[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
<?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。

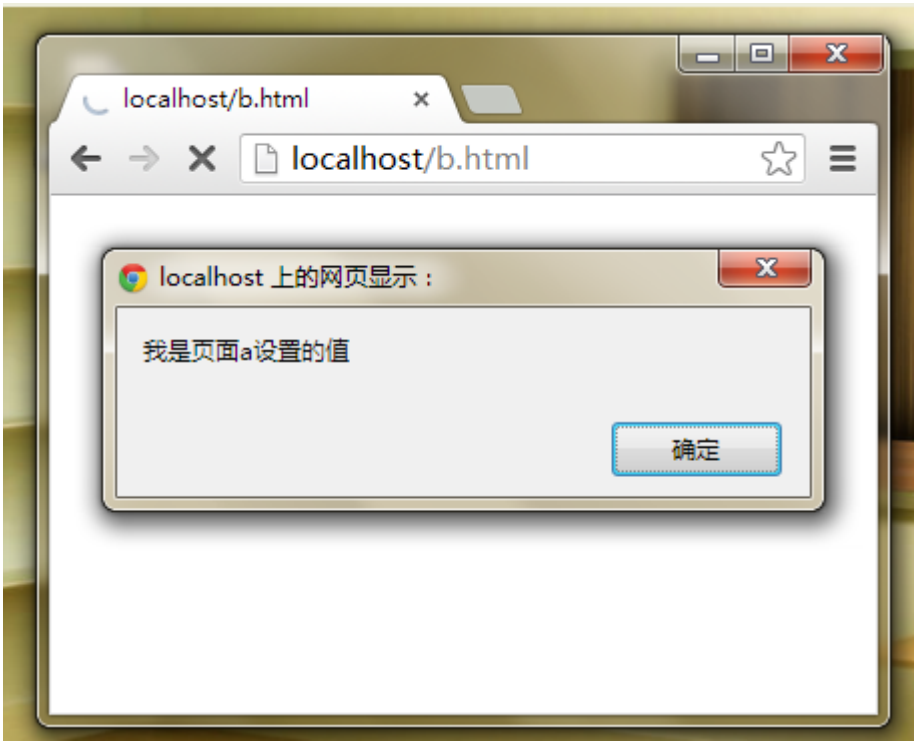
比如：有一个页面 a.html,它里面有这样的代码：

```
<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也行
    }
    </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 * 。

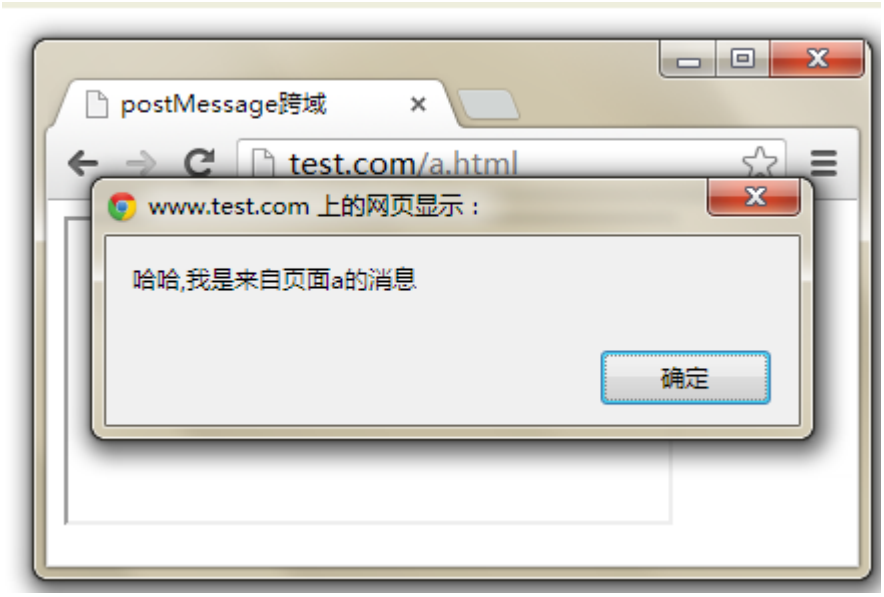
需要接收消息的 window 对象，可是通过监听自身的 message 事件来获取传过来的消息，消息内容储存在该事件对象的 data 属性中。

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。

转载至 http://www.cnblogs.com/2050/p/3191744.html