# 两个静态html页面传值方法的总结

## 问题

因最近尝试实现客户端与服务端分离，服务端只提供接口，客户端用html+js实现，分成两个独立的项目部署，因项目是个人项目，客户端展示不考虑使用像Angular、Vue、Native这种前端框架实现，于是全部使用静态页面实现，请求数据用ajax，在开发的过程中，就遇到了页面之间传值的问题。

## 解决方法

post提交表单跳转的方式肯定不行了，因服务端并不处理这些请求参数，连到达客户端的机会都没有，此时只能考虑get方式了，因get方式参数在url上，跳转后的页面肯定能获取到。还有一种思路，就是将参数存到一个公共的地方，页面都可以获取到，按这两种思路，总结了如下的解决方法。

### get方式，url携带参数

可以通过window.location.search获取url上的参数。如下面的示例。

a.html

<a href="./b.html?a=1&a=2&username=zhangsan&password=123&=">跳转</a>

b.html

<script type="text/javascript" src="./js/getUrlParam.js"></script>

<script>  
 **var** a = UrlParam.paramValues("a");  
 console.log(a);  
</script>

getUrlParam.js

UrlParam = **function**() { // url参数  
 **var** data, index;   
 (**function** init() {   
 data = []; //值，如[["1","2"],["zhangsan"],["lisi"]]  
 index = {}; //键:索引，如{a:0,b:1,c:2}  
 **var** u = window.location.search.substr(1);   
 **if** (u != '') {   
 **var** params = decodeURIComponent(u).split('&');  
 **for** (**var** i = 0, len = params.length; i < len; i++) {  
 **if** (params[i] != '') {  
 **var** p = params[i].split("=");  
 **if** (p.length == 1 || (p.length == 2 && p[1] == '')) {// p | p= | =  
 data.push(['']);   
 index[p[0]] = data.length - 1;   
 } **else if** (**typeof**(p[0]) == 'undefined' || p[0] == '') { // =c 舍弃  
 **continue**;  
 } **else if** (**typeof**(index[p[0]]) == 'undefined') { // c=aaa   
 data.push([p[1]]);   
 index[p[0]] = data.length - 1;   
 } **else** {// c=aaa   
 data[index[p[0]]].push(p[1]);   
 }   
 }   
 }   
 }   
 })();   
 **return** {   
 // 获得参数,类似request.getParameter()   
 param : **function**(o) { // o: 参数名或者参数次序  
 **try** {   
 **return** (**typeof**(o) == 'number' ? data[o][0] : data[index[o]][0]);   
 } **catch** (e) {   
 }   
 },   
 //获得参数组, 类似request.getParameterValues()   
 paramValues : **function**(o) { // o: 参数名或者参数次序  
 **try** {   
 **return** (**typeof**(o) == 'number' ? data[o] : data[index[o]]);   
 } **catch** (e) {}   
 },   
 //是否含有paramName参数  
 hasParam : **function**(paramName) {  
 **return typeof**(paramName) == 'string' ? **typeof**(index[paramName]) != 'undefined' : **false**;  
 },   
 // 获得参数Map ,类似request.getParameterMap()   
 paramMap : **function**() {  
 **var** map = {};   
 **try** {   
 **for** (**var** p **in** index) { map[p] = data[index[p]]; }   
 } **catch** (e) {}   
 **return** map;   
 }   
 }   
}();

此处代码参照：

<http://blog.csdn.net/caoyuan10036/article/details/7227214>

### 通过cookie，传递

cookie能够存储少量数据到客户端的磁盘中，特定的网页之间是可以共享cookie中的数据。

a.html

<script type="text/javascript" src="./js/jquery.min.js"></script>  
<script type="text/javascript" src="./js/jquery.cookie.js"></script>

<script>  
 $.cookie("a","12");  
</script>

b.html

<script type="text/javascript" src="./js/jquery.min.js"></script>  
<script type="text/javascript" src="./js/jquery.cookie.js"></script>

<script>

**var** param = $.cookie("a");  
 console.log(param);  
</script>

此处使用了jquery.cookie.js,具体用法可参照此篇文章：

<http://blog.csdn.net/csdn_ds/article/details/78022177>

### window.open和window.opener之间传值

window.open可以打开一个新的页面，在新的页面中可以通过window.opener获取父页面的窗口对象，从而可以获取父窗口中的参数。

a.html

<button id="open">打开新的页面</button>

<script>  
 $('#open').click(**function** () {  
 window.open('./b.html')  
 })  
</script>

b.html

<script>  
 **var** content = window.opener.document.getElementById("open").innerHTML;  
 alert(content);  
</script>

### **h5技术，window.localStorage存储数据**

在HTML5中，新加入了一个localStorage特性，这个特性主要是用来作为本地存储来使用的，解决了cookie存储空间不足的问题(cookie中每条cookie的存储空间为4k)，localStorage中一般浏览器支持的是5M大小，这个在不同的浏览器中localStorage会有所不同。此方法类似cookie，将数据存在一个公共的地方，实现页面之间传值。

a.html

<input type="text" name="username" />  
<input type="button" name="" value="post" onclick="set()"/>

<script>  
 **function** set() {  
 //由于是一个新的技术，你可以通过下面的代码检测你的浏览器是否支持  
 **if** (window.localStorage) {  
 //存储变量的值  
 localStorage.name = document.all.username.value;  
 location.href = './b.html';  
 } **else** {  
 alert("NOT SUPPORT");  
 }  
 }  
</script>

b.html

<script>  
 **var** value = localStorage["name"];  
 alert(value);  
</script>

## 总结

对于不同的解决方法，都有优缺点

1. url携带参数

优点：取值方便，可以跨域，利于页面分享，没有环境限制。

缺点：url携带参数值的长度有限制。

1. cookie方式

优点：可以在同源内的的任意网页中访问，存储数据的周期可以自由设置。

缺点：有长度限制。

1. 设置窗口之间的父子关联关系

优点：取值方便.只要window.opener指向父窗口,就可以访问所有对象.不仅可以访问值,还可以访问父窗口的方法.值长度无限制。  
缺点:两窗口要存在着关系.就是利用window.open打开的窗口。不能跨域。

1. h5技术，window.localStorage存储数据

优点：储存空间大，有5M存储空间。

缺点：不是所有浏览器都支持。

个人倾向第一种方式，主要是自己做的网页可以分享，在任何的地方都可以打开，其他的方式都有环境的要求。如果是做大型项目,架构是实现客户端与服务端的分离，建议还是引入客户端框架，框架实现了路由、参数的传递、以及安全问题，可以大大的提高开发效率。