WEB跨域

基于浏览器的同源策略/SOP（Same origin policy），两个通讯地址的**协议、域名、端口号中有一者不同。两个地址的通**讯将被浏览器视为不安全的，并被block下来，即无法直接进行通讯，产生跨域问题。

同源策略是一种约定，它是浏览器最核心也最基本的安全功能，如果缺少了同源策略，浏览器很容易受到XSS、CSFR等攻击。下面是具体的会产生跨域的例子：

URL	说明	跨域
http://www.a.com/a.js http://www.a.com/b.js	同一域名下	否
http://www.a.com/lab/a.js http://www.a.com/script/b.js	同一域名下不同文件夹	否
http://www.a.com:8000/a.js http://www.a.com/b.js	同一域名，不同端口	是
http://www.a.com/a.js https://www.a.com/b.js	同一域名，不同协议	是
http://www.a.com/a.js http://70.32.92.74/b.js	域名和域名对应ip	是
http://www.a.com/a.js http://script.a.com/b.js	主域相同，子域不同	是
http://www.a.com/a.js http://a.com/b.js	同一域名，不同二级域名（同上）	是
http://www.cnblogs.com/a.js http://www.a.com/b.js	不同域名	是

跨域的几种常见解决方案

1. 通过iframe + document.domain跨域 (只有在主域相同的时候才能使用该方法)
2. 通过iframe + location.hash跨域
3. 通过iframe + window.name跨域
4. 通过iframe + postMessage跨域（H5跨域方案）
5. WebSocket跨域（H5跨域方案，服务端支持<需要支持webSocket的服务器>）
6. script标签不受跨域限制（AJAX的jsonp底层实现）
7. 通过CORS跨域（服务端支持<主要>）
8. 通过反向代理服务器跨域（如Nginx，服务端支持）

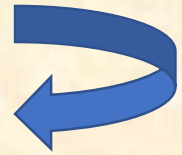
有一个页面，它的地址是http://www.damonare.cn/a.html， 在这个页面里面有一个iframe， 它的src是http://damonare.cn/b.html。通过iframe访问不同域的页面，是可以获取window对象的，但却无法获取相应的属性和方法，也无法互相通过js访问到，但是可以通过设置相同的document.domain来进行通讯。

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

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

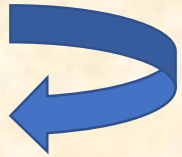


因为父窗口可以对iframe进行URL读写，iframe也可以读写父窗口的URL，URL有一部分被称为hash，就是#号及其后面的字符，它一般用于浏览器锚点定位，Server端并不关心这部分，应该说HTTP请求过程中不会携带hash，所以这部分的修改不会产生HTTP请求，但是会产生浏览器历史记录。

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。

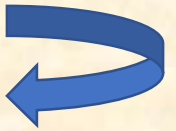
这个方式的通信会造成一些不必要的浏览器历史记录，而且有些浏览器不支持onhashchange事件，需要轮询来获知URL的改变，最后，这样做也存在缺点，诸如数据直接暴露在了url中，数据容量和类型都有限等。

A域名下a.html嵌入B域名下b.html，a.html通过hash把数据传到吧b.html#param，b.html通过url监听获取到hash值，处理完成后把response通过一个隐藏iframe放到A域名下的一个代理页面a2.html#response。因为a2.html和a.html是同一个域名，a2.html监听到hash值的变化，把hash值传到a.html的url里面。3个页面都需要监听url。



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。

```
<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>
```



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。

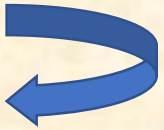
主页面：

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

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

子页面：

```
<script>
    window.name = '要传送的内容';
</script>
```



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

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

```
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);
}
```

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。

Header字段属性	说明
Access-Control-Allow-Origin	允许访问的origin
Access-Control-Expose-Headers	在跨域访问时，XMLHttpRequest对象的getResponseHeader()方法只能拿到一些最基本的响应头，Cache-Control、Content-Language、Content-Type、Expires、Last-Modified、Pragma，如果要访问其他头，则需要服务器设置本响应头
Access-Control-Max-Age	头指定了预检请求的结果能够被缓存多久
Access-Control-Allow-Credentials	为true时允许传cookies，但是Access-Control-Allow-Origin不允许为*
Access-Control-Allow-Methods	允许跨域请求的方法
Access-Control-Allow-Headers	首部字段用于预检请求的响应。其指明了实际请求中允许携带的首部字段，