13. 描述css 的reset的用途

<http://www.jianshu.com/p/69ba47248774>

众所周知，现在所使用的主流浏览器对一些标签的默认属性上并没有做到统一，所以我们偶尔会发现，某个页面在chrome浏览器上很正常，到了firefox上面却有意想不到的偏差。当然编程人员不喜欢这样的兼容性问题的，而reset.css就是解决默认样式不兼容问题的办法之一。

14. 实现请求跨域的几种方式 请简述其原理

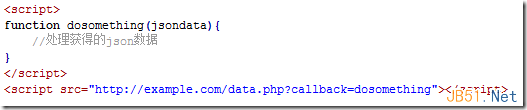
<http://www.jb51.net/article/48794.htm>

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**

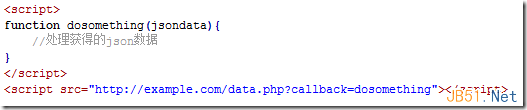
在js中，我们直接用XMLHttpRequest请求不同域上的数据时，是不可以的。但是，在页面上引入不同域上的js脚本文件却是可以的，jsonp正是利用这个特性来实现的。

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：

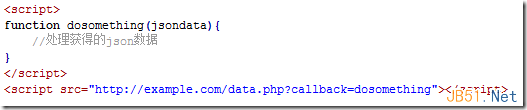


我们看到获取数据的地址后面还有一个callback参数，按惯例是用这个参数名，但是你用其他的也一样。当然如果获取数据的jsonp地址页面不是你自己能控制的，就得按照提供数据的那一方的规定格式来操作了。

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:



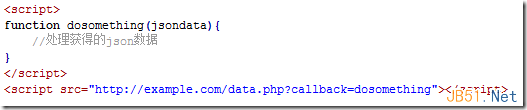
最终那个页面输出的结果是:



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。

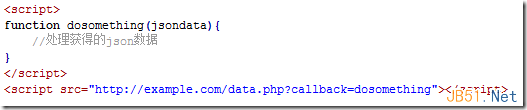
知道jsonp跨域的原理后我们就可以用js动态生成script标签来进行跨域操作了，而不用特意的手动的书写那些script标签。如果你的页面使用jquery，那么通过它封装的方法就能很方便的来进行jsonp操作了。



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。

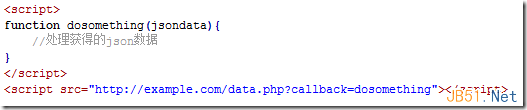
**2、通过修改document.domain来跨子域**

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：

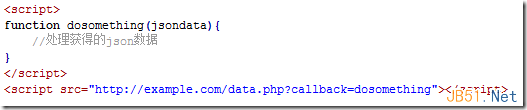
[](http://files.jb51.net/file_images/article/201404/201404081116576.png)

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:



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：

[](http://files.jb51.net/file_images/article/201404/201404081116578.png)

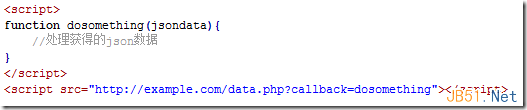
这样我们就可以通过js访问到iframe中的各种属性和对象了。

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。

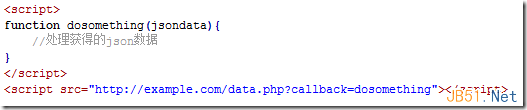
**3、使用window.name来进行跨域**

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。

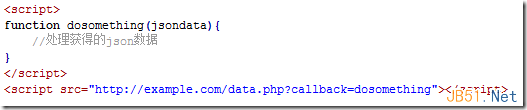
比如：有一个页面a.html,它里面有这样的代码：



再看看b.html页面的代码：



a.html页面载入后3秒，跳转到了b.html页面，结果为：



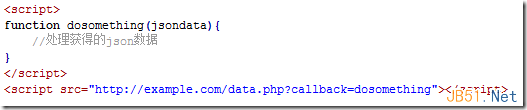
我们看到在b.html页面上成功获取到了它的上一个页面a.html给window.name设置的值。如果在之后所有载入的页面都没对 window.name进行修改的话，那么所有这些页面获取到的window.name的值都是a.html页面设置的那个值。当然，如果有需要，其中的 任何一个页面都可以对window.name的值进行修改。注意，window.name的值只能是字符串的形式，这个字符串的大小最大能允许2M左右甚 至更大的一个容量，具体取决于不同的浏览器，但一般是够用了。

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。

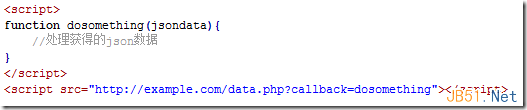
data.html页面里的代码很简单，就是给当前的window.name设置一个a.html页面想要得到的数据值。data.html里的代码：

[](http://files.jb51.net/file_images/article/201404/2014040811165712.png)

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：

[[](http://files.jb51.net/file_images/article/201404/2014040811165813.png)](http://files.jb51.net/file_images/article/201404/2014040811165813.png" \t "_blank)

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。

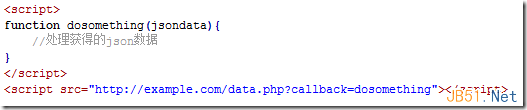
**4、使用HTML5中新引进的window.postMessage方法来跨域传送数据**

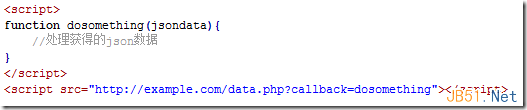
window.postMessage(message,targetOrigin)  方法是html5新引进的特性，可以使用它来向其它的window对象发送消息，无论这个window对象是属于同源或不同源，目前IE8+、 FireFox、Chrome、Opera等浏览器都已经支持window.postMessage方法。

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 \*  。

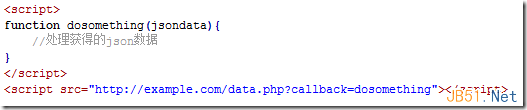
需要接收消息的window对象，可是通过监听自身的message事件来获取传过来的消息，消息内容储存在该事件对象的data属性中。

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

[](http://files.jb51.net/file_images/article/201404/2014040811165814.png)



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。

15. Css中 @import与link的区别

（1）**link**属于XHTML标签，而@import是CSS提供的; （2）页面被加载的时，**link**会同时被加载，而@import引用的CSS会等到页面被加载完再加载; （3）import只在IE5以上才能识别，而**link**是XHTML标签，无兼容问题; （4）**link**方式的样式的权重 高于@import的权重.

16. 常见的兼容性问题

**png24**位的图片在**iE6**浏览器上出现背景，解决方案是做成**PNG8**. \* 浏览器默认的**margin**和**padding**不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。 \* **IE6**双边距**bug**:块属性标签**float**后，又有横行的**margin**情况下，在**ie6**显示**margin**比设置的大。 浮动**ie**产生的双倍距离 **#box**{ float:left; width:10px; margin:0 0 0 100px;} 这种情况之下**IE**会产生20**px**的距离，解决方案是在**float**的标签样式控制中加入 ——\_**display**:inline;将其转化为行内属性。(\_这个符号只有**ie6**会识别) 渐进识别的方式，从总体中逐渐排除局部。 首先，巧妙的使用“\9”这一标记，将**IE**游览器从所有情况中分离出来。 接着，再次使用“+”将**IE8**和**IE7**、**IE6**分离开来，这样**IE8**已经独立识别。 **css** .bb{ background-color:#f1ee18;/\*所有识别\*/ .background-color:#00deff\9; /\*IE6、7、8识别\*/ +background-color:#a200ff;/\*IE6、7识别\*/ \_background-color:#1e0bd1;/\*IE6识别\*/ } \* **IE**下,可以使用获取常规属性的方法来获取自定义属性, 也可以使用**getAttribute**()获取自定义属性; **Firefox**下,只能使用**getAttribute**()获取自定义属性. 解决方法:统一通过**getAttribute**()获取自定义属性. \* **IE**下,**even**对象有**x**,**y**属性,但是没有**pageX**,**pageY**属性; **Firefox**下,**event**对象有**pageX**,**pageY**属性,但是没有**x**,**y**属性. \* 解决方法：（条件注释）缺点是在**IE**浏览器下可能会增加额外的**HTTP**请求数。 \* **Chrome** 中文界面下默认会将小于 12**px** 的文本强制按照 12**px** 显示, 可通过加入 **CSS** 属性 **-webkit-text-size-adjust**: **none**; 解决. 超链接访问过后**hover**样式就不出现了 被点击访问过的超链接样式不在具有**hover**和**active**了解决方法是改变**CSS**属性的排列顺序:**L-V-H-A** : **a**:link {} **a**:visited {} **a**:hover {} **a**:active {}