*jsonp 的跨域原理*

由于浏览器同源策略的限制，非同源下的请求，都会产生跨域问题，jsonp即是为了解决这个问题出现的一种简便解决方案。

同源策略即：同一协议，同一域名，同一端口号。当其中一个不满足时，我们的请求即会发生跨域问题。

举个简单的例子：

http://www.abc.com:3000到https://www.abc.com:3000的请求会跨域（域名、端口相同但协议不同）

http://www.abc.com:3000到http://www.abc.com:3001的请求会跨域（域名、协议相同但端口不同）

http://www.abc.com:3000到http://www.def.com:3000的请求会出现跨域（域名不同）

突破同源策略限制：

现在知道了同源策略，那我们就来看下jsonp是如何突破同源策略的限制实现跨域的

首先，不知道大家有没有注意，不管是我们的script标签的src还是img标签的src，或者说link标签的href他们没有被通源策略所限制，比如我们有可能使用一个网络上的图片，就可以请求得到

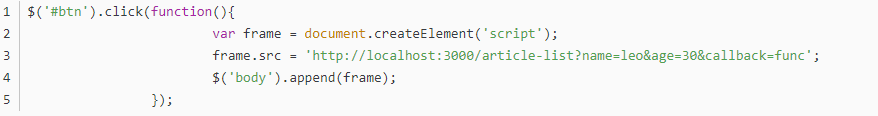
同理img标签的src和link标签的href也会发送一个get请求去请求静态资源。那么我们通过这点，是不是发现了点什么，这些标签的src和link属性，并没有受同源策略的限制。说到这里jsonp的实现原理就浮出水面了。

jsonp就是使用通源策略这一“漏洞”，实现的跨域请求（这也是jsonp跨域只能用get请求的原因所在）。

JSONP跨域实现：

根据上边所说的，我们要用过利用img、srcipt，link标签的src或href属性（到底使用那个标签无所谓）

来实现，那么我们如何做呢，我们来看一段简单的代码，为了方便，我这里使用jQuery：



可以看到，让我们点击按钮的时候，创建了一个script标签(即会发送一个get请求到src指向的地址)，src地址是"localhost:3000/article-list",这个src地址，就是我们请求的服务端接口。注意，这里我们有是那个参数，name,age和callback，name和age不说了，这跟我们平时普通的get请求参数无异。主要说下callback这个参数，callback参数就是核心所在。为什么要定义callback呢？首先我们知道，这个get请求已经被发出去了，那么我们如何接口请求回来的数据呢，callback=func则可以帮我们做这件事。我们继续看下边的代码



这里可以看到，我们声明了一个func函数，但没有执行，你可以想一下，如果服务端接口到get请求，返回的是**func({message:'hello'})，这样的话在服务端就可以把数据通过函数执行传参的方式实现数据传递了。**



这样下来，就完成了实现jsonp的跨域。

总结：

需要注意的是，callback参数定义的方法是需要前后端定义好的，具体什么名字，商讨好就可以了。其实jsonp的整个过程就类似于前端声明好一个函数，后端返回执行函数。执行函数参数中携带所需的数据，整个过程实际非常简单易懂，如有疑问大家可以在下方留言，我会及时回复。

*JS延时加载的方式*

