## jQuery简介

jQuery 库位于一个 JavaScript 文件中，其中包含了所有的 jQuery 函数。

可以通过下面的标记把 jQuery 添加到网页中：

<head>

<script type="text/javascript" src="jquery.js"></script>

</head>

请注意，<script> 标签应该位于页面的 <head> 部分。

## jQuery 语法

jQuery 语法是为 HTML 元素的选取编制的，可以对元素执行某些操作。

基础语法是：$(selector).action()

1、美元符号定义 jQuery

2、选择符（selector）“查询”和“查找” HTML 元素

3、jQuery 的 action() 执行对元素的操作

示例

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$(".test").hide() - 隐藏所有 class="test" 的所有元素

$("#test").hide() - 隐藏所有 id="test" 的元素

**提示：**jQuery 使用的语法是 XPath 与 CSS 选择器语法的组合。在本教程接下来的章节，您将学习到更多有关选择器的语法。

## 文档就绪函数

您也许已经注意到在我们的实例中的所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){

--- jQuery functions go here ----

});

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

如果在文档没有完全加载之前就运行函数，操作可能失败。下面是两个具体的例子：

1、试图隐藏一个不存在的元素

2、获得未完全加载的图像的大小

## jQuery noConflict() 方法

正如您已经了解到的，jQuery 使用 $ 符号作为 jQuery 的简写。

如果其他 JavaScript 框架也使用 $ 符号作为简写怎么办？

其他一些 JavaScript 框架包括：MooTools、Backbone、Sammy、Cappuccino、Knockout、JavaScript MVC、Google Web Toolkit、Google Closure、Ember、Batman 以及 Ext JS。

其中某些框架也使用 $ 符号作为简写（就像 jQuery），如果您在用的两种不同的框架正在使用相同的简写符号，有可能导致脚本停止运行。

noConflict() 方法会释放会 $ 标识符的控制，这样其他脚本就可以使用它了。

1、当然，您仍然可以通过全名替代简写的方式来使用 jQuery：

$.noConflict();

jQuery(document).ready(function(){

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

jQuery("p").text("jQuery 仍在运行！");

});

});

2、您也可以创建自己的简写。noConflict() 可返回对 jQuery 的引用，您可以把它存入变量，以供稍后使用。请看这个例子：

var jq = $.noConflict();

jq(document).ready(function(){

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

jq("p").text("jQuery 仍在运行！");

});

});

3、如果你的 jQuery 代码块使用 $ 简写，并且您不愿意改变这个快捷方式，那么您可以把 $ 符号作为变量传递给 ready 方法。这样就可以在函数内使用 $ 符号了 - 而在函数外，依旧不得不使用 "jQuery"：

$.noConflict();

jQuery(document).ready(function($){

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

$("p").text("jQuery 仍在运行！");

});

});

# jQuery 选择器

jQuery 元素选择器和属性选择器允许您通过标签名、属性名或内容对 HTML 元素进行选择。

选择器允许您对 HTML 元素组或单个元素进行操作。

在 HTML DOM 术语中：选择器允许您对 DOM 元素组或单个 DOM 节点进行操作。

## jQuery 元素选择器

jQuery 使用 CSS 选择器来选取 HTML 元素。

$("p") 选取 <p> 元素。

$("p.intro") 选取所有 class="intro" 的 <p> 元素。

$("p#demo") 选取所有 id="demo" 的 <p> 元素。

## jQuery 属性选择器

jQuery 使用 XPath 表达式来选择带有给定属性的元素。

$("[href]") 选取所有带有 href 属性的元素。

$("[href='#']") 选取所有带有 href 值等于 "#" 的元素。

$("[href!='#']") 选取所有带有 href 值不等于 "#" 的元素。

$("[href$='.jpg']") 选取所有 href 值以 ".jpg" 结尾的元素。

## jQuery CSS 选择器

jQuery CSS 选择器可用于改变 HTML 元素的 CSS 属性。

下面的例子把所有 p 元素的背景颜色更改为红色：

$("p").css("background-color","red");

# jQuery 事件

jQuery 事件处理方法是 jQuery 中的核心函数。

事件处理程序指的是当 HTML 中发生某些事件时所调用的方法。术语由事件“触发”（或“激发”）经常会被使用。

通常会把 jQuery 代码放到 <head>部分的事件处理方法中。

|  |  |
| --- | --- |
| 方法 | 描述 |
| [bind()](http://www.w3school.com.cn/jquery/event_bind.asp) | 向匹配元素附加一个或更多事件处理器 |
| [blur()](http://www.w3school.com.cn/jquery/event_blur.asp) | 触发、或将函数绑定到指定元素的 blur 事件 |
| [change()](http://www.w3school.com.cn/jquery/event_change.asp) | 触发、或将函数绑定到指定元素的 change 事件 |
| [click()](http://www.w3school.com.cn/jquery/event_click.asp) | 触发、或将函数绑定到指定元素的 click 事件 |
| [dblclick()](http://www.w3school.com.cn/jquery/event_dblclick.asp) | 触发、或将函数绑定到指定元素的 double click 事件 |
| [delegate()](http://www.w3school.com.cn/jquery/event_delegate.asp) | 向匹配元素的当前或未来的子元素附加一个或多个事件处理器 |
| [die()](http://www.w3school.com.cn/jquery/event_die.asp) | 移除所有通过 live() 函数添加的事件处理程序。 |
| [error()](http://www.w3school.com.cn/jquery/event_error.asp) | 触发、或将函数绑定到指定元素的 error 事件 |
| [event.isDefaultPrevented()](http://www.w3school.com.cn/jquery/event_isdefaultprevented.asp) | 返回 event 对象上是否调用了 event.preventDefault()。 |
| [event.pageX](http://www.w3school.com.cn/jquery/event_pagex.asp) | 相对于文档左边缘的鼠标位置。 |
| [event.pageY](http://www.w3school.com.cn/jquery/event_pagey.asp) | 相对于文档上边缘的鼠标位置。 |
| [event.preventDefault()](http://www.w3school.com.cn/jquery/event_preventdefault.asp) | 阻止事件的默认动作。 |
| [event.result](http://www.w3school.com.cn/jquery/event_result.asp) | 包含由被指定事件触发的事件处理器返回的最后一个值。 |
| [event.target](http://www.w3school.com.cn/jquery/event_target.asp) | 触发该事件的 DOM 元素。 |
| [event.timeStamp](http://www.w3school.com.cn/jquery/event_timeStamp.asp) | 该属性返回从 1970 年 1 月 1 日到事件发生时的毫秒数。 |
| [event.type](http://www.w3school.com.cn/jquery/event_type.asp) | 描述事件的类型。 |
| [event.which](http://www.w3school.com.cn/jquery/event_which.asp) | 指示按了哪个键或按钮。 |
| [focus()](http://www.w3school.com.cn/jquery/event_focus.asp) | 触发、或将函数绑定到指定元素的 focus 事件 |
| [keydown()](http://www.w3school.com.cn/jquery/event_keydown.asp) | 触发、或将函数绑定到指定元素的 key down 事件 |
| [keypress()](http://www.w3school.com.cn/jquery/event_keypress.asp) | 触发、或将函数绑定到指定元素的 key press 事件 |
| [keyup()](http://www.w3school.com.cn/jquery/event_keyup.asp) | 触发、或将函数绑定到指定元素的 key up 事件 |
| [live()](http://www.w3school.com.cn/jquery/event_live.asp) | 为当前或未来的匹配元素添加一个或多个事件处理器 |
| [load()](http://www.w3school.com.cn/jquery/event_load.asp) | 触发、或将函数绑定到指定元素的 load 事件 |
| [mousedown()](http://www.w3school.com.cn/jquery/event_mousedown.asp) | 触发、或将函数绑定到指定元素的 mouse down 事件 |
| [mouseenter()](http://www.w3school.com.cn/jquery/event_mouseenter.asp) | 触发、或将函数绑定到指定元素的 mouse enter 事件 |
| [mouseleave()](http://www.w3school.com.cn/jquery/event_mouseleave.asp) | 触发、或将函数绑定到指定元素的 mouse leave 事件 |
| [mousemove()](http://www.w3school.com.cn/jquery/event_mousemove.asp) | 触发、或将函数绑定到指定元素的 mouse move 事件 |
| [mouseout()](http://www.w3school.com.cn/jquery/event_mouseout.asp) | 触发、或将函数绑定到指定元素的 mouse out 事件 |
| [mouseover()](http://www.w3school.com.cn/jquery/event_mouseover.asp) | 触发、或将函数绑定到指定元素的 mouse over 事件 |
| [mouseup()](http://www.w3school.com.cn/jquery/event_mouseup.asp) | 触发、或将函数绑定到指定元素的 mouse up 事件 |
| [one()](http://www.w3school.com.cn/jquery/event_one.asp) | 向匹配元素添加事件处理器。每个元素只能触发一次该处理器。 |
| [ready()](http://www.w3school.com.cn/jquery/event_ready.asp) | 文档就绪事件（当 HTML 文档就绪可用时） |
| [resize()](http://www.w3school.com.cn/jquery/event_resize.asp) | 触发、或将函数绑定到指定元素的 resize 事件 |
| [scroll()](http://www.w3school.com.cn/jquery/event_scroll.asp) | 触发、或将函数绑定到指定元素的 scroll 事件 |
| [select()](http://www.w3school.com.cn/jquery/event_select.asp) | 触发、或将函数绑定到指定元素的 select 事件 |
| [submit()](http://www.w3school.com.cn/jquery/event_submit.asp) | 触发、或将函数绑定到指定元素的 submit 事件 |
| [toggle()](http://www.w3school.com.cn/jquery/event_toggle.asp) | 绑定两个或多个事件处理器函数，发生轮流click 事件时执行。 |
| [trigger()](http://www.w3school.com.cn/jquery/event_trigger.asp) | 所有匹配元素的指定事件 |
| [triggerHandler()](http://www.w3school.com.cn/jquery/event_triggerhandler.asp) | 第一个被匹配元素的指定事件 |
| [unbind()](http://www.w3school.com.cn/jquery/event_unbind.asp) | 从匹配元素移除一个被添加的事件处理器 |
| [undelegate()](http://www.w3school.com.cn/jquery/event_undelegate.asp) | 从匹配元素移除一个被添加的事件处理器，现在或将来 |
| [unload()](http://www.w3school.com.cn/jquery/event_unload.asp) | 触发、或将函数绑定到指定元素的 unload 事件 |

## 单独文件中的函数

如果您的网站包含许多页面，并且您希望您的 jQuery 函数易于维护，那么请把您的 jQuery 函数放到独立的 .js 文件中。

当我们在教程中演示 jQuery 时，会将函数直接添加到 <head> 部分中。不过，把它们放到一个单独的文件中会更好，就像这样（通过 src 属性来引用文件）：

<head>

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript" src="my\_jquery\_functions.js"></script>

</head>

## jQuery 名称冲突

jQuery 使用 $ 符号作为 jQuery 的简介方式，某些其他 JavaScript 库中的函数（比如 Prototype）同样使用 $ 符号。jQuery 使用名为 noConflict() 的方法来解决该问题。

var jq=jQuery.noConflict()，帮助您使用自己的名称（比如 jq）来代替 $ 符号。

## 结论

由于 jQuery 是为处理 HTML 事件而特别设计的，那么当您遵循以下原则时，您的代码会更恰当且更易维护：

1、把所有 jQuery 代码置于事件处理函数中

2、把所有事件处理函数置于文档就绪事件处理器中

3、把 jQuery 代码置于单独的 .js 文件中

4、如果存在名称冲突，则重命名 jQuery 库

# jQuery 效果

|  |  |
| --- | --- |
| 方法 | 描述 |
| [animate()](http://www.w3school.com.cn/jquery/effect_animate.asp) | 对被选元素应用“自定义”的动画 |
| [clearQueue()](http://www.w3school.com.cn/jquery/effect_clearqueue.asp) | 对被选元素移除所有排队的函数（仍未运行的） |
| delay() | 对被选元素的所有排队函数（仍未运行）设置延迟 |
| dequeue() | 运行被选元素的下一个排队函数 |
| [fadeIn()](http://www.w3school.com.cn/jquery/effect_fadein.asp) | 逐渐改变被选元素的不透明度，从隐藏到可见 |
| [fadeOut()](http://www.w3school.com.cn/jquery/effect_fadeout.asp) | 逐渐改变被选元素的不透明度，从可见到隐藏 |
| [fadeTo()](http://www.w3school.com.cn/jquery/effect_fadeto.asp) | 把被选元素逐渐改变至给定的不透明度 |
| [hide()](http://www.w3school.com.cn/jquery/effect_hide.asp) | 隐藏被选的元素 |
| queue() | 显示被选元素的排队函数 |
| [show()](http://www.w3school.com.cn/jquery/effect_show.asp) | 显示被选的元素 |
| [slideDown()](http://www.w3school.com.cn/jquery/effect_slidedown.asp) | 通过调整高度来滑动显示被选元素 |
| [slideToggle()](http://www.w3school.com.cn/jquery/effect_slidetoggle.asp) | 对被选元素进行滑动隐藏和滑动显示的切换 |
| [slideUp()](http://www.w3school.com.cn/jquery/effect_slideup.asp) | 通过调整高度来滑动隐藏被选元素 |
| [stop()](http://www.w3school.com.cn/jquery/effect_stop.asp) | 停止在被选元素上运行动画 |
| [toggle()](http://www.w3school.com.cn/jquery/effect_toggle.asp) | 对被选元素进行隐藏和显示的切换 |

## jQuery hide() 和 show()

通过 jQuery，您可以使用 hide() 和 show() 方法来隐藏和显示 HTML 元素：

$(selector).hide(speed,callback);

$(selector).show(speed,callback);

1、可选的 speed 参数规定隐藏/显示的速度，可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是隐藏或显示完成后所执行的函数名称。

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

$("p").hide(1000);

});

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

$("p").hide();

});

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

$("p").show();

});

## jQuery toggle()

通过 jQuery，您可以使用 toggle() 方法来切换 hide() 和 show() 方法。

$(selector).toggle(speed,callback);

1、可选的 speed 参数规定隐藏/显示的速度，可以取以下值："slow"、"fast" 或毫秒。2、可选的 callback 参数是 toggle() 方法完成后所执行的函数名称。

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

$("p").toggle();

});

## jQuery fadeIn()

jQuery fadeIn() 用于淡入已隐藏的元素。

$(selector).fadeIn(speed,callback);

1、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeIn() 方法：

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

$("#div1").fadeIn();

$("#div2").fadeIn("slow");

$("#div3").fadeIn(3000);

});

## jQuery fadeOut()

jQuery fadeOut() 方法用于淡出可见元素。

$(selector).fadeOut(speed,callback);

1、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeOut() 方法：

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

$("#div1").fadeOut();

$("#div2").fadeOut("slow");

$("#div3").fadeOut(3000);

});

## jQuery fadeToggle()

jQuery fadeToggle() 方法可以在 fadeIn() 与 fadeOut() 方法之间进行切换。

如果元素已淡出，则 fadeToggle() 会向元素添加淡入效果。

如果元素已淡入，则 fadeToggle() 会向元素添加淡出效果。

$(selector).fadeToggle(speed,callback);

1、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是 fading 完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeToggle() 方法：

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

$("#div1").fadeToggle();

$("#div2").fadeToggle("slow");

$("#div3").fadeToggle(3000);

});

## jQuery fadeTo()

jQuery fadeTo() 方法允许渐变为给定的不透明度（值介于 0 与 1 之间）。

$(selector).fadeTo(speed,opacity,callback);

1、必需的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、fadeTo() 方法中必需的 opacity 参数将淡入淡出效果设置为给定的不透明度（值介于 0 与 1 之间）。

3、可选的 callback 参数是该函数完成后所执行的函数名称。

下面的例子演示了带有不同参数的 fadeTo() 方法：

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

$("#div1").fadeTo("slow",0.15);

$("#div2").fadeTo("slow",0.4);

$("#div3").fadeTo("slow",0.7);

});

## jQuery slideDown() 方法

jQuery slideDown() 方法用于向下滑动元素。（display:none）

$(selector).slideDown(speed,callback);

1、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideDown() 方法：

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

$("#panel").slideDown();

});

## jQuery slideUp()

jQuery slideUp() 方法用于向上滑动元素。

$(selector).slideUp(speed,callback);

1、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideUp() 方法：

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

$("#panel").slideUp();

});

## jQuery slideToggle()

jQuery slideToggle() 方法可以在 slideDown() 与 slideUp() 方法之间进行切换。

1、如果元素向下滑动，则 slideToggle() 可向上滑动它们。

2、如果元素向上滑动，则 slideToggle() 可向下滑动它们。

$(selector).slideToggle(speed,callback);

1、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

2、可选的 callback 参数是滑动完成后所执行的函数名称。

下面的例子演示了 slideToggle() 方法：

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

$("#panel").slideToggle();

});

## jQuery 动画

jQuery animate() 方法用于创建自定义动画。

$(selector).animate({params},speed,callback);

1、必需的 params 参数定义形成动画的 CSS 属性。

2、可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

3、可选的 callback 参数是动画完成后所执行的函数名称。

下面的例子演示 animate() 方法的简单应用；它把 <div> 元素移动到左边，直到 left 属性等于 250 像素为止：

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

$("div").animate({left:'250px'});

});

**提示：**默认地，所有 HTML 元素都有一个静态位置，且无法移动。

如需对位置进行操作，要记得首先把元素的 CSS position 属性设置为 relative、fixed 或 absolute！

### animate() - 操作多个属性

请注意，生成动画的过程中可同时使用多个属性：

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

$("div").animate({

left:'250px',

opacity:'0.5',

height:'150px',

width:'150px'

});

});

**提示：**可以用 animate() 方法来操作所有 CSS 属性吗？

是的，几乎可以！不过，需要记住一件重要的事情：当使用 animate() 时，必须使用 Camel 标记法书写所有的属性名，比如，必须使用 paddingLeft 而不是 padding-left，使用 marginRight 而不是 margin-right，等等。

同时，色彩动画并不包含在核心 jQuery 库中。如果需要生成颜色动画，您需要从 jQuery.com 下载 Color Animations 插件。

### animate() - 使用相对值

也可以定义相对值（该值相对于元素的当前值）。需要在值的前面加上 += 或 -=：

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

$("div").animate({

left:'250px',

height:'+=150px',

width:'+=150px'

});

});

### animate() - 使用预定义的值

您甚至可以把属性的动画值设置为 "show"、"hide" 或 "toggle"：

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

$("div").animate({

height:'toggle'

});

});

### animate() - 使用队列功能

默认地，jQuery 提供针对动画的队列功能。这意味着如果您在彼此之后编写多个 animate() 调用，jQuery 会创建包含这些方法调用的“内部”队列。然后逐一运行这些 animate 调用。

实例 1

隐藏，如果您希望在彼此之后执行不同的动画，那么我们要利用队列功能：

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

var div=$("div");

div.animate({height:'300px',opacity:'0.4'},"slow");

div.animate({width:'300px',opacity:'0.8'},"slow");

div.animate({height:'100px',opacity:'0.4'},"slow");

div.animate({width:'100px',opacity:'0.8'},"slow");

});

实例 2

下面的例子把 <div> 元素移动到右边，然后增加文本的字号：

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

var div=$("div");

div.animate({left:'100px'},"slow");

div.animate({fontSize:'3em'},"slow");

});

## jQuery stop() 方法

jQuery stop() 方法用于停止动画或效果，在它们完成之前。

stop() 方法适用于所有 jQuery 效果函数，包括滑动、淡入淡出和自定义动画。

$(selector).stop(stopAll,goToEnd);

1、可选的 stopAll 参数规定是否应该清除动画队列。默认是 false，即仅停止活动的动画，允许任何排入队列的动画向后执行。

2、可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

下面的例子演示 stop() 方法，不带参数：

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

$("#panel").stop();

});

## jQuery Callback 函数

当动画 100% 完成后，即调用 Callback 函数。

典型的语法：

$(selector).hide(speed,callback)

*callback* 参数是一个在 hide 操作完成后被执行的函数。

$("p").hide(1000,function(){

alert("The paragraph is now hidden");

});

## jQuery 方法链接

有一种名为链接（chaining）的技术，允许我们在相同的元素上运行多条 jQuery 命令，一条接着另一条。如需链接一个动作，您只需简单地把该动作追加到之前的动作上。

**例子 1**

下面的例子把 css(), slideUp(), and slideDown() 链接在一起。"p1" 元素首先会变为红色，然后向上滑动，然后向下滑动：

$("#p1").css("color","red").slideUp(2000).slideDown(2000);

**例子 2**

这样写也可以运行，jQuery 会抛掉多余的空格，并按照一行长代码来执行上面的代码行：

$("#p1").css("color","red")

.slideUp(2000)

.slideDown(2000);

# jQuery HTML

## jQuery 文档操作方法

这些方法对于 XML 文档和 HTML 文档均是适用的，除了：html()。

|  |  |
| --- | --- |
| 方法 | 描述 |
| [appendTo()](http://www.w3school.com.cn/jquery/manipulation_appendto.asp) | 向目标结尾插入匹配元素集合中的每个元素。 |
| [clone()](http://www.w3school.com.cn/jquery/manipulation_clone.asp) | 创建匹配元素集合的副本。 |
| [detach()](http://www.w3school.com.cn/jquery/manipulation_detach.asp) | 从 DOM 中移除匹配元素集合。 |
| [hasClass()](http://www.w3school.com.cn/jquery/attributes_hasclass.asp) | 检查匹配的元素是否拥有指定的类。 |
| [html()](http://www.w3school.com.cn/jquery/manipulation_html.asp) | 设置或返回匹配的元素集合中的 HTML 内容。 |
| [insertAfter()](http://www.w3school.com.cn/jquery/manipulation_insertafter.asp) | 把匹配的元素插入到另一个指定的元素集合的后面。 |
| [insertBefore()](http://www.w3school.com.cn/jquery/manipulation_insertbefore.asp) | 把匹配的元素插入到另一个指定的元素集合的前面。 |
| [prependTo()](http://www.w3school.com.cn/jquery/manipulation_perpendto.asp) | 向目标开头插入匹配元素集合中的每个元素。 |
| [removeAttr()](http://www.w3school.com.cn/jquery/attributes_removeattr.asp) | 从所有匹配的元素中移除指定的属性。 |
| [replaceAll()](http://www.w3school.com.cn/jquery/manipulation_replaceall.asp) | 用匹配的元素替换所有匹配到的元素。 |
| [replaceWith()](http://www.w3school.com.cn/jquery/manipulation_replacewith.asp) | 用新内容替换匹配的元素。 |
| [unwrap()](http://www.w3school.com.cn/jquery/manipulation_unwrap.asp) | 移除并替换指定元素的父元素。 |
| [wrap()](http://www.w3school.com.cn/jquery/manipulation_wrap.asp) | 把匹配的元素用指定的内容或元素包裹起来。 |
| [wrapAll()](http://www.w3school.com.cn/jquery/manipulation_wrapall.asp) | 把所有匹配的元素用指定的内容或元素包裹起来。 |
| [wrapinner()](http://www.w3school.com.cn/jquery/manipulation_wrapinner.asp) | 将每一个匹配的元素的子内容用指定的内容或元素包裹起来。 |

## 获得内容text()、html()、val()

三个简单实用的用于 DOM 操作的 jQuery 方法：

1、text() - 设置或返回所选元素的文本内容

2、html() - 设置或返回所选元素的内容（包括 HTML 标记）

3、val() - 设置或返回表单字段的值

下面的例子演示如何通过 jQuery text() 和 html() 方法来获得内容：

$("#btn1").click(function(){

alert("Text: " + $("#test").text());

});

$("#btn2").click(function(){

alert("HTML: " + $("#test").html());

});

下面的例子演示如何通过 jQuery val() 方法获得输入字段的值：

$("#btn1").click(function(){

alert("Value: " + $("#test").val());

});

## 获取属性attr()

jQuery attr() 方法用于获取属性值。

下面的例子演示如何获得链接中 href 属性的值：

实例

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

alert($("#w3s").attr("href"));

});

## 设置内容text()、html()、val()

1、text() - 设置或返回所选元素的文本内容

2、html() - 设置或返回所选元素的内容（包括 HTML 标记）

3、val() - 设置或返回表单字段的值

下面的例子演示如何通过 text()、html() 以及 val() 方法来设置内容：

$("#btn1").click(function(){

$("#test1").text("Hello world!");

});

$("#btn2").click(function(){

$("#test2").html("<b>Hello world!</b>");

});

$("#btn3").click(function(){

$("#test3").val("Dolly Duck");

});

## text()、html()、val() 的回调函数

上面的三个 jQuery 方法：text()、html() 以及 val()，同样拥有回调函数。回调函数由两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回您希望使用的字符串。

下面的例子演示带有回调函数的 text() 和 html()：

实例

$("#btn1").click(function(){

$("#test1").text(function(i,origText){

return "Old text: " + origText + " New text: Hello world!

(index: " + i + ")";

});

});

$("#btn2").click(function(){

$("#test2").html(function(i,origText){

return "Old html: " + origText + " New html: Hello <b>world!</b>

(index: " + i + ")";

});

});

## 设置属性attr()

jQuery attr() 方法也用于设置/改变属性值。

下面的例子演示如何改变（设置）链接中 href 属性的值：

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

$("#w3s").attr("href","http://www.w3school.com.cn/jquery");

});

attr() 方法也允许您同时设置多个属性。

下面的例子演示如何同时设置 href 和 title 属性：

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

$("#w3s").attr({

"href" : "http://www.w3school.com.cn/jquery",

"title" : "W3School jQuery Tutorial"

});

});

## attr() 的回调函数

jQuery 方法 attr()，也提供回调函数。回调函数由两个参数：被选元素列表中当前元素的下标，以及原始（旧的）值。然后以函数新值返回您希望使用的字符串。

下面的例子演示带有回调函数的 attr() 方法：

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

$("#w3s").attr("href", function(i,origValue){

return origValue + "/jquery";

});

});

## jQuery append() 和prepend()

1、jQuery append() 方法在被选元素的结尾插入内容。

2、jQuery prepend() 方法在被选元素的开头插入内容。

$("p").append("Some appended text.");

$("p").prepend("Some prepended text.");

append() & prepend()是在元素内插入内容（该内容变成该元素的子元素或节点）

## 通过 append() 和 prepend() 添加若干新元素

append() 和 prepend() 方法能够通过参数接收无限数量的新元素。可以通过 jQuery 来生成文本/HTML（就像上面的例子那样），或者通过 JavaScript 代码和 DOM 元素。

在下面的例子中，我们创建若干个新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 append() 方法把这些新元素追加到文本中（对 prepend() 同样有效）：

function appendText()

{

var txt1="<p>Text.</p>"; // 以 HTML 创建新元素

var txt2=$("<p></p>").text("Text."); // 以 jQuery 创建新元素

var txt3=document.createElement("p"); // 以 DOM 创建新元素

txt3.innerHTML="Text.";

$("p").append(txt1,txt2,txt3); // 追加新元素

}

## jQuery after() 和 before()

1、jQuery after() 方法在被选元素之后插入内容。

2、jQuery before() 方法在被选元素之前插入内容。

$("img").after("Some text after");

$("img").before("Some text before");

after() & before()是在元素的外面插入内容（其内容变成元素的兄弟节点）

## 通过 after() 和 before() 添加若干新元素

after() 和 before() 方法能够通过参数接收无限数量的新元素。可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建新元素。

在下面的例子中，我们创建若干新元素。这些元素可以通过 text/HTML、jQuery 或者 JavaScript/DOM 来创建。然后我们通过 after() 方法把这些新元素插到文本中（对 before() 同样有效）：

function afterText()

{

var txt1="<b>I </b>"; // 以 HTML 创建新元素

var txt2=$("<i></i>").text("love "); // 通过 jQuery 创建新元素

var txt3=document.createElement("big"); // 通过 DOM 创建新元素

txt3.innerHTML="jQuery!";

$("img").after(txt1,txt2,txt3); // 在 img 之后插入新元素

}

## jQuery remove()

jQuery remove() 方法删除被选元素及其子元素。

$("#div1").remove();

jQuery remove() 方法也可接受一个参数，允许您对被删元素进行过滤。

该参数可以是任何 jQuery 选择器的语法。

下面的例子删除 class="italic" 的所有 <p> 元素：

$("p").remove(".italic");

## jQuery empty()

jQuery empty() 方法删除被选元素的子元素。

$("#div1").empty();

## jQuery addClass()

下面的例子展示如何向不同的元素添加 class 属性。当然，在添加类时，您也可以选取多个元素：

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

$("h1,h2,p").addClass("blue");

$("div").addClass("important");

});

您也可以在 addClass() 方法中规定多个类：

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

$("#div1").addClass("important blue");

});

## jQuery removeClass()

下面的例子演示如何不同的元素中删除指定的 class 属性：

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

$("h1,h2,p").removeClass("blue");

});

## jQuery toggleClass()

下面的例子将展示如何使用 jQuery toggleClass() 方法。该方法对被选元素进行添加/删除类的切换操作：

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

$("h1,h2,p").toggleClass("blue");

});

## jQuery css()

css() 方法设置或返回被选元素的一个或多个样式属性。

### 返回 CSS 属性

如需返回指定的 CSS 属性的值，请使用如下语法：

css("propertyname");

下面的例子将返回**首个匹配元素**的 background-color 值：

$("p").css("background-color");

### ****设置 CSS 属性****

如需设置指定的 CSS 属性，请使用如下语法：

css("propertyname","value");

下面的例子将为所有匹配元素设置 background-color 值：

$("p").css("background-color","yellow");

注：width和height属性可以直接设置，见下面“jQuery更多的width()和height()”

### 设置多个 CSS 属性

如需设置多个 CSS 属性，请使用如下语法：

css({"propertyname":"value","propertyname":"value",...});

下面的例子将为所有匹配元素设置 background-color 和 font-size：

$("p").css({"background-color":"yellow","font-size":"200%"});

## jQuery width() 和 height()

1、width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

2、height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

下面的例子返回指定的 <div> 元素的宽度和高度：

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

var txt="";

txt+="Width: " + $("#div1").width() + "</br>";

txt+="Height: " + $("#div1").height();

$("#div1").html(txt);

});

## jQuery innerWidth() 和 innerHeight()

1、innerWidth() 方法返回元素的宽度（包括内边距）。

2、innerHeight() 方法返回元素的高度（包括内边距）。

下面的例子返回指定的 <div> 元素的 inner-width/height：

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

var txt="";

txt+="Inner width: " + $("#div1").innerWidth() + "</br>";

txt+="Inner height: " + $("#div1").innerHeight();

$("#div1").html(txt);

});

## jQuery outerWidth() 和 outerHeight()

1、outerWidth() 方法返回元素的宽度（包括内边距和边框）。

2、outerHeight() 方法返回元素的高度（包括内边距和边框）。

下面的例子返回指定的 <div> 元素的 outer-width/height：

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

var txt="";

txt+="Outer width: " + $("#div1").outerWidth() + "</br>";

txt+="Outer height: " + $("#div1").outerHeight();

$("#div1").html(txt);

});

1、outerWidth(true) 方法返回元素的宽度（包括内边距、边框和外边距）。

2、outerHeight(true) 方法返回元素的高度（包括内边距、边框和外边距）。

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

var txt="";

txt+="Outer width (+margin): " + $("#div1").outerWidth(true) + "</br>";

txt+="Outer height (+margin): " + $("#div1").outerHeight(true);

$("#div1").html(txt);

});

## jQuery更多的 width() 和 height()

下面的例子返回文档（HTML 文档）和窗口（浏览器视口）的宽度和高度：

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

var txt="";

txt+="Document width/height: " + $(document).width();

txt+="x" + $(document).height() + "\n";

txt+="Window width/height: " + $(window).width();

txt+="x" + $(window).height();

alert(txt);

});

下面的例子设置指定的 <div> 元素的宽度和高度：

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

$("#div1").width(500).height(500);

});

# jQuery遍历

通过 jQuery 遍历，您能够从被选（当前的）元素开始，轻松地在家族树中向上移动（祖先），向下移动（子孙），水平移动（同胞）。这种移动被称为对 DOM 进行遍历。

## 遍历 DOM

jQuery 提供了多种遍历 DOM 的方法。

遍历方法中最大的种类是树遍历（tree-traversal）。

## jQuery parent()

parent() 方法返回**被选元素的直接父元素**。

该方法只会向上一级对 DOM 树进行遍历。

下面的例子返回每个 <span> 元素的的直接父元素：

$(document).ready(function(){

$("span").parent();

});

## jQuery parents()

parents() 方法返回**被选元素的所有祖先元素**，它一路向上直到文档的根元素 (<html>)。

下面的例子返回所有 <span> 元素的所有祖先：

$(document).ready(function(){

$("span").parents();

});

您也可以使用可选参数来过滤对祖先元素的搜索。

下面的例子返回所有 <span> 元素的所有祖先，并且它是 <ul> 元素：

$(document).ready(function(){

$("span").parents("ul");

});

## jQuery parentsUntil()

parentsUntil() 方法返回**介于两个给定元素之间的所有祖先元素**。

下面的例子返回介于 <span> 与 <div> 元素之间的所有祖先元素：

$(document).ready(function(){

$("span").parentsUntil("div");

});

## jQuery children()

children() 方法返回**被选元素的所有直接子元素**。

该方法只会向下一级对 DOM 树进行遍历。

下面的例子返回每个 <div> 元素的所有直接子元素：

$(document).ready(function(){

$("div").children();

});

您也可以使用可选参数来过滤对子元素的搜索。

下面的例子返回类名为 "1" 的所有 <p> 元素，并且它们是 <div> 的直接子元素：

$(document).ready(function(){

$("div").children("p.1");

});

## jQuery find()

find() 方法返回**被选元素的后代元素**，一路向下直到最后一个后代。

下面的例子返回属于 <div> 后代的所有 <span> 元素：

$(document).ready(function(){

$("div").find("span");

});

下面的例子返回 <div> 的所有后代：

$(document).ready(function(){

$("div").find("\*");

});

## jQuery siblings()

siblings() 方法返回被选元素的所有同胞元素。

下面的例子返回 <h2> 的所有同胞元素：

$(document).ready(function(){

$("h2").siblings();

});

您也可以使用可选参数来过滤对同胞元素的搜索。

下面的例子返回属于 <h2> 的同胞元素的所有 <p> 元素：

$(document).ready(function(){

$("h2").siblings("p");

});

## jQuery next() 、prev()

next() 方法返回被选元素的下一个同胞元素，prev()方法的工作方式类似，但方向相反。

该方法只返回一个元素。

下面的例子返回 <h2> 的下一个同胞元素：

$(document).ready(function(){

$("h2").next();

});

## jQuery nextAll()、prevAll()

nextAll() 方法返回被选元素的所有跟随的同胞元素，prevAll()方法的工作方式类似，但方向相反。

下面的例子返回 <h2> 的所有跟随的同胞元素：

$(document).ready(function(){

$("h2").nextAll();

});

## jQuery nextUntil()、prevUntil()

nextUntil() 方法返回介于两个给定参数之间的所有跟随的同胞元素，prevUntil ()方法的工作方式类似，但方向相反。

下面的例子返回介于 <h2> 与 <h6> 元素之间的所有同胞元素：

$(document).ready(function(){

$("h2").nextUntil("h6");

});

## jQuery first()

first() 方法返回被选元素的首个元素。

下面的例子选取首个 <div> 元素内部的第一个 <p> 元素：

$(document).ready(function(){

$("div p").first();

});

## jQuery last()

last() 方法返回被选元素的最后一个元素。

下面的例子选择最后一个 <div> 元素中的最后一个 <p> 元素：

$(document).ready(function(){

$("div p").last();

});

## jQuery eq()

eq() 方法返回被选元素中带有指定索引号的元素。

索引号从 0 开始，因此首个元素的索引号是 0 而不是 1。下面的例子选取第二个 <p> 元素（索引号 1）：

$(document).ready(function(){

$("p").eq(1);

});

## jQuery filter()

filter() 方法允许您规定一个标准。不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回。

下面的例子返回带有类名 "intro" 的所有 <p> 元素：

$(document).ready(function(){

$("p").filter(".intro");

});

注:貌似没什么用？

## jQuery not()

not() 方法返回不匹配标准的所有元素。

**提示：**not() 方法与 filter() 相反。

下面的例子返回不带有类名 "intro" 的所有 <p> 元素：

$(document).ready(function(){

$("p").not(".intro");

});

# jQuery AJAX

Query 提供多个与 AJAX 有关的方法。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

## jQuery load()

jQuery load() 方法是简单但强大的 AJAX 方法。load() 方法从服务器加载数据，并把返回的数据放入被选元素中。

$(selector).load(URL,data,callback);

1、必需的 *URL* 参数规定您希望加载的 URL。

2、可选的 *data* 参数规定与请求一同发送的查询字符串键/值对集合。

3、可选的 *callback* 参数是 load() 方法完成后所执行的函数名称。

这是示例文件（"demo\_test.txt"）的内容：

<h2>jQuery and AJAX is FUN!!!</h2>

<p id="p1">This is some text in a paragraph.</p>

1、下面的例子会把文件 "demo\_test.txt" 的内容加载到指定的 <div> 元素中：

$("#div1").load("demo\_test.txt");

2、也可以把 jQuery 选择器添加到 URL 参数。下面的例子把 "demo\_test.txt" 文件中 id="p1" 的元素的内容，加载到指定的 <div> 元素中：

$("#div1").load("demo\_test.txt #p1");

3、可选的 callback 参数规定当 load() 方法完成后所要允许的回调函数。回调函数可以设置不同的参数：

responseTxt - 包含调用成功时的结果内容

statusTXT - 包含调用的状态

xhr - 包含 XMLHttpRequest 对象

下面的例子会在 load() 方法完成后显示一个提示框。如果 load() 方法已成功，则显示“外部内容加载成功！”，而如果失败，则显示错误消息：

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

$("#div1").load("demo\_test.txt",function(responseTxt,statusTxt,xhr){

if(statusTxt=="success")

alert("外部内容加载成功！");

if(statusTxt=="error")

alert("Error: "+xhr.status+": "+xhr.statusText);

});

});

## HTTP 请求：GET vs. POST

两种在客户端和服务器端进行请求-响应的常用方法是：GET 和 POST。

1、GET - 从指定的资源请求数据

2、POST - 向指定的资源提交要处理的数据

GET 基本上用于从服务器获得（取回）数据。注释：GET 方法可能返回缓存数据。

POST 也可用于从服务器获取数据。不过，POST 方法不会缓存数据，并且常用于连同请求一起发送数据。

## jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

$.get(URL,callback);

1、必需的 *URL* 参数规定您希望请求的 URL。

2、可选的 *callback* 参数是请求成功后所执行的函数名。

下面的例子使用 $.get() 方法从服务器上的一个文件中取回数据：

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

$.get("demo\_test.asp",function(data,status){

alert("Data: " + data + "\nStatus: " + status);

});

});

$.get() 的第一个参数是我们希望请求的 URL（"demo\_test.asp"）。

第二个参数是回调函数，其中第一个回调参数存有被请求页面的内容，第二个回调参数存有请求的状态。

**提示：**这个 ASP 文件 ("demo\_test.asp") 类似这样：

<%

response.write("This is some text from an external ASP file.")

%>

## jQuery $.post() 方法

$.post() 方法通过 HTTP POST 请求从服务器上请求数据。

$.post(URL,data,callback);

1、必需的 *URL* 参数规定您希望请求的 URL。

2、可选的 *data* 参数规定连同请求发送的数据。

3、可选的 *callback* 参数是请求成功后所执行的函数名。

下面的例子使用 $.post() 连同请求一起发送数据：

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

$.post("demo\_test\_post.asp",

{

name:"Donald Duck",

city:"Duckburg"

},

function(data,status){

alert("Data: " + data + "\nStatus: " + status);

});

});

$.post() 的第一个参数是我们希望请求的 URL ("demo\_test\_post.asp")。

然后我们连同请求（name 和 city）一起发送数据。

"demo\_test\_post.asp" 中的 ASP 脚本读取这些参数，对它们进行处理，然后返回结果。

第三个参数是回调函数。第一个回调参数存有被请求页面的内容，而第二个参数存有请求的状态。

**提示：**这个 ASP 文件 ("demo\_test\_post.asp") 类似这样：

<%

dim fname,city

fname=Request.Form("name")

city=Request.Form("city")

Response.Write("Dear " & fname & ". ")

Response.Write("Hope you live well in " & city & ".")

%>