目录

[ajax() 方法通过 HTTP 请求加载远程数据。 2](#_Toc431408699)

[jQuery 文档操作 - after() 方法 7](#_Toc431408700)

[jQuery 遍历 - children() 方法 7](#_Toc431408701)

[jQuery 遍历 - last() 方法 7](#_Toc431408702)

[定义和用法 7](#_Toc431408703)

[详细说明 7](#_Toc431408704)

[jQuery DOM 元素方法 - index() 方法 8](#_Toc431408705)

[定义和用法 8](#_Toc431408706)

[第一个匹配元素的 index，相对于同胞元素 8](#_Toc431408707)

[语法 8](#_Toc431408708)

[jQuery 遍历 - siblings() 方法 8](#_Toc431408709)

[定义和用法 9](#_Toc431408710)

# ajax() 方法通过 HTTP 请求加载远程数据。

该方法是 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。

options

类型：Object

可选。AJAX 请求设置。所有选项都是可选的。

async

类型：Boolean

默认值: 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)

类型：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。如果返回 false 可以取消本次 ajax 请求。

cache

类型：Boolean

默认值: true，dataType 为 script 和 jsonp 时默认为 false。设置为 false 将不缓存此页面。

jQuery 1.2 新功能。

complete(XHR, TS)

类型：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 对象和一个描述请求类型的字符串。

这是一个 Ajax 事件。

contentType

类型：String

默认值: "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() 那么它必定会发送给服务器（即使没有数据要发送）。

context

类型：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");

}});

data

类型：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

类型：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。

dataType

类型：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 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。

"text": 返回纯文本字符串

error

类型：Function

默认值: 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"。

这是一个 Ajax 事件。

global

类型：Boolean

是否触发全局 AJAX 事件。默认值: true。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

类型：Boolean

仅在服务器数据改变时获取新数据。默认值: false。使用 HTTP 包 Last-Modified 头信息判断。在 jQuery 1.4 中，它也会检查服务器指定的 'etag' 来确定数据没有被修改过。

jsonp

类型：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

类型：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。

password

类型：String

用于响应 HTTP 访问认证请求的密码

processData

类型：Boolean

默认值: 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

类型：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。

success

类型：Function

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。

这是一个 Ajax 事件。

traditional

类型：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

类型：Number

设置请求超时时间（毫秒）。此设置将覆盖全局设置。

type

类型：String

默认值: "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

类型：String

默认值: 当前页地址。发送请求的地址。

username

类型：String

用于响应 HTTP 访问认证请求的用户名。

xhr

类型：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。

# jQuery 文档操作 - after() 方法

在每个 p 元素后插入内容：

$("button").click(function(){

$("p").after("<p>Hello world!</p>");

});

# jQuery 遍历 - children() 方法

add() 方法返回匹配元素集合中每个元素的子元素，添加可选参数可通过选择器进行过滤。如果给定表示 DOM 元素集合的 jQuery 对象，.children() 方法允许我们检索 DOM 树中的这些元素，并用匹配元素构造新的 jQuery 对象。[.find()](http://www.w3school.com.cn/jquery/traversing_find.asp) 和 .children() 方法类似，不过后者只沿着 DOM 树向下遍历单一层级。

请注意，与大多数 jQuery 方法一样，.children() 不返回文本节点；如果需要获得包含文本和注释节点在内的所有子节点，请使用 .contents()。

该方法接受一个选择器表达式作为可选参数，与我们传递到 $() 的参数的类型是相同的。如果应用该选择器，将测试元素是否匹配该表达式，以此筛选这些元素。

找到类名为 "selected" 的所有 div 的子元素，并将其设置为蓝色：

$("div").children(".selected").css("color", "blue");

# jQuery 遍历 - last() 方法

## 定义和用法

last() 将匹配元素集合缩减为集合中的最后一个元素。

高亮显示段落中的最后一个 span ：

$("p span").last().addClass('highlight');

### 详细说明

如果给定一个表示 DOM 元素集合的 jQuery 对象，.last() 方法会用最后一个匹配元素构造一个新的 jQuery 对象。

请思考下面这个带有简单列表的页面：

<ul>

<li>list item 1</li>

<li>list item 2</li>

<li>list item 3</li>

<li>list item 4</li>

<li>list item 5</li>

</ul>

我们可以对这个列表项集合应用该方法：

$('li').last().css('background-color', 'red');

# jQuery DOM 元素方法 - index() 方法

## 定义和用法

index() 方法返回指定元素相对于其他指定元素的 index 位置。

这些元素可通过 jQuery 选择器或 DOM 元素来指定。

**注释：**如果未找到元素，index() 将返回 -1。

## 第一个匹配元素的 index，相对于同胞元素

获得第一个匹配元素相对于其同胞元素的 index 位置。

### 语法

$(*selector*).index()

# jQuery 遍历 - siblings() 方法

## 定义和用法

siblings() 获得匹配集合中每个元素的同胞，通过选择器进行筛选是可选的。

如果给定一个表示 DOM 元素集合的 jQuery 对象，.siblings() 方法允许我们在 DOM 树中搜索这些元素的同胞元素，并用匹配元素构造一个新的 jQuery 对象。

该方法接受可选的选择器表达式，与我们向 $() 函数中传递的参数类型相同。如果应用这个选择器，则将通过检测元素是否匹配该选择器对元素进行筛选。

请思考这个带有基本的嵌套列表的页面：

<ul>

<li>list item 1</li>

<li>list item 2</li>

<li class="third-item">list item 3</li>

<li>list item 4</li>

<li>list item 5</li>

</ul>

如果我们从第三个项目开始，则可找到该元素的同胞元素：

$('li.third-item').siblings().css('background-color', 'red');