jQuery 语法是通过选取 HTML 元素，并对选取的元素执行某些操作。

基础语法： **$(*selector*).*action*()**

* 美元符号定义 jQuery
* 选择符（selector）"查询"和"查找" HTML 元素
* jQuery 的 action() 执行对元素的操作

实例:

* $(this).hide() - 隐藏当前元素
* $("p").hide() - 隐藏所有 <p> 元素
* $("p.test").hide() - 隐藏所有 class="test" 的 <p> 元素
* $("#test").hide() - 隐藏所有 id="test" 的元素

为了防止文档在完全加载（就绪）之前运行 jQuery 代码, 所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){  
   *// jQuery methods go here...*  
});

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：

* 试图隐藏一个不存在的元素
* 获得未完全加载的图像的大小

**提示：**简洁写法（与以上写法效果相同）:

$(function(){  
  
   *// jQuery methods go here...*  
  
});

jQuery 选择器允许您对 HTML 元素组或单个元素进行操作。

jQuery 选择器基于元素的 id、类、类型、属性、属性值等"查找"（或选择）HTML 元素。 它基于已经存在的 [CSS 选择器](http://www.runoob.com/cssref/css-selectors.html)，除此之外，它还有一些自定义的选择器。

jQuery 中所有选择器都以美元符号开头：$()。

jQuery 元素选择器基于元素名选取元素。

在页面中选取所有 <p> 元素:

$("p")

jQuery #id 选择器通过 HTML 元素的 id 属性选取指定的元素。

页面中元素的 id 应该是唯一的，所以您要在页面中选取唯一的元素需要通过 #id 选择器。

通过 id 选取元素语法如下：

$("#test")

jQuery 类选择器可以通过指定的 class 查找元素。

语法如下：

$(".test")

更多

|  |  |  |
| --- | --- | --- |
| $("\*") | 选取所有元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_all2) |
| $(this) | 选取当前 HTML 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_this) |
| $("p.intro") | 选取 class 为 intro 的 <p> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_pclass) |
| $("p:first") | 选取第一个 <p> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_pfirst) |
| $("ul li:first") | 选取第一个 <ul> 元素的第一个 <li> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_ullifirst) |
| $("ul li:first-child") | 选取每个 <ul> 元素的第一个 <li> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_ullifirstchild) |
| $("[href]") | 选取带有 href 属性的元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_hrefattr) |
| $("a[target='\_blank']") | 选取所有 target 属性值等于 "\_blank" 的 <a> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_hrefattrblank) |
| $("a[target!='\_blank']") | 选取所有 target 属性值不等于 "\_blank" 的 <a> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_hrefattrnotblank) |
| $(":button") | 选取所有 type="button" 的 <input> 元素 和 <button> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_button2) |
| $("tr:even") | 选取偶数位置的 <tr> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_even) |
| $("tr:odd") | 选取奇数位置的 <tr> 元素 | [**在线实例**](http://www.runoob.com/try/try.php?filename=tryjquery_sel_odd) |

**独立文件中使用 jQuery 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）：

**实例**

<head>  
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">  
</script>  
<script src="my\_jquery\_functions.js"></script>  
</head>

在事件中经常使用术语"触发"（或"激发"）例如： "当您按下按键时触发 keypress 事件"。

**jQuery 事件方法语法**

在 jQuery 中，大多数 DOM 事件都有一个等效的 jQuery 方法。

页面中指定一个点击事件：

$("p").click();

下一步是定义什么时间触发事件。您可以通过一个事件函数实现：

$("p").click(function(){  
  // action goes here!!  
});

常见 DOM 事件：

|  |  |  |  |
| --- | --- | --- | --- |
| **鼠标事件** | **键盘事件** | **表单事件** | **文档/窗口事件** |
| click | keypress | submit | load |
| Dblclick 双击 | keydown | change | resize |
| Mouseenter指针穿过元素时 | keyup | focus | scroll |
| Mouseleave指针呢离开元素食 |  | blur | unload |

**mousedown()**

当鼠标指针移动到元素上方，并按下鼠标按键时，会发生 mousedown 事件。

**mouseup()**

当在元素上松开鼠标按钮时，会发生 mouseup 事件。

**hover()**

hover()方法用于模拟光标悬停事件。

当鼠标移动到元素上时，会触发指定的第一个函数(mouseenter);当鼠标移出这个元素时，会触发指定的第二个函数(mouseleave)。

$("#p1").hover(function(){  
  alert("You entered p1!");  
  },  
  function(){  
  alert("Bye! You now leave p1!");  
});

**focus()**

当元素获得焦点时，发生 focus 事件。

当通过鼠标点击选中元素或通过 tab 键定位到元素时，该元素就会获得焦点。

focus() 方法触发 focus 事件，或规定当发生 focus 事件时运行的函数：

$("input").focus(function(){  
  $(this).css("background-color","#cccccc");  
});

**blur()**

当元素失去焦点时，发生 blur 事件。

blur() 方法触发 blur 事件，或规定当发生 blur 事件时运行的函数：

$("input").blur(function(){  
  $(this).css("background-color","#ffffff");  
});

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

**语法:**

$(*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);  
});

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：

语法:

$(selector).toggle(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 toggle() 方法完成后所执行的函数名称。

<body>

<button>隐藏/显示</button>

<p>这是一个文本段落。</p>

<p>这是另外一个文本段落。</p>

</body>

<script>

$(document).ready(function(){

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

$("p").toggle();

});

});

</script>

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：

**fadeIn() 用于淡入已隐藏的元素。**

**语法:**

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

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 fading 完成后所执行的函数名称。

$("button").click(function(){  
  $("#div1").fadeIn();  
  $("#div2").fadeIn("slow");  
  $("#div3").fadeIn(3000);  
});

**fadeOut() 方法用于淡出可见元素。**

**语法:**

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

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 fading 完成后所执行的函数名称。

$("button").click(function(){  
  $("#div1").fadeOut();  
  $("#div2").fadeOut("slow");  
  $("#div3").fadeOut(3000);  
});

**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。

**语法:**

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

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 fading 完成后所执行的函数名称。

$("button").click(function(){  
  $("#div1").fadeToggle();  
  $("#div2").fadeToggle("slow");  
  $("#div3").fadeToggle(3000);  
});

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

**语法:**

$(*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。

$("button").click(function(){  
  $("#div1").fadeTo("slow",0.15);  
  $("#div2").fadeTo("slow",0.4);  
  $("#div3").fadeTo("slow",0.7);  
});

jQuery 拥有以下滑动方法：

**slideDown() 方法用于向下滑动元素。**

**语法:**

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

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 参数是滑动完成后所执行的函数名称。

$("#flip").click(function(){  
  $("#panel").slideDown();  
});

**slideUp() 方法用于向上滑动元素。**

**语法:**

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

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 参数是滑动完成后所执行的函数名称。

$("#flip").click(function(){  
  $("#panel").slideUp();  
});

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

如果元素向下滑动，则 slideToggle() 可向上滑动它们。

如果元素向上滑动，则 slideToggle() 可向下滑动它们。

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

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 参数是滑动完成后所执行的函数名称。

$("#flip").click(function(){  
  $("#panel").slideToggle();  
});

## jQuery 动画

**animate() 方法用于创建自定义动画。**

**语法：**

$(*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。

下面的例子演示 animate() 方法的简单应用。它把 <div> 元素往右边移动了 250 像素：

**实例**

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

**请注意，生成动画的过程中可同时使用多个属性：**

**实例**

$("button").click(function(){  
  $("div").animate({  
    left:'250px',  
    opacity:'0.5',  
    height:'150px',  
    width:'150px'  
  });  
});

**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](http://jquery.com/download/) 下载 [Color Animations](http://plugins.jquery.com/project/color) 插件。

**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 -=：**

**实例**

$("button").click(function(){  
  $("div").animate({  
    left:'250px',  
    height:'+=150px',  
    width:'+=150px'  
  });  
});

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

$("button").click(function(){  
  $("div").animate({  
    height:'toggle'  
  });  
});

**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**

$("button").click(function(){  
  var div=$("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");  
});

下面的例子把 <div> 元素往右边移动了 100 像素，然后增加文本的字号：

实例 2

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

var div=$("div");

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

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

});

**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。

**语法:**

$(*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。

下面的例子演示 stop() 方法，不带参数：

**实例**

$("#stop").click(function(){  
  $("#panel").stop();  
});

停止滑动

<body>

<button id="stop">Stop sliding</button>

<div id="flip">Click to slide down panel</div>

<div id="panel">Hello world!</div>

</body>

<script>

$(document).ready(function(){

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

$("#panel").slideDown(5000);

});

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

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

});

});

</script>

<style type="text/css">

#panel,#flip

{

padding:5px;

text-align:center;

background-color:#e5eecc;

border:solid 1px #c3c3c3;

}

#panel

{

padding:50px;

display:none;

}

</style>

带参数：

<body>

<button id="start">Start</button>

<button id="stop">Stop</button>

<button id="stop2">Stop all</button>

<button id="stop3">Stop but finish</button>

<p>The "Start" button starts the animation.</p>

<p>The "Stop" button stops the current active animation, but allows the queued animations to be performed afterwards.</p>

<p>The "Stop all" button stops the current active animation and clears the

animation queue; so all animations on the element is stopped.</p>

<p>The "Stop but finish" rushes through the current active animation, then it stops.</p>

<div style="background:#98bf21;height:100px;width:200px;position:absolute;">HELLO</div>

</body>

<script>

$(document).ready(function(){

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

$("div").animate({left:'100px'},5000);

$("div").animate({fontSize:'3em'},5000);

});

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

$("div").stop();

});

$("#stop2").click(function(){

$("div").stop(true);

});

$("#stop3").click(function(){

$("div").stop(true,true);

});

});

</script>

**Callback 函数在当前动画 100% 完成之后执行。**

许多 jQuery 函数涉及动画。这些函数也许会将 *speed* 或 *duration* 作为可选参数。

例子：*$("p").hide("slow")*

*speed* 或 *duration* 参数可以设置许多不同的值，比如 "slow", "fast", "normal" 或毫秒。

以下实例在隐藏效果完全实现后回调函数:

$("button").click(function(){  
  $("p").hide("slow",function(){  
    alert("The paragraph is now hidden");  
  });  
});

以下实例没有回调函数，警告框会在隐藏效果完成前弹出：

$("button").click(function(){  
  $("p").hide(1000);  
  alert("The paragraph is now hidden");  
});

**Chaining 允许我们在一条语句中运行多个 jQuery 方法（在相同的元素上）。**

**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() 和 slideDown() 链接在一起。"p1" 元素首先会变为红色，然后向上滑动，再然后向下滑动：

**实例**

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

如果需要，我们也可以添加多个方法调用。

**提示：**当进行链接时，代码行会变得很差。不过，jQuery 语法不是很严格；您可以按照希望的格式来写，包含换行和缩进。

如下书写也可以很好地运行：

**实例**

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

jQuery 会抛掉多余的空格，并当成一行长代码来执行上面的代码行。

**获得内容 - text()、html() 以及 val()**

text() - 设置或返回所选元素的文本内容

C:\Users\lyh122508\AppData\Roaming\Tencent\Users\1942063705\QQ\WinTemp\RichOle\_(~F4S(BX%$6FTQ2]5}}RJO.png

html() - 设置或返回所选元素的内容（包括 HTML 标记）

C:\Users\lyh122508\AppData\Roaming\Tencent\Users\1942063705\QQ\WinTemp\RichOle\VE9X)MU4XACB$CX}GDLLSHN.png

val() - 设置或返回表单字段的值

C:\Users\lyh122508\AppData\Roaming\Tencent\Users\1942063705\QQ\WinTemp\RichOle\}%7OH10T$$$Z53@@4EID~M9.png

**text()和html()获取**

<body>

<p id="test">This is some <b>bold</b> text in a paragraph.</p>

<button id="btn1">Show Text</button>

<button id="btn2">Show HTML</button>

</body>

<script>

$(document).ready(function(){

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

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

});

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

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

});

});

**val()获取**

<body>

<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>

<button>Show Value</button>

</body>

<script>

$(document).ready(function(){

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

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

});

});

</script>

下面的例子演示如何通过 text()、html() 以及 val() 方法来**设置**内容：把原来全部覆盖掉！

$("#btn1").click(function(){  
  $("#test1").text("Hello world!");  
});  
$("#btn2").click(function(){  
  $("#test2").html("<b>Hello world!</b>");  
});  
$("#btn3").click(function(){  
  $("#test3").val("Dolly Duck");  
});

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()：

<body>

<p id="test1">This is a <b>bold</b> paragraph.</p>

<p id="test2">This is another <b>bold</b> paragraph.</p>

<button id="btn1">Show Old/New Text</button>

<button id="btn2">Show Old/New HTML</button>

</body>

<script>

$(document).ready(function(){

$("#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){

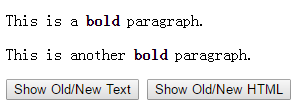
return "Old html: " + origText + " New html: Hello <b>world!</b> (index: " + i + ")";

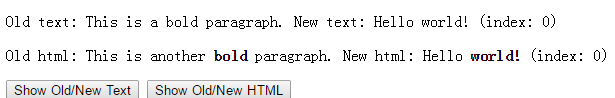
});

});

});

</script>





**attr() 方法用于获取属性值。**

**<body>**

**<p><a href="http://www.w3cschool.cc" id="w3s">W3Cschool.cc</a></p>**

**<button>Show href Value</button>**

**</body>**

<script>

$(document).ready(function(){

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

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

});

});

</script>

C:\Users\lyh122508\AppData\Roaming\Tencent\Users\1942063705\QQ\WinTemp\RichOle\YY$7U9952F`@PGB[[KBI`6N.png

**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：

<p><a href="http://www.w3cschool.cc" id="w3s">W3Cschool.cc</a></p>

<button>Change href Value</button>

$("button").click(function(){  
  $("#w3s").attr("href","http://www.w3cschool.cc/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.w3cschool.cc/jquery",  
    "title" : "W3Schools jQuery Tutorial"  
  });  
});

**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";

});

});

**用于添加新内容的四个 jQuery 方法：**

**append()** 方法在被选元素的结尾插入内容**。**

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

**prepend()** 方法在被选元素的开头插入内容。

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

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>";               // Create element with HTML    
var txt2=$("<p></p>").text("Text.");   // Create with jQuery  
var txt3=document.createElement("p");  // Create with DOM  
txt3.innerHTML="Text.";  
$("p").append(txt1,txt2,txt3);         // Append the new elements   
}

**after()** 方法在被选元素之后插入内容

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

**before()** 方法在被选元素之前插入内容

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

after() 和 before() 方法能够通过参数接收无限数量的新元素

function afterText()  
{  
var txt1="<b>I </b>";                    // Create element with HTML    
var txt2=$("<i></i>").text("love ");     // Create with jQuery  
var txt3=document.createElement("big");  // Create with DOM  
txt3.innerHTML="jQuery!";  
$("img").after(txt1,txt2,txt3);          // Insert new elements after img  
}

**remove()** 方法删除被选元素及其子元素。

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

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");

empty() 方法删除被选元素的子元素。

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

# 获取并设置 CSS 类

.important  
{  
font-weight:bold;  
font-size:xx-large;  
}  
  
.blue  
{  
color:blue;  
}

**addClass()** 方法

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");

});

**removeClass()** 方法

下面的例子演示如何在不同的元素中删除指定的 class 属性：

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

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

});

**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");  
});

**css()** 方法设置或返回被选元素的一个或多个样式属性

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");

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");

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 尺寸 方法**

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）。

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：

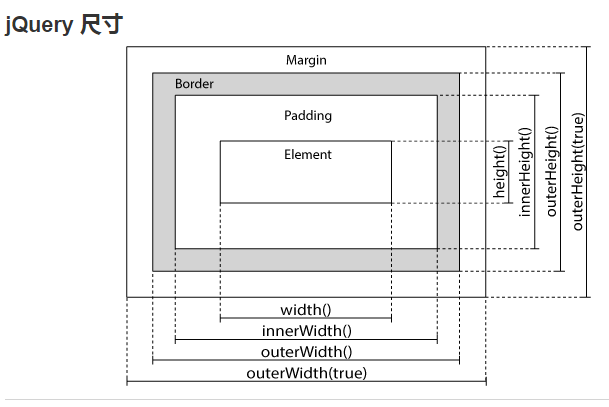
$("button").click(function(){  
  var txt="";  
  txt+="Width: " + $("#div1").width() + "</br>";  
  txt+="Height: " + $("#div1").height();  
  $("#div1").html(txt);  
});

nnerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

outerWidth() 方法返回元素的宽度（包括内边距和边框）。

outerHeight() 方法返回元素的高度（包括内边距和边框）。



**用于向上遍历 DOM 树**

**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>)。

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");

});

**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");  
});

**用于向下遍历 DOM 树的 jQuery 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();

});

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");

});

**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("\*");

});

**在 DOM 树中水平遍历**

**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();

});

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。

**nextAll()** 方法返回被选元素的所有跟随的同胞元素。

**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");

});

**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）。

**过滤**

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。

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);

});

**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 Ajax**

**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。

语法:

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

必需的 URL 参数规定您希望加载的 URL。

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。

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>

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");

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");

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("External content loaded successfully!");

if(statusTxt=="error")

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

});

});

**$.get()** 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

$.get(URL,callback);

必需的 URL 参数规定您希望请求的 URL。

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.php",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.php"）。

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

提示： 这个 PHP 文件 ("demo\_test.php") 类似这样：

<?php

echo "This is some text from an external PHP file.";

?>

**$.post()** 方法通过 HTTP POST 请求从服务器上请求数据。

语法:

$.post(URL,data,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 data 参数规定连同请求发送的数据。

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.html",

{

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.php")。

然后我们连同请求（name 和 city）一起发送数据。

"demo\_test\_post.php" 中的 PHP 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 PHP 文件 ("demo\_test\_post.php") 类似这样：

<?php

$name = isset($\_POST['name']) ? htmlspecialchars($\_POST['name']) : '';

$city = isset($\_POST['city']) ? htmlspecialchars($\_POST['city']) : '';

echo 'Dear ' . $name;

echo 'Hope you live well in ' . $city;

?>

**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。

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 is still working!");

});

});

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 is still working!");

});

});

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 is still working!");

});

});

**JSONP 应用**

如客户想访问 : http://www.runoob.com/try/ajax/jsonp.php?jsonp=callbackFunction。

假设客户期望返回JSON数据：["customername1","customername2"]。

真正返回到客户端的数据显示为: callbackFunction(["customername1","customername2"])。

服务端文件jsonp.php代码为：

<?php

header('Content-type: application/json');

//获取回调函数名

$jsoncallback = htmlspecialchars($\_REQUEST ['jsoncallback']);

//json数据

$json\_data = '["customername1","customername2"]';

//输出jsonp格式的数据

echo $jsoncallback . "(" . $json\_data . ")";

?>

客户端页面完整代码

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>JSONP 实例</title>

</head>

<body>

<div id="divCustomers"></div>

<script type="text/javascript">

function callbackFunction(result, methodName)

{

var html = '<ul>';

for(var i = 0; i < result.length; i++)

{

html += '<li>' + result[i] + '</li>';

}

html += '</ul>';

document.getElementById('divCustomers').innerHTML = html;

}

</script>

<script type="text/javascript" src="http://www.runoob.com/try/ajax/jsonp.php?jsoncallback=callbackFunction"></script>

</body>

</html>

以上代码可以使用 jQuery 代码实例：

<!DOCTYPE html>

<html>

<head>

<title>JSONP 实例</title>

<script src="http://apps.bdimg.com/libs/jquery/1.8.3/jquery.js"></script>

</head>

<body>

<div id="divCustomers"></div>

<script>

$.getJSON("http://www.runoob.com/try/ajax/jsonp.php?jsoncallback=?", function(data) {

var html = '<ul>';

for(var i = 0; i < data.length; i++)

{

html += '<li>' + data[i] + '</li>';

}

html += '</ul>';

$('#divCustomers').html(html);

});

</script>

</body>

</html>