jQuery中使用Ajax

# Ajax的优势和不足

Ajax中的优势：不需要插件支持，只需要允许javascript的浏览器上都能支持；优秀的用户体验；提高web程序的性能；Ajax工作在用户和服务器之间的一个中间层，有效的减轻服务器和宽带的负担。

Ajax中的不足：破坏浏览器前进和后退的正常功能；对搜索引擎的支持不足；

# JQuery中的Ajax

## load()方法

载入远程HTML文件代码并插入至DOM中，默认使用GET方式，传递参数时自动转换为POST方式。load(url,[data],[callback])，参数说明如下：

* url:要载入的远程url地址
* data:发送至服务器的key/value 数据
* callback:请求完成时的回调函数（无论成功还是失败）

示例代码如下：（load方法通常是取静态文件）

|  |
| --- |
| //无参数、无回调函数  $("#showload").load("load.htm");  //无回调函数  $("#showload").load("load.htm", {  "para": "para-value"  });  $("#showload").load("load.htm", {  "para": "para-value"  },  function(responseText,textStatus,XMLHttpRequest) {  // textStatus请求状态有success,error,notmodified,timeout  //处理  }); |

## $.get()方法

$.get(url, [data], [callback],[type])：使用get方式从服务器端获取数据。

参数如下：

* url：发送请求的URL地址
* data：要发送给服务器的数据
* callback：载入成功时回调函数，只有返回状态为success的时候
* type：服务器返回内容格式，xml,html,script,json,text

示例代码如下：

|  |
| --- |
| $("#send").click(function(){  $.get("get3.jsp", {  username: encodeURI($("#username").val()),  content: encodeURI($("#content").val())  }, function(data, textStatus) {  var username = data.username;  var content = data.content;  username = decodeURI(username);  content = decodeURI(content);  var txtHtml = "<div class='comment'><h6>" + username + ":</h6><p class='para'>" + content + "</p></div>";  $("#resText").html(txtHtml); // 把返回的数据添加到页面上  }, "json"); |

## $.post()方法

$.post(url, [data], [callback])：使用POST方式来进行异步请求。同jQuery.get()相比，差别在于请求的方式。

|  |
| --- |
| $.post("post3.jsp", {  username : $("#username").val() ,  content : $("#content").val()  }, function (data, textStatus){  var username = data.username;  var content = data.content;  var txtHtml = "<div class='comment'><h6>"+username+":</h6><p class='para'>"+content+"</p></div>";  $("#resText").html(txtHtml); // 把返回的数据添加到页面上  },"json"); |

## $.getScript()方法

$.getScript(url,[callback])：通过 GET 方式请求载入并执行一个JavaScript文件。也是jQuery.ajax的一种简单的使用方法。没有必要初次加载的js文件可以使用这种方式来延迟加载。

## $.getJson()方法

$.getJSON(url,[data],[callback])：通过get方式获取json格式的数据。

|  |
| --- |
| $('#send').click(function() {  $.getJSON("http://api.flickr.com/services/feeds/photos\_public.gne?tags=car&tagmode=any&format=json&jsoncallback=?",  function(data) {  $.each(data.items, function(i, item) {  $("<img class='para'/> ").attr("src", item.media.m).appendTo("#resText");  if (i == 3) {  return false;  }  });  }  );  }) |

## $.ajax()方法

$.ajax方法是jQuery最底层的Ajax实现。结构：$.ajax(options)，所有参数都是可选的，具体如下：

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| **url** | String | (默认: 当前页地址) 发送请求的地址。 |
| **type** | String | 请求方式 ("POST" 或 "GET")， 默认为 "GET"。注意：其它 HTTP 请求方法，如 PUT 和 DELETE 也可以使用，但仅部分浏览器支持。 |
| **timeout** | Number | 设置请求超时时间（毫秒）。此设置将覆盖全局设置。 |
| **async** | Boolean | (默认: true) 默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。 |
| **beforeSend** | Function | 发送请求前可修改 XMLHttpRequest 对象的函数，如添加自定义 HTTP 头。XMLHttpRequest 对象是唯一的参数。  function (XMLHttpRequest) {  this; // the options for this ajax [request](http://www.php100.com/tags.php/request/)} |
| **cache** | Boolean | (默认: true) jQuery 1.2 新功能，设置为 false 将不会从浏览器缓存中加载请求信息。 |
| **complete** | Function | 请求完成后回调函数 (请求成功或失败时均调用)。参数： XMLHttpRequest 对象，成功信息字符串。  function (XMLHttpRequest, textStatus) {  this; // the options for this ajax request} |
| **contentType** | String | (默认**: "application/x-www-form-urlencoded")** 发送信息至服务器时内容编码类型。默认值适合大多数应用场合。  **application/json** ：告诉服务端消息主体是序列化后的 JSON 字符串。  **multipart/form-data** ：这种方式一般用来上传文件 |
| **data** | Object, String | 发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称。如 {foo:["bar1", "bar2"]} 转换为 '&foo=bar1&foo=bar2'。 |
| **dataType** | String | 预期服务器返回的数据类型。如果不指定，jQuery 将自动根据 HTTP 包 MIME 信息返回 responseXML 或 responseText，并作为回调函数参数传递，可用值:  "xml": 返回 XML 文档，可用 jQuery 处理。  "html": 返回纯文本 HTML 信息；包含 script 元素。  "script": 返回纯文本 JavaScript 代码。不会自动缓存结果。  "json": 返回 JSON 数据 。  "jsonp": JSONP 格式。使用 JSONP 形式调用函数时，如 "myurl?callback=?" jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。 |
| **error** | Function | (默认: 自动判断 (xml 或 html)) 请求失败时将调用此方法。这个方法有三个参数：XMLHttpRequest 对象，错误信息，（可能）捕获的错误对象。  function (XMLHttpRequest, textStatus, errorThrown) { // 通常情况下textStatus和errorThown只有其中一个有值 this; // the options for this ajax request} |
| **global** | Boolean | (默认: true) 是否触发全局 AJAX 事件。设置为 false 将不会触发全局 AJAX 事件，如 ajaxStart 或 ajaxStop 。可用于控制不同的Ajax事件 |
| **ifModified** | Boolean | (默认: false) 仅在服务器数据改变时获取新数据。使用 HTTP 包 Last-Modified 头信息判断。 |
| **processData** | Boolean | (默认: true) 默认情况下，发送的数据将被转换为对象(技术上讲并非字符串) 以配合默认内容类型 "application/x-www-form-urlencoded"。如果要发送 DOM 树信息或其它不希望转换的信息，请设置为 false。 |
| **success** | Function | 请求成功后回调函数。这个方法有两个参数：服务器返回数据，返回状态  function (data, textStatus) { // data could be xmlDoc, jsonObj, html, text, etc... this; // the options for this ajax request} |

使用举例：

|  |
| --- |
| $('#send').click(function() {  $.ajax({  type: "GET",  url: "http://api.flickr.com/services/feeds/photos\_public.gne?tags=car&tagmode=any&format=json&jsoncallback=?",  dataType: "jsonp",  success : function(data){  $.each(data.items, function( i,item ){  $("<img class='para'/> ").attr("src", item.media.m ).appendTo("#resText");  if ( i == 3 ) {  return false;  }  });  }  });  }); |
| $('#send').click(function() {  $.ajax({  type: "GET",  url: "test.json",  dataType: "json",  success: function(data) {  $('#resText').empty();  var html = '';  $.each(data, function(commentIndex, comment) {  html += '<div class="comment"><h6>'  + comment['username']  + ':</h6><p class="para">'  + comment['content']  + '</p></div>';  })  $('#resText').html(html);  }  });  }); |

# 序列化元素

使用Ajax异步提交表单，可以防止整个浏览器被刷新

* serialize()方法：表单元素的名称和值序列化为字符串数据，返回值类似于single=1&multiple=2
* serializeArray()方法：将表单元素的名称和值序列化为json数据
* param()方法：创建一个序列化的数组或对象，适用于一个URL地址查询字符串或Ajax请求。是serialize()的核心方法

|  |
| --- |
| var obj={a:1，b:2}  var k = $.param(obj);//a=1&b=2 |
| #form表单元素序列化为json对象  var fields = $("select, :radio").serializeArray();  var o={};  jQuery.each(fields, function(i, fields){  if(o[this.name]){  if(!o[this.name].push){  o[this.name]=[o[this.name]];  }  o[this.name].push(this.value || ''); // 将值插入o[label]  }else{  o[this.name]=this.value || ''; // 第一次在o中插入o[label]  }  });  $("#results").append(JSON.stringify(o));  console.log(o); |

# javascript中的跨域访问

第一种方式：使用代理服务器。

第二种方式：使用jsonp，它只能使用get方法。

第三种方式：使用XHL2