[**jsonp详解**](https://www.cnblogs.com/JinQuanLi/p/6551415.html)

json相信大家都用的多，jsonp我就一直没有机会用到，但也经常看到，只知道是“用来跨域的”，一直不知道具体是个什么东西。今天总算搞明白了。下面一步步来搞清楚jsonp是个什么玩意。

**同源策略**

首先基于安全的原因，浏览器是存在[同源策略](https://developer.mozilla.org/zh-CN/docs/JavaScript%E7%9A%84%E5%90%8C%E6%BA%90%E7%AD%96%E7%95%A5)这个机制的，同源策略阻止从一个源加载的文档或脚本获取或设置另一个源加载的文档的属性。看起来不知道什么意思，实践一下就知道了。

**1.随便建两个网页**

一个端口是2698，一个2701，按照定义它们是不同源的。

[](http://images.cnblogs.com/cnblogs_com/lemontea/201212/201212110107145414.png)

**2.用jQuery发起不同源的请求**

在2698端口的网页上添加一个按钮，Click事件随便发起两个向端口为2701域的请求。

$("#getOtherDomainThings").click(function () {

$.get("http://localhost:2701/Scripts/jquery-1.4.4.min.js", function (data) {

console.log(data)

})

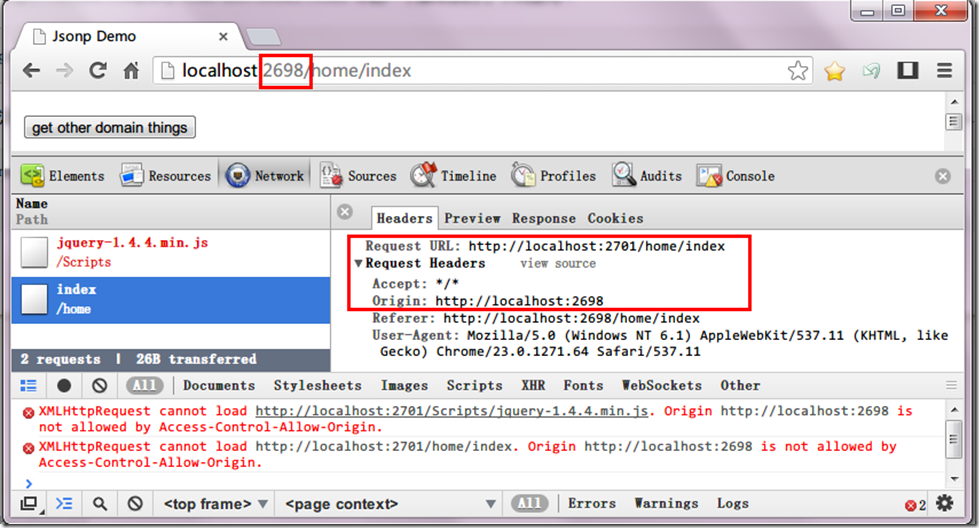
$.get("http://localhost:2701/home/index", function (data) {

console.log(data)

})

})

根据同源策略，很明显会悲剧了。浏览器会阻止，根本不会发起这个请求。（not allowed by Access-Control-Allow-Origin）

[](http://images.cnblogs.com/cnblogs_com/lemontea/201212/201212110107241500.png)

OK，原来jsonp是要解决这个问题的。

**script标签的跨域能力**

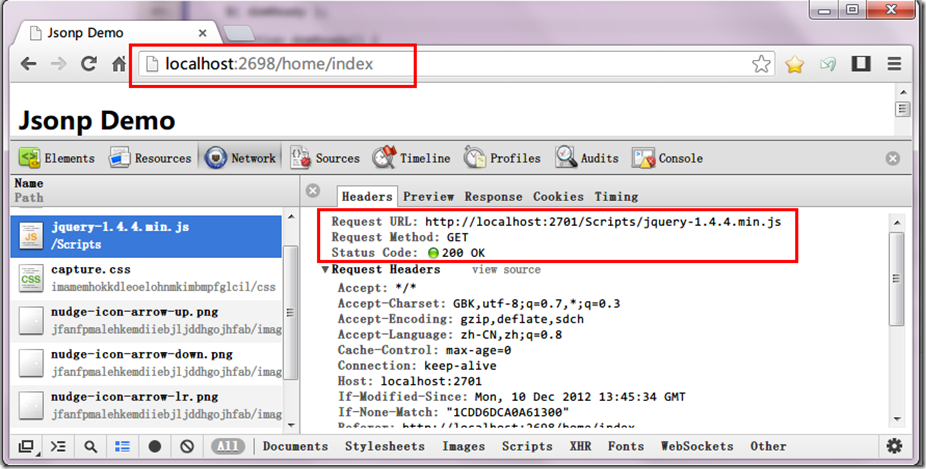
不知道大家知不知道CDN这个东西，例如[微软的CDN](http://www.asp.net/ajaxlibrary/cdn.ashx)，使用它，我们的网页可以不提供jQuery，由微软的网站帮我们提供：

<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.8.0.js" type="text/javascript"></script>

回到我们的2698端口的网页，上面我们在Click事件里有一个对2701端口域的jQuery文件的请求，这次使用script标签来请求。

<script type="text/javascript" src="http://localhost:2701/Scripts/jquery-1.4.4.min.js"></script>

当然，200，OK了

[](http://images.cnblogs.com/cnblogs_com/lemontea/201212/20121211010739821.png)

同样是端口2698的网页发起对2701域的请求，放在script里设置scr属性的OK了，另一个方式就悲剧。利用script的跨域能力，这就是jsonp的基础。

**利用script获取不同源的json**

既然它叫jsonp，很明显目的还是json，而且是跨域获取。根据上面的分析，很容易想到：利用js构造一个script标签，把json的url赋给script的scr属性，把这个script插入到dom里，让浏览器去获取。实践：

function CreateScript(src) {

$("<script><//script>").attr("src", src).appendTo("body")

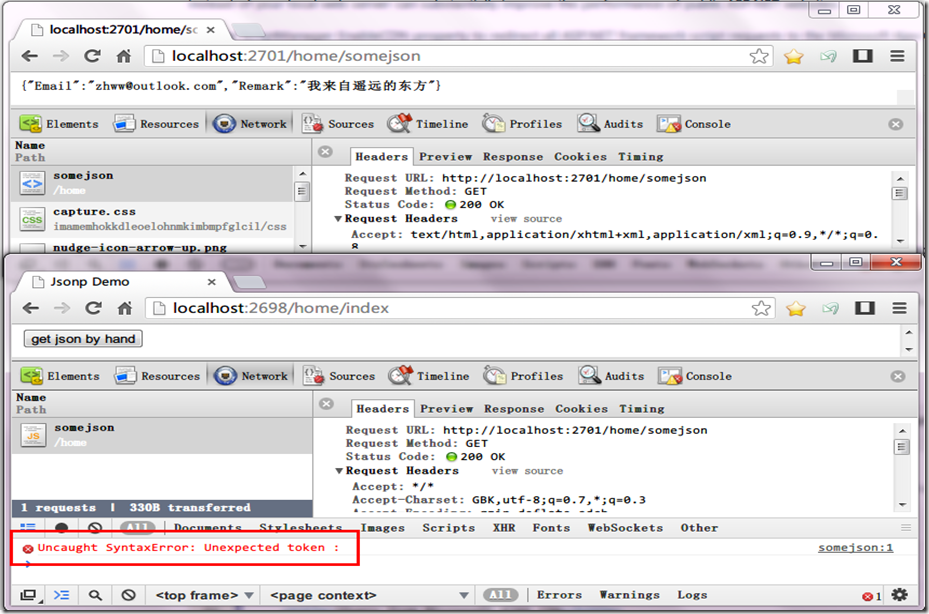
}

添加一个按钮事件来测试一下：

$("#getJsonByHand").click(function () {

CreateScript("http://localhost:2701/home/somejson")

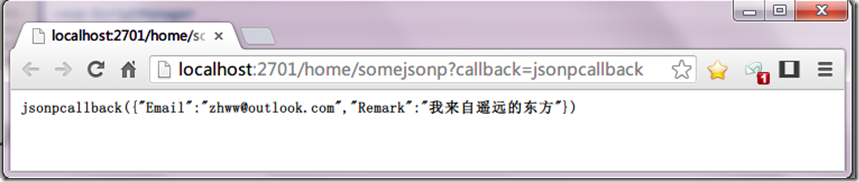
})

[](http://images.cnblogs.com/cnblogs_com/lemontea/201212/201212110107554328.png)

首先，第一个浏览器，http://localhost:2701/home/somejson这个Url的确是存在一个json的，而且在2698网页上用script标签来请求这个2701这个Url也是200OK的，但是最下面报js语法错误了。原来用script标签加载完后，会立即把响应当js去执行，很明显{"Email":"zhww@outlook.com","Remark":"我来自遥远的东方"}不是合法的js语句。

**利用script获取异域的jsonp**

显然，把上面的json放到一个回调方法里是最简单的方法。例如，变成这样：

[](http://images.cnblogs.com/cnblogs_com/lemontea/201212/201212110108065963.png)

如果存在jsonpcallback这个方法，那么jsonpcallback({"Email":"zhww@outlook.com","Remark":"我来自遥远的东方"})就是合法的js语句。

由于服务器不知道客户端的回调是什么，不可能hard code成jsonpcallback，所以就带一个QueryString让客户端告诉服务端，回调方法是什么，当然，QueryString的key要遵从服务端的约定，上面的是”callback“。

添加回调函数：

function jsonpcallback(json) {

console.log(json)

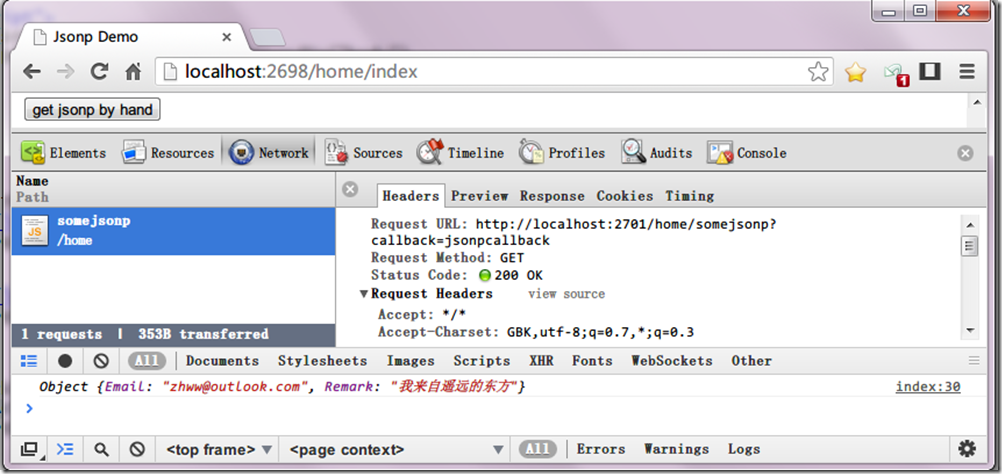
}

把前面的方法稍微改改参数：

$("#getJsonpByHand").click(function () {

CreateScript("http://localhost:2701/home/somejsonp?callback=jsonpcallback")

})

[](http://images.cnblogs.com/cnblogs_com/lemontea/201212/201212110108144789.png)

200OK，服务器返回jsonpcallback({"Email":"zhww@outlook.com","Remark":"我来自遥远的东方"})，我们也写了jsonpcallback方法，当然会执行。OK顺利获得了json。没错，到这里就是jsonp的全部。

**利用jQuery获取jsonp**

上面的方式中，又要插入script标签，又要定义一个回调，略显麻烦，利用jQuery可以直接得到想要的json数据，同样是上面的jsonp：

$("#getJsonpByJquery").click(function () {

$.ajax({

url: 'http://localhost:2701/home/somejsonp',

dataType: "jsonp",

jsonp: "callback",

success: function (data) {

console.log(data)

}

})

})

得到的结果跟上面类似。

**总结**

一句话就是利用script标签绕过同源策略，获得一个类似这样的数据，jsonpcallback是页面存在的回调方法，参数就是想得到的json。

jsonpcallback({"Email":"zhww@outlook.com","Remark":"我来自遥远的东方"})

到这里不免产生一个疑惑，同源策略是基于安全的原因建立的，发起不同源的请求是被认为不安全的，不可以接受的，但为什么这样绕一个小圈子就可以接受呢？这样就安全了吗？感觉就像洗澡时怕被人看见，把门关上，但侧面的大窗户却是打开的，这样做有意义吗？元芳你怎么看？