不想限定域，可以使用通配符 * 。

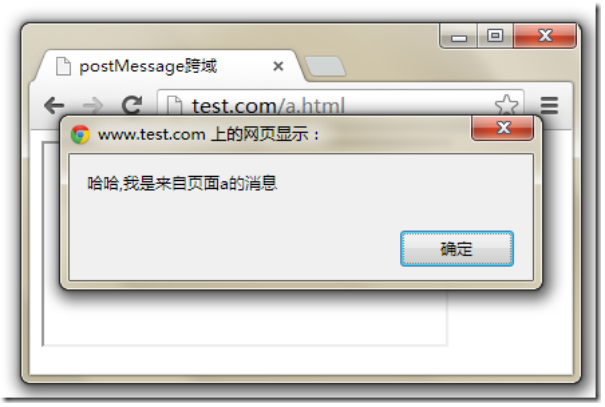
需要接收消息的window对象，可是通过监听自身的message事件来获取传过来的消息，消息内容储存在该事件对象的data属性中。

上面所说的向其他window对象发送消息，其实就是指一个页面有几个框架的那种情况，因为每一个框架都有一个window对象。在讨论第二种方法的时候，我们说过，不同域的框架间是可以获取到对方的window对象的，而且也可以使用window.postMessage这个方法。下面看一个简单的示例，有两个页面

```
<!-- 这是 页面 http://test.com/a.html 的代码 -->
<script>
function onLoad(){
    var iframe = document.getElementById('iframe');
    var win = iframe.contentWindow; //获取window对象
    win.postMessage('哈哈,我是来自页面a的消息','*'); //向不同域的http://www.test.com/b.html页面发送消息
}
</script>
<iframe id="iframe" src="http://www.test.com/b.html" onload="onLoad()"></iframe>
```

```
<!-- 这是 页面 http://www.test.com/b.html 的代码 -->
<script>
window.onmessage = function(e){ //注册message事件用来接收消息
    e = e || event; //获取事件对象
    alert(e.data);  //通过data属性得到传送的消息
}
</script>
```

我们运行a页面后得到的结果：



我们看到b页面成功的收到了消息。

使用postMessage来跨域传送数据还是比较直观和方便的，但是缺点是IE6、IE7不支持，所以用不用还得根据实际需要来决定。

结语：

除了以上几种方法外，还有flash、在服务器上设置代理页面等跨域方式，这里就不做介绍了。

以上四种方法，可以根据项目的实际情况来进行选择应用，个人认为window.name的方法既不复杂，也能兼容到几乎所有浏览器，这真是极好的一种跨域方法。

分类: js

无双
关注 - 3
粉丝 - 852
+加关注

« 上一篇：简述“用新浪微博账号登录”功能开发流程
» 下一篇：安卓 9.png 图片的制作

posted on 2013-07-15 18:47 无双 阅读(102500) 评论(31) 编辑 收藏

评论

#1楼 2013-07-15 19:06 lihuabest

总结的还不错

支持(0) 反对(0)

#2楼 2013-07-15 20:35 零度冷

涨见识了。。。

支持(0) 反对(0)

#3楼 2013-07-15 22:58 Never_say

介绍的很全面，强大。而且写的也很细心，周详，收货很多。
不过感觉结合jquery，jsonp的方式最方便了。

支持(0) 反对(0)

#4楼 2013-07-15 23:22 无心花

学习了，呵呵

支持(0) 反对(0)

#5楼 2013-07-16 09:27 寒@鹏

jsonp最方便

支持(0) 反对(0)

#6楼 2013-07-16 12:54 liujb

不错，jsonp肯定是最广的

支持(0) 反对(0)

#7楼 2013-07-16 17:24 ningmj00

好文章，之前只听过jsonp，现在见识过了。哈哈

支持(0) 反对(0)

#8楼 2013-07-23 13:01 要有好的心情

收藏

支持(0) 反对(0)

#9楼 2013-07-29 00:37 MyNameIsJim

和楼上的小伙伴们一样

支持(0) 反对(0)

#10楼 2013-10-30 11:20 iam_mackong

还不错

支持(0) 反对(0)

#11楼 2014-02-04 14:59 沙漠中的绿洲

收获颇多啊，谢谢分享。

支持(0) 反对(0)

#12楼 2014-03-06 09:53 gaojun

长知识，非常感谢分享！！

支持(0) 反对(0)

#13楼 2014-05-28 18:33 JsonID

学习了

支持(0) 反对(0)

#14楼 2014-08-12 20:19 PHSeven

不错。

支持(0) 反对(0)

#15楼 2014-12-02 14:17 royalrover

@ rainFish05
这只是本地文件访问，无所谓同源不同源吧。

支持(0) 反对(0)

#16楼 2014-12-02 14:21 rainFish05

@ royalrover
嗯嗯，之前请教人已经弄明白了，还是很感谢你的回答^_^

支持(0) 反对(0)

#17楼 2015-03-10 11:27 VinceChueng

window.name中 iframe的src为什么是www.example.com中的data.html呢, data.html不应该是另一个域中的么

支持(1) 反对(0)

#18楼 2015-04-10 18:34 草木不凋零

多谢分享--秋叶

支持(0) 反对(0)

#19楼 2015-08-04 11:29 毛已煜

讲的不错 谢谢楼主
jsonp 不适用 所有的 跨越请求，这个讲的全面

支持(0) 反对(0)

#20楼 2015-08-10 18:06 Zakero

谢谢分享，好人多福。

支持(0) 反对(0)

#21楼 2015-08-25 19:28 陌上微语

讲的太好了，谢谢楼主分享

支持(0) 反对(0)

#22楼 2015-12-07 15:48 ITCode

同问！！！@未来的未未来的来
window.name中 iframe的src为什么是www.example.com中的data.html呢, data.html不应该是另一个域中的么？

支持(0) 反对(0)

#23楼[楼主] 2015-12-07 16:12 无双

@ ITCode
@未来的未未来的来

那里确实是写错了，应该是www.cnblogs.com/data.html，图片已经ps过来了，感谢指出:)

支持(0) 反对(0)

#24楼 2016-01-07 22:13 阿林十一

好文章，之前只听过jsonp，不太懂，现在见识了。哈哈~

支持(0) 反对(0)

#25楼 2016-03-11 10:51 檀紮

1’还必须把这个iframe的src设成跟a.html页面同一个域才行，不然根据前面讲的同源策略，a.html是不能访问到iframe里的window.name属性的‘。 2‘上面的例子中，我们用到的页面a.html和b.html是处于同一个域的，但是即使a.html与b.html处于不同的域中，上述结论同样是适用的，这也正是利用window.name进行跨域的原理。’ 这两句话冲突了

支持(0) 反对(1)

#26楼 2016-03-17 17:52 ihankang

@ 檀紮
没有冲突，iframe先设置为与目标相同的源，获取ajax数据存入window.name，此时iframe指向的是目标地址，与父框架不同源，父子存在跨域调用，所以再获取ajax数据后将iframe的src指回与父窗口同源的地址，这样iframe与父窗口才不会产生跨域问题。

支持(0) 反对(0)

#27楼 2016-03-24 11:18 王朋

又见识了一种新方式，window.name

支持(0) 反对(0)

#28楼 2016-03-26 20:38 venoral

感谢楼主！通俗易懂！！

支持(0) 反对(0)

#29楼 2016-05-17 16:55 jllezhou

mark

支持(0) 反对(0)

#30楼 2016-06-13 10:39 张泰峰

目前又多了一种方式 cors 服务端主动指定允许跨域请求的地址

```
//跨域请求实例
    function CORSRequest(method,url,opation,callback) {
        var xhr = new XMLHttpRequest();
        if ("withCredentials" in xhr) {
        // 此时即支持CORS的情况
        // 检查XMLHttpRequest对象是否有“withCredentials”属性
        // “withCredentials”仅存在于XMLHTTPRequest level 2对象里
        } else {
        // 否则检查是否支持XDomainRequest
        // XDomainRequest仅存在于IE中，是IE用于支持CORS请求的方式
        xhr = new XDomainRequest();
        }
        xhr.open(method, url, true);
        if(method=="POST"){
            xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
            xhr.send(opation);
        }else{
            xhr.send();
        }

        xhr.onload = function(){
            callback(xhr.responseText);

        }
    }
```

支持(0) 反对(0)

#31楼 2016-06-28 10:25 沐浴露

学习了

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新知识库文章：

· 程序猿媳妇儿注意事项
· 可是姑娘，你为什么要编程呢？
· 知其所以然（以算法学习为例）
· 如何给变量取个简短且无歧义的名字
· 编程的智慧
» 更多知识库文章...