

jQuery Ajax 全解析

 本文地址: [jQuery Ajax 全解析](#)

本文作者: [QLeelulu](#)

转载请标明出处!

jQuery 确实是一个挺好的轻量级的 JS 框架,能帮助我们快速的开发 JS 应用,并在一定程度上改变了我们写 JavaScript 代码的习惯。

废话少说,直接进入正题,我们先来看一些简单的方法,这些方法都是对 `jQuery.ajax()` 进行封装以方便我们使用的方法,当然,如果要处理复杂的逻辑,还是需要用到 `jQuery.ajax()` 的(这个后面会说到)。

1. load(url, [data], [callback]) : 载入远程 HTML 文件代码并插入至 DOM 中。

url (String) : 请求的 HTML 页的 URL 地址。

data (Map) : (可选参数) 发送至服务器的 key/value 数据。

callback (Callback) : (可选参数) 请求完成时(不需要是 **success** 的)的回调函数。

这个方法默认使用 GET 方式来传递的,如果 `[data]` 参数有传递数据进去,就会自动转换为 POST 方式的。jQuery 1.2 中,可以指定选择符,来筛选载入的 HTML 文档,DOM 中将仅插入筛选出的 HTML 代码。语法形如 `"url #some > selector"`。

这个方法可以很方便的动态加载一些 HTML 文件,例如表单。

示例代码:

```
$(".ajax.load").load("http://www.cnblogs.com/QLeelulu/archive/2008/03/30/1130270.html .post",  
    function (responseText, textStatus, XMLHttpRequest) {  
        this;//在这里 this 指向的是当前的 DOM 对象,即  
        $(".ajax.load")[0]  
        //alert(responseText);//请求返回的内容  
        //alert(textStatus);//请求状态: success, error  
        //alert(XMLHttpRequest);//XMLHttpRequest 对象
```

```
});
```

这里将显示结果。

注：不知道为什么 URL 写绝对路径在 FF 下会出错，知道的麻烦告诉下。下面的 `get()`和 `post()` 示例使用的是绝对路径，所以在 FF 下你将会出错并不会看到返回结果。还有 `get()`和 `post()` 示例都是跨域调用的，发现传上来后没办法获取结果，所以把运行按钮去掉了。

2. `jQuery.get(url, [data], [callback])`：使用 GET 方式来进行异步请求

参数：

url (String)：发送请求的 URL 地址。

data (Map)：(可选) 要发送给服务器的数据，以 Key/value 的键值对形式表示，会做为 QueryString 附加到请求 URL 中。

callback (Function)：(可选) 载入成功时回调函数(只有当 Response 的返回状态是 success 才是调用该方法)。

这是一个简单的 GET 请求功能以取代复杂 `$.ajax` 。请求成功时可调用回调函数。如果需要
在出错时执行函数，请使用 `$.ajax`。示例代码：

```
$.get("./Ajax.aspx", {Action:"get",Name:"lulu"},
function (data, textStatus) {
    //返回的 data 可以是 xmlDoc, jsonObj,
    html, text, 等等.

    this; // 在这里 this 指向的是 Ajax 请求的
    选项配置信息，请参考下图

    alert(data);
    //alert(textStatus); //请求状态：
    success, error 等等。
```


当然这里捕捉不到 error，因为 error 的时候根本不会运行该回调函数

```
        //alert(this);  
    });
```

点击发送请求：

jQuery.get()回调函数里面的 this，指向的是 Ajax 请求的选项配置信息：



3. jQuery.post(url, [data], [callback], [type])：使用 POST 方式来进行异步请求

参数：

url (String)：发送请求的 URL 地址。

data (Map)：(可选) 要发送给服务器的数据，以 Key/value 的键值对形式表示。

callback (Function)：(可选) 载入成功时回调函数(只有当 Response 的返回状态是 success 才是调用该方法)。

type (String)：(可选)官方的说明是：Type of data to be sent。其实应该为客户端请求的类型(JSON,XML,等等)

这是一个简单的 POST 请求功能以取代复杂 \$.ajax。请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 \$.ajax。示例代码：

Ajax.aspx:

```
Response.ContentType = "application/json";
```

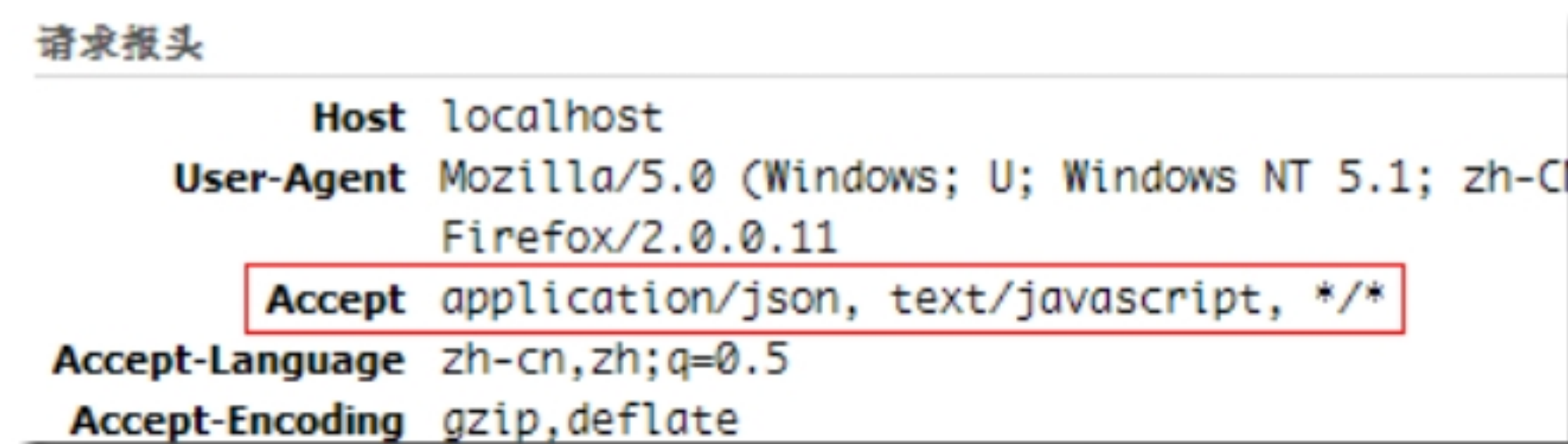
```
Response.Write("{result: '" + Request["Name"] + "', 你好! (这消息来自服务器)'}");
```

jQuery 代码:

```
$.post("Ajax.aspx", { Action: "post", Name: "lulu" },  
    function (data, textStatus) {  
        // data 可以是 xmlDoc, jsonObj, html, text, 等等.  
        //this; // 这个 Ajax 请求的选项配置信息, 请参考  
        //jQuery.get() 说到的 this  
        alert(data.result);  
    }, "json");
```

点击提交:

这里设置了请求的格式为"json":



请求报头

Host	localhost
User-Agent	Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.2.13) Gecko/20100308 Firefox/2.0.0.11
Accept	application/json, text/javascript, */*
Accept-Language	zh-cn,zh;q=0.5
Accept-Encoding	gzip,deflate

如果你设置了请求的格式为"json", 此时你没有设置 Response 回来的 ContentType 为:

Response.ContentType = "application/json"; 那么你将无法捕捉到返回的数据。

注意一下, alert(data.result); 由于设置了 Accept 报头为"json", 这里返回的 data 就是一个对象, 并不需要用 eval()来转换为对象。

4. jQuery.getScript(url, [callback]) : 通过 GET 方式请求载入并执行一个 JavaScript 文件。

参数

url (String) : 待载入 JS 文件地址。

callback (Function) : (可选) 成功载入后回调函数。

jQuery 1.2 版本之前, `getScript` 只能调用同域 JS 文件。1.2 中, 您可以跨域调用 JavaScript 文件。注意: Safari 2 或更早的版本不能在全局作用域中同步执行脚本。如果通过 `getScript` 加入脚本, 请加入延时函数。

这个方法可以用在例如当只有编辑器 `focus()` 的时候才去加载编辑器需要的 JS 文件. 下面看一些示例代码:

加载并执行 `test.js`。

jQuery 代码:

```
$.getScript("test.js");
```

加载并执行 `AjaxEvent.js` , 成功后显示信息。

jQuery 代码:

```
$.getScript("AjaxEvent.js", function() {  
    alert("AjaxEvent.js 加载完成并执行完成. 你再点击上面的  
Get 或 Post 按钮看看有什么不同?");  
});
```

加载完后请重新点击一下上面的 `Load` 请求看看有什么不同。

jQuery Ajax 事件

Ajax 请求会产生若干不同的事件, 我们可以订阅这些事件并在其中处理我们的逻辑。在 jQuery 这里有两种 Ajax 事件: 局部事件 和 全局事件。

局部事件就是在每次的 Ajax 请求时在方法内定义的, 例如:

```
$.ajax({  
    beforeSend: function() {  
        // Handle the beforeSend event
```

```

    },
    complete: function() {
        // Handle the complete event
    }
    // ...
});

```

全局事件是每次的 Ajax 请求都会触发的,它会向 DOM 中的所有元素广播,在上面 `getScript()` 示例中加载的脚本就是全局 Ajax 事件。全局事件可以如下定义:

```

$("#loading").bind("ajaxSend", function() {
    $(this).show();
}).bind("ajaxComplete", function() {
    $(this).hide();
});

```

或者:

```

$("#loading").ajaxStart(function() {
    $(this).show();
});

```

我们可以在特定的请求将全局事件禁用,只要设置下 `global` 选项就可以了:

```

$.ajax({
    url: "test.html",
    global: false, // 禁用全局 Ajax 事件.
    // ...
});

```

下面是 jQuery 官方给出的完整的 Ajax 事件列表:

- **ajaxStart** (Global Event)

This event is broadcast if an Ajax request is started and no other Ajax requests are currently running.

- **beforeSend** (Local Event)

This event, which is triggered before an Ajax request is started, allows you to modify the XMLHttpRequest object (setting additional headers, if need be.)

- **ajaxSend** (Global Event)

This global event is also triggered before the request is run.

- **success** (Local Event)

This event is only called if the request was successful (no errors from the server, no errors with the data).

- **ajaxSuccess** (Global Event)

This event is also only called if the request was successful.

- **error** (Local Event)

This event is only called if an error occurred with the request (you can never have both an error and a success callback with a request).

- **ajaxError** (Global Event)

This global event behaves the same as the local error event.

- **complete** (Local Event)

This event is called regardless of if the request was successful, or not. You will always receive a complete callback, even for synchronous requests.

- **ajaxComplete** (Global Event)

This event behaves the same as the complete event and will be triggered every time an Ajax request finishes.

- **ajaxStop** (Global Event)

This global event is triggered if there are no more Ajax requests being processed.

具体的全局事件请参考 **API** 文档。

好了，下面开始说 jQuery 里面功能最强的 Ajax 请求方法 `$.ajax()`;

jQuery.ajax(options) : 通过 HTTP 请求加载远程数据

这个是 jQuery 的底层 AJAX 实现。简单易用的高层实现见 `$.get`, `$.post` 等。

`$.ajax()` 返回其创建的 `XMLHttpRequest` 对象。大多数情况下你无需直接操作该对象，但特殊情况下可用于手动终止请求。

注意： 如果你指定了 `dataType` 选项，请确保服务器返回正确的 `MIME` 信息，(如 `xml` 返回 `"text/xml"`)。错误的 `MIME` 类型可能导致不可预知的错误。见 [Specifying the Data Type for AJAX Requests](#) 。

当设置 `datatype` 类型为 `'script'` 的时候，所有的远程(不在同一个域中)`POST` 请求都回转换为 `GET` 方式。

`$.ajax()` 只有一个参数：参数 `key/value` 对象，包含各配置及回调函数信息。详细参数选项见下。

jQuery 1.2 中，您可以跨域加载 `JSON` 数据，使用时需将数据类型设置为 `JSONP`。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。数据类型设置为 `"jsonp"` 时，jQuery 将自动调用回调函数。(这个我不是很懂)

参数列表：

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	(默认: "GET") 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 <code>XMLHttpRequest</code> 对象的函数，如添加自定义 HTTP 头。 <code>XMLHttpRequest</code> 对象是唯一的参数。 <pre>function (XMLHttpRequest) { this; // the options for this ajax request }</pre>

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。 <pre>function (XMLHttpRequest, textStatus) { this; // the options for this ajax request }</pre>
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。
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。 <pre>function (XMLHttpRequest, textStatus, errorThrown) { // 通常情况下 textStatus 和 errorThrown 只有其中一个</pre>

		有值 <pre> this; // the options for this ajax request </pre>
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 <pre> function (data, textStatus) { // data could be xmlDoc, jsonObj, html, text, etc... this; // the options for this ajax request } </pre>

这里有几个 Ajax 事件参数: **beforeSend** , **success** , **complete** , **error** 。我们可以定义这些事件来很好的处理我们的每一次的 Ajax 请求。注意一下, 这些 Ajax 事件里面的 **this** 都是指向 Ajax 请求的选项信息的(请参考说 **get()** 方法时的 **this** 的图片)。

请认真阅读上面的参数列表, 如果你要用 jQuery 来进行 Ajax 开发, 那么这些参数你都必需熟知的。

示例代码, 获取博客园首页的文章题目:

```

$.ajax({
  type: "get",
  url: "http://www.cnblogs.com/rss",
  beforeSend: function(XMLHttpRequest) {
    //ShowLoading();
  },

```

```

        success: function(data, textStatus) {
            $(".ajax.ajaxResult").html("");
            $(".item", data).each(function(i, domEle) {

                $(".ajax.ajaxResult").append("<li>" + $(domEle).children("title")
                .text() + "</li>");

            });
        },
        complete: function(XMLHttpRequest, textStatus) {
            //HideLoading();
        },
        error: function() {
            //请求出错处理
        }
    });

```

这里将显示首页文章列表。

其他

jQuery.ajaxSetup(options)：设置全局 AJAX 默认选项。

设置 AJAX 请求默认地址为 `"/xmlhttp/"`，禁止触发全局 AJAX 事件，用 POST 代替默认 GET 方法。其后的 AJAX 请求不再设置任何选项参数。

jQuery 代码：

```

$.ajaxSetup({
    url: "/xmlhttp/",
    global: false,
    type: "POST"
});

```



```
$.ajax({ data: myData });
```

serialize() 与 serializeArray()

serialize()：序列表表格内容为字符串。

serializeArray()：序列化表格元素（类似 '.serialize()' 方法）返回 JSON 数据结构数据。

示例：

HTML 代码：

```
<p id="results"><b>Results: </b> </p>

<form>

  <select name="single">
    <option>Single</option>
    <option>Single2</option>
  </select>

  <select name="multiple" multiple="multiple">
    <option selected="selected">Multiple</option>
    <option>Multiple2</option>
    <option selected="selected">Multiple3</option>
  </select><br/>

  <input type="checkbox" name="check" value="check1"/> check1
  <input type="checkbox" name="check" value="check2"
checked="checked"/> check2

  <input type="radio" name="radio" value="radio1"
checked="checked"/> radio1
  <input type="radio" name="radio" value="radio2"/> radio2
</form>
```

serialize()结果: single=Single&multiple=Multiple&multiple=Multiple3&check=check2

☐ check1
 ☒ check2
 ☒ radio1
 ☐ radio2

serializeArray() 结果为:

[-] json	[Object name=single value=Single, Object name=multiple value=Multiple, Object name=multiple value=Multiple3, 2 more...]
[-] 0	Object name=single value=Single
name	"single"
value	"Single"
[-] 1	Object name=multiple value=Multiple
name	"multiple"
value	"Multiple"
[-] 2	Object name=multiple value=Multiple3
name	"multiple"
value	"Multiple3"
[-] 3	Object name=check value=check2
name	"check"
value	"check2"
[-] 4	Object name=radio value=radio1
name	"radio"
value	"radio1"

一些资源

一个 jQuery 的 Ajax Form 表单插件: <http://www.malsup.com/jquery/form/>

一个专门生成 Loading 图片的站点: <http://ajaxload.info/> 大家觉得那些 Loading 比较炫

的可以在这里跟帖晒一下, 方便大家取用, 嘎嘎