**Jquery中的ajax异步调用讲解**

## **jQuery.ajax(url,[settings])**

### 概述

通过 HTTP 请求加载远程数据。

jQuery 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函数，除非你需要操作不常用的选项，以获得更多的灵活性。

最简单的情况下，$.ajax()可以不带任何参数直接使用。

****注意****，所有的选项都可以通过$.ajaxSetup()函数来全局设置。

****回调函数****

如果要处理$.ajax()得到的数据，则需要使用回调函数。beforeSend、error、dataFilter、success、complete。

* beforeSend 在发送请求之前调用，并且传入一个XMLHttpRequest作为参数。
* error 在请求出错时调用。传入XMLHttpRequest对象，描述错误类型的字符串以及一个异常对象（如果有的话）
* dataFilter 在请求成功之后调用。传入返回的数据以及"dataType"参数的值。并且必须返回新的数据（可能是处理过的）传递给success回调函数。
* success 当请求之后调用。传入返回后的数据，以及包含成功代码的字符串。
* complete 当请求完成之后调用这个函数，无论成功或失败。传入XMLHttpRequest对象，以及一个包含成功或错误代码的字符串。

****数据类型****

$.ajax()函数依赖服务器提供的信息来处理返回的数据。如果服务器报告说返回的数据是XML，那么返回的结果就可以用普通的XML方法或者jQuery的选择器来遍历。如果见得到其他类型，比如HTML，则数据就以文本形式来对待。

通过dataType选项还可以指定其他不同数据处理方式。除了单纯的XML，还可以指定 html、json、jsonp、script或者text。

其中，text和xml类型返回的数据不会经过处理。数据仅仅简单的将XMLHttpRequest的responseText或responseHTML属性传递给success回调函数，

'''注意'''，我们必须确保网页服务器报告的MIME类型与我们选择的dataType所匹配。比如说，XML的话，服务器端就必须声明 text/xml 或者 application/xml 来获得一致的结果。

如果指定为html类型，任何内嵌的JavaScript都会在HTML作为一个字符串返回之前执行。类似的，指定script类型的话，也会先执行服务器端生成JavaScript，然后再把脚本作为一个文本数据返回。

如果指定为json类型，则会把获取到的数据作为一个JavaScript对象来解析，并且把构建好的对象作为结果返回。为了实现这个目的，他首先尝试使用JSON.parse()。如果浏览器不支持，则使用一个函数来构建。JSON数据是一种能很方便通过JavaScript解析的结构化数据。如果获取的数据文件存放在远程服务器上（域名不同，也就是跨域获取数据），则需要使用jsonp类型。使用这种类型的话，会创建一个查询字符串参数 callback=? ，这个参数会加在请求的URL后面。服务器端应当在JSON数据前加上回调函数名，以便完成一个有效的JSONP请求。如果要指定回调函数的参数名来取代默认的callback，可以通过设置$.ajax()的jsonp参数。

****注意****，JSONP是JSON格式的扩展。他要求一些服务器端的代码来检测并处理查询字符串参数。更多信息可以参阅 [最初的文章](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/)。

如果指定了script或者jsonp类型，那么当从服务器接收到数据时，实际上是用了&lt;script&gt;标签而不是XMLHttpRequest对象。这种情况下，$.ajax()不再返回一个XMLHttpRequest对象，并且也不会传递事件处理函数，比如beforeSend。

****发送数据到服务器****

默认情况下，Ajax请求使用GET方法。如果要使用POST方法，可以设定type参数值。这个选项也会影响data选项中的内容如何发送到服务器。

data选项既可以包含一个查询字符串，比如 key1=value1&amp;key2=value2 ，也可以是一个映射，比如 {key1: 'value1', key2: 'value2'} 。如果使用了后者的形式，则数据再发送器会被转换成查询字符串。这个处理过程也可以通过设置processData选项为false来回避。如果我们希望发送一个XML对象给服务器时，这种处理可能并不合适。并且在这种情况下，我们也应当改变contentType选项的值，用其他合适的MIME类型来取代默认的 application/x-www-form-urlencoded 。

****高级选项****

global选项用于阻止响应注册的回调函数，比如.ajaxSend，或者ajaxError，以及类似的方法。这在有些时候很有用，比如发送的请求非常频繁且简短的时候，就可以在ajaxSend里禁用这个。更多关于这些方法的详细信息，请参阅下面的内容。

如果服务器需要HTTP认证，可以使用用户名和密码可以通过username和password选项来设置。

Ajax请求是限时的，所以错误警告被捕获并处理后，可以用来提升用户体验。请求超时这个参数通常就保留其默认值，要不就通过jQuery.ajaxSetup来全局设定，很少为特定的请求重新设置timeout选项。

默认情况下，请求总会被发出去，但浏览器有可能从他的缓存中调取数据。要禁止使用缓存的结果，可以设置cache参数为false。如果希望判断数据自从上次请求后没有更改过就报告出错的话，可以设置ifModified为true。

scriptCharset允许给&lt;script&gt;标签的请求设定一个特定的字符集，用于script或者jsonp类似的数据。当脚本和页面字符集不同时，这特别好用。

Ajax的第一个字母是asynchronous的开头字母，这意味着所有的操作都是并行的，完成的顺序没有前后关系。$.ajax()的async参数总是设置成true，这标志着在请求开始后，其他代码依然能够执行。强烈不建议把这个选项设置成false，这意味着所有的请求都不再是异步的了，这也会导致浏览器被锁死。

$.ajax函数返回他创建的XMLHttpRequest对象。通常jQuery只在内部处理并创建这个对象，但用户也可以通过xhr选项来传递一个自己创建的xhr对象。返回的对象通常已经被丢弃了，但依然提供一个底层接口来观察和操控请求。比如说，调用对象上的.abort()可以在请求完成前挂起请求。

### 参数

#### ****url,[settings]****ObjectV1.5

****url****:一个用来包含发送请求的URL字符串。

****settings****:AJAX 请求设置。所有选项都是可选的。

### V1.0****settings****:选项

#### ****accepts****Map

默认： 取决于数据类型。

内容类型发送请求头，告诉服务器什么样的响应会接受返回。如果accepts设置需要修改，推荐在$.ajaxSetup()方法中做一次。

#### ****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(XHR)****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。这是一个 [Ajax 事件](http://docs.jquery.com/Ajax_Events" \o "Ajax Events)。如果返回false可以取消本次ajax请求。

function (XMLHttpRequest) {

this; // 调用本次AJAX请求时传递的options参数

}

#### ****cache****Boolean

(默认: true,dataType为script和jsonp时默认为false) jQuery 1.2 新功能，设置为 false 将不缓存此页面。

#### ****complete(XHR, TS)****Function

请求完成后回调函数 (请求成功或失败之后均调用)。参数： XMLHttpRequest 对象和一个描述成功请求类型的字符串。 [Ajax 事件](http://docs.jquery.com/Ajax_Events" \o "Ajax Events)。

function (XMLHttpRequest, textStatus) {

this; // 调用本次AJAX请求时传递的options参数

}

#### ****contents****MapV1.5

一个以"{字符串:正则表达式}"配对的对象，用来确定jQuery将如何解析响应，给定其内容类型。

#### ****contentType****String

(默认: "application/x-www-form-urlencoded") 发送信息至服务器时内容编码类型。默认值适合大多数情况。如果你明确地传递了一个content-type给 $.ajax() 那么他必定会发送给服务器（即使没有数据要发送）

#### ****context****Object

这个对象用于设置Ajax相关回调函数的上下文。也就是说，让回调函数内this指向这个对象（如果不设定这个参数，那么this就指向调用本次AJAX请求时传递的options参数）。比如指定一个DOM元素作为context参数，这样就设置了success回调函数的上下文为这个DOM元素。就像这样：

$.ajax({ url: "test.html", context: document.body, success: function(){

$(this).addClass("done");

}});

#### ****converters****mapV1.5

默认： {"\* text": window.String, "text html": true, "text json": jQuery.parseJSON, "text xml": jQuery.parseXML}

一个数据类型对数据类型转换器的对象。每个转换器的值是一个函数，返回响应的转化值

#### ****crossDomain****mapV1.5

默认： 同域请求为false

跨域请求为true如果你想强制跨域请求（如JSONP形式）同一域，设置crossDomain为true。这使得例如，服务器端重定向到另一个域

#### ****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"。

#### ****dataFilter****Function

给Ajax返回的原始数据的进行预处理的函数。提供data和type两个参数：data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回的值将由jQuery进一步处理。

function (data, type) {

// 对Ajax返回的原始数据进行预处理

return data // 返回处理后的数据

}

#### ****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 信息来智能判断，比如XML MIME类型就被识别为XML。在1.4中，JSON就会生成一个JavaScript对象，而script则会执行这个脚本。随后服务器端返回的数据会根据这个值解析后，传递给回调函数。可用值:

"xml": 返回 XML 文档，可用 jQuery 处理。

"html": 返回纯文本 HTML 信息；包含的script标签会在插入dom时执行。

"script": 返回纯文本 JavaScript 代码。不会自动缓存结果。除非设置了"cache"参数。'''注意：'''在远程请求时(不在同一个域下)，所有POST请求都将转为GET请求。(因为将使用DOM的script标签来加载)

"json": 返回 JSON 数据 。

"jsonp": [JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/" \o "http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/) 格式。使用 [JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/" \o "http://bob.pythonmac.org/archives/2005/12/05/remote-json-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。

"text": 返回纯文本字符串

#### ****error****Function

(默认: 自动判断 (xml 或 html)) 请求失败时调用此函数。有以下三个参数：XMLHttpRequest 对象、错误信息、（可选）捕获的异常对象。如果发生了错误，错误信息（第二个参数）除了得到null之外，还可能是"timeout", "error", "notmodified" 和 "parsererror"。[Ajax 事件](http://docs.jquery.com/Ajax_Events" \o "Ajax Events)。

function (XMLHttpRequest, textStatus, errorThrown) {

// 通常 textStatus 和 errorThrown 之中

// 只有一个会包含信息

this; // 调用本次AJAX请求时传递的options参数

}

#### ****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](http://docs.jquery.com/Ajax_Events" \o "Ajax Events)。

#### ****headers****mapV1.5

Default: {}

一个额外的"{键:值}"对映射到请求一起发送。此设置被设置之前beforeSend函数被调用;因此，消息头中的值设置可以在覆盖beforeSend函数范围内的任何设置。

#### ****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。在jQuery 1.4中，他也会检查服务器指定的'etag'来确定数据没有被修改过。

#### ****isLocal****mapV1.5.1

默认: 取决于当前的位置协议

允许当前环境被认定为“本地”，（如文件系统），即使jQuery默认情况下不会承认它。以下协议目前公认为本地：file, \*-extension, and widget。如果isLocal设置需要修改，建议在$.ajaxSetup()方法中这样做一次。

#### ****jsonp****String

在一个jsonp请求中重写回调函数的名字。这个值用来替代在"callback=?"这种GET或POST请求中URL参数里的"callback"部分，比如{jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。

#### ****jsonpCallback****String

为jsonp请求指定一个回调函数名。这个值将用来取代jQuery自动生成的随机函数名。这主要用来让jQuery生成度独特的函数名，这样管理请求更容易，也能方便地提供回调函数和错误处理。你也可以在想让浏览器缓存GET请求的时候，指定这个回调函数名。

#### ****mimeType****StringV1.5.1

一个mime类型用来覆盖XHR的 MIME类型。

#### ****password****String

用于响应HTTP访问认证请求的密码

#### ****processData****Boolean

(默认: true) 默认情况下，通过data选项传递进来的数据，如果是一个对象(技术上讲只要不是字符串)，都会处理转化成一个查询字符串，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。

#### ****scriptCharset****String

只有当请求时dataType为"jsonp"或"script"，并且type是"GET"才会用于强制修改charset。通常只在本地和远程的内容编码不同时使用。

#### ****statusCode****mapV1.5

默认: {}

一组数值的HTTP代码和函数对象，当响应时调用了相应的代码。例如，如果响应状态是404，将触发以下警报：

$.ajax({

statusCode: {404: function() {

alert('page not found');

}

});

#### ****success(data, textStatus, jqXHR)****Function,Array

请求成功后的回调函数。参数：由服务器返回，并根据dataType参数进行处理后的数据；描述状态的字符串。还有 jqXHR（在jQuery 1.4.x的中，XMLHttpRequest） 对象 。在jQuery 1.5， 成功设置可以接受一个函数数组。每个函数将被依次调用。 [Ajax 事件](http://docs.jquery.com/Ajax_Events" \o "Ajax Events)。

function (data, textStatus) {

// data 可能是 xmlDoc, jsonObj, html, text, 等等...

this; // 调用本次AJAX请求时传递的options参数

}

#### ****traditional****Boolean

如果你想要用传统的方式来序列化数据，那么就设置为true。请参考工具分类下面的jQuery.param 方法。

#### ****timeout****Number

设置请求超时时间（毫秒）。此设置将覆盖全局设置。

#### ****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。

#### ****url****String

(默认: 当前页地址) 发送请求的地址。

#### ****username****String

用于响应HTTP访问认证请求的用户名

#### ****xhr****Function

需要返回一个XMLHttpRequest 对象。默认在IE下是ActiveXObject 而其他情况下是XMLHttpRequest 。用于重写或者提供一个增强的XMLHttpRequest 对象。这个参数在jQuery 1.3以前不可用。

#### ****xhrFields****mapV1.5

一对“文件名-文件值”在本机设置XHR对象。例如，如果需要的话，你可以用它来设置withCredentials为true的跨域请求。

### 示例

#### 描述:

加载并执行一个 JS 文件。

##### jQuery 代码:

$.ajax({

type: "GET",

url: "test.js",

dataType: "script"

});

#### 描述:

保存数据到服务器，成功时显示信息。

##### jQuery 代码:

$.ajax({

type: "POST",

url: "some.php",

data: "name=John&location=Boston",

success: function(msg){

alert( "Data Saved: " + msg );

}

});

#### 描述:

装入一个 HTML 网页最新版本。

##### jQuery 代码:

$.ajax({

url: "test.html",

cache: false,

success: function(html){

$("#results").append(html);

}

});

#### 描述:

同步加载数据。发送请求时锁住浏览器。需要锁定用户交互操作时使用同步方式。

##### jQuery 代码:

var html = $.ajax({

url: "some.php",

async: false

}).responseText;

#### 描述:

发送 XML 数据至服务器。设置 processData 选项为 false，防止自动转换数据格式。

##### jQuery 代码:

var xmlDocument = [create xml document];

$.ajax({

url: "page.php",

processData: false,

data: xmlDocument,

success: handleResponse

});

## **load(url, [data], [callback])**

### 概述

载入远程 HTML 文件代码并插入至 DOM 中。

默认使用 GET 方式 - 传递附加参数时自动转换为 POST 方式。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"。请查看示例。

### 参数

#### ****url,[data,[callback]]****String,Map/String,CallbackV1.0

****url****:待装入 HTML 网页网址。

****data****:发送至服务器的 key/value 数据。在jQuery 1.3中也可以接受一个字符串了。

****callback****:载入成功时回调函数。

### 示例

#### 描述:

加载文章侧边栏导航部分至一个无序列表。

##### HTML 代码:

<b>jQuery Links:</b>

<ul id="links"></ul>

##### jQuery 代码:

$("#links").load("/Main\_Page #p-Getting-Started li");

#### 描述:

加载 feeds.html 文件内容。

##### jQuery 代码:

$("#feeds").load("feeds.html");

#### 描述:

同上，但是以 POST 形式发送附加参数并在成功时显示信息。

##### jQuery 代码:

$("#feeds").load("feeds.php", {limit: 25}, function(){

alert("The last 25 entries in the feed have been loaded");

});

## **jQuery.get(url, [data], [callback], [type])**

### 概述

通过远程 HTTP GET 请求载入信息。

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。

### 参数

#### ****url,[data],[callback],[type]****String,Map,Function,StringV1.0

****url****:待载入页面的URL地址

****data****:待发送 Key/value 参数。

****callback****:载入成功时回调函数。

****type****:返回内容格式，xml, html, script, json, text, \_default。

### 示例

#### 描述:

请求 test.php 网页，忽略返回值。

##### jQuery 代码:

$.get("test.php");

#### 描述:

请求 test.php 网页，传送2个参数，忽略返回值。

##### jQuery 代码:

$.get("test.php", { name: "John", time: "2pm" } );

#### 描述:

显示 test.php 返回值(HTML 或 XML，取决于返回值)。

##### jQuery 代码:

$.get("test.php", function(data){

alert("Data Loaded: " + data);

});

#### 描述:

显示 test.cgi 返回值(HTML 或 XML，取决于返回值)，添加一组请求参数。

##### jQuery 代码:

$.get("test.cgi", { name: "John", time: "2pm" },

function(data){

alert("Data Loaded: " + data);

});

## **jQuery.getJSON(url, [data], [callback])**

### 概述

通过 HTTP GET 请求载入 JSON 数据。

在 jQuery 1.2 中，您可以通过使用[JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/" \o "http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/)形式的回调函数来加载其他网域的JSON数据，如 "myurl?callback=?"。jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。 注意：此行以后的代码将在这个回调函数执行前执行。

### 参数

#### ****url,[data],[callback]****String,Map,FunctionV1.0

****url****:发送请求地址。

****data****:待发送 Key/value 参数。

****callback****:载入成功时回调函数。

### 示例

#### 描述:

从 Flickr JSONP API 载入 4 张最新的关于猫的图片。

##### HTML 代码:

<div id="images"></div>

##### jQuery 代码:

$.getJSON("http://api.flickr.com/services/feeds/photos\_public.gne?tags=cat&tagmode=any&format=json&jsoncallback=?", function(data){

$.each(data.items, function(i,item){

$("<img/>").attr("src", item.media.m).appendTo("#images");

if ( i == 3 ) return false;

});

});

#### 描述:

从 test.js 载入 JSON 数据并显示 JSON 数据中一个 name 字段数据。

##### jQuery 代码:

$.getJSON("test.js", function(json){

alert("JSON Data: " + json.users[3].name);

});

#### 描述:

从 test.js 载入 JSON 数据，附加参数，显示 JSON 数据中一个 name 字段数据。

##### jQuery 代码:

$.getJSON("test.js", { name: "John", time: "2pm" }, function(json){

alert("JSON Data: " + json.users[3].name);

});

## **jQuery.getScript(url, [callback])**

### 概述

通过 HTTP GET 请求载入并执行一个 JavaScript 文件。

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。

### 参数

#### ****url,[callback]****String,FunctionV1.0

****url****:待载入 JS 文件地址。

****callback****:成功载入后回调函数。

### 示例

#### 描述:

载入 <a title="http://jquery.com/plugins/project/color" class="external text" href="http://jquery.com/plugins/project/color">jQuery 官方颜色动画插件</a> 成功后绑定颜色变化动画。

##### HTML 代码:

<button id="go">» Run</button>

<div class="block"></div>

##### jQuery 代码:

jQuery.getScript("http://dev.jquery.com/view/trunk/plugins/color/jquery.color.js", function(){

$("#go").click(function(){

$(".block").animate( { backgroundColor: 'pink' }, 1000)

.animate( { backgroundColor: 'blue' }, 1000);

});

});

#### 描述:

加载并执行 test.js。

##### jQuery 代码:

$.getScript("test.js");

#### 描述:

加载并执行 test.js ，成功后显示信息。

##### jQuery 代码:

$.getScript("test.js", function(){

alert("Script loaded and executed.");

});

## **jQuery.post(url, [data], [callback], [type])**

### 概述

通过远程 HTTP POST 请求载入信息。

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。

### 参数

#### ****url,[data],[callback],[type]****String,Map,Function,StringV1.0

****url****:发送请求地址。

****data****:待发送 Key/value 参数。

****callback****:发送成功时回调函数。

****type****:返回内容格式，xml, html, script, json, text, \_default。

### 示例

#### 1描述:

请求 test.php 网页，忽略返回值：

##### jQuery 代码:

$.post("test.php");

#### 2描述:

请求 test.php 页面，并一起发送一些额外的数据（同时仍然忽略返回值）：

##### jQuery 代码:

$.post("test.php", { name: "John", time: "2pm" } );

#### 3描述:

向服务器传递数据数组（同时仍然忽略返回值）：

##### jQuery 代码:

$.post("test.php", { 'choices[]': ["Jon", "Susan"] });

#### 4描述:

使用 ajax 请求发送表单数据：

##### jQuery 代码:

$.post("test.php", $("#testform").serialize());

#### 5描述:

输出来自请求页面 test.php 的结果（HTML 或 XML，取决于所返回的内容）：

##### jQuery 代码:

$.post("test.php", function(data){

alert("Data Loaded: " + data);

});

#### 6描述:

向页面 test.php 发送数据，并输出结果（HTML 或 XML，取决于所返回的内容）：

##### jQuery 代码:

$.post("test.php", { name: "John", time: "2pm" },

function(data){

alert("Data Loaded: " + data);

});

#### 7描述:

获得 test.php 页面的内容，并存储为 XMLHttpResponse 对象，并通过 process() 这个 JavaScript 函数进行处理：

##### jQuery 代码:

$.post("test.php", { name: "John", time: "2pm" },

function(data){

process(data);

}, "xml");

#### 8描述:

获得 test.php 页面返回的 json 格式的内容：：

##### jQuery 代码:

$.post("test.php", { "func": "getNameAndTime" },

function(data){

alert(data.name); // John

console.log(data.time); // 2pm

}, "json");

## **ajaxComplete(callback)**

### 概述

AJAX 请求完成时执行函数。Ajax 事件。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

### 参数

#### ****callback****FunctionV1.0

待执行函数

### 示例

#### 描述:

AJAX 请求完成时执行函数。

##### jQuery 代码:

$("#msg").ajaxComplete(function(event,request, settings){

$(this).append("<li>请求完成.</li>");

});

#### 描述:

当 AJAX 请求正在进行时显示“正在加载”的指示：

##### jQuery 代码:

$("#txt").ajaxStart(function(){

$("#wait").css("display","block");

});

$("#txt").ajaxComplete(function(){

$("#wait").css("display","none");

});

## **ajaxError(callback)**

### 概述

AJAX 请求发生错误时执行函数。Ajax 事件。

XMLHttpRequest 对象和设置作为参数传递给回调函数。捕捉到的错误可作为最后一个参数传递。

### 参数

#### ****callback****FunctionV1.0

待执行函数

function (event, XMLHttpRequest, ajaxOptions, thrownError) {

// thrownError 只有当异常发生时才会被传递

this; // 监听的 dom 元素

}

### 示例

#### 描述:

AJAX 请求失败时显示信息。

##### jQuery 代码:

$("#msg").ajaxError(function(event,request, settings){

$(this).append("<li>出错页面:" + settings.url + "</li>");

});

## **ajaxSend(callback)**

### 概述

AJAX 请求发送前执行函数。Ajax 事件。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

### 参数

#### ****callback****FunctionV1.0

待执行函数

### 示例

#### 描述:

AJAX 请求发送前显示信息。

##### jQuery 代码:

$("#msg").ajaxSend(function(evt, request, settings){

$(this).append("<li>开始请求: " + settings.url + "</li>");

});

## **ajaxStart(callback)**

### 概述

AJAX 请求开始时执行函数。Ajax 事件。

### 参数

#### ****callback****FunctionV1.0

待执行函数

### 示例

#### 描述:

AJAX 请求开始时显示信息。

##### jQuery 代码:

$("#loading").ajaxStart(function(){

$(this).show();

});

## **ajaxStop(callback)**

### 概述

AJAX 请求结束时执行函数。Ajax 事件。

### 参数

#### ****callback****FunctionV1.0

待执行函数

### 示例

#### 描述:

AJAX 请求结束后隐藏信息。

##### jQuery 代码:

$("#loading").ajaxStop(function(){

$(this).hide();

});

## **ajaxSuccess(callback)**

### 概述

AJAX 请求成功时执行函数。Ajax 事件。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

### 参数

#### ****callback****FunctionV1.0

待执行函数

### 示例

#### 描述:

当 AJAX 请求成功后显示消息。

##### jQuery 代码:

$("#msg").ajaxSuccess(function(evt, request, settings){

$(this).append("<li>请求成功!</li>");

});

## **jQuery.ajaxSetup([options])**

### 概述

设置全局 AJAX 默认选项。

参数见 '$.ajax' 说明。

### 参数

#### ****options****ObjectV1.1

选项设置。所有设置项均为可选设置。.

### 示例

#### 描述:

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()**

### V1.0概述

序列表表格内容为字符串。

### 示例

#### 描述:

序列表表格内容为字符串，用于 Ajax 请求。

##### 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>

##### jQuery 代码:

$("#results").append( "<tt>" + $("form").serialize() + "</tt>" );