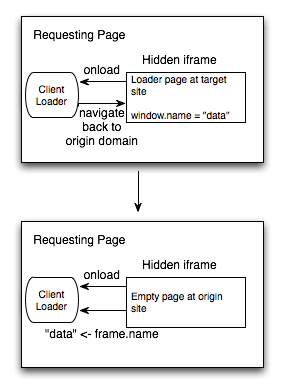
<http://www.planabc.net/2008/09/01/window_name_transport/>

**使用 window.name 解决跨域问题**

window.name 传输技术，原本是 [Thomas Frank](http://www.thomasfrank.se/about.html) 用于解决 cookie 的一些劣势（每个域名 4 x 20 Kb 的限制、数据只能是字符串、设置和获取 cookie 语法的复杂等等）而发明的（详细见原文：[《Session variables without cookies》](http://www.thomasfrank.se/sessionvars.html)），后来 [Kris Zyp](http://www.sitepen.com/blog/2008/07/22/windowname-transport/) 在此方法的基础上强化了 window.name 传输 ，并引入到了 [Dojo](http://dojotoolkit.org/)（[dojox.io.windowName](http://bugs.dojotoolkit.org/ticket/6893)），用来解决跨域数据传输问题。

window.name 的美妙之处：**name 值在不同的页面（甚至不同域名）加载后依旧存在，并且可以支持非常长的 name 值（2MB）。**

window.name 传输技术的基本原理和步骤为：



name 在浏览器环境中是一个全局/window对象的属性，且当在 frame 中加载新页面时，name 的属性值依旧保持不变。通过在 iframe 中加载一个资源，该目标页面将设置 frame 的 name 属性。此 name 属性值可被获取到，以访问 Web 服务发送的信息。但 name 属性仅对相同域名的 frame 可访问。这意味着为了访问 name 属性，当远程 Web 服务页面被加载后，必须导航 frame 回到原始域。同源策略依旧防止其他 frame 访问 name 属性。一旦 name 属性获得，销毁 frame 。

在最顶层，name 属性是不安全的，对于所有后续页面，设置在 name 属性中的任何信息都是可获得的。然而 windowName 模块总是在一个 iframe 中加载资源，并且一旦获取到数据，或者当你在最顶层浏览了一个新页面，这个 iframe 将被销毁，所以其他页面永远访问不到 window.name 属性。

基本实现代码，基于 [YUI](http://developer.yahoo.com/yui/)，源自[克军写的样例](http://hikejun.com/demo/windowname/demo_windowname.html)：

(function(){  
    var YUD = YAHOO.util.Dom, YUE = YAHOO.util.Event;  
  
    dataRequest = {  
        \_doc: document,  
        cfg: {  
            proxyUrl: 'proxy.html'  
        }  
    };  
  
    dataRequest.send = function(sUrl, fnCallBack){  
        if(!sUrl || typeof sUrl !== 'string'){  
            return;  
        }  
  
        sUrl += (sUrl.indexOf('?') > 0 ? '&' : '?') + 'windowname=true';  
  
        var frame = this.\_doc.createElement('iframe'), state = 0, self = this;  
        this.\_doc.body.appendChild(frame);  
        frame.style.display = 'none';  
  
        var clear = function(){  
            try{  
                frame.contentWindow.document.write('');  
                frame.contentWindow.close();  
                self.\_doc.body.removeChild(frame);  
            }catch(e){}  
        };  
  
        var getData = function(){  
            try{  
                var da = frame.contentWindow.name;  
            }catch(e){}  
            clear();  
            if(fnCallBack && typeof fnCallBack === 'function'){  
                fnCallBack(da);  
            }  
        };  
  
        YUE.on(frame, 'load', function(){  
            if(state === 1){  
                getData();  
            } else if(state === 0){  
                state = 1;  
                frame.contentWindow.location = self.cfg.proxyUrl;  
            }  
        });  
  
        frame.src = sUrl;  
    };  
})();

**Web 服务器如何提供 window.name 数据**

为了让 Web 服务器实现 window.name，服务器应该只寻找请求中是否包含 windowname 参数。如果包含了 windowname 参数，服务器应该返回一个设置了 window.name 字符串值的 HTML 文档，回应此请求并传送到客户端。例如：  
http://www.planabc.net/getdata.html?windowname=true

如果服务器想用 Hello 响应客服端，它应该返回一个 HTML 页面：

<html>  
    <script type="text/javascript">  
        window.name="Hello";  
    </script>  
</html>

同样也可以转换为 JSON 数据：

<html>  
    <script type="text/javascript">  
        window.name='{"foo":"bar"}';  
    </script>  
</html>

如果你手动创建资源，书写大量的多行的 JSON 对象为一个引用的字符串应该是比较困难的并且易于出错的。可以使用这样的 HTML 样例简单的创建 JSON 数据，将会转换为一个 JSON 字符串而无需手动转义 JSON 为字符串：

<html>  
    <script type="\'text/javascript\'">  
        window.name = document.getElementsByTagName("script")[0].innerHTML.match(/temp\s\*=([\w\W]\*)/)[1];  
        temp= {  
            foo:"bar", // put json data here  
            baz:"foo"  
        }  
    </script>  
</html>

同样的，如果你想传递 HTML/XML 数据，这里有一个样例实现，而无需手动将这些数据转换成字符串：

<html>  
    <body>  
        <p id="content">  
            some <strong>html/xml-style</strong>data  
        </p>  
    </body>  
    <script type="text/javascript">  
        window.name = document.getElementById("content").innerHTML;  
    </script>  
</html>

window.name 传输技术相比其他的跨域传输的一些优势：

1. 它是安全的。也就是说，它和其他的基于安全传输的 frame 一样安全，例如 Fragment Identifier messaging （FIM）和 [Subspace](http://research.microsoft.com/~helenw/papers/subspace.pdf)。(I)Frames 也有他们自己的安全问题，由于 frame 可以改变其他 frame 的 location，但是这个是非常不同的安全溢出，通常不太严重。
2. 它比 FIM 更快，因为它不用处理小数据包大小的 Fragment Identifier ，并且它不会有更多的 IE 上的“机关枪”声音效果。它也比 Subspace 快，Subspace 需要加载两个 Iframe 和两个本地的 HTML 文件来处理一个请求。window.name 仅需要一个 Iframe 和一个本地文件。
3. 它比 FIM 和 Subspace 更简单和安全。FIM 稍微复杂，而 Subspace 非常复杂。Subspace 也有一些额外的限制和安装要求，如预先声明所有的目标主机和拥有针对若干不同特殊主机的 DNS 入口。window.name 非常简单和容易使用。
4. 它不需要任何插件（比如 Flash）或者替代技术（例如 Java）。