# [核心函数 7](#_Toc283569072)

# [jQuery核心函数 7](#_Toc283569073)

# [jQuery(expression,[context]) 7](#_Toc283569074)

# [jQuery(html) 创建DOM元素 8](#_Toc283569075)

# [jQuery(elements) 将一个或多个DOM元素转化为jQuery对象 9](#_Toc283569076)

# [jQuery(callback) 在DOM文档载入完成后执行的函数 10](#_Toc283569077)

# [jQuery 对象访问 11](#_Toc283569078)

# [each(callback) 遍历数组 11](#_Toc283569079)

# [size() 这个函数的返回值与 jQuery 对象的'length' 属性一致 13](#_Toc283569080)

# [length 当前匹配的元素个数。 size 将返回相同的值。 14](#_Toc283569081)

# [get() 取得所有匹配的 DOM 元素集合。如果你想要直接操作 DOM 对象而不是 jQuery 对象，这个函数非常有用. 14](#_Toc283569082)

# [get(index) 取得其中一个匹配的元素(实际的DOM 元素)。num表示取得第几个匹配的元素。 15](#_Toc283569083)

# [index(subject) 搜索与参数表示的对象匹配的元素，并返回相应元素的索引值值。 16](#_Toc283569084)

# [插件机制 16](#_Toc283569085)

# [jQuery.fn.extend(object) 扩展 jQuery 元素集来提供新的方法（通常用来制作插件）。 16](#_Toc283569086)

# [jQuery.extend(object) 扩展jQuery对象本身。 17](#_Toc283569087)

# [多库共存 18](#_Toc283569088)

# [jQuery.noConflict() 这有助于确保jQuery不会与其他库的$对象发生冲突。在运行这个函数后，就只能使用jQuery变量访问jQuery对象。 18](#_Toc283569089)

# [jQuery.noConflict(extreme) 将$和jQuery的控制权都交还给原来的库。 19](#_Toc283569090)

# [Revert control of both the $ and jQuery variables to their original owners. Use with discretion. 19](#_Toc283569091)

# [选择器 20](#_Toc283569092)

# [基本 20](#_Toc283569093)

# [$("#idName"); 20](#_Toc283569094)

# [$("p"); 21](#_Toc283569095)

# [$(".className"); 22](#_Toc283569096)

# [$("\*") 22](#_Toc283569097)

# [$("div,span,p.myClass") 23](#_Toc283569098)

# [层级 24](#_Toc283569099)

# [$("form input")(所有子孙元素) 24](#_Toc283569100)

# [$("form > input")(所有子元素) 25](#_Toc283569101)

# [$("label + input")(挨着我, 右边是XX的元素) 25](#_Toc283569102)

# [$("form ~ input")(我右边所有是XX的元素) 26](#_Toc283569103)

# [简单 27](#_Toc283569104)

# [$("tr:first") 27](#_Toc283569105)

# [$("tr:last") 28](#_Toc283569106)

# [$("input:not(:checked)") 28](#_Toc283569107)

# [$("tr:even") 29](#_Toc283569108)

# [$("tr:odd") 30](#_Toc283569109)

# [$("tr:eq(1)") 30](#_Toc283569110)

# [$("tr:gt(0)") 31](#_Toc283569111)

# [$("tr:lt(2)") 32](#_Toc283569112)

# [$(":header")( h1, h2, h3之类的标题元素) 33](#_Toc283569113)

# [:animated(所有没有在执行动画效果中的元素) 33](#_Toc283569114)

# [内容 34](#_Toc283569115)

# [$("div:contains('John')") 34](#_Toc283569116)

# [$("td:empty") 35](#_Toc283569117)

# [$("div:has(p)") 35](#_Toc283569118)

# [:parent(含有子元素或者文本的元素) 36](#_Toc283569119)

# [可见性 37](#_Toc283569120)

# [:hidden 37](#_Toc283569121)

# [:visible 37](#_Toc283569122)

# [属性 38](#_Toc283569123)

# [[attribute]( 包含给定属性) 38](#_Toc283569124)

# [[attribute=value] 39](#_Toc283569125)

# [[attribute!=value] 40](#_Toc283569126)

# [[attribute^=value]\_\_ 给定的属性是以某些值开始的元素 40](#_Toc283569127)

# [[attribute$=value]\_\_ 给定的属性是以某些值结尾 41](#_Toc283569128)

# [[attribute\*=value]\_\_ 给定的属性的值包含某字符串的元素 42](#_Toc283569129)

# [[selector1][selector2][selectorN]\_\_ 同时满足多个条件 43](#_Toc283569130)

# [子元素 44](#_Toc283569131)

# [:nth-child(index/even/odd/equation)\_\_ 第N个子或奇偶元素 44](#_Toc283569132)

# [:first-child 45](#_Toc283569133)

# [:last-child 46](#_Toc283569134)

# [:only-child\_\_如果某个元素是父元素中唯一的子元素，那将会被匹配, 47](#_Toc283569135)

# [表单 48](#_Toc283569136)

# [:input 48](#_Toc283569137)

# [:text 49](#_Toc283569138)

# [:password 49](#_Toc283569139)

# [:radio 50](#_Toc283569140)

# [:checkbox\_\_复选框 51](#_Toc283569141)

# [:submit 52](#_Toc283569142)

# [:image\_\_图像域<input type="image" /> 53](#_Toc283569143)

# [:reset 54](#_Toc283569144)

# [:button 55](#_Toc283569145)

# [:file\_\_文件域 56](#_Toc283569146)

# [:hidden 57](#_Toc283569147)

# [表单对象属性 58](#_Toc283569148)

# [:enabled\_\_可用 58](#_Toc283569149)

# [:disabled 58](#_Toc283569150)

# [:checked\_\_选中的复选框 59](#_Toc283569151)

# [:selected\_\_选中的选项 60](#_Toc283569152)

# [属性 61](#_Toc283569153)

# [属性 61](#_Toc283569154)

# [attr(name)\_\_ 第一个匹配元素的属性值 61](#_Toc283569155)

# [attr(properties) 62](#_Toc283569156)

# [attr(key,value) 62](#_Toc283569157)

# [attr(key,fn) 63](#_Toc283569158)

# [removeAttr(name) 64](#_Toc283569159)

# [CSS类 65](#_Toc283569160)

# [addClass(class) 65](#_Toc283569161)

# [removeClass(class) 66](#_Toc283569162)

# [toggleClass(class) 67](#_Toc283569163)

# [Html 67](#_Toc283569164)

# [html() 67](#_Toc283569165)

# [html(val) 68](#_Toc283569166)

# [text() 68](#_Toc283569167)

# [text(val) 69](#_Toc283569168)

# [值 70](#_Toc283569169)

# [val() 70](#_Toc283569170)

# [val(val) 71](#_Toc283569171)

# [val(val) 72](#_Toc283569172)

# [筛选 73](#_Toc283569173)

# [过滤 73](#_Toc283569174)

# [eq(index) 73](#_Toc283569175)

# [hasClass(class) 73](#_Toc283569176)

# [filter(expr) 74](#_Toc283569177)

# [filter(fn) 75](#_Toc283569178)

# [is(expr) 76](#_Toc283569179)

# [map(callback) 77](#_Toc283569180)

# [not(expr) 78](#_Toc283569181)

# [slice(start,[end]) 78](#_Toc283569182)

# [查找 80](#_Toc283569183)

# [@add(expr) 80](#_Toc283569184)

# [children([expr]) 82](#_Toc283569185)

# [contents() 83](#_Toc283569186)

# [find(expr) 84](#_Toc283569187)

# [next([expr]) 84](#_Toc283569188)

# [nextAll([expr]) 85](#_Toc283569189)

# [parent([expr]) 86](#_Toc283569190)

# [parents([expr]) 87](#_Toc283569191)

# [prev([expr]) 88](#_Toc283569192)

# [prevAll([expr]) 89](#_Toc283569193)

# [siblings([expr]) 90](#_Toc283569194)

# [串联 91](#_Toc283569195)

# [andSelf() 91](#_Toc283569196)

# [end() 91](#_Toc283569197)

# [文档处理 92](#_Toc283569198)

# [内部插入 92](#_Toc283569199)

# [append(content)\_\_ 内部追加 92](#_Toc283569200)

# [appendTo(content)\_\_追加到另一个元素中 93](#_Toc283569201)

# [prepend(content)\_\_ 内部前置 94](#_Toc283569202)

# [prependTo(content)\_\_ 前置到另一个元素中 95](#_Toc283569203)

# [after(content)\_\_ 元素之后插入 96](#_Toc283569204)

# [before(content)\_\_ 在每个匹配的元素之前插入 97](#_Toc283569205)

# [insertAfter(content)\_\_ 插入到另一个后面 98](#_Toc283569206)

# [insertBefore(content)\_\_ 插入到另一个元素前面 99](#_Toc283569207)

# [wrap(html)\_\_ 用其他元素包裹 100](#_Toc283569208)

# [wrap(elem)\_\_ 用其他元素 101](#_Toc283569209)

# [wrapAll(html)\_\_ 将所有匹配的元素用单个元素包裹起来 102](#_Toc283569210)

# [wrapAll(elem)\_\_ 将所有匹配的元素用单个元素包裹起来 103](#_Toc283569211)

# [wrapInner(html) 103](#_Toc283569212)

# [wrapInner(elem) 104](#_Toc283569213)

# [replaceWith(content) 105](#_Toc283569214)

# [replaceAll(selector) 106](#_Toc283569215)

# [empty()\_\_清空元素内容 106](#_Toc283569216)

# [remove([expr])\_\_删除元素 107](#_Toc283569217)

# [clone() 108](#_Toc283569218)

# [clone(true) 109](#_Toc283569219)

# [CSS 109](#_Toc283569220)

# [CSS 109](#_Toc283569221)

# [css(name)\_\_ 访问第一个匹配元素的样式属性 109](#_Toc283569222)

# [css(properties)\_\_设置css值(名/值对) 110](#_Toc283569223)

# [$("p").css({ color: "#ff0011", background: "blue" }); 110](#_Toc283569224)

# [css(name,value)\_\_ 设置一个样式属性的值 111](#_Toc283569225)

# [位置 111](#_Toc283569226)

# [offset()\_\_top 和 left 111](#_Toc283569227)

# [height() 112](#_Toc283569228)

# [height(val) 113](#_Toc283569229)

# [width() 113](#_Toc283569230)

# [width(val) 114](#_Toc283569231)

# [事件 114](#_Toc283569232)

# [页面载入 114](#_Toc283569233)

# [ready(fn) 114](#_Toc283569234)

# [事件处理 116](#_Toc283569235)

# [bind(type,[data],fn) 116](#_Toc283569236)

# [one(type,[data],fn) 117](#_Toc283569237)

# [trigger(type,[data]) 118](#_Toc283569238)

# [triggerHandler(type,[data]) 120](#_Toc283569239)

# [unbind([type],[data]) 121](#_Toc283569240)

# [hover(over,out) 122](#_Toc283569241)

# [toggle(fn,fn) 123](#_Toc283569242)

# [blur() 124](#_Toc283569243)

# [blur(fn) 124](#_Toc283569244)

# [change() 125](#_Toc283569245)

# [change(fn) 125](#_Toc283569246)

# [click() 126](#_Toc283569247)

# [click(fn) 126](#_Toc283569248)

# [dblclick() 127](#_Toc283569249)

# [dblclick(fn) 128](#_Toc283569250)

# [error() 128](#_Toc283569251)

# [error(fn) 129](#_Toc283569252)

# [focus() 130](#_Toc283569253)

# [focus(fn) 131](#_Toc283569254)

# [keydown() 131](#_Toc283569255)

# [keydown(fn) 132](#_Toc283569256)

# [keypress() 133](#_Toc283569257)

# [keypress(fn) 133](#_Toc283569258)

# [keyup() 134](#_Toc283569259)

# [keyup(fn) 134](#_Toc283569260)

# [load(fn) 134](#_Toc283569261)

# [mousedown(fn) 135](#_Toc283569262)

# [mousemove(fn) 135](#_Toc283569263)

# [mouseout(fn) 136](#_Toc283569264)

# [mouseover(fn) 136](#_Toc283569265)

# [mouseup(fn) 137](#_Toc283569266)

# [resize(fn) 137](#_Toc283569267)

# [scroll(fn (Function)) 137](#_Toc283569268)

# [select() 138](#_Toc283569269)

# [select(fn) 138](#_Toc283569270)

# [submit() 139](#_Toc283569271)

# [submit(fn) 139](#_Toc283569272)

# [unload(fn) 140](#_Toc283569273)

# [效果 140](#_Toc283569274)

# [show() 141](#_Toc283569275)

# [show(speed,[callback]) 141](#_Toc283569276)

# [hide() 142](#_Toc283569277)

# [hide(speed,[callback]) 143](#_Toc283569278)

# [toggle() 144](#_Toc283569279)

# [slideDown(speed,[callback]) 144](#_Toc283569280)

# [slideUp(speed,[callback]) 145](#_Toc283569281)

# [slideToggle(speed,[callback]) 146](#_Toc283569282)

# [fadeIn(speed,[callback]) 147](#_Toc283569283)

# [fadeOut(speed,[callback]) 148](#_Toc283569284)

# [fadeTo(speed,opacity,[callback]) 148](#_Toc283569285)

# [animate(params[,duration[,easing[,callback]]]) 149](#_Toc283569286)

# [animate(params,options) 152](#_Toc283569287)

# [stop() 154](#_Toc283569288)

# [queue() 155](#_Toc283569289)

# [queue(callback) 156](#_Toc283569290)

# [queue(queue) 157](#_Toc283569291)

# [dequeue() 158](#_Toc283569292)

# [Ajax 159](#_Toc283569293)

# [Ajax 请求 159](#_Toc283569294)

# [jQuery.ajax(options) 159](#_Toc283569295)

# [load(url,[data],[callback]) 163](#_Toc283569296)

# [jQuery.get(url,[data],[callback]) 164](#_Toc283569297)

# [jQuery.getJSON(url,[data],[callback]) 165](#_Toc283569298)

# [jQuery.getScript(url,[callback]) 166](#_Toc283569299)

# [jQuery.post(url,[data],[callback]) 167](#_Toc283569300)

# [ajaxComplete(callback) 168](#_Toc283569301)

# [ajaxError(callback) 168](#_Toc283569302)

# [ajaxSend(callback) 169](#_Toc283569303)

# [ajaxStart(callback) 170](#_Toc283569304)

# [ajaxStop(callback) 170](#_Toc283569305)

# [ajaxSuccess(callback) 171](#_Toc283569306)

# [jQuery.ajaxSetup(options) 171](#_Toc283569307)

# [serialize() 172](#_Toc283569308)

# [serializeArray() 173](#_Toc283569309)

# [工具 174](#_Toc283569310)

# [jQuery.browser 174](#_Toc283569311)

# [Map 175](#_Toc283569312)

# [jQuery.browser.version 175](#_Toc283569313)

# [jQuery.boxModel 176](#_Toc283569314)

# [jQuery.each(obj,callback) 176](#_Toc283569315)

# [jQuery.extend(target,obj1,[objN]) 177](#_Toc283569316)

# [jQuery.grep(array,callback,[invert]) 178](#_Toc283569317)

# [jQuery.makeArray(obj) 179](#_Toc283569318)

# [jQuery.map(array,callback) 180](#_Toc283569319)

# [jQuery.unique(array) 181](#_Toc283569320)

# [jQuery.isFunction(obj) 182](#_Toc283569321)

# [jQuery.trim(str) 183](#_Toc283569322)

# 核心函数

## jQuery核心函数

### jQuery(expression,[context])

这个函数接收一个包含 CSS 选择器的字符串，然后用这个字符串去匹配一组元素。

jQuery 的核心功能都是通过这个函数实现的。 jQuery中的一切都构建于这个函数之上，或者说都是在以某种方式使用这个函数。这个函数最基本的用法就是向它传递一个表达式（通常由 CSS 选择器组成），然后根据这个表达式来查找所有匹配的元素。

默认情况下, 如果没有指定context参数，$()将在当前的 HTML 文档中查找 DOM 元素；如果指定了 context 参数，如一个 DOM 元素集或 jQuery 对象，那就会在这个 context 中查找。

参考 Selectors 获取更多用于 expression 参数的 CSS 语法的信息。

This function accepts a string containing a CSS selector which is then used to match a set of elements.

The core functionality of jQuery centers around this function. Everything in jQuery is based upon this, or uses this in some way. The most basic use of this function is to pass in an expression (usually consisting of CSS), which then finds all matching elements.

By default, if no context is specified, $() looks for DOM elements within the context of the current HTML document. If you do specify a context, such as a DOM element or jQuery object, the expression will be matched against the contents of that context.

See Selectors for the allowed CSS syntax for expressions.

**返回值**

jQuery

**参数**

**expression** (String) : 用来查找的字符串

**context** (Element, jQuery) : (可选) 作为待查找的 DOM 元素集、文档或 jQuery 对象。

**示例**

|  |
| --- |
| 找到所有 p 元素，并且这些元素都必须是 div 元素的子元素。  **HTML 代码:**  <p>one</p> <div><p>two</p></div> <p>three</p>  **jQuery 代码:**  $("div > p");  **结果:**  [ <p>two</p> ] |
| 在文档的第一个表单中，查找所有的单选按钮(即: type 值为 radio 的 input 元素)。  **jQuery 代码:**  $("input:radio", document.forms[0]); |
| 在一个由 AJAX 返回的 XML 文档中，查找所有的 div 元素。  **jQuery 代码:**  $("div", xml.responseXML); |

### jQuery(html) 创建DOM元素

根据提供的原始 HTML 标记字符串，动态创建 由 jQuery 对象包装的 DOM 元素。

你可以传递一个手写的 HTML 字符串，或者由某些模板引擎或插件创建的字符串，也可以是通过 AJAX 加载过来的字符串。但是在你创建 input 元素的时会有限制，可以参考第二个示例。当然这个字符串可以包含斜杠 (比如一个图像地址)，还有反斜杠。当你创建单个元素时，请使用闭合标签或 XHTML 格式。例如，创建一个 span ，可以用 $("<span/>") 或 $("<span></span>") ，但不推荐 $("<span>")

Create DOM elements on-the-fly from the provided String of raw HTML.

You can pass in plain HTML Strings written by hand, create them using some template engine or plugin, or load them via AJAX. There are limitations when creating input elements, see the second example. Also when passing strings that may include slashes (such as an image path), escape the slashes. When creating single elements use the closing tag or XHTML format. For example, to create a span use $("<span/>") or $("<span></span>") instead of without the closing slash/tag.

**返回值**

jQuery

**参数**

**html** (String) : 用于动态创建DOM元素的HTML标记字符串

**示例**

|  |
| --- |
| 动态创建一个 div 元素（以及其中的所有内容），并将它追加到 body 元素中。在这个函数的内部，是通过临时创建一个元素，并将这个元素的 innerHTML 属性设置为给定的标记字符串，来实现标记到 DOM 元素转换的。所以，这个函数既有灵活性，也有局限性。  **jQuery 代码:**  $("<div><p>Hello</p></div>").appendTo("body"); |
| 创建一个 <input> 元素必须同时设定 type 属性。因为微软规定 <input> 元素的 type 只能写一次。  **jQuery 代码:**  // 在 IE 中无效:  $("<input>").attr("type", "checkbox");  // 在 IE 中有效:  $("<input type='checkbox'>");  jQuery(elements) |

### jQuery(elements) 将一个或多个DOM元素转化为jQuery对象

将一个或多个DOM元素转化为jQuery对象。

这个函数也可以接收XML文档和Window对象（虽然它们不是DOM元素）作为有效的参数。

Wrap jQuery functionality around a single or multiple DOM Element(s).

This function also accepts XML Documents and Window objects as valid arguments (even though they are not DOM Elements).

**返回值**

jQuery

**参数**

**elements** (Element, Array<Element>) : 用于封装成jQuery对象的DOM元素

**示例**

|  |
| --- |
| 设置页面背景色。  **jQuery 代码:**  $(document.body).css( "background", "black" ); |
| 隐藏一个表单中所有元素。  **jQuery 代码:**  $(myForm.elements).hide() |

### jQuery(callback) 在DOM文档载入完成后执行的函数

$(document).ready() 的简写。

允许你绑定一个在DOM文档载入完成后执行的函数。这个函数的作用如同$(document).ready()一样，只不过用这个函数时，需要把页面中所有需要在 DOM 加载完成时执行的$()操作符都包装到其中来。从技术上来说，这个函数是可链接的－－但真正以这种方式链接的情况并不多。

你可以在一个页面中使用任意多个$(document).ready事件。

参考 ready(Function) 获取更多 ready 事件的信息。

A shorthand for $(document).ready().

Allows you to bind a function to be executed when the DOM document has finished loading. This function behaves just like $(document).ready(), in that it should be used to wrap other $() operations on your page that depend on the DOM being ready to be operated on. While this function is, technically, chainable - there really isn't much use for chaining against it.

You can have as many $(document).ready events on your page as you like.

See ready(Function) for details about the ready event.

**返回值**

jQuery

**参数**

**callback** (Function) : 当DOM加载完成后要执行的函数

**示例**

|  |
| --- |
| 当DOM加载完成后，执行其中的函数。  **jQuery 代码:**  $(function(){  // Document is ready  }); |
| Uses both the shortcut for $(document).ready() and the argument to write failsafe jQuery code using the $ alias, without relying on the global alias.  **jQuery 代码:**  jQuery(function($) {  // Your code using failsafe $ alias here...  }); |

## jQuery 对象访问

### each(callback) 遍历数组

以每一个匹配的元素作为上下文来执行一个函数。

意味着，每次执行传递进来的函数时，函数中的this关键字都指向一个不同的DOM元素（每次都是一个不同的匹配元素）。

而且，在每次执行函数时，都会给函数传递一个表示作为执行环境的元素在匹配的元素集合中所处位置的数字值作为参数（从零开始的整形）。

返回 'false' 将停止循环 (就像在普通的循环中使用 'break')。返回 'true' 跳至下一个循环(就像在普通的循环中使用'continue')。

Execute a function within the context of every matched element.

This means that every time the passed-in function is executed (which is once for every element matched) the 'this' keyword points to the specific DOM element.

Additionally, the function, when executed, is passed a single argument representing the position of the element in the matched set (integer, zero-index).

Returning 'false' from within the each function completely stops the loop through all of the elements (this is like using a 'break' with a normal loop). Returning 'true' from within the loop skips to the next iteration (this is like using a 'continue' with a normal loop).

**返回值**

jQuery

**参数**

**callback** (Function) : 对于每个匹配的元素所要执行的函数

**示例**

|  |
| --- |
| 迭代两个图像，并设置它们的 src 属性。注意:此处 this 指代的是 DOM 对象而非 jQuery 对象。  **HTML 代码:**  <img/><img/>  **jQuery 代码:**  $("img").each(function(i){  this.src = "test" + i + ".jpg";   });  **结果:**  [ <img src="test0.jpg" />, <img src="test1.jpg" /> ] |
| 如果你想得到 jQuery对象，可以使用 $(this) 函数。  **jQuery 代码:**  $("img").each(function(){  $(this).toggleClass("example");  }); |
| 你可以使用 'return false' 来提前跳出 each() 循环。  **HTML 代码:**  <button>Change colors</button>  <span></span>  <div></div>  <div></div>  <div></div>  <div></div>  <div id="stop">Stop here</div>  <div></div>  <div></div>  <div></div>  **jQuery 代码:**  $("button").click(function () {  $("div").each(function (index, domEle) {  // domEle == this  $(domEle).css("backgroundColor", "yellow");  if ($(this).is("#stop")) {  $("span").text("Stopped at div index #" + index);  return false;  }  });  }); |

### size() 这个函数的返回值与 jQuery 对象的'length' 属性一致

jQuery 对象中元素的个数。

这个函数的返回值与 jQuery 对象的'length' 属性一致。

The number of elements in the jQuery object.

This returns the same number as the 'length' property of the jQuery object.

**返回值**

Number

**示例**

|  |
| --- |
| 计算文档中所有图片数量  **HTML 代码:**  <img src="test1.jpg"/> <img src="test2.jpg"/>  **jQuery 代码:**  $("img").size();  **结果:**  2 |

### length 当前匹配的元素个数。 size 将返回相同的值。

jQuery 对象中元素的个数。

当前匹配的元素个数。 size 将返回相同的值。

The number of elements in the jQuery object.

The number of elements currently matched. The size function will return the same value.

**返回值**Number

**示例**

|  |
| --- |
| 计算文档中所有图片数量  **HTML 代码:**  <img src="test1.jpg"/> <img src="test2.jpg"/>  **jQuery 代码:**  $("img").length;  **结果:**  2  get() |

### get() 取得所有匹配的 DOM 元素集合。如果你想要直接操作 DOM 对象而不是 jQuery 对象，这个函数非常有用.

取得所有匹配的 DOM 元素集合。

这是取得所有匹配元素的一种向后兼容的方式（不同于jQuery对象，而实际上是元素数组）。

如果你想要直接操作 DOM 对象而不是 jQuery 对象，这个函数非常有用。

Access all matched DOM elements.

This serves as a backwards-compatible way of accessing all matched elements (other than the jQuery object itself, which is, in fact, an array of elements). It is useful if you need to operate on the DOM elements themselves instead of using built-in jQuery functions.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 选择文档中所有图像作为元素数组，并用数组内建的 reverse 方法将数组反向。  **HTML 代码:**  <img src="test1.jpg"/> <img src="test2.jpg"/>  **jQuery 代码:**  $("img").get().reverse(); // reverse()方法将数组反向  **结果:**  [ <img src="test2.jpg"/> <img src="test1.jpg"/> ] |

### get(index) 取得其中一个匹配的元素(实际的DOM 元素)。num表示取得第几个匹配的元素。

取得其中一个匹配的元素。num表示取得第几个匹配的元素。

这能够让你选择一个实际的DOM 元素并且对他直接操作，而不是通过 jQuery 函数。$(this).get(0)与$(this)[0]等价。

Access a single matched DOM element at a specified index in the matched set.

This allows you to extract the actual DOM element and operate on it directly without necessarily using jQuery functionality on it. This function called as $(this).get(0) is the equivalent of using square bracket notation on the jQuery object itself like $(this)[0].

**返回值**Element

**参数**

**index** (Number) :取得第 index 个位置上的元素

**示例**

|  |
| --- |
| **HTML 代码:**  <img src="test1.jpg"/> <img src="test2.jpg"/>  **jQuery 代码:**  $("img").get(0);  **结果:**  [ <img src="test1.jpg"/> ] |

### index(subject) 搜索与参数表示的对象匹配的元素，并返回相应元素的索引值值。

搜索与参数表示的对象匹配的元素，并返回相应元素的索引值值。

如果找到了匹配的元素，从0开始返回；如果没有找到匹配的元素，返回-1。

Searches every matched element for the object and returns the index of the element, if found, starting with zero.

Returns -1 if the object wasn't found.

**返回值**Number

**参数**

**subject** (Element) : 要搜索的对象

**示例**

|  |
| --- |
| 返回ID值为foobar的元素的索引值值。  **HTML 代码:**  <div id="foobar"><b></b><span id="foo"></span></div>  **jQuery 代码:**  $("\*").index($('#foobar')[0])  **结果:**  5 |

## 插件机制

### jQuery.fn.extend(object) 扩展 jQuery 元素集来提供新的方法（通常用来制作插件）。

扩展 jQuery 元素集来提供新的方法（通常用来制作插件）。

查看这里[Plugins/Authoring](http://docs.jquery.com/Plugins/Authoring)可以获取更多信息。

Extends the jQuery element set to provide new methods (used to make a typical jQuery plugin).

Can be used to add functions into the to add [plugin methods (plugins)](http://docs.jquery.com/Plugins/Authoring).

**返回值**

jQuery

**参数**

**object** (Object) :用来扩充 jQuery 对象。

**示例**

|  |
| --- |
| 增加两个插件方法。  **jQuery 代码:**  jQuery.fn.extend({    check: function() {      return this.each(function() { this.checked = true; });    },    uncheck: function() {      return this.each(function() { this.checked = false; });    }  });  **结果:**  $("input[@type=checkbox]").check();  $("input[@type=radio]").uncheck(); |

### jQuery.extend(object) 扩展jQuery对象本身。

扩展jQuery对象本身。

用来在jQuery命名空间上增加新函数。 查看 'jQuery.fn.extend' 获取更多添加[插件](http://docs.jquery.com/Plugins/Authoring)的信息。

Extends the jQuery object itself.

Can be used to add functions into the jQuery namespace. See 'jQuery.fn.extend' for more information on using this method to add [Plugins](http://docs.jquery.com/Plugins/Authoring).

**返回值**

jQuery

**参数**

**object** (Object) : 用以扩展 jQuery 对象

**示例**

|  |
| --- |
| 在jQuery命名空间上增加两个函数。  **jQuery 代码:**  jQuery.extend({    min: function(a, b) { return a < b ? a : b; },    max: function(a, b) { return a > b ? a : b; }  });  **结果:**  jQuery.min(2,3); // => 2  jQuery.max(4,5); // => 5 |

## 多库共存

### jQuery.noConflict() 这有助于确保jQuery不会与其他库的$对象发生冲突。在运行这个函数后，就只能使用jQuery变量访问jQuery对象。

运行这个函数将变量$的控制权让渡给第一个实现它的那个库。

这有助于确保jQuery不会与其他库的$对象发生冲突。在运行这个函数后，就只能使用jQuery变量访问jQuery对象。例如，在要用到$("div p")的地方，就必须换成jQuery("div p")。

Run this function to give control of the $ variable back to whichever library first implemented it.

This helps to make sure that jQuery doesn't conflict with the $ object of other libraries. By using this function, you will only be able to access jQuery using the 'jQuery' variable. For example, where you used to do $("div p"), you now must do jQuery("div p").

**返回值**

jQuery

**示例**

|  |
| --- |
| 将$引用的对象映射回原始的对象。  **jQuery 代码:**  jQuery.noConflict();  // 使用 jQuery  jQuery("div p").hide();  // 使用其他库的 $()  $("content").style.display = 'none'; |
| 恢复使用别名$，然后创建并执行一个函数，在这个函数的作用域中仍然将$作为jQuery的别名来使用。在这个函数中，原来的$对象是无效的。这个函数对于大多数不依赖于其他库的插件都十分有效。  **jQuery 代码:**  jQuery.noConflict();  (function($) {    $(function() {      // 使用 $ 作为 jQuery 别名的代码    });  })(jQuery);  // 其他用 $ 作为别名的库的代码 |
| 创建一个新的别名用以在接下来的库中使用jQuery对象。  **jQuery 代码:**  var j = jQuery.noConflict();  // 基于 jQuery 的代码  j("div p").hide();  // 基于其他库的 $() 代码  $("content").style.display = 'none'; |

### jQuery.noConflict(extreme) 将$和jQuery的控制权都交还给原来的库。

将$和jQuery的控制权都交还给原来的库。**用之前请考虑清楚!**

这是相对于简单的 **noConflict** 方法更极端的版本，因为这将完全重新定义jQuery。这通常用于一种极端的情况，比如你想要将jQuery嵌入一个高度冲突的环境。**注意:**调用此方法后极有可能导致插件失效。

### Revert control of both the $ and jQuery variables to their original owners. **Use with discretion.**

This is a more-extreme version of the simple **noConflict** method, as this one will completely undo what jQuery has introduced. This is to be used in an extreme case where you'd like to embed jQuery into a high-conflict environment. **NOTE:** It's very likely that plugins won't work after this particular method has been called.

返回值

jQuery

**参数**

**extreme** (Boolean) : 传入 true 来允许彻底将jQuery变量还原

**示例**

|  |
| --- |
| 完全将 jQuery 移到一个新的命名空间。  **jQuery 代码:**  var dom = {};  dom.query = jQuery.noConflict(true);  **结果:**  // 新 jQuery 的代码  dom.query("div p").hide();  // 另一个库 $() 的代码  $("content").style.display = 'none';  // 另一个版本 jQuery 的代码  jQuery("div > p").hide(); |

# 选择器

## 基本

### $("#idName");

**#id**

根据给定的ID匹配一个元素。

Matches a single element with the given id attribute.

**返回值**Element

**参数**

**id** (String) : 用于搜索的，通过元素的 id 属性中给定的值

**示例**

|  |
| --- |
| 查找 ID 为"myDiv"的元素。  **HTML 代码:**  <div id="notMe"><p>id="notMe"</p></div>  <div id="myDiv">id="myDiv"</div>  **jQuery 代码:**  $("#myDiv");  **结果:**  [ <div id="myDiv">id="myDiv"</div> ] |

### $("p");

根据给定的元素名匹配所有元素

Matches all elements with the given name.

**返回值**Array<Element>

**参数**

**element** (String) : 一个用于搜索的元素。指向 DOM 节点的标签名。

**示例**

|  |
| --- |
| 查找一个 DIV 元素。  **HTML 代码:**  <div>DIV1</div>  <div>DIV2</div>  <span>SPAN</span>  **jQuery 代码:**  $("div");  **结果:**  [ <div>DIV1</div>, <div>DIV2</div> ] |

### $(".className");

根据给定的类匹配元素。

Matches all elements with the given class.

**返回值**Array<Element>

**参数**

**class** (String) : 一个用以搜索的类。一个元素可以有多个类，只要有一个符合就能被匹配到。

**示例**

|  |
| --- |
| 查找所有类是 "myClass" 的元素.  **HTML 代码:**  <div class="notMe">div class="notMe"</div>  <div class="myClass">div class="myClass"</div>  <span class="myClass">span class="myClass"</span>  **jQuery 代码:**  $(".myClass");  **结果:**  [ <div class="myClass">div class="myClass"</div>, <span class="myClass">span class="myClass"</span> ] |

### $("\*")

匹配所有元素, 多用于结合上下文来搜索。

Matches all elements.

Most useful when combined with a context to search in.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 找到每一个元素  **HTML 代码:**  <div>DIV</div>  <span>SPAN</span>  <p>P</p>  **jQuery 代码:**  $("\*")  **结果:**  [ <div>DIV</div>, <span>SPAN</span>, <p>P</p> ] |

### $("div,span,p.myClass")

将每一个选择器匹配到的元素合并后一起返回。你可以指定任意多个选择器，并将匹配到的元素合并到一个结果内。

Matches the combined results of all the specified selectors.

You can specify any number of selectors to combine into a single result.

**返回值**Array<Element>

**参数**

**selector1** (Selector) : 一个有效的选择器

**selector2** (Selector) : 另一个有效的选择器

**selectorN** (Selector) : (可选) 任意多个有效选择器

**示例**

|  |
| --- |
| 找到匹配任意一个类的元素。  **HTML 代码:**  <div>div</div>  <p class="myClass">p class="myClass"</p>  <span>span</span>  <p class="notMyClass">p class="notMyClass"</p>  **jQuery 代码:**  $("div,span,p.myClass")  **结果:**  [ <div>div</div>, <p class="myClass">p class="myClass"</p>, <span>span</span> ] |

## 层级

### $("form input")(所有子孙元素)

在给定的祖先元素下匹配所有的后代元素(所有子孙元素)

Matches all descendant elements specified by descendant of elements specified by ancestor.

**返回值**Array<Element>

**参数**

**ancestor** (Selector) : 任何有效选择器

**descendant** (Selector) : 用以匹配元素的选择器，并且它是第一个选择器的后代元素

**示例**

|  |
| --- |
| 找到表单中所有的 input 元素  **HTML 代码:**  <form>    <label>Name:</label>    <input name="name" />    <fieldset>        <label>Newsletter:</label>        <input name="newsletter" />   </fieldset>  </form>  <input name="none" />  **jQuery 代码:**  $("form input")  **结果:**  [ <input name="name" />, <input name="newsletter" /> ] |

### $("form > input")(所有子元素)

在给定的父元素下匹配所有的子元素(所有子元素)

Matches all child elements specified by child of elements specified by parent.

**返回值**Array<Element>

**参数**

**parent** (Selector) : 任何有效选择器

**child** (Selector) : 用以匹配元素的选择器，并且它是第一个选择器的子元素

**示例**

|  |
| --- |
| 匹配表单中所有的子级input元素。  **HTML 代码:**  <form>    <label>Name:</label>    <input name="name" />    <fieldset>        <label>Newsletter:</label>        <input name="newsletter" />   </fieldset>  </form>  <input name="none" />  **jQuery 代码:**  $("form > input")  **结果:**  [ <input name="name" /> ] |

### $("label + input")(挨着我, 右边是XX的元素)

匹配所有紧接在 prev 元素后的 next 元素()(挨着我, 右边是XX的元素)

Matches all next elements specified by next that are next to elements specified by prev.

**返回值**Array<Element>

**参数**

**prev** (Selector) : 任何有效选择器

**next** (Selector) :一个有效选择器并且紧接着第一个选择器

**示例**

|  |
| --- |
| 匹配所有跟在 label 后面的 input 元素  **HTML 代码:**  <form>    <label>Name:</label>    <input name="name" />    <fieldset>        <label>Newsletter:</label>        <input name="newsletter" />   </fieldset>  </form>  <input name="none" />  **jQuery 代码:**  $("label + input")  **结果:**  [ <input name="name" />, <input name="newsletter" /> ] |

### $("form ~ input")(我右边所有是XX的元素)

匹配 prev 元素之后的所有 siblings 元素(我右边所有的是XX的元素)

Matches all sibling elements after the "prev" element that match the filtering "siblings" selector.

**返回值**Array<Element>

**参数**

**prev** (Selector) : 任何有效选择器

**siblings** (Selector) : 一个选择器，并且它作为第一个选择器的同辈

**示例**

|  |
| --- |
| 找到所有与表单同辈的 input 元素  **HTML 代码:**  <form>    <label>Name:</label>    <input name="name" />    <fieldset>        <label>Newsletter:</label>        <input name="newsletter" />   </fieldset>  </form>  <input name="none" />  **jQuery 代码:**  $("form ~ input")  **结果:**  [ <input name="none" /> ] |

## 简单

### $("tr:first")

匹配找到的第一个元素

Matches the first selected element.

**返回值**Element

**示例**

|  |
| --- |
| 查找表格的第一行  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:first")  **结果:**  [ <tr><td>Header 1</td></tr> ] |

### $("tr:last")

匹配找到的最后一个元素

Matches the last selected element.

**返回值** Element

**示例**

|  |
| --- |
| 查找表格的最后一行  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:last")  **结果:**  [ <tr><td>Value 2</td></tr> ]  :not(selector) |

### $("input:not(:checked)")

去除所有与给定选择器匹配的元素

Removes all elements matching the given selector.

**返回值**Array<Element>

**参数**

**selector** (Selector) : 用于筛选的选择器

**示例**

|  |
| --- |
| 查找所有未选中的 input 元素  **HTML 代码:**  <input name="apple" />  <input name="flower" checked="checked" />  **jQuery 代码:**  $("input:not(:checked)")  **结果:**  [ <input name="apple" /> ] |

### $("tr:even")

匹配所有索引值为偶数的元素，从 0 开始计数

Matches even elements, zero-indexed.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找表格的1、3、5...行（即索引值0、2、4...）  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:even")  **结果:**  [ <tr><td>Header 1</td></tr>, <tr><td>Value 2</td></tr> ] |

### $("tr:odd")

匹配所有索引值为奇数的元素，从 0 开始计数

Matches odd elements, zero-indexed.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找表格的2、4、6行（即索引值1、3、5...）  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:odd")  **结果:**  [ <tr><td>Value 1</td></tr> ] |

### $("tr:eq(1)")

匹配一个给定索引值的元素

Matches a single element by its index.

**返回值**Element

**参数**

**index** (Number) : 从 0 开始计数

**示例**

|  |
| --- |
| 查找第二行  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:eq(1)")  **结果:**  [ <tr><td>Value 1</td></tr> ] |

### $("tr:gt(0)")

匹配所有大于给定索引值的元素

Matches all elements with an index above the given one.

**返回值**Array<Element>

**参数**

**index** (Number) : 从 0 开始计数

**示例**

|  |
| --- |
| 查找第二第三行，即索引值是1和2，也就是比0大  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:gt(0)")  **结果:**  [ <tr><td>Value 1</td></tr>, <tr><td>Value 2</td></tr> ] |

### $("tr:lt(2)")

匹配所有小于给定索引值的元素

Matches all elements with an index below the given one.

**返回值**Array<Element>

**参数**

**index** (Number) : 从 0 开始计数

**示例**

|  |
| --- |
| 查找第一第二行，即索引值是0和1，也就是比2小  **HTML 代码:**  <table>    <tr><td>Header 1</td></tr>    <tr><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:lt(2)")  **结果:**  [ <tr><td>Header 1</td></tr>, <tr><td>Value 1</td></tr> ] |

### $(":header")( h1, h2, h3之类的标题元素)

匹配如 h1, h2, h3之类的标题元素

Matches all elements that are headers, like h1, h2, h3 and so on.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 给页面内所有标题加上背景色  **HTML 代码:**  <h1>Header 1</h1>  <p>Contents 1</p>  <h2>Header 2</h2>  <p>Contents 2</p>  **jQuery 代码:**  $(":header").css("background", "#EEE");  **结果:**  [ <h1 style="background:#EEE;">Header 1</h1>, <h2 style="background:#EEE;">Header 2</h2> ] |

### :animated(所有没有在执行动画效果中的元素)

匹配所有没有在执行动画效果中的元素

Matches all elements that are currently being animated.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 只有对不在执行动画效果的元素执行一个动画特效  **HTML 代码:**  <button id="run">Run</button><div></div>  **jQuery 代码:**  $("#run").click(function(){    $("div:not(:animated)").animate({ left: "+20" }, 1000);  }); |

## 内容

### $("div:contains('John')")

匹配包含给定文本的元素

Matches elements which contain the given text.

**返回值**Array<Element>

**参数**

**text** (String) : 一个用以查找的字符串

**示例**

|  |
| --- |
| 查找所有包含 "John" 的 div 元素  **HTML 代码:**  <div>John Resig</div>  <div>George Martin</div>  <div>Malcom John Sinclair</div>  <div>J. Ohn  **jQuery 代码:**  $("div:contains('John')")  **结果:**  [ <div>John Resig</div>, <div>Malcom John Sinclair</div> ] |

### $("td:empty")

匹配所有 不包含子元素或者文本的空元素

Matches all elements that are empty, be it elements or text.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有不包含子元素或者文本的空元素  **HTML 代码:**  <table>    <tr><td>Value 1</td><td></td></tr>    <tr><td>Value 2</td><td></td></tr>  </table>  **jQuery 代码:**  $("td:empty")  **结果:**  [ <td></td>, <td></td> ] |

### $("div:has(p)")

匹配含有选择器所匹配的元素的元素

Matches elements which contain at least one element that matches the specified selector.

**返回值**Array<Element>

**参数**

**selector** (Selector) : 一个用于筛选的选择器

**示例**

|  |
| --- |
| 给所有包含 p 元素的 div 元素添加一个 text 类  **HTML 代码:**  <div><p>Hello</p></div>  <div>Hello again!</div>  **jQuery 代码:**  $("div:has(p)").addClass("test");  **结果:**  [ <div class="test"><p>Hello</p></div> ] |

### :parent(含有子元素或者文本的元素)

匹配含有子元素或者文本的元素

Matches all elements that are parents - they have child elements, including text.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有含有子元素或者文本的 td 元素  **HTML 代码:**  <table>    <tr><td>Value 1</td><td></td></tr>    <tr><td>Value 2</td><td></td></tr>  </table>  **jQuery 代码:**  $("td:parent")  **结果:**  [ <td>Value 1</td>, <td>Value 2</td> ] |

## 可见性

### :hidden

匹配所有的不可见元素，input 元素的 type 属性为 "hidden" 的话也会被匹配到

Matches all elements that are hidden, or input elements of type "hidden".

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有不可见的 tr 元素  **HTML 代码:**  <table>    <tr style="display:none"><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:hidden")  **结果:**  [ <tr style="display:none"><td>Value 1</td></tr> ] |

### :visible

匹配所有的可见元素

Matches all elements that are visible.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有可见的 tr 元素  **HTML 代码:**  <table>    <tr style="display:none"><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:visible")  **结果:**  [ <tr><td>Value 2</td></tr> ] |

## 属性

### [attribute]( 包含给定属性)

匹配包含给定属性的元素

Matches elements that have the specified attribute.

**返回值**Array<Element>

**参数**

**attribute** (String) : 属性名

**示例**

|  |
| --- |
| 查找所有含有 id 属性的 div 元素  **HTML 代码:**  <div>    <p>Hello!</p>  </div>  <div id="test2"></div>  **jQuery 代码:**  $("div[id]")  **结果:**  [ <div id="test2"></div> ]  [attribute=value] |

### [attribute=value]

匹配给定的属性是某个特定值的元素

Matches elements that have the specified attribute with a certain value.

**返回值**Array<Element>

**参数**

**attribute** (String) : 属性名

**value** (String) : 属性值。引号在大多数情况下是可选的。但在遇到诸如属性值包含"]"时，用以避免冲突。

**示例**

|  |
| --- |
| 查找所有 name 属性是 newsletter 的 input 元素  **HTML 代码:**  '<input type="checkbox" name="newsletter" value="Hot Fuzz" />  <input type="checkbox" name="newsletter" value="Cold Fusion" />  <input type="checkbox" name="accept" value="Evil Plans" />  **jQuery 代码:**  $("input[name='newsletter']").attr("checked", true);  **结果:**  [ <input type="checkbox" name="newsletter" value="Hot Fuzz" checked="true" />, <input type="checkbox" name="newsletter" value="Cold Fusion" checked="true" /> ] |

### [attribute!=value]

匹配给定的属性是不包含某个特定值的元素

Matches elements that don't have the specified attribute with a certain value.

**返回值**Array<Element>

**参数**

**attribute** (String) : 属性名

**value** (String) : 属性值。引号在大多数情况下是可选的。但在遇到诸如属性值包含"]"时，用以避免冲突。

**示例**

|  |
| --- |
| 查找所有 name 属性不是 newsletter 的 input 元素  **HTML 代码:**  '<input type="checkbox" name="newsletter" value="Hot Fuzz" />  <input type="checkbox" name="newsletter" value="Cold Fusion" />  <input type="checkbox" name="accept" value="Evil Plans" />  **jQuery 代码:**  $("input[name!='newsletter']").attr("checked", true);  **结果:**  [ <input type="checkbox" name="accept" value="Evil Plans" checked="true" /> ] |

### [attribute^=value]\_\_ 给定的属性是以某些值开始的元素

匹配给定的属性是以某些值开始的元素

Matches elements that have the specified attribute and it starts with a certain value.

**返回值**Array<Element>

**参数**

**attribute** (String) : 属性名

**value** ( String) : 属性值。引号在大多数情况下是可选的。但在遇到诸如属性值包含"]"时，用以避免冲突。

**示例**

|  |
| --- |
| 查找所有 name 以 'news' 开始的 input 元素  **HTML 代码:**  <input name="newsletter" />  <input name="milkman" />  <input name="newsboy" />  **jQuery 代码:**  $("input[name^='news']")  **结果:**  [ <input name="newsletter" />, <input name="newsboy" /> ] |

### [attribute$=value]\_\_ 给定的属性是以某些值结尾

匹配给定的属性是以某些值结尾的元素

Matches elements that have the specified attribute and it ends with a certain value.

**返回值**Array<Element>

**参数**

**attribute** (String) : 属性名

**value** (String) : 属性值。引号在大多数情况下是可选的。但在遇到诸如属性值包含"]"时，用以避免冲突。

**示例**

|  |
| --- |
| 查找所有 name 以 'letter' 结尾的 input 元素  **HTML 代码:**  <input name="newsletter" />  <input name="milkman" />  <input name="jobletter" />  **jQuery 代码:**  $("input[name$='letter']")  **结果:**  [ <input name="newsletter" />, <input name="jobletter" /> ] |

### [attribute\*=value]\_\_ 给定的属性的值包含某字符串的元素

匹配给定的属性的值包含某字符串的元素

Matches elements that have the specified attribute and it contains a certain value.

**返回值**Array<Element>

**参数**

**attribute** (String) : 属性名

**value** (String) : 属性值。引号在大多数情况下是可选的。但在遇到诸如属性值包含"]"时，用以避免冲突。

**示例**

|  |
| --- |
| 查找所有 name 包含 'man' 的 input 元素  **HTML 代码:**  <input name="man-news" />  <input name="milkman" />  <input name="letterman2" />  <input name="newmilk" />  **jQuery 代码:**  $("input[name\*='man']")  **结果:**  [ <input name="man-news" />, <input name="milkman" />, <input name="letterman2" /> ] |

### [selector1][selector2][selectorN]\_\_ 同时满足多个条件

复合属性选择器，需要同时满足多个条件时使用。

Matches elements that have the specified attribute and it contains a certain value.

**返回值**Array<Element>

**参数**

**selector1** (Selector) : 属性选择器

**selector2** (Selector) : 另一个属性选择器，用以进一步缩小范围

**selectorN** (Selector) : 任意多个属性选择器

**示例**

|  |
| --- |
| 找到所有含有 id 属性，并且它的 name 属性是以 man 结尾的  **HTML 代码:**  <input id="man-news" name="man-news" />  <input name="milkman" />  <input id="letterman" name="new-letterman" />  <input name="newmilk" />  **jQuery 代码:**  $("input[id][name$='man']")  **结果:**  [ <input id="letterman" name="new-letterman" /> ] |

## 子元素

### :nth-child(index/even/odd/equation)\_\_ 第N个子或奇偶元素

匹配其父元素下的第N个子或奇偶元素

“:eq(index)” 只匹配一个元素，而这个将为每一个父元素匹配子元素。

:nth-child从1开始的，而:eq()是从0算起的！

可以使用

:nth-child(even)

:nth-child(odd)

:nth-child(2)

:nth-child(3n)

:nth-child(3n+1)

:nth-child(3n+2)

Matches the nth-child of its parent.

While ':eq(index)' matches only a single element, this matches more then one: One for each parent. The specified index is one-indexed, in contrast to :eq() which starst at zero.

**返回值**Array<Element>

**参数**

**index** (Number) : 要匹配元素的序号，从1开始

**示例**

|  |
| --- |
| 在每个 ul 查找第 2 个li  **HTML 代码:**  <ul>    <li>John</li>    <li>Karl</li>    <li>Brandon</li>  </ul>  <ul>    <li>Glen</li>    <li>Tane</li>    <li>Ralph</li>  </ul>  **jQuery 代码:**  $("ul li:nth-child(2)")  **结果:**  [ <li>Karl</li>,   <li>Tane</li> ] |

### :first-child

匹配第一个子元素

“:first” 只匹配一个元素，而此选择符将为每个父元素匹配一个子元素

Matches the first child of its parent.

While ':first' matches only a single element, this matches more then one: One for each parent.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 在每个 ul 中查找第一个 li  **HTML 代码:**  <ul>    <li>John</li>    <li>Karl</li>    <li>Brandon</li>  </ul>  <ul>    <li>Glen</li>    <li>Tane</li>    <li>Ralph</li>  </ul>  **jQuery 代码:**  $("ul li:first-child")  **结果:**  [ <li>John</li>, <li>Glen</li> ] |

### :last-child

匹配最后一个子元素

':last'只匹配一个元素，而此选择符将为每个父元素匹配一个子元素

Matches the last child of its parent.

While ':last' matches only a single element, this matches more then one: One for each parent.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 在每个 ul 中查找最后一个 li  **HTML 代码:**  <ul>    <li>John</li>    <li>Karl</li>    <li>Brandon</li>  </ul>  <ul>    <li>Glen</li>    <li>Tane</li>    <li>Ralph</li>  </ul>  **jQuery 代码:**  $("ul li:last-child")  **结果:**  [ <li>Brandon</li>, <li>Ralph</li> ] |

### :only-child\_\_如果某个元素是父元素中唯一的子元素，那将会被匹配,

如果某个元素是父元素中唯一的子元素，那将会被匹配, 如果父元素中含有其他元素，那将不会被匹配。

Matches the only child of its parent.

If the parent has other child elements, nothing is matched.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 在 ul 中查找是唯一子元素的 li  **HTML 代码:**  <ul>    <li>John</li>    <li>Karl</li>    <li>Brandon</li>  </ul>  <ul>    <li>Glen</li>  **jQuery 代码:**  $("ul li:only-child")  **结果:**  [ <li>Glen</li> ] |

## 表单

### :input

匹配所有 input, textarea, select 和 button 元素

Matches all input, textarea, select and button elements.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有的input元素  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":input")  **结果:**  [ <input type="text" />, <input type="checkbox" />, <input type="radio" />, <input type="image" />, <input type="file" />, <input type="submit" />, <input type="reset" />, <input type="password" />, <input type="button" /> ] |

### :text

匹配所有的单行文本框

Matches all input elements of type text.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有文本框  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":text")  **结果:**  [ <input type="text" /> ] |

### :password

匹配所有密码框

Matches all input elements of type password.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有密码框  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":password")  **结果:**  [ <input type="password" /> ]  :radio |

### :radio

匹配所有单选按钮

Matches all input elements of type radio.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有单选按钮  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":radio")  **结果:**  [ <input type="radio" /> ] |

### :checkbox\_\_复选框

匹配所有复选框

Matches all input elements of type checkbox.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有复选框  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":checkbox")  **结果:**  [ <input type="checkbox" /> ]  :submit |

### :submit

匹配所有提交按钮

Matches all input elements of type submit.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有提交按钮  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":submit")  **结果:**  [ <input type="submit" /> ]  :image |

### :image\_\_图像域<input type="image" />

匹配所有图像域

Matches all input elements of type image.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 匹配所有图像域  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":image")  **结果:**  [ <input type="image" /> ]  :reset |

### :reset

匹配所有重置按钮

Matches all input elements of type reset.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有重置按钮  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":reset")  **结果:**  [ <input type="reset" /> ]  :button |

### :button

匹配所有按钮

包括

|  |
| --- |
| <input type="button" />  <button></button> |

Matches all input elements of type button.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有按钮.  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":button")  **结果:**  [ <input type="button" />,<button></button> ] |

### :file\_\_文件域

匹配所有文件域

Matches all input elements of type file.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有文件域  **HTML 代码:**  <form>    <input type="text" />    <input type="checkbox" />    <input type="radio" />    <input type="image" />    <input type="file" />    <input type="submit" />    <input type="reset" />    <input type="password" />    <input type="button" />    <select><option/></select>    <textarea></textarea>    <button></button>  </form>  **jQuery 代码:**  $(":file")  **结果:**  [ <input type="file" /> ] |

### :hidden

匹配所有不可见元素，或者type为hidden的元素

Matches all elements that are hidden, or input elements of type "hidden".

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找隐藏的 tr  **HTML 代码:**  <table>    <tr style="display:none"><td>Value 1</td></tr>    <tr><td>Value 2</td></tr>  </table>  **jQuery 代码:**  $("tr:hidden")  **结果:**  [ <tr style="display:none"><td>Value 1</td></tr> ] |
| 匹配type为hidden的元素  **HTML 代码:**  <form>    <input type="text" name="email" />    <input type="hidden" name="id" />  </form>  **jQuery 代码:**  $("input:hidden")  **结果:**  [ <input type="hidden" name="id" /> ] |

## 表单对象属性

### :enabled\_\_可用

匹配所有可用元素

Matches all elements that are enabled.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有可用的input元素  **HTML 代码:**  <form>    <input name="email" disabled="disabled" />    <input name="id" />  </form>  **jQuery 代码:**  $("input:enabled")  **结果:**  [ <input name="id" /> ] |

### :disabled

匹配所有不可用元素

Matches all elements that are disabled.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有不可用的input元素  **HTML 代码:**  <form>    <input name="email" disabled="disabled" />    <input name="id" />  </form>  **jQuery 代码:**  $("input:disabled")  **结果:**  [ <input name="email" disabled="disabled" /> ] |

### :checked\_\_选中的复选框

匹配所有选中的复选框元素

Matches all elements that are checked.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有选中的复选框元素  **HTML 代码:**  <form>    <input type="checkbox" name="newsletter" checked="checked" value="Daily" />    <input type="checkbox" name="newsletter" value="Weekly" />    <input type="checkbox" name="newsletter" checked="checked" value="Monthly" />  </form>  **jQuery 代码:**  $("input:checked")  **结果:**  [ <input type="checkbox" name="newsletter" checked="checked" value="Daily" />, <input type="checkbox" name="newsletter" checked="checked" value="Monthly" /> ] |

### :selected\_\_选中的选项

匹配所有选中的选项元素

Matches all elements that are selected.

**返回值**Array<Element>

**示例**

|  |
| --- |
| 查找所有选中的选项元素  **HTML 代码:**  <select>    <option value="1">Flowers</option>    <option value="2" selected="selected">Gardens</option>    <option value="3">Trees</option>  </select>  **jQuery 代码:**  $("select option:selected")  **结果:**  [ <option value="2" selected="selected">Gardens</option> ] |
| Finds all option elements that are selected.  **HTML 代码:**  <select multiple="multiple">    <option value="1">Flowers</option>    <option value="2" selected="selected">Gardens</option>    <option value="3" selected="selected">Trees</option>  </select>  **jQuery 代码:**  $("select option:selected")  **结果:**  [ <option value="2" selected="selected">Gardens</option>, <option value="3" selected="selected">Trees</option> ] |

# 属性

## 属性

### attr(name)\_\_ 第一个匹配元素的属性值

取得第一个匹配元素的属性值。通过这个方法可以方便地从第一个匹配元素中获取一个属性的值。如果元素没有相应属性，则返回 undefined 。

Access a property on the first matched element. This method makes it easy to retrieve a property value from the first matched element. If the element does not have an attribute with such a name, undefined is returned.

**返回值**Object

**参数**

**name** (String) : 属性名称

**示例**

|  |
| --- |
| 返回文档中第一个图像的src属性值。  **HTML 代码:**  <img src="test.jpg"/>  **jQuery 代码:**  $("img").attr("src");  **结果:**  test.jpg |

### attr(properties)

将一个“名/值”形式的对象设置为所有匹配元素的属性。

这是一种在所有匹配元素中批量设置很多属性的最佳方式。 注意，如果你要设置对象的class属性，你必须使用'className' 作为属性名。或者你可以直接使用.addClass( class ) 和 .removeClass( class ).

Set a key/value object as properties to all matched elements.

This serves as the best way to set a large number of properties on all matched elements. Note that you must use 'className' as key if you want to set the class-Attribute. Or use .addClass( class ) or .removeClass( class ).

**返回值**

jQuery

**参数**

**properties** (Map) : 作为属性的“名/值对”对象

**示例**

|  |
| --- |
| 为所有图像设置src和alt属性。  **HTML 代码:**  <img/>  **jQuery 代码:**  $("img").attr({ src: "test.jpg", alt: "Test Image" });  **结果:**  [ <img src= "test.jpg"  alt:="Test Image" /> ] |

### attr(key,value)

为所有匹配的元素设置一个属性值。

Set a single property to a value, on all matched elements.

**返回值**

jQuery

**参数**

**key** (String) : 属性名称

**value** (Object) : 属性值

**示例**

|  |
| --- |
| 为所有图像设置src属性。  **HTML 代码:**  <img/>  <img/>  **jQuery 代码:**  $("img").attr("src","test.jpg");  **结果:**  [ <img src= "test.jpg" /> , <img src= "test.jpg" /> ] |

### attr(key,fn)

为所有匹配的元素设置一个计算的属性值。

不提供值，而是提供一个函数，由这个函数计算的值作为属性值。

Set a single property to a computed value, on all matched elements.

Instead of supplying a string value as described 'above', a function is provided that computes the value.

**返回值**

jQuery

**参数**

**key** (String) : 属性名称

**fn** (Function) : 返回值的函数 范围:当前元素, 参数: 当前元素的索引值

**示例**

|  |
| --- |
| 把src属性的值设置为title属性的值。  **HTML 代码:**  <img src="test.jpg"/>  **jQuery 代码:**  $("img").attr("title", function() { return this.src });  **结果:**  <img src="test.jpg" title="test.jpg" /> |

### removeAttr(name)

从每一个匹配的元素中删除一个属性

Remove an attribute from each of the matched elements.

**返回值**

jQuery

**参数**

**name** (String) : 要删除的属性名

**示例**

|  |
| --- |
| 将文档中图像的src属性删除  **HTML 代码:**  <img src="test.jpg"/>  **jQuery 代码:**  $("img").removeAttr("src");  **结果:**  [ <img /> ]  类  addClass(class) |

## CSS类

### addClass(class)

为每个匹配的元素添加指定的类名。

Adds the specified class(es) to each of the set of matched elements.

**返回值**

jQuery

**参数**

**class** (String) : 一个或多个要添加到元素中的CSS类名，请用空格分开

**示例**

|  |
| --- |
| 为匹配的元素加上 'selected' 类  **HTML 代码:**  <p>Hello</p>  **jQuery 代码:**  $("p").addClass("selected");  **结果:**  [ <p class="selected">Hello</p> ] |
| 为匹配的元素加上 selected highlight 类  **HTML 代码:**  <p>Hello</p>  **jQuery 代码:**  $("p").addClass("selected highlight");  **结果:**  [ <p class="selected highlight">Hello</p> ] |

### removeClass(class)

从所有匹配的元素中删除全部或者指定的类。

Removes all or the specified class(es) from the set of matched elements.

**返回值**

jQuery

**参数**

**class** (String) : (可选) 一个或多个要删除的CSS类名，请用空格分开

**示例**

|  |
| --- |
| 从匹配的元素中删除 'selected' 类  **HTML 代码:**  <p class="selected first">Hello</p>  **jQuery 代码:**  $("p").removeClass("selected");  **结果:**  [ <p>Hello</p> ] |
| 删除匹配元素的所有类  **HTML 代码:**  <p class="selected first">Hello</p>  **jQuery 代码:**  $("p").removeClass();  **结果:**  [ <p>Hello</p> ]  toggleClass(class) |

### toggleClass(class)

如果存在（不存在）就删除（添加）一个类。

Adds the specified class if it is not present, removes the specified class if it is present.

**返回值**

jQuery

**参数**

**class** (String) :CSS类名

**示例**

|  |
| --- |
| 为匹配的元素切换 'selected' 类  **HTML 代码:**  <p>Hello</p><p class="selected">Hello Again</p>  **jQuery 代码:**  $("p").toggleClass("selected");  **结果:**  [ <p class="selected">Hello</p>, <p>Hello Again</p> ]  Html代码 |

## Html

### html()

取得第一个匹配元素的html内容。这个函数不能用于XML文档。但可以用于XHTML文档。

Get the html contents of the first matched element. This property is not available on XML documents (although it will work for XHTML documents).

**返回值**String

**示例**

|  |
| --- |
| **HTML 代码:**  <div><p>Hello</p></div>  **jQuery 代码:**  $("div").html();  **结果:**  Hello |

### html(val)

设置每一个匹配元素的html内容。这个函数不能用于XML文档。但可以用于XHTML文档。

Set the html contents of every matched element. This property is not available on XML documents (although it will work for XHTML documents).

**返回值**

jQuery

**参数**

**val** (String) : 用于设定HTML内容的值

**示例**

|  |
| --- |
| **HTML 代码:**  <div></div>  **jQuery 代码:**  $("div").html("<p>Hello Again</p>");  **结果:**  [ <div><p>Hello Again</p></div> ] |

文本

### text()

取得所有匹配元素的内容。

结果是由所有匹配元素包含的文本内容组合起来的文本。这个方法对HTML和XML文档都有效。

Get the text contents of all matched elements.

The result is a string that contains the combined text contents of all matched elements. This method works on both HTML and XML documents.

**返回值**String

**示例**

|  |
| --- |
| **HTML 代码:**  <p><b>Test</b> Paragraph.</p><p>Paraparagraph</p>  **jQuery 代码:**  $("p").text();  **结果:**  Test Paragraph.Paraparagraph |

text(val)

### text(val)

设置所有匹配元素的文本内容

与 html() 类似, 但将编码 HTML (将 "<" 和 ">" 替换成相应的HTML实体).

Set the text contents of all matched elements.

Similar to html(), but escapes HTML (replace "<" and ">" with their HTML entities).

**返回值**

jQuery

**参数**

**val** (String) : 用于设置元素内容的文本

**示例**

|  |
| --- |
| **HTML 代码:**  <p>Test Paragraph.</p>  **jQuery 代码:**  $("p").text("<b>Some</b> new text.");  **结果:**  [ <p><b>Some</b> new text.</p> ] |

值

### val()

获得第一个匹配元素的当前值。

在 jQuery 1.2 中,可以返回任意元素的值了。包括select。如果多选，将返回一个数组，其包含所选的值。

Get the content of the value attribute of the first matched element.

In jQuery 1.2, a value is now returned for all elements, including selects. For multiple selects an array of values is returned.

**返回值**String,Array

**示例**

|  |
| --- |
| 获得单个select的值和多选select的值。  **HTML 代码:**  <p></p><br/>  <select id="single">    <option>Single</option>    <option>Single2</option>  </select>  <select id="multiple" multiple="multiple">    <option selected="selected">Multiple</option>    <option>Multiple2</option>    <option selected="selected">Multiple3</option>  </select>  **jQuery 代码:**  $("p").append(    "<b>Single:</b> "   + $("#single").val() +    " <b>Multiple:</b> " + $("#multiple").val().join(", ")  );  **结果:**  [ <p><b>Single:</b>Single<b>Multiple:</b>Multiple, Multiple3</p>] |
| 获取文本框中的值  **HTML 代码:**  <input type="text" value="some text"/>  **jQuery 代码:**  $("input").val();  **结果:**  some text |

### val(val)

设置每一个匹配元素的值。

在 jQuery 1.2, 这也可以为select元件赋值

Set the value attribute of every matched element.

In jQuery 1.2, this is also able to set the value of select elements, but selecting the appropriate options.

**返回值**

jQuery

**参数**

**val** (String) : 要设置的值。

**示例**

|  |
| --- |
| 设定文本框的值  **HTML 代码:**  <input type="text"/>  **jQuery 代码:**  $("input").val("hello world!"); |

### val(val)

check,select,radio等都能使用为之赋值

**返回值**

jQuery

**参数**

**val** (Array<String>) : 用于 check/select 的值

**示例**

|  |
| --- |
| 设定一个select和一个多选的select的值  **HTML 代码:**  <select id="single">   <option>Single</option>   <option>Single2</option> </select>  <select id="multiple" multiple="multiple">   <option selected="selected">Multiple</option>   <option>Multiple2</option>   <option selected="selected">Multiple3</option> </select>  <input type="checkbox" value="check1"/> check1 <input type="checkbox" value="check2"/> check2  <input type="radio" value="radio1"/> radio1 <input type="radio" value="radio2"/> radio2  **jQuery 代码:**  $("#single").val("Single2"); //当选 $("#multiple").val(["Multiple2", "Multiple3"]); //多选 $("input").val(["check2", "radio1"]); //input类型 |

# 筛选

## 过滤

### eq(index)

获取第N个元素

这个元素的位置是从0算起。

Reduce the set of matched elements to a single element.

The position of the element in the set of matched elements starts at 0 and goes to length - 1.

**返回值**

jQuery

**参数**

**index** (Integer) :元素在jQuery对象中的索引

**示例**

|  |
| --- |
| 获取匹配的第二个元素  **HTML 代码:**  <p> This is just a test.</p> <p> So is this</p>  **jQuery 代码:**  $("p").eq(1)  **结果:**  [ <p> So is this</p> ] |

### hasClass(class)

检查当前的元素是否含有某个特定的类，如果有，则返回true。

这其实就是 is("." + class)。

Checks the current selection against a class and returns true, if at least one element of the selection has the given class.

This is an alternative to is("." + class).

**返回值**Boolean

**参数**

**class** (String) : 用于匹配的类名

**示例**

|  |
| --- |
| 给包含有某个类的元素进行一个动画。  **HTML 代码:**  <div class="protected"></div><div></div>  **jQuery 代码:**  $("div").click(function(){    if ( $(this).hasClass("protected") )      $(this)        .animate({ left: -10 })        .animate({ left: 10 })        .animate({ left: -10 })        .animate({ left: 10 })        .animate({ left: 0 });  }); |

### filter(expr)

筛选出与指定表达式匹配的元素集合。

这个方法用于缩小匹配的范围。用逗号分隔多个表达式

Removes all elements from the set of matched elements that do not match the specified expression(s).

This method is used to narrow down the results of a search. Provide a comma-separated list of expressions to apply multiple filters at once.

**返回值**

jQuery

**参数**

**expr** (Expression) : 表达式

**示例**

|  |
| --- |
| 保留带有select类的元素  **HTML 代码:**  <p>Hello</p><p>Hello Again</p><p class="selected">And Again</p>  **jQuery 代码:**  $("p").filter(".selected")  **结果:**  [ <p class="selected">And Again</p> ] |
| 保留第一个以及带有select类的元素  **HTML 代码:**  <p>Hello</p><p>Hello Again</p><p class="selected">And Again</p>  **jQuery 代码:**  $("p").filter(".selected, :first")  **结果:**  [ <p>Hello</p>, <p class="selected">And Again</p> ] |

### filter(fn)

筛选出与指定函数返回值匹配的元素集合

这个函数内部将对每个对象计算一次 (正如 '$.each'). 如果调用的函数返回false则这个元素被删除，否则就会保留。

Removes all elements from the set of matched elements that does not match the specified function.

The function is called with a context equal to the current element (just like '$.each'). If the function returns false, then the element is removed - anything else and the element is kept.

**返回值**

jQuery

**参数**

**fn** (Function) : 传递进filter的函数

**示例**

|  |
| --- |
| 保留子元素中不含有ol的元素。  **HTML 代码:**  <p><ol><li>Hello</li></ol></p><p>How are you?</p>  **jQuery 代码:**  $("p").filter(function(index) {    return $("ol", this).length == 0;  });  **结果:**  [ <p>How are you?</p> ] |

### is(expr)

用一个表达式来检查当前选择的元素集合，如果其中至少有一个元素符合这个给定的表达式就返回true。

如果没有元素符合，或者表达式无效，都返回'false'. 'filter' 内部实际也是在调用这个函数，所以，filter()函数原有的规则在这里也适用。

Checks the current selection against an expression and returns true, if at least one element of the selection fits the given expression.

If no element fits, or the expression is not valid, then the response will be 'false'. 'filter' is used internally, therefore all rules that apply there apply here, as well.

**返回值**Boolean

**参数**

**expr** (String) :用于筛选的表达式

**示例**

|  |
| --- |
| 由于input元素的父元素是一个表单元素，所以返回true。  **HTML 代码:**  <form><input type="checkbox" /></form>  **jQuery 代码:**  $("input[type='checkbox']").parent().is("form")  **结果:**  true |

### map(callback)

将一组元素转换成其他数组（不论是否是元素数组）

你可以用这个函数来建立一个列表，不论是值、属性还是CSS样式，或者其他特别形式。这都可以用'$.map()'来方便的建立。

Translate a set of elements into another set of values (which may, or may not, be elements).

You could use this to build lists of values, attributes, css values - or even perform special, custom, selector transformations. This is provided as a convenience method for using '$.map()'.

**返回值**

jQuery

**参数**

**callback** (Function) : 给每个元素执行的函数

**示例**

|  |
| --- |
| 把form中的每个input元素的值建立一个列表。  **HTML 代码:**  <p><b>Values: </b></p>  <form>    <input type="text" name="name" value="John"/>    <input type="text" name="password" value="password"/>    <input type="text" name="url" value="http://ejohn.org/"/>  </form>  **jQuery 代码:**  $("p").append( $("input").map(function(){    return $(this).val();  }).get().join(", ") );  **结果:**  [ <p>John, password, http://ejohn.org/</p> ] |

### not(expr)

删除与指定表达式匹配的元素

Removes elements matching the specified expression from the set of matched elements.

**返回值**

jQuery

**参数**

**expr** (String, DOMElement, Array<DOMElement>) : 一个表达式、一个元素或者一组元素

**示例**

|  |
| --- |
| 从p元素中删除带有 select 的ID的元素  **HTML 代码:**  <p>Hello</p><p id="selected">Hello Again</p>  **jQuery 代码:**  $("p").not( $("#selected")[0] )  **结果:**  [ <p>Hello</p> ] |

### slice(start,[end])

选取一个匹配的子集

与原来的slice方法类似

Selects a subset of the matched elements.

Behaves exactly like the built-in Array slice method.

**返回值**

jQuery

**参数**

**start** (Integer) :开始选取子集的位置。第一个元素是0.如果是负数，则可以从集合的尾部开始选起。

**end** (Integer) : (可选) 结束选取自己的位置，如果不指定，则就是本身的结尾。

**示例**

|  |
| --- |
| 选择第一个p元素  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").slice(0, 1).wrapInner("<b></b>");  **结果:**  [ <p><b>Hello</b></p> ] |
| 选择前两个p元素  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").slice(0, 2).wrapInner("<b></b>");  **结果:**  [ <p><b>Hello</b></p>,<p><b>cruel</b></p> ] |
| 只选取第二个p元素  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").slice(1, 2).wrapInner("<b></b>");  **结果:**  [ <p><b>cruel</b></p> ] |
| 只选取第二第三个p元素  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").slice(1).wrapInner("<b></b>");  **结果:**  [ <p><b>cruel</b></p>, <p><b>World</b></p> ] |
| Selects all paragraphs, then slices the selection to include only the third element.  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").slice(-1).wrapInner("<b></b>");  **结果:**  [ <p><b>World</b></p> ]  查找 |

## 查找

### @add(expr)

把与表达式匹配的元素添加到jQuery对象中。这个函数可以用于连接分别与两个表达式匹配的元素结果集。

Adds more elements, matched by the given expression, to the set of matched elements.

**返回值**

jQuery

**参数**

**expr** (String, DOMElement, Array<DOMElement>) : 用于匹配元素并添加的表达式字符串，或者用于动态生成的HTML代码，如果是一个字符串数组则返回多个元素

**示例**

|  |
| --- |
| 添加一个新元素到一组匹配的元素中，并且这个新元素能匹配给定的表达式。  **HTML 代码:**  <p>Hello</p><span>Hello Again</span>  **jQuery 代码:**  $("p").add("span")  **结果:**  [ <p>Hello</p>, <span>Hello Again</span> ] |
| 动态生成一个元素并添加至匹配的元素中  **HTML 代码:**  <p>Hello</p>  **jQuery 代码:**  $("p").add("<span>Again</span>")  **结果:**  [ <p>Hello</p>, <span>Hello Again</span> ] |
| 为匹配的元素添加一个或者多个元素  **HTML 代码:**  <p>Hello</p><p><span id="a">Hello Again</span></p>  **jQuery 代码:**  $("p").add(document.getElementById("a"))  **结果:**  [ <p>Hello</p>, <p><span id="a">Hello Again</span></p>, <span id="a">Hello Again</span> ] |

### children([expr])

取得一个包含匹配的元素集合中每一个元素的所有子元素的元素集合。

可以通过可选的表达式来过滤所匹配的子元素。注意：parents()将查找所有祖辈元素，而children()之考虑子元素而不考虑所有后代元素。

Get a set of elements containing all of the unique children of each of the matched set of elements.

This set can be filtered with an optional expression that will cause only elements matching the selector to be collected. Also note: while parents() will look at all ancestors, children() will only consider immediate child elements.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用以过滤子元素的表达式

**示例**

|  |
| --- |
| 查找DIV中的每个子元素。  **HTML 代码:**  <p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>  **jQuery 代码:**  $("div").children()  **结果:**  [ <span>Hello Again</span> ] |
| 在每个div中查找 .selected 的类。  **HTML 代码:**  <div><span>Hello</span><p class="selected">Hello Again</p><p>And Again</p></div>  **jQuery 代码:**  $("div").children(".selected")  **结果:**  [ <p class="selected">Hello Again</p> ] |

### contents()

查找匹配元素内部所有的子节点（包括文本节点）。如果元素是一个iframe，则查找文档内容

Find all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe.

**返回值**

jQuery

**示例**

|  |
| --- |
| 查找所有文本节点并加粗  **HTML 代码:**  <p>Hello <a href="http://ejohn.org/">John</a>, how are you doing?</p>  **jQuery 代码:**  $("p").contents().not("[@nodeType=1]").wrap("<b/>");  **结果:**  <p><b>Hello</b> <a href="http://ejohn.org/">John</a>, <b>how are you doing?</b></p> |
| 往一个空框架中加些内容  **HTML 代码:**  <iframe src="/index-blank.html" width="300" height="100"></iframe>  **jQuery 代码:**  $("iframe").contents().find("body")   .append("I'm in an iframe!"); |

### find(expr)

搜索所有与指定表达式匹配的元素。这个函数是找出正在处理的元素的后代元素的好方法。

所有搜索都依靠jQuery表达式来完成。这个表达式可以使用CSS1-3的选择器语法来写。

Searches for all elements that match the specified expression. This method is a good way to find additional descendant elements with which to process.

All searching is done using a jQuery expression. The expression can be written using CSS 1-3 Selector syntax.

**返回值**

jQuery

**参数**

**expr** (String) :用于查找的表达式

**示例**

|  |
| --- |
| 从所有的段落开始，进一步搜索下面的span元素。与$("p span")相同。  **HTML 代码:**  <p><span>Hello</span>, how are you?</p>  **jQuery 代码:**  $("p").find("span")  **结果:**  [ <span>Hello</span> ] |

### next([expr])

取得一个包含匹配的元素集合中每一个元素紧邻的后面同辈元素的元素集合。

这个函数只返回后面那个紧邻的同辈元素，而不是后面所有的同辈元素（可以使用nextAll）。可以用一个可选的表达式进行筛选。

Get a set of elements containing the unique next siblings of each of the matched set of elements.

It only returns the very next sibling for each element, not all next siblings (nor does it return the next closest sibling). You may provide an optional expression to filter the match.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用于筛选的表达式

**示例**

|  |
| --- |
| 找到每个段落的后面紧邻的同辈元素。  **HTML 代码:**  <p>Hello</p><p>Hello Again</p><div><span>And Again</span></div>  **jQuery 代码:**  $("p").next()  **结果:**  [ <p>Hello Again</p>, <div><span>And Again</span></div> ] |
| 找到每个段落的后面紧邻的同辈元素中类名为selected的元素。  **HTML 代码:**  <p>Hello</p><p class="selected">Hello Again</p><div><span>And Again</span></div>  **jQuery 代码:**  $("p").next(".selected")  **结果:**  [ <p class="selected">Hello Again</p> ] |

### nextAll([expr])

查找当前元素之后的所有元素。

可以用表达式过滤

Find all sibling elements after the current element.

Use an optional expression to filter the matched set.

**返回值**

jQuery

**参数**

**expr** (String) : (可选)用来过滤的表达式

**示例**

|  |
| --- |
| 给第一个div之后的所有元素加个类  **HTML 代码:**  <div></div><div></div><div></div><div></div>  **jQuery 代码:**  $("div:first").nextAll().addClass("after");  **结果:**  [ <div class="after"></div>, <div class="after"></div>, <div class="after"></div> ] |

### parent([expr])

取得一个包含着所有匹配元素的唯一父元素的元素集合。

你可以使用可选的表达式来筛选。

Get a set of elements containing the unique parents of the matched set of elements.

You may use an optional expression to filter the set of parent elements that will match.

**返回值**

jQuery

**参数**

**expr** (String) : (可选)用来筛选的表达式

**示例**

|  |
| --- |
| 查找每个段落的父元素  **HTML 代码:**  <div><p>Hello</p><p>Hello</p></div>  **jQuery 代码:**  $("p").parent()  **结果:**  [ <div><p>Hello</p><p>Hello</p></div> </body> ] |
| 查找段落的父元素中每个类名为selected的父元素。  **HTML 代码:**  <div><p>Hello</p></div><div class="selected"><p>Hello Again</p></div>  **jQuery 代码:**  $("p").parent(".selected")  **结果:**  [ <div class="selected"><p>Hello Again</p></div> ] |

### parents([expr])

取得一个包含着所有匹配元素的祖先元素的元素集合（不包含根元素）。可以通过一个可选的表达式进行筛选。

Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element). The matched elements can be filtered with an optional expression.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用于筛选祖先元素的表达式

**示例**

|  |
| --- |
| 找到每个span元素的所有祖先元素。  **HTML 代码:**  <html><body><div><p><span>Hello</span></p><span>Hello Again</span></div></body></html>  **jQuery 代码:**  $("span").parents()  找到每个span的所有是p元素的祖先元素。  **jQuery 代码:**  $("span").parents("p") |

### prev([expr])

取得一个包含匹配的元素集合中每一个元素紧邻的前一个同辈元素的元素集合。

可以用一个可选的表达式进行筛选。只有紧邻的同辈元素会被匹配到，而不是前面所有的同辈元素。

Get a set of elements containing the unique previous siblings of each of the matched set of elements.

Use an optional expression to filter the matched set. Only the immediately previous sibling is returned, not all previous siblings.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用于筛选前一个同辈元素的表达式

**示例**

|  |
| --- |
| 找到每个段落紧邻的前一个同辈元素。  **HTML 代码:**  <p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>  **jQuery 代码:**  $("p").prev()  **结果:**  [ <div><span>Hello Again</span></div> ] |
| 找到每个段落紧邻的前一个同辈元素中类名为selected的元素。  **HTML 代码:**  <div><span>Hello</span></div><p class="selected">Hello Again</p><p>And Again</p>  **jQuery 代码:**  $("p").prev(".selected")  **结果:**  [ <p class="selected">Hello Again</p> ] |

### prevAll([expr])

查找当前元素之前所有的同辈元素

可以用表达式过滤。

Find all sibling elements before the current element.

Use an optional expression to filter the matched set.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用于过滤的表达式

**示例**

|  |
| --- |
| 给最后一个之前的所有div加上一个类  **HTML 代码:**  <div></div><div></div><div></div><div></div>  **jQuery 代码:**  $("div:last").prevAll().addClass("before");  **结果:**  [ <div class="before"></div>, <div class="before"></div>, <div class="before"></div> ] |

### siblings([expr])

取得一个包含匹配的元素集合中每一个元素的所有唯一同辈元素的元素集合。可以用可选的表达式进行筛选。

Get a set of elements containing all of the unique siblings of each of the matched set of elements. Can be filtered with an optional expressions.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用于筛选同辈元素的表达式

**示例**

|  |
| --- |
| 找到每个div的所有同辈元素。  **HTML 代码:**  <p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>  **jQuery 代码:**  $("div").siblings()  **结果:**  [ <p>Hello</p>, <p>And Again</p> ] |
| 找到每个div的所有同辈元素中带有类名为selected的元素。  **HTML 代码:**  <div><span>Hello</span></div><p class="selected">Hello Again</p><p>And Again</p>  **jQuery 代码:**  $("p").siblings(".selected")  **结果:**  [ <p class="selected">Hello Again</p> ]  串联 |

## 串联

### andSelf()

加入先前所选的加入当前元素中

对于筛选或查找后的元素，要加入先前所选元素时将会很有用。

Add the previous selection to the current selection.

Useful for traversing elements, and then adding something that was matched before the last traversion.

**返回值**

jQuery

**示例**

|  |
| --- |
| 选取所有div以及内部的p，并加上border类  **HTML 代码:**  <div><p>First Paragraph</p><p>Second Paragraph</p></div>  **jQuery 代码:**  $("div").find("p").andSelf().addClass("border");  **结果:**  <div class="border"><p class="border">First Paragraph</p><p class="border">Second Paragraph</p></div> |

### end()

回到最近的一个"破坏性"操作之前。即，将匹配的元素列表变为前一次的状态。

如果之前没有破坏性操作，则返回一个空集。所谓的"破坏性"就是指任何改变所匹配的jQuery元素的操作。这包括在 Traversing 中任何返回一个jQuery对象的函数--'add', 'andSelf', 'children', 'filter', 'find', 'map', 'next', 'nextAll', 'not', 'parent', 'parents', 'prev', 'prevAll', 'siblings' and 'slice'--再加上 Manipulation 中的 'clone'。

**Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state (right before the destructive operation).**

**If there was no destructive operation before, an empty set is returned. A 'destructive' operation is any operation that changes the set of matched jQuery elements, which means any Traversing function that returns a jQuery object - including 'add', 'andSelf', 'children', 'filter', 'find', 'map', 'next', 'nextAll', 'not', 'parent', 'parents', 'prev', 'prevAll', 'siblings' and slice - plus the 'clone' function (from Manipulation).**

**返回值**

jQuery

**示例**

|  |
| --- |
| 选取所有的p元素，查找并选取span子元素，然后再回过来选取p元素  **HTML 代码:**  <p><span>Hello</span>,how are you?</p>  **jQuery 代码:**  $("p").find("span").end()  **结果:**  [ <p><span>Hello</span> how are you?</p> ] |

# 文档处理

## 内部插入

append(content)

### append(content)\_\_ 内部追加

向每个匹配的元素内部追加内容。

这个操作与对指定的元素执行appendChild方法，将它们添加到文档中的情况类似。

**Append content to the inside of every matched element.**

**This operation is similar to doing an appendChild to all the specified elements, adding them into the document.**

**返回值**

jQuery

**参数**

**content** (String, Element, jQuery) : C要追加到目标中的内容

**示例**

|  |
| --- |
| 向所有段落中追加一些HTML标记。  **HTML 代码:**  <p>I would like to say: </p>  **jQuery 代码:**  $("p").append("<b>Hello</b>");  **结果:**  [ <p>I would like to say: <b>Hello</b></p> ] |

### appendTo(content)\_\_追加到另一个元素中

把所有匹配的元素追加到另一个、指定的元素元素集合中。

实际上，使用这个方法是颠倒了常规的$(A).append(B)的操作，即不是把B追加到A中，而是把A追加到B中。

**返回值**

jQuery

**参数**

**content** (String) :用于被追加的内容

**示例**

|  |
| --- |
| 把所有段落追加到ID值为foo的元素中。  **HTML 代码:**  <p>I would like to say: </p><div id="foo"></div>  **jQuery 代码:**  $("p").appendTo("#foo");  **结果:**  <div id="foo"><p>I would like to say: </p></div> |

**appendTo(content)**

把所有匹配的元素追加到另一个、指定的元素元素集合中。

实际上，使用这个方法是颠倒了常规的$(A).append(B)的操作，即不是把B追加到A中，而是把A追加到B中。

Append all of the matched elements to another, specified, set of elements.

This operation is, essentially, the reverse of doing a regular $(A).append(B), in that instead of appending B to A, you're appending A to B.

**返回值**

jQuery

**参数**

**content** (String) :用于被追加的内容

**示例**

把所有段落追加到ID值为foo的元素中。

**HTML 代码:**

<p>I would like to say: </p><div id="foo"></div>

**jQuery 代码:**

$("p").appendTo("#foo");

**结果:**

<div id="foo"><p>I would like to say: </p></div>

### prepend(content)\_\_ 内部前置

向每个匹配的元素内部前置内容。

这是向所有匹配元素内部的开始处插入内容的最佳方式。

**返回值**

jQuery

**参数**

**content** (String, Element, jQuery) : 要插入到目标元素内部前端的内容

**示例**

|  |
| --- |
| 向所有段落中前置一些HTML标记代码。  **HTML 代码:**  <p>I would like to say: </p>  **jQuery 代码:**  $("p").prepend("<b>Hello</b>");  **结果:**  [ <b>Hello</b><p>I would like to say: </p> ] |
| 将一个DOM元素前置入所有段落  **HTML 代码:**  <p>I would like to say: </p> <p>I would like to say: </p> <b class="foo">Hello</b> <b class="foo">Good Bye</b>  **jQuery 代码:**  $("p").prepend( $(".foo")[0] );  **结果:**  <p><b class="foo">Hello</b>I would like to say: </p> <p><b class="foo">Hello</b>I would like to say: </p> <b class="foo">Hello</b> <b class="foo">Good Bye</b> |
| 向所有段落中前置一个jQuery对象(类似于一个DOM元素数组)。  **HTML 代码:**  <p>I would like to say: </p><b>Hello</b>  **jQuery 代码:**  $("p").prepend( $("b") );  **结果:**  <p><b>Hello</b>I would like to say: </p> |

**prepend(content)**

向每个匹配的元素内部前置内容。

这是向所有匹配元素内部的开始处插入内容的最佳方式。

Prepend content to the inside of every matched element.

This operation is the best way to insert elements inside, at the beginning, of all matched elements.

**返回值**

jQuery

**参数**

**content** (String, Element, jQuery) : 要插入到目标元素内部前端的内容

**示例**

向所有段落中前置一些HTML标记代码。

**HTML 代码:**

<p>I would like to say: </p>

**jQuery 代码:**

$("p").prepend("<b>Hello</b>");

**结果:**

[ <b>Hello</b><p>I would like to say: </p> ]

将一个DOM元素前置入所有段落

**HTML 代码:**

<p>I would like to say: </p>  
<p>I would like to say: </p>  
<b class="foo">Hello</b>  
<b class="foo">Good Bye</b>

**jQuery 代码:**

$("p").prepend( $(".foo")[0] );

**结果:**

<p><b class="foo">Hello</b>I would like to say: </p>  
<p><b class="foo">Hello</b>I would like to say: </p>  
<b class="foo">Hello</b>  
<b class="foo">Good Bye</b>

向所有段落中前置一个jQuery对象(类似于一个DOM元素数组)。

**HTML 代码:**

<p>I would like to say: </p><b>Hello</b>

**jQuery 代码:**

$("p").prepend( $("b") );

**结果:**

<p><b>Hello</b>I would like to say: </p>

### prependTo(content)\_\_ 前置到另一个元素中

把所有匹配的元素前置到另一个、指定的元素元素集合中。

实际上，使用这个方法是颠倒了常规的$(A).prepend(B)的操作，即不是把B前置到A中，而是把A前置到B中。

Prepend all of the matched elements to another, specified, set of elements.

This operation is, essentially, the reverse of doing a regular $(A).prepend(B), in that instead of prepending B to A, you're prepending A to B.

**返回值**

jQuery

**参数**

**content** (String) :用于匹配元素的jQuery表达式

**示例**

|  |
| --- |
| 把所有段落追加到ID值为foo的元素中。  **HTML 代码:**  <p>I would like to say: </p><div id="foo"></div>  **jQuery 代码:**  $("p").prependTo("#foo");  **结果:**  <div id="foo"><p>I would like to say: </p></div>  外部插入 |

### after(content)\_\_ 元素之后插入

在每个匹配的元素之后插入内容。

Insert content after each of the matched elements.

**返回值**

jQuery

**参数**

**content** (String, Element, jQuery) : 插入到每个目标后的内容

**示例**

|  |
| --- |
| 在所有段落之后插入一些HTML标记代码。  **HTML 代码:**  <p>I would like to say: </p>  **jQuery 代码:**  $("p").after("<b>Hello</b>");  **结果:**  <p>I would like to say: </p><b>Hello</b> |
| 在所有段落之后插入一个DOM元素。  **HTML 代码:**  <b id="foo">Hello</b><p>I would like to say: </p>  **jQuery 代码:**  $("p").after( $("#foo")[0] );  **结果:**  <p>I would like to say: </p><b id="foo">Hello</b> |
| 在所有段落中后插入一个jQuery对象(类似于一个DOM元素数组)。  **HTML 代码:**  <b>Hello</b><p>I would like to say: </p>  **jQuery 代码:**  $("p").after( $("b") );  **结果:**  <p>I would like to say: </p><b>Hello</b> |

### before(content)\_\_ 在每个匹配的元素之前插入

在每个匹配的元素之前插入内容。

Insert content before each of the matched elements.

**返回值**

jQuery

**参数**

**content** (String, Element, jQuery) : 插入到每个目标前的内容

**示例**

|  |
| --- |
| 在所有段落之前插入一些HTML标记代码。  **HTML 代码:**  <p>I would like to say: </p>  **jQuery 代码:**  $("p").before("<b>Hello</b>");  **结果:**  [ <b>Hello</b><p>I would like to say: </p> ] |
| 在所有段落之前插入一个元素。  **HTML 代码:**  <p>I would like to say: </p><b id="foo">Hello</b>  **jQuery 代码:**  $("p").before( $("#foo")[0] );  **结果:**  <b id="foo">Hello</b><p>I would like to say: </p> |
| 在所有段落中前插入一个jQuery对象(类似于一个DOM元素数组)。  **HTML 代码:**  <p>I would like to say: </p><b>Hello</b>  **jQuery 代码:**  $("p").before( $("b") );  **结果:**  <b>Hello</b><p>I would like to say: </p> |

### insertAfter(content)\_\_ 插入到另一个后面

把所有匹配的元素插入到另一个、指定的元素元素集合的后面。

实际上，使用这个方法是颠倒了常规的$(A).after(B)的操作，即不是把B插入到A后面，而是把A插入到B后面。

Insert all of the matched elements after another, specified, set of elements.

This operation is, essentially, the reverse of doing a regular $(A).after(B), in that instead of inserting B after A, you're inserting A after B.

**返回值**

jQuery

**参数**

**content** (String) : 用于匹配元素的jQuery表达式

**示例**

|  |
| --- |
| 在所有段落之后插入一个元素。与 $("#foo").after("p")相同  **HTML 代码:**  <p>I would like to say: </p><div id="foo">Hello</div>  **jQuery 代码:**  $("p").insertAfter("#foo");  **结果:**  <div id="foo">Hello</div><p>I would like to say: </p> |

### insertBefore(content)\_\_ 插入到另一个元素前面

把所有匹配的元素插入到另一个、指定的元素元素集合的前面。

实际上，使用这个方法是颠倒了常规的$(A).before(B)的操作，即不是把B插入到A前面，而是把A插入到B前面。

Insert all of the matched elements before another, specified, set of elements.

This operation is, essentially, the reverse of doing a regular $(A).before(B), in that instead of inserting B before A, you're inserting A before B.

**返回值**

jQuery

**参数**

**content** (String) : 用于匹配元素的jQuery表达式

**示例**

|  |
| --- |
| 在所有段落之前插入一个元素。与 $("#foo").before("p")相同。  **HTML 代码:**  <div id="foo">Hello</div><p>I would like to say: </p>  **jQuery 代码:**  $("p").insertBefore("#foo");  **结果:**  <p>I would like to say: </p><div id="foo">Hello</div>  包裹 |

### wrap(html)\_\_ 用其他元素包裹

把所有匹配的元素用其他元素的结构化标记包裹起来。

这种包装对于在文档中插入额外的结构化标记最有用，而且它不会破坏原始文档的语义品质。

这个函数的原理是检查提供的第一个元素（它是由所提供的HTML标记代码动态生成的），并在它的代码结构中找到最上层的祖先元素－－这个祖先元素就是包裹元素。

当HTML标记代码中的元素包含文本时无法使用这个函数。因此，如果要添加文本应该在包裹完成之后再行添加。

Wrap all matched elements with a structure of other elements.

This wrapping process is most useful for injecting additional structure into a document, without ruining the original semantic qualities of a document.

This works by going through the first element provided (which is generated, on the fly, from the provided HTML) and finds the deepest ancestor element within its structure -- it is that element that will enwrap everything else.

This does not work with elements that contain text. Any necessary text must be added after the wrapping is done.

**返回值**

jQuery

**参数**

**html** (String) : HTML标记代码字符串，用于动态生成元素并包裹目标元素

**示例**

|  |
| --- |
| 把所有的段落用一个新创建的div包裹起来  **HTML 代码:**  <p>Test Paragraph.</p>  **jQuery 代码:**  $("p").wrap("<div class='wrap'></div>");  **结果:**  <div class="wrap"><p>Test Paragraph.</p></div> |

### wrap(elem)\_\_ 用其他元素

把所有匹配的元素用其他元素的结构化标记包装起来。

Wrap all matched elements with a structure of other elements.

**返回值**

jQuery

**参数**

**elem** (Element) : 用于包装目标元素的DOM元素

**示例**

|  |
| --- |
| 用ID是"content"的div将每一个段落包裹起来  **HTML 代码:**  <p>Test Paragraph.</p>  <div id="content"></div>  **jQuery 代码:**  $("p").wrap(document.getElementById('content'));  **结果:**  <div id="content"><p>Test Paragraph.</p></div>  <div id="content"></div> |

### wrapAll(html)\_\_ 将所有匹配的元素用单个元素包裹起来

将所有匹配的元素用单个元素包裹起来

这于 '.wrap()' 是不同的，'.wrap()'为每一个匹配的元素都包裹一次。

这种包装对于在文档中插入额外的结构化标记最有用，而且它不会破坏原始文档的语义品质。

这个函数的原理是检查提供的第一个元素并在它的代码结构中找到最上层的祖先元素－－这个祖先元素就是包装元素。

Wrap all the elements in the matched set into a single wrapper element.

This is different from '.wrap()' where each element in the matched set would get wrapped with an element.

This wrapping process is most useful for injecting additional structure into a document, without ruining the original semantic qualities of a document.

This works by going through the first element provided (which is generated, on the fly, from the provided HTML) and finds the deepest ancestor element within its structure -- it is that element that will enwrap everything else.

**返回值**

jQuery

**参数**

**html** (String) : TML标记代码字符串，用于动态生成元素并包装目标元素

**示例**

|  |
| --- |
| 用一个生成的div将所有段落包裹起来  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").wrapAll("<div></div>");  **结果:**  <div><p>Hello</p><p>cruel</p><p>World</p></div> |

### wrapAll(elem)\_\_ 将所有匹配的元素用单个元素包裹起来

将所有匹配的元素用单个元素包裹起来

这于 '.wrap()' 是不同的，'.wrap()'为每一个匹配的元素都包裹一次。

Wrap all the elements in the matched set into a single wrapper element.

This is different from '.wrap()' where each element in the matched set would get wrapped with an element.

**返回值**

jQuery

**参数**

**elem** (Element) : 用于包装目标元素的DOM元素

**示例**

|  |
| --- |
| 用一个生成的div将所有段落包裹起来  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("p").wrapAll(document.createElement("div"));  **结果:**  <div><p>Hello</p><p>cruel</p><p>World</p></div> |

### wrapInner(html)

将每一个匹配的元素的子内容(包括文本节点)用一个HTML结构包裹起来

这个函数的原理是检查提供的第一个元素（它是由所提供的HTML标记代码动态生成的），并在它的代码结构中找到最上层的祖先元素－－这个祖先元素就是包装元素。

Wrap the inner child contents of each matched element (including text nodes) with an HTML structure.

This wrapping process is most useful for injecting additional structure into a document, without ruining the original semantic qualities of a document. This works by going through the first element provided (which is generated, on the fly, from the provided HTML) and finds the deepest ancestor element within its structure -- it is that element that will enwrap everything else.

**返回值**

jQuery

**参数**

**html** (String) : HTML标记代码字符串，用于动态生成元素并包装目标元素

**示例**

|  |
| --- |
| 把所有段落内的每个子内容加粗  **HTML 代码:**  <p>Hello</p>  <p>cruel</p>  <p>World</p>  **jQuery 代码:**  $("p").wrapInner("<b></b>");  **结果:**  <p><b>Hello</b></p>  <p><b>cruel</b></p>  <p><b>World</b></p> |

### wrapInner(elem)

将每一个匹配的元素的子内容(包括文本节点)用DOM元素包裹起来

Wrap the inner child contents of each matched element (including text nodes) with a DOM element.

**返回值**

jQuery

**参数**

**elem** (Element) : 用于包装目标元素的DOM元素

**示例**

|  |
| --- |
| 把所有段落内的每个子内容加粗  **HTML 代码:**  <p>Hello</p>  <p>cruel</p>  <p>World</p>  **jQuery 代码:**  $("p").wrapInner(document.createElement("b"));  **结果:**  <p><b>Hello</b></p>  <p><b>cruel</b></p>  <p><b>World</b></p>  替换 |

### replaceWith(content)

将所有匹配的元素替换成指定的HTML或DOM元素。

Replaces all matched elements with the specified HTML or DOM elements.

**返回值**

jQuery

**参数**

**content** (String, Element, jQuery) : 用于将匹配元素替换掉的内容

**示例**

|  |
| --- |
| 把所有的段落标记替换成加粗的标记。  **HTML 代码:**  <p>Hello</p>  <p>cruel</p>  <p>World</p>  **jQuery 代码:**  $("p").replaceWith("<b>Paragraph. </b>");  **结果:**  <b>Paragraph. </b>  <b>Paragraph. </b>  <b>Paragraph. </b> |

### replaceAll(selector)

用匹配的元素替换掉所有 selector匹配到的元素。

Replaces the elements matched by the specified selector with the matched elements.

**返回值**

jQuery

**参数**

**selector** (选择器) : 用于查找所要被替换的元素

**示例**

|  |
| --- |
| 把所有的段落标记替换成加粗标记  **HTML 代码:**  <p>Hello</p><p>cruel</p><p>World</p>  **jQuery 代码:**  $("<b>Paragraph. </b>").replaceAll("p");  **结果:**  <b>Paragraph. </b>  <b>Paragraph. </b>  <b>Paragraph. </b>  删除  empty() |

### empty()\_\_清空元素内容

删除匹配的元素集合中所有的子节点。

Remove all child nodes from the set of matched elements.

**返回值**

jQuery

**示例**

|  |
| --- |
| 把所有段落的子元素（包括文本节点）删除  **HTML 代码:**  <p>Hello, <span>Person</span> <a href="#">and person</a></p>  **jQuery 代码:**  $("p").empty();  **结果:**  <p></p> |

### remove([expr])\_\_删除元素

从DOM中删除所有匹配的元素。

这个方法不会把匹配的元素从jQuery对象中删除，因而可以在将来再使用这些匹配的元素。

Removes all matched elements from the DOM.

This does NOT remove them from the jQuery object, allowing you to use the matched elements further. Can be filtered with an optional expression.

**返回值**

jQuery

**参数**

**expr** (String) : (可选) 用于筛选元素的jQuery表达式

**示例**

|  |
| --- |
| 从DOM中把所有段落删除  **HTML 代码:**  <p>Hello</p> how are <p>you?</p>  **jQuery 代码:**  $("p").remove();  **结果:**  how are |
| 从DOM中把带有hello类的段落删除  **HTML 代码:**  <p class="hello">Hello</p> how are <p>you?</p>  **jQuery 代码:**  $("p").remove(".hello");  **结果:**  how are <p>you?</p>  复制 |

### clone()

克隆匹配的DOM元素并且选中这些克隆的副本。

在想把DOM文档中元素的副本添加到其他位置时这个函数非常有用。

Clone matched DOM Elements and select the clones.

This is useful for moving copies of the elements to another location in the DOM.

**返回值**

jQuery

**示例**

|  |
| --- |
| 克隆所有b元素（并选中这些克隆的副本），然后将它们前置到所有段落中。  **HTML 代码:**  <b>Hello</b><p>, how are you?</p>  **jQuery 代码:**  $("b").clone().prependTo("p");  **结果:**  <b>Hello</b><p><b>Hello</b>, how are you?</p> |

### clone(true)

元素以及其所有的事件处理并且选中这些克隆的副本

在想把DOM文档中元素的副本添加到其他位置时这个函数非常有用。

Clone matched DOM Elements, and all their event handlers, and select the clones.

**This is useful for moving copies of the elements, and their events, to another location in the DOM.**

**返回值**

jQuery

**参数**

**true** (Boolean) : 设置为true以便复制元素的所有事件处理

**示例**

|  |
| --- |
| 创建一个按钮，他可以复制自己，并且他的副本也有同样功能。  **HTML 代码:**  <button>Clone Me!</button>  **jQuery 代码:**  $("button").click(function(){   $(this).clone(true).insertAfter(this); }); |

# CSS

## CSS

### css(name)\_\_ 访问第一个匹配元素的样式属性

访问第一个匹配元素的样式属性。

Return a style property on the first matched element.

**返回值**String

**参数**

**name** (String) : 要访问的属性名称

**示例**

|  |
| --- |
| 取得第一个段落的color样式属性的值。  **jQuery 代码:**  $("p").css("color"); |

### css(properties)\_\_设置css值(名/值对)

把一个“名/值对”对象设置为所有匹配元素的样式属性。

这是一种在所有匹配的元素上设置大量样式属性的最佳方式。

Set a key/value object as style properties to all matched elements.

This is the best way to set several style properties on all matched elements.

**返回值**

jQuery

**参数**

**properties** (Map) : 要设置为样式属性的名/值对

**示例**

|  |
| --- |
| 将所有段落的字体颜色设为红色并且背景为蓝色。  **jQuery 代码:**  $("p").css({ color: "#ff0011", background: "blue" }); |
| 如果属性名包含 "-"的话，必须使用引号:  **jQuery 代码:**  $("p").css({ "margin-left": "10px", "background-color": "blue" }); |

### css(name,value)\_\_ 设置一个样式属性的值

在所有匹配的元素中，设置一个样式属性的值。

数字将自动转化为像素值

Set a single style property to a value on all matched elements.

If a number is provided, it is automatically converted into a pixel value.

**返回值**

jQuery

**参数**

**name** (value) : 属性名

**value** (String, Number) : 属性值

**示例**

|  |
| --- |
| 将所有段落字体设为红色  **jQuery 代码:**  $("p").css("color","red"); |

## 位置

### offset()\_\_top 和 left

获取匹配元素在当前视口的相对偏移。

返回的对象包含两个整形属性：top 和 left。此方法只对可见元素有效。

Get the current offset of the first matched element relative to the viewport.

The returned object contains two Integer properties, top and left. The method works only with visible elements.

**返回值**Object{top,left}

**示例**

|  |
| --- |
| 获取第二段的偏移  **HTML 代码:**  <p>Hello</p><p>2nd Paragraph</p>  **jQuery 代码:**  var p = $("p:last");  var offset = p.offset();  p.html( "left: " + offset.left + ", top: " + offset.top );  **结果:**  <p>Hello</p><p>left: 0, top: 35</p> |

**常见的求宽和高**

|  |
| --- |
| 例子：  alert($(document).width());  alert($(window).width());    alert($(document).height());  alert($(window).height());    alert($("#classNode").height());  alert($("#classNode").width()); |

### height()

取得第一个匹配元素当前计算的高度值（px）。

在 jQuery 1.2 以后可以用来获取 window 和 document 的高

Get the current computed, pixel, height of the first matched element.

In jQuery 1.2, this method is able to find the height of the window and document.

**返回值**Integer

**示例**

|  |
| --- |
| 获取第一段的高  **jQuery 代码:**  $("p").height(); |
| 获取文档的高  **jQuery 代码:**  $(document).height();  $(window).height(); |

### height(val)

为每个匹配的元素设置CSS高度(hidth)属性的值。如果没有明确指定单位（如：em或%），使用px。

如果没有明确指定单位（如：em或%），使用px。

Set the CSS height of every matched element.

If no explicit unit was specified (like 'em' or '%') then "px" is concatenated to the value.

**返回值**

jQuery

**参数**

**val** (String, Number) : 设定CSS中 'height' 的值

**示例**

|  |
| --- |
| 把所有段落的高设为 20:  **jQuery 代码:**  $("p").height(20); |

### width()

取得第一个匹配元素当前计算的宽度值（px）。

在 jQuery 1.2 以后可以用来获取 window 和 document 的宽

Get the current computed, pixel, width of the first matched element.

In jQuery 1.2, this method is able to find the width of the window and document.

**返回值**Integer

**示例**

|  |
| --- |
| 获取第一段的宽  **jQuery 代码:**  $("p").width(); |
| 获取当前窗口的宽  **jQuery 代码:**  $(window).width();  $(document).width(); |

### width(val)

为每个匹配的元素设置CSS宽度(width)属性的值。

如果没有明确指定单位（如：em或%），使用px。

Set the CSS width of every matched element.

If no explicit unit was specified (like 'em' or '%') then "px" is concatenated to the value.

**返回值**

jQuery

**参数**

**val** (String, Number) : 设定 CSS 'width' 的属性值

**示例**

|  |
| --- |
| 将所有段落的宽设为 20:  **jQuery 代码:**  $("p").width(20); |

# 事件

## 页面载入

### ready(fn)

当DOM载入就绪可以查询及操纵时绑定一个要执行的函数。

这是事件模块中最重要的一个函数，因为它可以极大地提高web应用程序的响应速度。

简单地说，这个方法纯粹是对向window.load事件注册事件的替代方法。通过使用这个方法，可以在DOM载入就绪能够读取并操纵时立即调用你所绑定的函数，而99.99%的JavaScript函数都需要在那一刻执行。

有一个参数－－对jQuery函数的引用－－会传递到这个ready事件处理函数中。可以给这个参数任意起一个名字，并因此可以不再担心命名冲突而放心地使用$别名。

请确保在 <body> 元素的onload事件中没有注册函数，否则不会触发$(document).ready()事件。

可以在同一个页面中无限次地使用$(document).ready()事件。其中注册的函数会按照（代码中的）先后顺序依次执行。

Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.

This is probably the most important function included in the event module, as it can greatly improve the response times of your web applications.

In a nutshell, this is a solid replacement for using window.onload, and attaching a function to that. By using this method, your bound function will be called the instant the DOM is ready to be read and manipulated, which is when what 99.99% of all JavaScript code needs to run.

There is one argument passed to the ready event handler: A reference to the jQuery function. You can name that argument whatever you like, and can therefore stick with the $ alias without risk of naming collisions.

Please ensure you have no code in your <body> onload event handler, otherwise $(document).ready() may not fire.

You can have as many $(document).ready events on your page as you like. The functions are then executed in the order they were added.

**返回值**

jQuery

**参数**

**fn** (Function) : 要在DOM就绪时执行的函数

**示例**

|  |
| --- |
| 在DOM加载完成时运行的代码，可以这样写：  **jQuery 代码:**  $(document).ready(function(){    // Your code here...  }); |
| 使用$(document).ready()的简写，并且具有安全保障（failsafe）的jQuery代码。安全保障能够让你的$别名不依赖于全局的$.  **jQuery 代码:**  jQuery(function($) {    // Your code using failsafe $ alias here...  }); |

## 事件处理

### bind(type,[data],fn)

为每一个匹配元素的特定事件（像click）绑定一个事件处理器函数。

这个事件处理函数会接收到一个事件对象，可以通过它来阻止（浏览器）默认的行为。如果既想取消默认的行为，又想阻止事件起泡，这个事件处理函数必须返回false。多数情况下，可以把事件处理器函数定义为匿名函数（见示例一）。在不可能定义匿名函数的情况下，可以传递一个可选的数据对象作为第二个参数（而事件处理器函数则作为第三个参数），见示例二。

Binds a handler to a particular event (like click) for each matched element.

The event handler is passed an event object that you can use to prevent default behaviour. To stop both default action and event bubbling, your handler has to return false. In most cases, you can define your event handlers as anonymous functions (see first example). In cases where that is not possible, you can pass additional data as the second parameter (and the handler function as the third), see second example.

**返回值**

jQuery

**参数**

**type** (String) : 事件类型

**data** (Object) : (可选) 作为event.data属性值传递给事件对象的额外数据对象

**fn** ( Function) : 绑定到每个匹配元素的事件上面的处理函数

**示例**

|  |
| --- |
| 当每个段落被点击的时候，弹出其文本。  **jQuery 代码:**  $("p").bind("click", function(){    alert( $(this).text() );  }); |
| 你可以在事件处理之前传递一些附加的数据。  **jQuery 代码:**  function handler(event) {    alert(event.data.foo);  }  $("p").bind("click", {foo: "bar"}, handler) |
| 通过返回false来取消默认的行为并阻止事件起泡。  **jQuery 代码:**  $("form").bind("submit", function() { return false; }) |
| 通过使用 preventDefault() 方法只取消默认的行为。  **jQuery 代码:**  $("form").bind("submit", function(event){    event.preventDefault();  }); |
| 通过使用 stopPropagation() 方法只阻止一个事件起泡。  **jQuery 代码:**  $("form").bind("submit", function(event){    event.stopPropagation();  }); |

### one(type,[data],fn)

为每一个匹配元素的特定事件（像click）绑定一个一次性的事件处理函数。

在每个对象上，这个事件处理函数只会被执行一次。其他规则与bind()函数相同。这个事件处理函数会接收到一个事件对象，可以通过它来阻止（浏览器）默认的行为。如果既想取消默认的行为，又想阻止事件起泡，这个事件处理函数必须返回false。

多数情况下，可以把事件处理函数定义为匿名函数（见示例一）。在不可能定义匿名函数的情况下，可以传递一个可选的数据对象作为第二个参数（而事件处理函数则作为第三个参数），见示例二。

Binds a handler to a particular event to be executed once for each matched element.

The handler is executed only once for each element. Otherwise, the same rules as described in 'bind()' apply. The event handler is passed an event object that you can use to prevent default behaviour. To stop both default action and event bubbling, your handler should return false.

In most cases, you can define your event handlers as anonymous functions (see first example). In cases where that is not possible, you can pass additional data as the second paramter (and the handler function as the third), see second example.

**返回值**

jQuery

**参数**

**type** (String) : 事件类型

**data** (Object) : (可选) 作为event.data属性值传递给事件对象的额外数据对象

**fn** (Function) : 绑定到每个匹配元素的事件上面的处理函数

**示例**

|  |
| --- |
| 当所有段落被第一次点击的时候，显示所有其文本。  **jQuery 代码:**  $("p").one("click", function(){    alert( $(this).text() );  }); |

### trigger(type,[data])

在每一个匹配的元素上触发某类事件。

这个函数也会导致浏览器同名的默认行为的执行。比如，如果用trigger()触发一个'submit'，则同样会导致浏览器提交表单。如果要阻止这种默认行为，应返回false。

你也可以触发由bind()注册的自定义事件

Trigger a type of event on every matched element.

This will also cause the default action of the browser with the same name (if one exists) to be executed. For example, passing 'submit' to the trigger() function will also cause the browser to submit the form. This default action can be prevented by returning false from one of the functions bound to the event.

You can also trigger custom events registered with bind.

**返回值**

jQuery

**参数**

**type** (String) : 要触发的事件类型

**data** (Array) : (可选)传递给事件处理函数的附加参数

**示例**

|  |
| --- |
| 提交第一个表单，但不用submit()  **jQuery 代码:**  $("form:first").trigger("submit") |
| 给一个事件传递参数  **jQuery 代码:**  $("p").click( function (event, a, b) {    // 一个普通的点击事件时，a和b是undefined类型    // 如果用下面的语句触发，那么a指向"foo",而b指向"bar"  } ).trigger("click", ["foo", "bar"]); |
| 下面的代码可以显示一个"Hello World"  **jQuery 代码:**  $("p").bind("myEvent", function (event, message1, message2) {    alert(message1 + ' ' + message2);  });  $("p").trigger("myEvent", ["Hello","World!"]);  triggerHandler(type,data) |

### triggerHandler(type,[data])

这个特别的方法将会触发指定的事件类型上所有绑定的处理函数。但**不会执行**浏览器默认动作.

This particular method triggers all bound event handlers on an element (for a specific event type) WITHOUT executing the browsers default actions.

**返回值**

jQuery

**参数**

**type** (String) : 要触发的事件类型

**data** (Array) : (可选)传递给事件处理函数的附加参数

**示例**

|  |
| --- |
| 如果你对一个focus事件执行了 .triggerHandler() ，浏览器默认动作将不会被触发，只会触发你绑定的动作。  **HTML 代码:**  <button id="old">.trigger("focus")</button>  <button id="new">.triggerHandler("focus")</button><br/><br/>  <input type="text" value="To Be Focused"/>  **jQuery 代码:**  $("#old").click(function(){    $("input").trigger("focus");  });  $("#new").click(function(){    $("input").triggerHandler("focus");  });  $("input").focus(function(){   $("<span>Focused!</span>").appendTo("body").fadeOut(1000); });  unbind(type,data) |

### unbind([type],[data])

bind()的反向操作，从每一个匹配的元素中删除绑定的事件。

如果没有参数，则删除所有绑定的事件。

你可以将你用bind()注册的自定义事件取消绑定。

I如果提供了事件类型作为参数，则只删除该类型的绑定事件。

如果把在绑定时传递的处理函数作为第二个参数，则只有这个特定的事件处理函数会被删除。

This does the opposite of bind, it removes bound events from each of the matched elements.

Without any arguments, all bound events are removed.

You can also unbind custom events registered with bind.

If the type is provided, all bound events of that type are removed.

If the function that was passed to bind is provided as the second argument, only that specific event handler is removed.

**返回值**

jQuery

**参数**

**type** (String) : (可选) 事件类型

**data** (Function) : (可选) 要从每个匹配元素的事件中反绑定的事件处理函数

**示例**

|  |
| --- |
| 把所有段落的所有事件取消绑定  **jQuery 代码:**  $("p").unbind() |
| 将段落的click事件取消绑定  **jQuery 代码:**  $("p").unbind( "click" ) |
| 删除特定函数的绑定，将函数作为第二个参数传入  **jQuery 代码:**  var foo = function () {    // code to handle some kind of event  };  $("p").bind("click", foo); // ... now foo will be called when paragraphs are clicked ...  $("p").unbind("click", foo); // ... foo will no longer be called.  交互处理 |

### hover(over,out)

一个模仿悬停事件（鼠标移动到一个对象上面及移出这个对象）的方法。这是一个自定义的方法，它为频繁使用的任务提供了一种“保持在其中”的状态。

当鼠标移动到一个匹配的元素上面时，会触发指定的第一个函数。当鼠标移出这个元素时，会触发指定的第二个函数。而且，会伴随着对鼠标是否仍然处在特定元素中的检测（例如，处在div中的图像），如果是，则会继续保持“悬停”状态，而不触发移出事件（修正了使用mouseout事件的一个常见错误）。

Simulates hovering (moving the mouse on, and off, an object). This is a custom method which provides an 'in' to a frequent task.

Whenever the mouse cursor is moved over a matched element, the first specified function is fired. Whenever the mouse moves off of the element, the second specified function fires. Additionally, checks are in place to see if the mouse is still within the specified element itself (for example, an image inside of a div), and if it is, it will continue to 'hover', and not move out (a common error in using a mouseout event handler).

**返回值**

jQuery

**参数**

**over** (Function) : 鼠标移到元素上要触发的函数

**out** (Function) : 鼠标移出元素要触发的函数

**示例**

|  |
| --- |
| 鼠标悬停的表格加上特定的类  **jQuery 代码:**  $("td").hover(    function () {      $(this).addClass("hover");    },    function () {      $(this).removeClass("hover");    }  ); |

### toggle(fn,fn)

每次点击时切换要调用的函数。

如果点击了一个匹配的元素，则触发指定的第一个函数，当再次点击同一元素时，则触发指定的第二个函数。随后的每次点击都重复对这两个函数的轮番调用。

可以使用unbind("click")来删除。

Toggle between two function calls every other click.

Whenever a matched element is clicked, the first specified function is fired, when clicked again, the second is fired. All subsequent clicks continue to rotate through the two functions.

Use unbind("click") to remove.

**返回值**

jQuery

**参数**

**fn** (Function) : 第奇数次点击时要执行的函数。

**fn** (Function) : 第偶数次点击时要执行的函数。

**示例**

|  |
| --- |
| 对表格的切换一个类  **jQuery 代码:**  $("td").toggle(    function () {      $(this).addClass("selected");    },    function () {      $(this).removeClass("selected");    }  ); |

### blur()

触发每一个匹配元素的blur事件。

这个函数会调用执行绑定到blur事件的所有函数，包括浏览器的默认行为。可以通过返回false来防止触发浏览器的默认行为。blur事件会在元素失去焦点的时候触发，既可以是鼠标行为，也可以是按tab键离开的

Triggers the blur event of each matched element.

This causes all of the functions that have been bound to that blur event to be executed, and calls the browser's default blur action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the blur event. The blur event usually fires when an element loses focus either via the pointing device or by tabbing navigation

**返回值**

jQuery

**示例**

|  |
| --- |
| 触发所有段落的blur事件  **jQuery 代码:**  $("p").blur(); |

### blur(fn)

在每一个匹配元素的blur事件中绑定一个处理函数。

blur事件会在元素失去焦点的时候触发，既可以是鼠标行为，也可以是按tab键离开的

Bind a function to the blur event of each matched element.

The blur event fires when an element loses focus either via the pointing device or by tabbing navigation.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的blur事件中绑定的处理函数。

**示例**

|  |
| --- |
| 任何段落失去焦点时弹出一个 "Hello World!"在每一个匹配元素的blur事件中绑定的处理函数。  **jQuery 代码:**  $("p").blur( function () { alert("Hello World!"); } ); |

### change()

触发每个匹配元素的change事件

这个函数会调用执行绑定到change事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。change事件会在元素失去焦点的时候触发，也会当其值在获得焦点后改变时触发。

Triggers the change event of each matched element.

This causes all of the functions that have been bound to that change event to be executed, and calls the browser's default change action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the change event. The change event usually fires when a control loses the input focus and its value has been modified since gaining focus.

**返回值**

jQuery

change(fn)

### change(fn)

在每一个匹配元素的change事件中绑定一个处理函数。

change事件会在元素失去焦点的时候触发，也会当其值在获得焦点后改变时触发。

Binds a function to the change event of each matched element.

The change event fires when a control loses the input focus and its value has been modified since gaining focus.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的change事件中绑定的处理函数。

**示例**

|  |
| --- |
| 给所有的文本框增加输入验证  **jQuery 代码:**  $("input[@type='text']").change( function() {    // check input for validity here  }); |

### click()

触发每一个匹配元素的click事件。

这个函数会调用执行绑定到click事件的所有函数。

Triggers the click event of each matched element.

Causes all of the functions that have been bound to that click event to be executed.

**返回值**

jQuery

**示例**

|  |
| --- |
| 触发页面内所有段落的点击事件  **jQuery 代码:**  $("p").click(); |

### click(fn)

在每一个匹配元素的click事件中绑定一个处理函数。

点击事件会在你的指针设备的按钮在元素上单击时触发。单击的定义是在屏幕的同一点触发了mousedown和mouseup.常用的事件有如下的

mousedown

mouseup

click

Binds a function to the click event of each matched element.

The click event fires when the pointing device button is clicked over an element. A click is defined as a mousedown and mouseup over the same screen location. The sequence of these events is:

mousedown

mouseup

click

**返回值**

jQuery

**参数**

**fn** (Function) : 绑定到click事件的函数

**示例**

|  |
| --- |
| 将页面内所有段落点击后隐藏。  **jQuery 代码:**  $("p").click( function () { $(this).hide(); }); |

### dblclick()

触发每一个匹配元素的dbclick事件。

这个函数会调用执行绑定到dblclick事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。dblclick事件会在元素的同一点双击时触发。

Triggers the dblclick event of each matched element.

This causes all of the functions that have been bound to that dblclick event to be executed, and calls the browser's default dblclick action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the dblclick event. The dblclick event usually fires when the pointing device button is double clicked over an element.

**返回值**

jQuery

dblclick(fn)

### dblclick(fn)

在每一个匹配元素的dblclick事件中绑定一个处理函数。

的那个在某个元素上双击的时候就会触发dblclick事件

Binds a function to the dblclick event of each matched element.

The dblclick event fires when the pointing device button is double clicked over an element

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的dblclick事件中绑定的处理函数。

**示例**

|  |
| --- |
| To bind a "Hello World!" alert box the dblclick event on every paragraph on the page:  **jQuery 代码:**  $("p").dblclick( function () { alert("Hello World!"); }); |

### error()

这个函数会调用执行绑定到error事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。

Triggers the error event of each matched element.

This causes all of the functions that have been bound to that error event to be executed, and calls the browser's default error action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the error event. The error event usually fires when an element loses focus either via the pointing device or by tabbing navigation.

**返回值**

jQuery

error(fn)

### error(fn)

在每一个匹配元素的error事件中绑定一个处理函数。

对于error事件，没有一个公众的标准。在大多数浏览器中，当页面的JavaScript发生错误时，window对象会触发error事件;当图像的src属性无效时，比如文件不存在或者图像数据错误时，也会触发图像对象的error事件。

如果异常是由window对象抛出，事件处理函数将会被传入三个参数：

1. 描述事件的信息 ("varName is not defined", "missing operator in expression", 等等.),

2. 包含错误的文档的完整URL

3. 异常发生的行数

如果事件处理函数返回true，则表示事件已经被处理，浏览器将认为没有异常。

更多相关信息:

Binds a function to the error event of each matched element.

There is no public standard for the error event. In most browsers, the window object's error event is triggered when a JavaScript error occurs on the page. An image object's error event is triggered when it is set an invalid src attribute - either a non-existent file, or one with corrupt image data.

If the event is thrown by the window object, the event handler will be passed three parameters:

1. A message describing the event ("varName is not defined", "missing operator in expression", etc.),

2. the full URL of the document containing the error, and

3. the line number on which the error occured.

If the event handler returns true, it signifies that the event was handled, and the browser raises no errors.

For more information, see:

**返回值**

jQuery

**参数**

**fn** (Function) :在每一个匹配元素的error事件中绑定的处理函数。2008-2-22

**示例**

|  |
| --- |
| 在服务器端记录JavaScript错误日志:  **jQuery 代码:**  $(window).error(function(msg, url, line){    jQuery.post("js\_error\_log.php", { msg: msg, url: url, line: line });  }); |
| 隐藏JavaScript错误:  **jQuery 代码:**  $(window).error(function(){    return true;  }); |
| 给你IE的用户隐藏无效的图像:  **jQuery 代码:**  $("img").error(function(){    $(this).hide();  }); |

### focus()

触发每一个匹配元素的focus事件。

这将触发所有绑定的focus函数，注意，某些对象不支持focus方法。

Triggers the focus event of each matched element.

This causes all of the functions that have been bound to the focus event to be executed. Note that this does not execute the focus method of the underlying elements.

**返回值**

jQuery

**示例**

|  |
| --- |
| To focus on a login input box with id 'login' on page startup, try:  **jQuery 代码:**  $(document).ready(function(){    $("#login").focus();  }); |

### focus(fn)

在每一个匹配元素的focus事件中绑定一个处理函数。

focus事件可以通过鼠标点击或者键盘上的TAB导航触发

Binds a function to the focus event of each matched element.

The focus event fires when an element receives focus either via the pointing device or by tab navigation.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的focus事件中绑定的处理函数。

**示例**

|  |
| --- |
| 使人无法使用文本框:  **jQuery 代码:**  $("input[@type=text]").focus(function(){    this.blur();  }); |

### keydown()

触发每一个匹配元素的keydown事件

这个函数会调用执行绑定到keydown事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。keydown事件会在键盘按下时触发。

Triggers the keydown event of each matched element.

This causes all of the functions that have been bound to the keydown event to be executed, and calls the browser's default keydown action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the keydown event. The keydown event usually fires when a key on the keyboard is pressed.

**返回值**

jQuery

keydown(fn)

### keydown(fn)

在每一个匹配元素的keydown事件中绑定一个处理函数。

keydown事件会在键盘按下时触发。

Bind a function to the keydown event of each matched element.

The keydown event fires when a key on the keyboard is pressed.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的keydown事件中绑定的处理函数。

**示例**

|  |
| --- |
| 在页面内对键盘按键做出回应，可以使用如下代码:  **jQuery 代码:**  $(window).keydown(function(event){    switch(event.keyCode) {      // ...      // 不同的按键可以做不同的事情      // 不同的浏览器的keycode不同      // 更多详细信息:      http://unixpapa.com/js/key.html      // ...    }  }); |

### keypress()

触发每一个匹配元素的keypress事件

T这个函数会调用执行绑定到keydown事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。keydown事件会在键盘按下时触发

Triggers the keypress event of each matched element.

This causes all of the functions that have been bound to the keypress event to be executed, and calls the browser's default keypress action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the keypress event. The keypress event usually fires when a key on the keyboard is pressed.

**返回值**

jQuery

keypress(fn)

### keypress(fn)

在每一个匹配元素的keypress事件中绑定一个处理函数。

keypress事件会在敲击按键时触发。 敲击按键的定义为按下并抬起同一个按键。这几个事件发生的顺序是:

keydown

keyup

keypress

Binds a function to the keypress event of each matched element.

The keypress event fires when a key on the keyboard is "clicked". A keypress is defined as a keydown and keyup on the same key. The sequence of these events is:

keydown

keyup

keypress

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的keypress事件中绑定的处理函数。

keyup()

### keyup()

触发每一个匹配元素的keyup事件

这个函数会调用执行绑定到keyup事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。keyup事件会在按键释放时触发。

Triggers the keyup event of each matched element.

This causes all of the functions that have been bound to the keyup event to be executed, and calls the browser's default keyup action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the keyup event. The keyup event usually fires when a key on the keyboard is released.

**返回值**

jQuery

keyup(fn)

### keyup(fn)

在每一个匹配元素的keyup事件中绑定一个处理函数。

keyup 事件会在键盘按下时触发。

Bind a function to the keyup event of each matched element.

The keyup event fires when a key on the keyboard is released.

**返回值**

jQuery

**参数**

**fn** (Function) :在每一个匹配元素的keyup事件中绑定的处理函数。

load(fn)

### load(fn)

在每一个匹配元素的load事件中绑定一个处理函数。

如果绑定给window对象，则会在所有内容加载后触发，包括窗口，框架，对象和图像。如果绑定在元素上，则当元素的内容加载完毕后触发。

Binds a function to the load event of each matched element.

When bound to the window element, the event fires when the user agent finishes loading all content within a document, including window, frames, objects and images. For elements, it fires when the target element and all of its content has finished loading.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的load事件中绑定的处理函数。

mousedown(fn)

### mousedown(fn)

在每一个匹配元素的mousedown事件中绑定一个处理函数。

mousedown事件在鼠标在元素上点击后会触发

Binds a function to the mousedown event of each matched element.

The mousedown event fires when the pointing device button is pressed over an element.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的mousedown事件中绑定的处理函数。

mousemove(fn)

### mousemove(fn)

在每一个匹配元素的mousemove事件中绑定一个处理函数。

mousemove 事件通过鼠标在元素上移动来触发。事件处理函数会被传递一个变量——事件对象，其.clientX 和 .clientY 属性代表鼠标的坐标

Bind a function to the mousemove event of each matched element.

The mousemove event fires when the pointing device is moved while it is over an element. The event handler is passed one variable - the event object. Its .clientX and .clientY properties represent the coordinates of the mouse.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的mousemove事件中绑定的处理函数。

mouseout(fn)

### mouseout(fn)

在每一个匹配元素的mouseout事件中绑定一个处理函数。

mousedown事件在鼠标从元素上离开后会触发

Bind a function to the mouseout event of each matched element.

The mouseout event fires when the pointing device is moved away from an element.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的mouseout事件中绑定的处理函数。

mouseover(fn)

### mouseover(fn)

在每一个匹配元素的mouseover事件中绑定一个处理函数。

mouseout事件会在鼠标移入对象时触发

Bind a function to the mouseover event of each matched element.

The mouseout event fires when the pointing device is moved onto an element.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的mouseover事件中绑定的处理函数。

mouseup(fn)

### mouseup(fn)

在每一个匹配元素的mouseup事件中绑定一个处理函数。

mouseup事件会在鼠标点击对象释放时

Bind a function to the mouseup event of each matched element.

The mouseup event fires when the pointing device button is released over an element.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的mouseup事件中绑定的处理函数。

resize(fn)

### resize(fn)

在每一个匹配元素的resize事件中绑定一个处理函数。

当文档窗口改变大小时触发

**返回值**

jQuery

**参数**

**fn** (Function) :在每一个匹配元素的resize事件中绑定的处理函数。

**示例**

|  |
| --- |
| 让人每次改变页面窗口的大小时很郁闷的方法:  **jQuery 代码:**  $(document).resize(function(){    alert("Stop it!");  }); |

### scroll(fn (Function))

在每一个匹配元素的resize事件中绑定一个处理函数。

当滚动条发生变化时触发

Bind a function to the scroll event of each matched element.

The scroll event fires when a document view is scrolled.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的resize事件中绑定的处理函数。

**示例**

|  |
| --- |
| 当页面滚动条变化时，执行的函数:  **jQuery 代码:**  $(document).scroll( function() { /\* ...do something... \*/ } ); |

### select()

触发每一个匹配元素的select事件

这个函数会调用执行绑定到select事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。

Trigger the select event of each matched element.

This causes all of the functions that have been bound to that select event to be executed, and calls the browser's default select action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the select event.

**返回值**

jQuery

**示例**

|  |
| --- |
| 触发所有input元素的select事件:  **jQuery 代码:**  $("input").select(); |

### select(fn)

在每一个匹配元素的select事件中绑定一个处理函数。

当用户在文本框(包括input和textarea)中选中某段文本时会触发select事件。

Bind a function to the select event of each matched element.

The select event fires when a user selects some text in a text field, including input and textarea.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的select事件中绑定的处理函数。

**示例**

|  |
| --- |
| 当文本框中文本被选中时执行的函数:  **jQuery 代码:**  $(document).select( function () { /\* ...do something... \*/ } ); |

### submit()

触发每一个匹配元素的submit事件。

这个函数会调用执行绑定到submit事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。

Trigger the submit event of each matched element.

This causes all of the functions that have been bound to that submit event to be executed, and calls the browser's default submit action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the submit event.

**返回值**

jQuery

**示例**

|  |
| --- |
| 提交本页的第一个表单:  **jQuery 代码:**  $("form:first").submit(); |

### submit(fn)

在每一个匹配元素的submit事件中绑定一个处理函数。

select事件将会在表单提交时触发

**返回值**

jQuery

**参数**

**fn** (Function) :在每一个匹配元素的submit事件中绑定的处理函数

**示例**

|  |
| --- |
| 如果你要阻止表单提交:  **jQuery 代码:**  $("form").submit( function () {    return false;  } ); |

### unload(fn)

在每一个匹配元素的unload事件中绑定一个处理函数。

Binds a function to the unload event of each matched element.

**返回值**

jQuery

**参数**

**fn** (Function) : 在每一个匹配元素的unload事件中绑定的处理函数。

**示例**

|  |
| --- |
| 页面卸载的时候弹出一个警告框:  **jQuery 代码:**  $(document).unload( function () { alert("Bye now!"); } ); |

# 效果

## 基本

show()

### show()

显示隐藏的匹配元素。

这个就是 'show( speed, [callback] )' 无动画的版本。如果选择的元素是可见的，这个方法将不会改变任何东西。无论这个元素是通过hide()方法隐藏的还是在CSS里设置了display:none;，这个方法都将有效。

Displays each of the set of matched elements if they are hidden.

Same as 'show( speed, [callback] )' without animations. Doesn't change anything if the selected elements are all visible. It doesn't matter if the element is hidden via a hide() call, or via a display:none in a stylesheet.

**返回值**

jQuery

**示例**

|  |
| --- |
| 显示所有段落  **HTML 代码:**  <p style="display: none">Hello</p>  **jQuery 代码:**  $("p").show() |

### show(speed,[callback])

以优雅的动画显示所有匹配的元素，并在显示完成后可选地触发一个回调函数。

可以根据指定的速度动态地改变每个匹配元素的高度、宽度和不透明度。

Show all matched elements using a graceful animation and firing an optional callback after completion.

The height, width, and opacity of each of the matched elements are changed dynamically according to the specified speed.

**返回值**

jQuery

**参数**

**speed** (String,Number) : 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (Function) : (Optional) 在动画完成时执行的函数，每个元素执行一次。

**示例**

|  |
| --- |
| 用缓慢的动画将隐藏的段落显示出来，历时600毫秒。  **HTML 代码:**  <p style="display: none">Hello</p>  **jQuery 代码:**  $("p").show("slow"); |
| 用迅速的动画将隐藏的段落显示出来，历时200毫秒。并在之后执行反馈！  **HTML 代码:**  <p style="display: none">Hello</p>  **jQuery 代码:**  $("p").show("fast",function(){     $(this).text("Animation Done!");   }); |
| 将隐藏的段落用将近4秒的时间显示出来。。。并在之后执行一个反馈。。。  **HTML 代码:**  <p style="display: none">Hello</p>  **jQuery 代码:**  $("p").show(4000,function(){     $(this).text("Animation Done...");   }); |

### hide()

隐藏显示的元素

这个就是 'hide( speed, [callback] )' 的无动画版。如果选择的元素是隐藏的，这个方法将不会改变任何东西。

Hides each of the set of matched elements if they are shown.

Same as 'hide( speed, [callback] )' without animations. Doesn't change anything if the selected elements are all hidden.

**返回值**

jQuery

**示例**

|  |
| --- |
| 隐藏所有段落  **jQuery 代码:**  $("p").hide() |

### hide(speed,[callback])

以优雅的动画隐藏所有匹配的元素，并在显示完成后可选地触发一个回调函数。

可以根据指定的速度动态地改变每个匹配元素的高度、宽度和不透明度。

Hide all matched elements using a graceful animation and firing an optional callback after completion.

The height, width, and opacity of each of the matched elements are changed dynamically according to the specified speed.

**返回值**

jQuery

**参数**

**speed** (String,Number) :三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (FunctionFunction) : (Optional) 在动画完成时执行的函数，每个元素执行一次。

**示例**

|  |
| --- |
| 用600毫秒的时间将段落缓慢的隐藏  **jQuery 代码:**  $("p").hide("slow"); |
| 用200毫秒将段落迅速隐藏，之后弹出一个对话框。  **jQuery 代码:**  $("p").hide("fast",function(){     alert("Animation Done.");   }); |

### toggle()

切换元素的可见状态。

如果元素是可见的，切换为隐藏的；如果元素是隐藏的，切换为可见的。

Toggles each of the set of matched elements.

If they are shown, toggle makes them hidden. If they are hidden, toggle makes them shown.

**返回值**

jQuery

**示例**

|  |
| --- |
| 切换所有段落的可见状态。  **HTML 代码:**  <p>Hello</p><p style="display: none">Hello Again</p>  **jQuery 代码:**  $("p").toggle()  **结果:**  <p tyle="display: none">Hello</p><p style="display: block">Hello Again</p> |
|  |

### slideDown(speed,[callback])

通过高度变化（向下增大）来动态地显示所有匹配的元素，在显示完成后可选地触发一个回调函数。

这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式显示出来。

Reveal all matched elements by adjusting their height and firing an optional callback after completion.

Only the height is adjusted for this animation, causing all matched elements to be revealed in a "sliding" manner.

**返回值**

jQuery

**参数**

**speed** (String,Number) :三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (FunctionFunction) : (可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 用600毫秒缓慢的将段落滑下  **jQuery 代码:**  $("p").slideDown("slow"); |
| 用200毫秒快速将段落滑下，之后弹出一个对话框  **jQuery 代码:**  $("p").slideDown("fast",function(){     alert("Animation Done.");   }); |

### slideUp(speed,[callback])

通过高度变化（向上减小）来动态地隐藏所有匹配的元素，在隐藏完成后可选地触发一个回调函数。

这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式隐藏起来。

Hide all matched elements by adjusting their height and firing an optional callback after completion.

Only the height is adjusted for this animation, causing all matched elements to be hidden in a "sliding" manner.

**返回值**

jQuery

**参数**

**speed** (String,Number) : 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (Function) : (可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 用600毫秒缓慢的将段落滑上  **jQuery 代码:**  $("p").slideUp("slow"); |
| 用200毫秒快速将段落滑上，之后弹出一个对话框  **jQuery 代码:**  $("p").slideUp("fast",function(){     alert("Animation Done.");   }); |

### slideToggle(speed,[callback])

通过高度变化来切换所有匹配元素的可见性，并在切换完成后可选地触发一个回调函数。

这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式隐藏或显示。

Toggle the visibility of all matched elements by adjusting their height and firing an optional callback after completion.

Only the height is adjusted for this animation, causing all matched elements to be hidden or shown in a "sliding" manner.

**返回值**

jQuery

**参数**

**speed** (String,Number) : 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (Function) : (可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 用600毫秒缓慢的将段落滑上或滑下  **jQuery 代码:**  $("p").slideToggle("slow"); |
| 用200毫秒快速将段落滑上或滑下，之后弹出一个对话框  **jQuery 代码:**  $("p").slideToggle("fast",function(){     alert("Animation Done.");   });  淡入淡出 |

### fadeIn(speed,[callback])

通过不透明度的变化来实现所有匹配元素的淡入效果，并在动画完成后可选地触发一个回调函数。

这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化。

Fade in all matched elements by adjusting their opacity and firing an optional callback after completion.

Only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

**返回值**

jQuery

**参数**

**speed** (String,Number) : 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (Function) : (Optional) (可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 用600毫秒缓慢的将段落淡入  **jQuery 代码:**  $("p").fadeIn("slow"); |
| 用200毫秒快速将段落淡入，之后弹出一个对话框  **jQuery 代码:**  $("p").fadeIn("fast",function(){     alert("Animation Done.");   }); |

### fadeOut(speed,[callback])

通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数。

这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化。

Fade out all matched elements by adjusting their opacity and firing an optional callback after completion.

Only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

**返回值**

jQuery

**参数**

**speed** (String,Number) :三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**callback** (Function) :(可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 用600毫秒缓慢的将段落淡出  **jQuery 代码:**  $("p").fadeOut("slow"); |
| 用200毫秒快速将段落淡出，之后弹出一个对话框  **jQuery 代码:**  $("p").fadeOut("fast",function(){     alert("Animation Done.");   }); |

### fadeTo(speed,opacity,[callback])

把所有匹配元素的不透明度以渐进方式调整到指定的不透明度，并在动画完成后可选地触发一个回调函数。

这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化。

Fade the opacity of all matched elements to a specified opacity and firing an optional callback after completion.

Only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

**返回值**

jQuery

**参数**

**speed** (String,Number) : 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**opacity** (Number) : 要调整到的不透明度值(0到1之间的数字).

**callback** (Function) : (可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 用600毫秒缓慢的将段落的透明度调整到0.66，大约2/3的可见度  **jQuery 代码:**  $("p").fadeTo("slow", 0.66);$("p").fadeTo("slow", 0.66); |
| 用200毫秒快速将段落的透明度调整到0.25，大约1/4的可见度，之后弹出一个对话框  **jQuery 代码:**  $("p").fadeTo("fast", 0.25, function(){     alert("Animation Done.");   }); |

### animate(params[,duration[,easing[,callback]]])

用于创建自定义动画的函数。

这个函数的关键在于指定动画形式及结果样式属性对象。这个对象中每个属性都表示一个可以变化的样式属性（如“height”、“top”或“opacity”）。

注意：所有指定的属性必须用骆驼形式，比如用marginLeft代替margin-left.

而每个属性的值表示这个样式属性到多少时动画结束。如果是一个数值，样式属性就会从当前的值渐变到指定的值。如果使用的是“hide”、“show”或“toggle”这样的字符串值，则会为该属性调用默认的动画形式。

在 jQuery 1.2 中，你可以使用 em 和 % 单位。另外，在 jQuery 1.2 中，你可以通过在属性值前面指定 "+=" 或 "-=" 来让元素做相对运动。

A function for making your own, custom animations.

The key aspect of this function is the object of style properties that will be animated, and to what end. Each key within the object represents a style property that will also be animated (for example: "height", "top", or "opacity").

Note that properties should be specified using camel case eg. marginLeft instead of margin-left.

The value associated with the key represents to what end the property will be animated. If a number is provided as the value, then the style property will be transitioned from its current state to that new number. Otherwise if the string "hide", "show", or "toggle" is provided, a default animation will be constructed for that property.

As of jQuery 1.2 you can now animate properties by em and % (where applicable). Additionally, in jQuery 1.2, you can now do relative animations - specifying a "+=" or "-=" in front of the property value to move the element positively, or negatively, relative to the current position.

**返回值**

jQuery

**参数**

**params** (Options) : 一组包含作为动画属性和终值的样式属性和及其值的集合

**duration** (String,Number) : (可选) 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**easing** (String) : (可选) 要使用的擦除效果的名称(需要插件支持).默认jQuery提供"linear" 和 "swing".

**callback** (Function) : (可选) 在动画完成时执行的函数

**示例**

|  |
| --- |
| 点击按钮后div元素的几个不同属性一同变化  **HTML 代码:**  <button id="go"> Run</button>  <div id="block">Hello!</div>  **jQuery 代码:**  // 在一个动画中同时应用三种类型的效果  $("#go").click(function(){    $("#block").animate({      width: "90%",      height: "100%",      fontSize: "10em",      borderWidth: 10    }, 1000 );  }); |
| 让指定元素左右移动  **HTML 代码:**  <button id="left">«</button> <button id="right">»</button>  <div class="block"></div>  **jQuery 代码:**  $("#right").click(function(){    $(".block").animate({left: '+50px'}, "slow");  });  $("#left").click(function(){    $(".block").animate({left: '-50px'}, "slow");  }); |
| 在600毫秒内切换段落的高度和透明度  **jQuery 代码:**  $("p").animate({     height: 'toggle', opacity: 'toggle'   }, "slow"); |
| 用500毫秒将段落移到left为50的地方并且完全清晰显示出来（透明度为1）  **jQuery 代码:**  $("p").animate({     left: 50, opacity: 'show'   }, 500); |
| 一个使用“easein”函数提供不同动画样式的例子。只有使用了插件来提供这个“easein”函数，这个参数才起作用。  **jQuery 代码:**  $("p").animate({     opacity: 'show'   }, "slow", "easein"); |

### animate(params,options)

用于创建自定义动画的函数。

这个函数的关键在于指定动画形式及结果样式属性对象。这个对象中每个属性都表示一个可以变化的样式属性（如“height”、“top”或“opacity”）。

注意：所有指定的属性必须用骆驼形式，比如用marginLeft代替margin-left.

而每个属性的值表示这个样式属性到多少时动画结束。如果是一个数值，样式属性就会从当前的值渐变到指定的值。如果使用的是“hide”、“show”或“toggle”这样的字符串值，则会为该属性调用默认的动画形式。

在 jQuery 1.2 中，你可以使用 em 和 % 单位。另外，在 jQuery 1.2 中，你可以通过在属性值前面指定 "+=" 或 "-=" 来让元素做相对运动。

A function for making your own, custom animations.

The key aspect of this function is the object of style properties that will be animated, and to what end. Each key within the object represents a style property that will also be animated (for example: "height", "top", or "opacity").

Note that properties should be specified using camel case eg. marginLeft instead of margin-left.

The value associated with the key represents to what end the property will be animated. If a number is provided as the value, then the style property will be transitioned from its current state to that new number. Otherwise if the string "hide", "show", or "toggle" is provided, a default animation will be constructed for that property.

**返回值**

jQuery

**参数**

**params** (Options) : 一组包含作为动画属性和终值的样式属性和及其值的集合

**options** (Options) : 一组包含动画选项的值的集合。

**选项**

**duration** (String,Number) : (默认值: "normal") 三种预定速度之一的字符串("slow", "normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**easing** (String) : (默认值: "swing") 要使用的擦除效果的名称(需要插件支持).默认jQuery提供"linear" 和 "swing".

**complete** (Function) : 在动画完成时执行的函数

**step** (Callback) : [Template:APICallback](http://docs.jquery.com/action/edit/Template:APICallback)

**queue** (Boolean) : (默认值: true) 设定为false将使此动画不进入动画队列 (jQuery 1.2中新增)

**示例**

|  |
| --- |
| 第一个按钮按了之后展示了不在队列中的动画。在div扩展到90%的同时也在增加字体，一旦字体改变完毕后，边框的动画才开始。  第二个按钮按了之后就是一个传统的链式动画，即等前一个动画完成后，后一个动画才会开始.  **HTML 代码:**  <button id="go1">» Animate Block1</button>  <button id="go2">» Animate Block2</button>  <div id="block1">Block1</div><div id="block2">Block2</div>  **jQuery 代码:**  $("#go1").click(function(){    $("#block1").animate( { width: "90%"}, { queue: false, duration: 5000 } )       .animate( { fontSize: '10em' } , 1000 )       .animate( { borderWidth: 5 }, 1000);  });  $("#go2").click(function(){    $("#block2").animate( { width: "90%"}, 1000 )       .animate( { fontSize: '10em' } , 1000 )       .animate( { borderWidth: 5 }, 1000);  }); |
| 用600毫秒切换段落的高度和透明度  **jQuery 代码:**  $("p").animate({     height: 'toggle', opacity: 'toggle'   }, { duration: "slow" }); |
| 用500毫秒将段落移到left为50的地方并且完全清晰显示出来（透明度为1）  **jQuery 代码:**  $("p").animate({     left: 50, opacity: 'show'   }, { duration: 500 }); |
| 一个使用“easein”函数提供不同动画样式的例子。只有使用了插件来提供这个“easein”函数，这个参数才起作用。  **jQuery 代码:**  $("p").animate({     opacity: 'show'   }, { duration: "slow", easing: "easein" }); |

### stop()

停止所有在指定元素上正在运行的动画。

如果队列中有等待执行的动画，他们将被马上执行

Stops all the currently running animations on all the specified elements.

If any animations are queued to run, then they will begin immediately.

**返回值**

jQuery

**示例**

|  |
| --- |
| 点击Go之后开始动画,点Stop之后会在当前位置停下来  **HTML 代码:**  <button id="go">Go</button> <button  id="stop">STOP!</button>  <div class="block"></div>  <button id="go">Go</button> <button  id="stop">STOP!</button>  <div class="block"></div>  **jQuery 代码:**  // Start animation  $("#go").click(function(){    $(".block").animate({left: '+200px'}, 5000);  });  // Stop animation when button is clicked  $("#stop").click(function(){    $(".block").stop();  }); |

### queue()

返回指向第一个匹配元素的队列(将是一个函数数组)

Returns a reference to the first element's queue (which is an array of functions).

**返回值**Array<Function>

**示例**

|  |
| --- |
| 显示队列长度  **HTML 代码:**  <style>  div { margin:3px; width:40px; height:40px;  position:absolute; left:0px; top:30px;  background:green; display:none; }  div.newcolor { background:blue; }  span { color:red; }  </style>  <button id="show">Show Length of Queue</button>  <span></span>  <div></div>  **jQuery 代码:**  $("#show").click(function () {  var n = $("div").queue("fx");  $("span").text("Queue length is: " + n.length);  });  function runIt() {  $("div").show("slow");  $("div").animate({left:'+=200'},2000);  $("div").slideToggle(1000);  $("div").slideToggle("fast");  $("div").animate({left:'-=200'},1500);  $("div").hide("slow");  $("div").show(1200);  $("div").slideUp("normal", runIt);  }  runIt(); |

### queue(callback)

在匹配的元素的动画队列中添加一个函数

Adds a new function, to be executed, onto the end of the queue of all matched elements.

**返回值**

jQuery

**参数**

**callback** (Function) : 要添加进队列的函数

**示例**

|  |
| --- |
| 插入一个自定义函数 如果函数执行后要继续队列，则执行 jQuery(this).dequeue();  **HTML 代码:**  <style>  div { margin:3px; width:40px; height:40px;  position:absolute; left:0px; top:30px;  background:green; display:none; }  div.newcolor { background:blue; }  </style>  Click here...  <div></div>  **jQuery 代码:**  $(document.body).click(function () {  $("div").show("slow");  $("div").animate({left:'+=200'},2000);  $("div").queue(function () {  $(this).addClass("newcolor");  $(this).dequeue();  });  $("div").animate({left:'-=200'},500);  $("div").queue(function () {  $(this).removeClass("newcolor");  $(this).dequeue();  });  $("div").slideUp();  }); |

### queue(queue)

将匹配元素的动画队列用新的一个队列来代替(函数数组).

Replaces the queue of all matched element with this new queue (the array of functions).

**返回值**

queue

**参数**

**queue** (Array<Function>) : 用于替换的队列。所有函数都有同一个参数，这个值与queue(callback)相同

**示例**

|  |
| --- |
| 通过设定队列数组来删除动画队列  **HTML 代码:**  <style>  div { margin:3px; width:40px; height:40px;  position:absolute; left:0px; top:30px;  background:green; display:none; }  div.newcolor { background:blue; }  </style>  <button id="start">Start</button>  <button id="stop">Stop</button>  <div></div>  **jQuery 代码:**  $("#start").click(function () {  $("div").show("slow");  $("div").animate({left:'+=200'},5000);  $("div").queue(function () {  $(this).addClass("newcolor");  $(this).dequeue();  });  $("div").animate({left:'-=200'},1500);  $("div").queue(function () {  $(this).removeClass("newcolor");  $(this).dequeue();  });  $("div").slideUp();  });  $("#stop").click(function () {  $("div").queue("fx", []);  $("div").stop();  }); |

### dequeue()

从动画队列中移除一个队列函数.

Removes a queued function from the front of the queue and executes it.

**返回值**

jQuery

**示例**

|  |
| --- |
| **HTML 代码:**  <style>  div { margin:3px; width:50px; position:absolute;  height:50px; left:10px; top:30px;  background-color:yellow; }  div.red { background-color:red; }  </style>  <button>Start</button>  <div></div>  **jQuery 代码:**  $("button").click(function () {  $("div").animate({left:'+=200px'}, 2000);  $("div").animate({top:'0px'}, 600);  $("div").queue(function () {  $(this).toggleClass("red");  $(this).dequeue();  });  $("div").animate({left:'10px', top:'30px'}, 700);  }); |

# Ajax请求

jQuery.ajax(options)

### jQuery.ajax(options)

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](http://docs.jquery.com/Specifying_the_Data_Type_for_AJAX_Requests) 。

$.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](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/)。使用 [JSONP](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。数据类型设置为 "jsonp" 时，jQuery 将自动调用回调函数。

Load a remote page using an HTTP request.

This is jQuery's low-level AJAX implementation. See $.get, $.post etc. for higher-level abstractions that are often easier to understand and use, but don't offer as much functionality (such as error callbacks).

$.ajax() returns the XMLHttpRequest that it creates. In most cases you won't need that object to manipulate directly, but it is available if you need to abort the request manually.

**Note:** If you specify the dataType option described below, make sure the server sends the correct MIME type in the response (eg. xml as "text/xml"). Sending the wrong MIME type can lead to unexpected problems in your script. See [Specifying the Data Type for AJAX Requests](http://docs.jquery.com/Specifying_the_Data_Type_for_AJAX_Requests) for more information.

$.ajax() takes one argument, an object of key/value pairs, that are used to initialize and handle the request. See below for a full list of the key/values that can be used.

As of jQuery 1.2, you can load JSON data located on another domain if you specify a [JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/) callback, which can be done like so: "myurl?callback=?". jQuery automatically replaces the ? with the correct method name to call, calling your specified callback. Or, if you set the dataType to "jsonp" a callback will be automatically added to your Ajax request.

**返回值**XMLHttpRequest

**参数**

**options** (可选) : AJAX 请求设置。所有选项都是可选的。

**选项**

**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。 [Ajax Event](http://docs.jquery.com/Ajax_Events).

**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。 [Ajax 事件](http://docs.jquery.com/Ajax_Events)。

**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](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/) 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)) 请求失败时调用时间。参数：XMLHttpRequest 对象，错误信息，（可能）捕获的错误对象。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

**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)。

**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。参数：服务器返回数据，数据格式。 [Ajax 事件](http://docs.jquery.com/Ajax_Events)。

**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。

**示例**

|  |
| --- |
| 加载并执行一个 JS 文件。  **jQuery 代码:**  $.ajax({    type: "GET",    url: "test.js",    dataType: "script"  }); |
| 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 );     }  }); |
| 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);    }  }); |
| 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; |
| 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) |

### load(url,[data],[callback])

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。

Load HTML from a remote file and inject it into the DOM.

A GET request will be performed by default - but if you pass in any extra parameters then a POST will occur. In jQuery 1.2 you can now specify a jQuery selector in the URL. Doing so will filter the incoming HTML document, only injecting the elements that match the selector. The syntax looks something like "url #some > selector". See the examples for more information.

**返回值**

jQuery

**参数**

**url** (String) : 待装入 HTML 网页网址。

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

**callback** (Callback) : (可选) 载入成功时回调函数。

**示例**

|  |
| --- |
| 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"); |
| 加载 feeds.html 文件内容。  **jQuery 代码:**  $("#feeds").load("feeds.html"); |
| 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])

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。

Load a remote page using an HTTP GET request.

This is an easy way to send a simple GET request to a server without having to use the more complex $.ajax function. It allows a single callback function to be specified that will be executed when the request is complete (and only if the response has a successful response code). If you need to have both error and success callbacks, you may want to use $.ajax.

**返回值**XMLHttpRequest

**参数**

**url** (String) : The URL of the page to load.

**data** (Map) : (可选) 待发送 Key/value 参数。

**callback** (Function) : (可选) 载入成功时回调函数。

**示例**

|  |
| --- |
| 请求 test.php 网页，忽略返回值。  **jQuery 代码:**  $.get("test.php"); |
| $.get("test.php");  **jQuery 代码:**  $.get("test.php", { name: "John", time: "2pm" } ); |
| 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);  }); |
| 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])

通过 HTTP GET 请求载入 JSON 数据。

在 jQuery 1.2 中，您可以跨域加载 JSON 数据。使用 [JSONP](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。

注意：此行以后的代码将在这个回调函数执行前执行。

Load JSON data using an HTTP GET request.

As of jQuery 1.2, you can load JSON data located on another domain if you specify a [JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/) callback, which can be done like so: "myurl?callback=?". jQuery automatically replaces the ? with the correct method name to call, calling your specified callback.

Note: Keep in mind, that lines after this function will be executed before callback.

**返回值**XMLHttpRequest

**参数**

**url** (String) : 发送请求地址。

**data** (Map) : (可选) 待发送 Key/value 参数。

**callback** (Function) : (可选) 载入成功时回调函数。

**示例**

|  |
| --- |
| 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;    });  }); |
| 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);  }); |
| 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])

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。

Loads, and executes, a local JavaScript file using an HTTP GET request.

Before jQuery 1.2, getScript was only able to load scripts from the same domain as the original page. As of 1.2, you can now load JavaScript files from any domain. Warning: Safari 2 and older is unable to evaluate scripts in a global context synchronously. If you load functions via getScript, make sure to call them after a delay.

**返回值**XMLHttpRequest

**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。

**示例**

|  |
| --- |
| 载入 [jQuery 官方颜色动画插件](http://jquery.com/plugins/project/color) 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);    });  });  加载并执行 test.js。  **jQuery 代码:**  $.getScript("test.js"); |
| 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])

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。

Load a remote page using an HTTP POST request.

This is an easy way to send a simple POST request to a server without having to use the more complex $.ajax function. It allows a single callback function to be specified that will be executed when the request is complete (and only if the response has a successful response code). If you need to have both error and success callbacks, you may want to use $.ajax.

**返回值**XMLHttpRequest

**参数**

**url** (String) : 发送请求地址。

**data** (Map) : (可选) 待发送 Key/value 参数。

**callback** (Function) : (可选) 发送成功时回调函数。

**示例**

Ajax 事件

ajaxComplete(callback)

### ajaxComplete(callback)

AJAX 请求完成时执行函数。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

Attach a function to be executed whenever an AJAX request completes. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

**返回值**

jQuery

**参数**

**callback** (Function) : 待执行函数

**示例**

|  |
| --- |
| AJAX 请求完成时执行函数。  **jQuery 代码:**   $("#msg").ajaxComplete(function(request, settings){     $(this).append("<li>Request Complete.</li>");   }); |

### ajaxError(callback)

AJAX 请求发生错误时执行函数。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

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。

Attach a function to be executed whenever an AJAX request fails. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

The XMLHttpRequest and settings used for that request are passed as arguments to the callback. A third argument, an exception object, is passed if an exception occured while processing the request.

**返回值**

jQuery

**参数**

**callback** (Function) : 待执行函数

**示例**

|  |
| --- |
| AJAX 请求失败时显示信息。  **jQuery 代码:**   $("#msg").ajaxError(function(request, settings){     $(this).append("<li>Error requesting page " + settings.url +  "</li>");   }); |

### ajaxSend(callback)

AJAX 请求发送前执行函数。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

Attach a function to be executed before an AJAX request is sent. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

**返回值**

jQuery

**参数**

**callback** (Function) : 待执行函数

**示例**

|  |
| --- |
| AJAX 请求发送前显示信息。  **jQuery 代码:**   $("#msg").ajaxSend(function(evt, request, settings){     $(this).append("<li>Starting request at " + settings.url +  "</li>");   }); |

### ajaxStart(callback)

AJAX 请求开始时执行函数。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

Attach a function to be executed whenever an AJAX request begins and there is none already active. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

**返回值**

jQuery

**参数**

**callback** (Function) : 待执行函数

**示例**

|  |
| --- |
| AJAX 请求开始时显示信息。  **jQuery 代码:**   $("#loading").ajaxStart(function(){     $(this).show();   }); |

### ajaxStop(callback)

AJAX 请求结束时执行函数。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

Attach a function to be executed whenever all AJAX requests have ended. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

**返回值**

jQuery

**参数**

**callback** (Function) : 待执行函数

**示例**

|  |
| --- |
| AJAX 请求结束后隐藏信息。  **jQuery 代码:**   $("#loading").ajaxStop(function(){     $(this).hide();   }); |

### ajaxSuccess(callback)

AJAX 请求成功时执行函数。[Ajax 事件](http://docs.jquery.com/Ajax_Events)。

XMLHttpRequest 对象和设置作为参数传递给回调函数。

Attach a function to be executed whenever an AJAX request completes successfully. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

**返回值**

jQuery

**参数**

**callback** (Function) : 待执行函数

**示例**

|  |
| --- |
| 当 AJAX 请求成功后显示消息。  **jQuery 代码:**   $("#msg").ajaxSuccess(function(evt, request, settings){     $(this).append("<li>Successful Request!</li>");   });  其它 |

### jQuery.ajaxSetup(options)

设置全局 AJAX 默认选项。

参数见 '$.ajax' 说明。

Setup global settings for AJAX requests.

See '$.ajax' for a description of all available options.

**返回值**

jQuery

**参数**

**options** (可选) : 选项设置。所有设置项均为可选设置。.

**示例**

|  |
| --- |
| 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()

序列表表格内容为字符串。

Serializes a set of input elements into a string of data. This will serialize all given elements.

**返回值**

jQuery

**示例**

|  |
| --- |
| 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>" ); |

### serializeArray()

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

Serializes all forms and form elements (like the '.serialize()' method) but returns a JSON data structure for you to work with.

**返回值**

jQuery

**示例**

|  |
| --- |
| 取得表单内容并插入到网页中。  **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:**  var fields = $("select, :radio").serializeArray(); jQuery.each( fields, function(i, field){   $("#results").append(field.value + " "); }); |

# 工具

浏览器

jQuery.browser

### jQuery.browser

浏览器内核标识。依据 navigator.userAgent 判断。

可用值:

safari

opera

msie

mozilla

此属性在 DOM 树加载完成前即有效，可用于为特定浏览器设置 ready 事件。

浏览器对象检测技术与此属性共同使用可提供可靠的浏览器检测支持。

Contains flags for the useragent, read from navigator.userAgent.

Available flags are:

safari

opera

msie

mozilla

This property is available before the DOM is ready, therefore you can use it to add ready events only for certain browsers.

There are situations where object detection is not reliable enough, in such cases it makes sense to use browser detection.

A combination of browser and object detection yields quite reliable results.

**返回值**

# Map

**示例**

|  |
| --- |
| 在 Microsoft's Internet Explorer 浏览器中返回 true。  **jQuery 代码:**  $.browser.msie |
| 仅在 Safari 中提示 "this is safari!" 。  **jQuery 代码:**  if ($.browser.safari) {     alert("this is safari!");  } |

### jQuery.browser.version

浏览器渲染引擎版本号。

典型结果:

Internet Explorer: 6.0, 7.0

Opera: 9.20

Safari/Webkit: 312.8, 418.9

Mozilla/Firefox/Flock/Camino: 1.7.12, 1.8.1.3

Opera: 9.20

Safari/Webkit: 312.8, 418.9

**返回值**String

**示例**

|  |
| --- |
| 显示当前 IE 浏览器版本号。  **jQuery 代码:**  if ( $.browser.msie )   alert( $.browser.version );  jQuery.boxModel |

### jQuery.boxModel

当前页面中浏览器是否使用标准盒模型渲染页面。 [W3C CSS 盒模型](http://www.w3.org/TR/REC-CSS2/box.html).

States if the current page, in the user's browser, is being rendered using the [W3C CSS Box Model](http://www.w3.org/TR/REC-CSS2/box.html).

**返回值**Boolean

**示例**

|  |
| --- |
| 在 Internet Explorer 怪癖模式（QuirksMode）中返回 False。  **jQuery 代码:**  $.boxModel  数组和对象操作 |

### jQuery.each(obj,callback)

通用例遍方法，可用于例遍对象和数组。

不同于例遍 jQuery 对象的 $().each() 方法，此方法可用于例遍任何对象。

回调函数拥有两个参数：第一个为对象的成员或数组的索引，第二个为对应变量或内容。

如果需要退出 each 循环可使回调函数返回 false，其它返回值将被忽略。

A generic iterator function, which can be used to seamlessly iterate over both objects and arrays.

This function is not the same as $().each() - which is used to iterate, exclusively, over a jQuery object. This function can be used to iterate over anything.

The callback has two arguments:the key (objects) or index (arrays) as the first, and the value as the second.

If you wish to break the each() loop at a particular iteration you can do so by making your function return false. Other return values are ignored.

**返回值**Object

**参数**

**object** (Object) : 需要例遍的对象或数组。

**callback** (Function) : (可选) 每个成员/元素执行的回调函数。

**示例**

|  |
| --- |
| 例遍数组，同时使用元素索引和内容。  **jQuery 代码:**  $.each( [3,5,8], function(i, n){ //i是脚标, n是元素值    alert( "Item #" + i + ": " + n );  }); |
| 例遍对象，同时使用成员名称和变量内容。  **jQuery 代码:**  $.each( { name: "John", lang: "JS" }, function(i, n){ //i是名称; n是值    alert( "Name: " + i + ", Value: " + n );  }); |

### jQuery.extend(target,obj1,[objN])

用一个或多个其他对象来扩展一个对象，返回被扩展的对象。

用于简化继承。

Extend one object with one or more others, returning the original, modified, object.

This is a great utility for simple inheritance.

**返回值**Object

**参数**

**target** (Object) : 待修改对象。

**object1** (Object) : 待合并到第一个对象的对象。

**objectN** (Object) : (可选) 待合并到第一个对象的对象。

**示例**

|  |
| --- |
| 合并 settings 和 options，修改并返回 settings。  **jQuery 代码:**  var settings = { validate: false, limit: 5, name: "foo" };  var options = { validate: true, name: "bar" };  jQuery.extend(settings, options);  **结果:**  settings == { validate: true, limit: 5, name: "bar" } |
| 合并 defaults 和 options, 不修改 defaults。  **jQuery 代码:**  var empty = {}  var defaults = { validate: false, limit: 5, name: "foo" };  var options = { validate: true, name: "bar" };  var settings = jQuery.extend(empty, defaults, options);  **结果:**  settings == { validate: true, limit: 5, name: "bar" }  empty == { validate: true, limit: 5, name: "bar" } |

### jQuery.grep(array,callback,[invert])

使用过滤函数 过滤数组元素。

此函数至少传递两个参数：待过滤数组和过滤函数。过滤函数必须返回 true 以保留元素或 false 以删除元素。

Filter items out of an array, by using a filter function.

The specified function will be passed two arguments: The current array item and the filter function. The function must return 'true' to keep the item in the array, false to remove it.

**返回值**Array

**参数**

**array** (Array) : 待过滤数组。

**callback** (Function) : 此函数将处理数组每个元素。第一个参数为当前元素，第二个参数而元素索引值。此函数应返回一个布尔值。另外，此函数可设置为一个字符串，当设置为字符串时，将视为“lambda-form”（缩写形式？），其中 a 代表数组元素，i 代表元素索引值。如“a > 0”代表“function(a){ return a > 0; }”。

**invert** (Boolean) : (可选) 如果 "invert" 为 false 或为设置，则函数返回数组中由过滤函数返回 true 的元素，当"invert" 为 true，则返回过滤函数中返回 false 的元素集。

**示例**

|  |
| --- |
| 过滤数组中小于 0 的元素。  **jQuery 代码:**  $.grep( [0,1,2], function(n,i){    return n > 0;  });  **结果:**  [1, 2] |
| 排除数组中大于 0 的元素，使用第三个参数进行排除。  **jQuery 代码:**  $.grep( [0,1,2], function(n,i){    return n > 0;  }, true);  **结果:**  [0] |

### jQuery.makeArray(obj)

将类数组对象 转换为数组对象。

类数组对象有 length 属性，其成员索引为 0 至 length - 1。实际中此函数在 jQuery 中将自动使用而无需特意转换。

Turns an array-like object into a true array.

Array-like objects have a length property and its properties are numbered from 0 to length - 1. Typically it will be unnecessary to use this function if you are using jQuery which uses this function internally.

**返回值**Array

**参数**

**obj** (Object) : 类数组对象。

**示例**

|  |
| --- |
| 过滤数组中小于 0 的元素。  **HTML 代码:**  <div>First</div><div>Second</div><div>Third</div><div>Fourth</div>  **jQuery 代码:**  var arr = jQuery.makeArray(document.getElementsByTagName("div"));  arr.reverse(); // 使用数组翻转函数  **结果:**  Fourth  Third  Second  First |

### jQuery.map(array,callback)

将一个数组中的元素 转换到另一个数组中。

作为参数的转换函数会为每个数组元素调用，而且会给这个转换函数传递一个表示被转换的元素作为参数。转换函数可以返回转换后的值、null（删除数组中的项目）或一个包含值的数组，并扩展至原始数组中。

Translate all items in an array to another array of items.

The translation function that is provided to this method is called for each item in the array and is passed one argument: The item to be translated. The function can then return the translated value, 'null' (to remove the item), or an array of values - which will be flattened into the full array.

**返回值**Array

**参数**

**array** (Array) : 待转换数组。

**callback** (Function) : 为每个数组元素调用，而且会给这个转换函数传递一个表示被转换的元素作为参数。函数可返回任何值。另外，此函数可设置为一个字符串，当设置为字符串时，将视为“lambda-form”（缩写形式？），其中 a 代表数组元素。如“a \* a”代表“function(a){ return a \* a; }”。

**示例**

|  |
| --- |
| 将原数组中每个元素加 4 转换为一个新数组。  **jQuery 代码:**  $.map( [0,1,2], function(n){    return n + 4;  });  **结果:**  [4, 5, 6] |
| 原数组中大于 0 的元素加 1 ，否则删除。  **jQuery 代码:**  $.map( [0,1,2], function(n){    return n > 0 ? n + 1 : null;  });  **结果:**  [2, 3] |
| 原数组中每个元素扩展为一个包含其本身和其值加 1 的数组，并转换为一个新数组。  **jQuery 代码:**  $.map( [0,1,2], function(n){    return [ n, n + 1 ];  });  **结果:**  [0, 1, 1, 2, 2, 3] |

### jQuery.unique(array)

删除数组中重复元素。

**返回值**

jQuery

**参数**

**array** (Array) : 待处理数组。

**示例**

|  |
| --- |
| 删除重复 div 标签。  **jQuery 代码:**  $.unique(document.getElementsByTagName("div"));  **结果:**  [<div>, <div>, ...] |

### jQuery.isFunction(obj)

测试对象是否为函数。

Determine if the parameter passed is a function.

**返回值**

jQuery

**参数**

**obj** (Object) : 用于测试是否为函数的对象

**示例**

|  |
| --- |
| 检测是否为函数  **jQuery 代码:**  function stub() {  }  var objs = [  function () {},  { x:15, y:20 },  null,  stub,  "function"  ];  jQuery.each(objs, function (i) {  var isFunc = **jQuery.isFunction**(objs[i]);  $("span:eq( " + i + ")").text(isFunc);  });  **结果:**  [ true,false,false,true,false ] |

字符串操作

### jQuery.trim(str)

去掉字符串起始和结尾的空格。

Remove the whitespace from the beginning and end of a string.

**返回值**

jQuery

**参数**

**str** (String) : 需要处理的字符串

**示例**

|  |
| --- |
| 去掉字符串起始和结尾的空格。  **jQuery 代码:**  $.trim(" hello, how are you? ");  **结果:**  "hello, how are you?" |