前端跨域知识总结

2017-08-20 前端大全

（点击上方公众号，可快速关注）

> 作者：伯乐在线/秦至
> http://web.jobbole.com/88519/
> 如有好文章投稿，请点击 → 这里了解详情

前言

相信每一个前端er对于跨域这两个字都不会陌生，在实际项目中应用也是比较多的。但跨域方法的多种多样实在让人目不暇接。老规矩，碰到这种情况，就只能自己总结一篇博客，作为记录。

正文

1. 什么是跨域?

跨域一词从字面意思看，就是跨域名嘛，但实际上跨域的范围绝对不止那么狭隘。具体概念如下：只要协议、域名、端口有任何一个不同，都被当作是不同的域。之所以会产生跨域这个问题呢，其实也很容易想明白，要是随便引用外部文件，不同标签下的页面引用类似的彼此的文件，浏览器很容易懵逼的，安全也得不到保障了就。什么事，都是安全第一嘛。但在安全限制的同时也给注入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能够操作其他域的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）。下面是具体的跨域情况详解：

> URL 说明 是否允许通信
> http://www.a.com/a.js
> http://www.a.com/b.js *同一域名下* *允许*
> http://www.a.com/lab/a.js
> http://www.a.com/script/b.js *同一域名下不同文件夹* *允许*
> http://www.a.com:8000/a.js
> http://www.a.com/b.js *同一域名，不同端口* *不允许*
> http://www.a.com/a.js
> https://www.a.com/b.js *同一域名，不同协议* *不允许*
> http://www.a.com/a.js
> http://70.32.92.74/b.js *域名和域名对应ip* *不允许*
> http://www.a.com/a.js
> http://script.a.com/b.js *主域相同，子域不同* *不允许（cookie这种情况下也不允许访问）*
> http://www.a.com/a.js

> http://a.com/b.js 同一域名，不同二级域名（同上） 不允许（cookie这种情况下也不允许访问）
> http://www.cnblogs.com/a.js
> http://www.a.com/b.js 不同域名 不允许

这里我们需要注意两点:

1. 如果是协议和端口造成的跨域问题“前台”是无能为力的；
2. 在跨域问题上，域仅仅是通过“URL的首部”来识别而不会去尝试判断相同的ip地址对应着两个域或两个域是否在同一个ip上。
   （“URL的首部”指window.location.protocol +window.location.host，也可以理解为“Domains, protocols and ports must match”。）

2. 通过document.domain跨域

前面说过了，浏览器有一个同源策略，其限制之一是不能通过ajax的方法去请求不同源中的文档。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。不同的框架之间是可以获取window对象的，但却无法获取相应的属性和方法。比如，有一个页面，它的地址是http://www.damonare.cn/a.html， 在这个页面里面有一个iframe，它的src是http://damonare.cn/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 type="text/javascript">
  function test(){
    var iframe = document.getElementById('ifame');
    var win = document.contentWindow;//可以获取到iframe里的window对象，但该window对象的属性和方法几乎是不可用的
    var doc = win.document;//这里获取不到iframe里的document对象
    var name = win.name;//这里同样获取不到window对象的name属性
  }
</script>
<iframe id = "iframe" src="http://damonare.cn/b.html" onload = "test()"></iframe>
```

这个时候，document.domain就可以派上用场了，我们只要把http://www.damonare.cn/a.html和http://damonare.cn/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。

- 在页面http://www.damonare.cn/a.html 中设置
- document.domain:

```
<iframe id = "iframe" src="http://damonare.cn/b.html" onload = "test()"></iframe>
<script type="text/javascript">
  document.domain = 'damonare.cn';//设置成主域
```

```
    function test(){
        alert(document.getElementById('iframe').contentWindow);//contentWindow 可取得子窗口的 window 对象
    }
</script>
```

- 在页面http://damonare.cn/b.html 中也设置document.domain:

```
<script type="text/javascript">
    document.domain = 'damonare.cn';//在iframe载入这个页面也设置document.domain，使之与主页面的document.domain相同
</script>
```

修改document.domain的方法只适用于不同子域的框架间的交互。

3. 通过location.hash跨域

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请求过程中不会携带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。此方法的原理就是改变URL的hash部分来进行双向通信。每个window通过改变其他window的location来发送消息（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，所以要借助于父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一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的改变，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属性），要实现此两个页面间的通信可以通过以下方法。

- a.html传送数据到b.html
- a.html下修改iframe的src为google.com/b.html#paco
- b.html监听到url发生变化，触发相应操作
- 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.hash的值，所以要借助于父窗口域名下的一个代理iframe
  - 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数据，如src=”http://www.baidu.com/proxy.html#data”
  - 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.html可修改a.html的url hash）
  - 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:

```
try {
```

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

proxy.html页面的关键代码如下：

```
//因为parent.parent（即baidu.com/a.html）和baidu.com/proxy.html属于同一个域，所以可以改变其location.hash的值
parent.parent.location.hash = self.location.hash.substring(1);
```

4. 通过HTML5的postMessage方法跨域

高级浏览器Internet Explorer 8+, chrome，Firefox , Opera 和 Safari 都将支持这个功能。这个功能主要包括接受信息的”message”事件和发送消息的”postMessage”方法。比如damonare.cn域的A页面通过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：

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

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 window.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:

```
var onmessage = function (event) {
  var data = event.data;//消息
```

```
  var origin = event.origin;//消息来源地址
  var source = event.source;//源Window对象
  if(origin=="http://www.baidu.com"){
console.log(data);//hello world!
  }
};
if (typeof window.addEventListener != 'undefined') {
  window.addEventListener('message', onmessage, false);
} else if (typeof window.attachEvent != 'undefined') {
  //for ie
  window.attachEvent('onmessage', onmessage);
}
```

同理，也可以B页面发送消息，然后A页面监听并接受消息。

5.通过jsonp跨域

刚才说的这几种都是双向通信的，即两个iframe，页面与iframe或是页面与页面之间的，下面说几种单项跨域的（一般用来获取数据），因为通过script标签引入的js是不受同源策略的限制的。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，jsp等）的文件，此文件返回一个js函数的调用。
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://damonare.cn/data.php,那么a.html中的代码就可以这样：

```
<script type="text/javascript">
   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://damonare.cn/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(一定要和后端约定好哦)：

```
<?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，输出结果为：dosomething([‘a’,’b’,’c’]);

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

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。

- JSONP的优缺点
  - JSONP的优点是：它不像XMLHttpRequest对象实现的Ajax请求那样受到同源策略的限制；它的兼容性更好，在更加古老的浏览器中都可以运行，不需要XMLHttpRequest或ActiveX的支持；并且在请求完毕后可以通过调用callback的方式回传结果。
  - 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；它只支持跨域HTTP请求这种情况，不能解决不同域的两个页面之间如何进行JavaScript调用的问题。

6. 通过CORS跨域

CORS（Cross-Origin Resource Sharing）跨域资源共享，定义了必须在访问跨域资源时，浏览器与服务器应该如何沟通。CORS背后的基本思想就是使用自定义的HTTP头部让浏览器与服务器进行沟通，从而决定请求或响应是应该成功还是失败。目前，所有浏览器都支持该功能，IE浏览器不能低于IE10。整个CORS通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。

因此，实现CORS通信的关键是服务器。只要服务器实现了CORS接口，就可以跨源通信。

平时的ajax请求可能是这样的:

```
<script type="text/javascript">
  var xhr = new XMLHttpRequest();
  xhr.open("POST", "/damonare",true);
  xhr.send();
</script>
```

以上damonare部分是相对路径，如果我们要使用CORS，相关Ajax代码可能如下所示：

```
<script type="text/javascript">
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "http://segmentfault.com/u/trigkit4/",true);
  xhr.send();
</script>
```

代码与之前的区别就在于相对路径换成了其他域的绝对路径，也就是你要跨域访问的接口地址。

服务器端对于CORS的支持，主要就是通过设置Access-Control-Allow-Origin来进行的。如果浏览器检测到相应的设置，就可以允许Ajax进行跨域的访问。关于CORS更多了解可以看下阮一峰老师的这一篇文章：跨域资源共享 CORS 详解

- CORS和JSONP对比
  - JSONP只能实现GET请求，而CORS支持所有类型的HTTP请求。
  - 使用CORS，开发者可以使用普通的XMLHttpRequest发起请求和获得数据，比起JSONP有更好的错误处理。
  - JSONP主要被老的浏览器支持，它们往往不支持CORS，而绝大多数现代浏览器都已经支持了CORS）。

CORS与JSONP相比，无疑更为先进、方便和可靠。

7. 通过window.name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：我们在任意一个页面输入

```
window.name = "My window's name";
setTimeout(function(){
    window.location.href = "http://damonare.cn/";
},1000)
```

进入damonare.cn页面后我们再检测再检测 window.name :

```
window.name; // My window's name
```

可以看到，如果在一个标签里面跳转网页的话，我们的 window.name 是不会改变的。
基于这个思想，我们可以在某个页面设置好 window.name 的值，然后跳转到另外一个页面。在这个页面中就可以获取到我们刚刚设置的 window.name 了。

> 由于安全原因，浏览器始终会保持 window.name 是string 类型。

同样这个方法也可以应用到和iframe的交互来：

比如：我的页面(http://damonare.cn/index.html)中内嵌了一个iframe：

```
<iframe id="iframe" src="http://www.google.com/iframe.html"></iframe>
```

在 iframe.html 中设置好了 window.name 为我们要传递的字符串。
我们在 index.html 中写了下面的代码：

```
var iframe = document.getElementById('iframe');
var data = '';

iframe.onload = function() {
    data = iframe.contentWindow.name;
};
```

Boom!报错！肯定的，因为两个页面不同源嘛，想要解决这个问题可以这样干：

```
var iframe = document.getElementById('iframe');
var data = '';

iframe.onload = function() {
    iframe.onload = function(){
        data = iframe.contentWindow.name;
    }
    iframe.src = 'about:blank';
};
```

或者将里面的 about:blank 替换成某个同源页面（about:blank，javascript: 和 data: 中的内容，继承了载入他们的页面的源。）

这种方法与 document.domain 方法相比，放宽了域名后缀要相同的限制，可以从任意页面获取 string 类型的数据。

后记

> 其它诸如中间件跨域，服务器代理跨域，Flash URLLoader跨域，动态创建script标签（简化版本的jsonp）不作讨论。

参考文章：