**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。

## 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。

<script>

$.noConflict();

jQuery(document).ready(function(){

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

jQuery("p").text("jQuery 仍在运行！");

});

});

</script>

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

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

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。

可选的 callback 参数是隐藏或显示完成后所执行的函数名称。

下面的例子演示了带有 speed 参数的 hide() 方法：

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

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

});

## jQuery toggle()

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

显示被隐藏的元素，并隐藏已显示的元素：

<script type="text/javascript">

$(document).ready(function(){

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

$("p").toggle(1100);

});

});

</script>

## jQuery Fading 方法

通过 jQuery，您可以实现元素的淡入淡出效果。

jQuery 拥有下面四种 fade 方法：

* jQuery fadeIn() 用于淡入已隐藏的元素。
* jQuery fadeOut() 方法用于淡出可见元素。
* jQuery fadeToggle() 方法可以在 fadeIn() 与 fadeOut() 方法之间进行切换。
* jQuery fadeTo() 方法允许渐变为给定的不透明度（值介于 0 与 1 之间）。

所谓的淡入淡出就是透明度由1至0的变化吧

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

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。

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）。

可选的 callback 参数是该函数完成后所执行的函数名称。

fade其余三种方法的参数就是少个opacity

**jQuery 滑动方法**

通过 jQuery，您可以在元素上创建滑动效果。

jQuery 拥有以下滑动方法：

* jQuery slideDown() 方法用于向下滑动元素。
* jQuery slideUp() 方法用于向上滑动元素
* jQuery slideToggle() 方法可以在 slideDown() 与 slideUp() 方法之间进行切换。

## *jQuery 动画 - animate() 方法*

jQuery animate() 方法用于创建自定义动画。

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

必需的 params 参数定义形成动画的 CSS 属性。

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。

可选的 callback 参数是动画完成后所执行的函数名称。

Camel标记法，类似于“骆驼拼写法”（CamelCase），词语中的第二个单词首字母大写，而正规的CamelCase所有单词都要首字母大写。

**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。

## jQuery 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'

});

});

## jQuery 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。

如果您希望在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");

});

能否同时运行几个动画？

Query 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);

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。

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。

## jQuery 方法链接

直到现在，我们都是一次写一条 jQuery 语句（一条接着另一条）。

不过，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。

**提示：**这样的话，浏览器就不必多次查找相同的元素。

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。

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);

**获得内容 - text()、html() 以及 val()**

三个简单实用的用于 DOM 操作的 jQuery 方法：

* text() - 设置或返回所选元素的文本内容
* html() - 设置或返回所选元素的内容（包括 HTML 标记）
* val() - 设置或返回表单字段的值

## text()、html() 以及 val() 的回调函数

|  |  |
| --- | --- |
| *function(index,oldcontent)* | 必需。规定返回被选元素的新文本内容的函数。   * index - 可选。接受选择器的 index 位置。 * html - 可选。接受选择器的当前内容。 |

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。

## 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：

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

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

});

## 设置属性 - attr()

jQuery attr() 方法也用于设置/改变单个属性值。

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"

});

});

**通过 jQuery，可以很容易地添加新元素/内容。**

## 添加新的 HTML 内容

* append() - 在被选元素的结尾插入内容
* prepend() - 在被选元素的开头插入内容
* after() - 在被选元素之后插入内容
* before() - 在被选元素之前插入内容

append(),prepend():是在被选的这个元素里操作

after(),before()：是在被选元素外面

两个的区别在于一个在里，一个在外操作，通过观察源代码更易察觉两者的区别

**通过 jQuery，可以很容易地删除已有的 HTML 元素。**

jQuery remove() 方法删除被选元素及其子元素。

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

jQuery empty() 方法删除被选元素的子元素。

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

## 过滤被删除的元素

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。

下面的例子删除所有 <p> 元素中class="italic" 的p元素：

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

# jQuery 操作 CSS

jQuery 拥有若干进行 CSS 操作的方法。我们将学习下面这些：

* addClass() - 向被选元素添加一个或多个类，即添加 class 属性
* removeClass() - 从被选元素删除一个或多个类
* toggleClass() - 对被选元素进行添加/删除类的切换操作
* css() - 设置或返回样式属性

## jQuery addClass() 方法

$("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() 方法

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() 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() 方法

css() 方法设置或返回被选元素的一个或多个样式属性。

下面的例子将返回首个匹配元素的 background-color 值：

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

下面的例子将为所有匹配元素设置 background-color 值：

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

## 设置多个 CSS 属性

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 尺寸 方法

jQuery 提供多个处理尺寸的重要方法：

* width()
* height()
* innerWidth()
* innerHeight()
* outerWidth()
* outerHeight()

# jQuery 遍历 - 祖先

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();

});

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();

});

下面的例子返回所有 <span> 元素的所有特定的<ul> 元素祖先

$(document).ready(function(){

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

});

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() 方法

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();

});

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");

});

find() 方法返回被选元素的所有后代元素中，想要选择的特定元素

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("\*");

});

## 在 DOM 树中水平遍历

有许多有用的方法让我们在 DOM 树进行水平遍历：

* siblings()
* next()
* nextAll()
* nextUntil()
* prev()
* prevAll()
* prevUntil()

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");

})

next() 方法返回被选元素的下一个同胞元素。

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();

});

nextAll() 方法返回被选元素之后的所有跟随的同胞元素。

下面的例子返回 <h2> 后的所有同胞元素：

$(document).ready(function(){

$("h2").nextAll();

});

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。

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 prev(), prevAll() & prevUntil() 方法

prev(), prevAll() 以及 prevUntil() 方法的工作方式与上面的方法类似，只不过方向相反而已：它们返回的是前面的同胞元素（在 DOM 树中沿着同胞元素向后遍历，而不是向前）。

# jQuery 遍历 - 过滤

## 缩写搜索元素的范围

三个最基本的过滤方法是：first(), last() 和 eq()，它们允许您基于其在一组元素中的位置来选择一个特定的元素。

其他过滤方法，比如 filter() 和 not() 允许您选取匹配或不匹配某项指定标准的元素。

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();

});

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();

});

eq() 方法返回被选元素中带有指定索引号的元素比起first和last我更喜欢这个

索引号从 0 开始，-1结束，下面的例子选取第二个 <p> 元素（索引号 1）：

$(document).ready(function(){

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

});

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");

});

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 遍历 函数

## each() 方法规定为每个匹配元素规定运行的函数。

提示：返回 false 可用于及早停止循环。语法

$(*selector*).each(*function(index,element)*)

|  |  |
| --- | --- |
| **参数** | **描述** |
| *function(index,element)* | 必需。为每个匹配元素规定运行的函数。   * *index* - 选择器的 index 位置 * *element* - 当前的元素（也可使用 "this" 选择器） |

## 实例

输出每个 li 元素的文本：

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

$("li").each(function(){

alert($(this).text())

});

});