链滴

Jsonp 原理

作者：liumapp

原文链接：https://ld246.com/article/1509714970177

来源网站：链滴

核心内容转载自cnblogs博主:随它去吧的帖子，这篇博文看后感触很深，难得一见的好文。

定义

其实网上关于JSONP的讲解有很多，但却千篇一律，而且云里雾里，对于很多刚接触的人来讲理解起有些困难，小可不才，试着用自己的方式来阐释一下这个问题，看看是否有帮助。

● 一个众所周知的问题，Ajax直接请求普通文件存在跨域无权限访问的问题，甭管你是静态页面、动网页、web服务、WCF，只要是跨域请求，一律不准；

● 不过我们又发现，Web页面上调用js文件时则不受是否跨域的影响（不仅如此，我们还发现凡是拥有src"这个属性的标签都拥有跨域的能力，比如

<script>、<img>、<iframe>

● 于是可以判断，当前阶段如果想通过纯web端（ActiveX控件、服务端代理、属于未来的HTML5之ebsocket等方式不算）跨域访问数据就只有一种可能，那就是在远程服务器上设法把数据装进js格式文件里，供客户端调用和进一步处理；

● 恰巧我们已经知道有一种叫做JSON的纯字符数据格式可以简洁的描述复杂数据，更妙的是JSON还js原生支持，所以在客户端几乎可以随心所欲的处理这种格式的数据；

● 这样子解决方案就呼之欲出了，web客户端通过与调用脚本一模一样的方式，来调用跨域服务器上态生成的js格式文件（一般以JSON为后缀），显而易见，服务器之所以要动态生成JSON文件，目的在于把客户端需要的数据装入进去。

● 客户端在对JSON文件调用成功之后，也就获得了自己所需的数据，剩下的就是按照自己需求进行理和展现了，这种获取远程数据的方式看起来非常像AJAX，但其实并不一样。

● 为了便于客户端使用数据，逐渐形成了一种非正式传输协议，人们把它称作JSONP，该协议的一个点就是允许用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为数名来包裹住JSON数据，这样客户端就可以随意定制自己的函数来自动处理返回数据了。

如果对于callback参数如何使用还有些模糊的话，我们后面会有具体的实例来讲解。

案例

1.

不管jQuery也好，extjs也罢，又或者是其他支持jsonp的框架，他们幕后所做的工作都是一样的，下我来循序渐进的说明一下jsonp在客户端的实现。

我们知道，哪怕跨域js文件中的代码（当然指符合web脚本安全策略的），web页面也是可以无条件行的。

远程服务器remoteserver.com根目录下有个remote.js文件代码如下：

```
alert('我是远程文件');
```

本地服务器localserver.com下有个jsonp.html页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title></title>
    <script type="text/javascript" src="http://remoteserver.com/remote.js"></script>
</head>
<body>

</body>
</html>
```

毫无疑问，页面将会弹出一个提示窗体，显示跨域调用成功。

2. 现在我们在jsonp.html页面定义一个函数，然后在远程remote.js中传入数据进行调用。

jsonp.html页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
    var localHandler = function(data){
        alert('我是本地函数，可以被跨域的remote.js文件调用，远程js带来的数据是：' + data.result);
    };
    </script>
    <script type="text/javascript" src="http://remoteserver.com/remote.js"></script>
</head>
<body>

</body>
</html>
```

remote.js文件代码如下：

```
localHandler({"result":"我是远程js带来的数据"});
```

运行之后查看结果，页面成功弹出提示窗口，显示本地函数被跨域的远程js调用成功，并且还接收到远程js带来的数据。很欣喜，跨域远程获取数据的目的基本实现了，但是又一个问题出现了，我怎么远程js知道它应该调用的本地函数叫什么名字呢？毕竟是jsonp的服务者都要面对很多服务对象，而这服务对象各自的本地函数都不相同啊？我们接着往下看。

3.

聪明的开发者很容易想到，只要服务端提供的js脚本是动态生成的就行了呗，这样调用者可以传一个数过去告诉服务端“我想要一段调用XXX函数的js代码，请你返回给我”，于是服务器就可以按照客端的需求来生成js脚本并响应了。

看jsonp.html页面的代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/T
/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
```

```
    <script type="text/javascript">
    // 得到航班信息查询结果后的回调函数
    var flightHandler = function(data){
        alert('你查询的航班结果是：票价 ' + data.price + ' 元，' + '余票 ' + data.tickets + ' 张。');
    };
    // 提供jsonp服务的url地址（不管是什么类型的地址，最终生成的返回值都是一段javascript代
)
    var url = "http://flightQuery.com/jsonp/flightResult.aspx?code=CA1998&callback=flight
andler";
    // 创建script标签，设置其属性
    var script = document.createElement('script');
    script.setAttribute('src', url);
    // 把script标签加入head，此时调用开始
    document.getElementsByTagName('head')[0].appendChild(script);
    </script>
  </head>
  <body>

  </body>
  </html>
```

这次的代码变化比较大，不再直接把远程js文件写死，而是编码实现动态查询，而这也正是jsonp客户实现的核心部分，本例中的重点也就在于如何完成jsonp调用的全过程。

我们看到调用的url中传递了一个code参数，告诉服务器我要查的是CA1998次航班的信息，而callbac参数则告诉服务器，我的本地回调函数叫做flightHandler，所以请把查询结果传入这个函数中进行调。

OK，服务器很聪明，这个叫做flightResult.aspx的页面生成了一段这样的代码提供给jsonp.html（服端的实现这里就不演示了，与你选用的语言无关，说到底就是拼接字符串）：

```
flightHandler({
  "code": "CA1998",
  "price": 1780,
  "tickets": 5
});
```

我们看到，传递给flightHandler函数的是一个json，它描述了航班的基本信息。运行一下页面，成功出提示窗口，jsonp的执行全过程顺利完成

4. 到这里为止的话，相信你已经能够理解jsonp的客户端实现原理了吧？剩下的就是如何把代码封装下，以便于与用户界面交互，从而实现多次和重复调用。

什么？你用的是jQuery，想知道jQuery如何实现jsonp调用？好吧，那我就好人做到底，再给你一段jquery使用jsonp的代码（我们依然沿用上面那个航班信息查询的例子，假定返回jsonp结果不变）：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR
xhtml1/DTD/xhtml1-transitional.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml" >
  <head>
     <title>Untitled Page</title>
      <script type="text/javascript" src=jquery.min.js"></script>
      <script type="text/javascript">
     jQuery(document).ready(function(){
        $.ajax({
            type: "get",
```

```
            async: false,
            url: "http://flightQuery.com/jsonp/flightResult.aspx?code=CA1998",
            dataType: "jsonp",
            jsonp: "callback",//传递给请求处理程序或页面的，用以获得jsonp回调函数名的参数名(一
默认为:callback)
            jsonpCallback:"flightHandler",//自定义的jsonp回调函数名称，默认为jQuery自动生成的
机函数名，也可以写"?"，jQuery会自动为你处理数据
            success: function(json){
                alert('您查询到航班信息：票价： ' + json.price + ' 元，余票： ' + json.tickets + ' 张。')

            },
            error: function(){
                alert('fail');
            }
        });
    });
    </script>
    </head>
  <body>
  </body>
 </html>
```

是不是有点奇怪？为什么我这次没有写flightHandler这个函数呢？而且竟然也运行成功了！哈哈，这是jQuery的功劳了，jquery在处理jsonp类型的ajax时（还是忍不住吐槽，虽然jquery也把jsonp归入ajax，但其实它们真的不是一回事儿），自动帮你生成回调函数并把数据取出来供success属性方法调用，是不是很爽呀？

jsonp与json的异同

● ajax和jsonp这两种技术在调用方式上“看起来”很像，目的也一样，都是请求一个url，然后把服器返回的数据进行处理，因此jquery和ext等框架都把jsonp作为ajax的一种形式进行了封装；

● 但ajax和jsonp其实本质上是不同的东西。ajax的核心是通过XmlHttpRequest获取非本页内容，而jonp的核心则是动态添加

```
 <script>
```

标签来调用服务器提供的js脚本。

● 所以说，其实ajax与jsonp的区别不在于是否跨域，ajax通过服务端代理一样可以实现跨域，jsonp身也不排斥同域的数据的获取。

● 还有就是，jsonp是一种方式或者说非强制性协议，如同ajax一样，它也不一定非要用json格式来递数据，如果你愿意，字符串都行，只不过这样不利于用jsonp提供公开服务。

总而言之，jsonp不是ajax的一个特例，哪怕jquery等巨头把jsonp封装进了ajax，也不能改变着一点！