[第二章基础选择器 3](#_Toc410935890)

[2-1#id 选择器(使用身份证号来找人) 3](#_Toc410935891)

[2-2element 选择器(寻找铅笔) 3](#_Toc410935892)

[2-3.class 选择器(寻找红色铅笔) 4](#_Toc410935893)

[2-4\* 选择器(取走全部铅笔) 4](#_Toc410935894)

[2-5sele1,sele2,seleN选择器 5](#_Toc410935895)

[2-6ance desc选择器 6](#_Toc410935896)

[2-7parent > child选择器 6](#_Toc410935897)

[2-8prev + next选择器 6](#_Toc410935898)

[第3章 jQuery过滤性选择器 7](#_Toc410935899)

[3-1:first过滤选择器 7](#_Toc410935900)

[3-2:eq(index)过滤选择器 7](#_Toc410935901)

[3-3:contains(text)过滤选择器 8](#_Toc410935902)

[3-4:has(selector)过滤选择器 9](#_Toc410935903)

[3-5:hidden过滤选择器 10](#_Toc410935904)

[3-6:visible过滤选择器 10](#_Toc410935905)

[3-7[attribute=value]属性选择器 10](#_Toc410935906)

[3-8[attribute!=value]属性选择器 11](#_Toc410935907)

[3-9[attribute\*=value]属性选择器 12](#_Toc410935908)

[3-10 :first-child子元素过滤选择器 12](#_Toc410935909)

[3-11 :last-child子元素过滤选择器 13](#_Toc410935910)

[第4章jQuery表单选择器 13](#_Toc410935911)

[4-1:input表单选择器 13](#_Toc410935912)

[4-2:text表单文本选择器（空格） 14](#_Toc410935913)

[4-3:password表单密码选择器 14](#_Toc410935914)

[4-4:radio单选按钮选择器 15](#_Toc410935915)

[4-5:checkbox复选框选择器 15](#_Toc410935916)

[4-6:submit提交按钮选择器 16](#_Toc410935917)

[4-7:image图像域选择器 16](#_Toc410935918)

[4-8:button表单按钮选择器 17](#_Toc410935919)

[4-9:checked选中状态选择器checked选择器与元素之间没有空格 17](#_Toc410935920)

[4-10:selected选中状态选择器 17](#_Toc410935921)

[第5章jQuery操作DOM元素 18](#_Toc410935922)

[5-1使用attr()方法控制元素的属性 18](#_Toc410935923)

[5-2html()和text()方法操作元素的内容 19](#_Toc410935924)

[5-3addClass()和css()操作元素中的样式 20](#_Toc410935925)

[5-4removeAttr(name)和removeClass(class)移除元素的属性和样式 20](#_Toc410935926)

[5-5使用append()方法向元素内追加内容 20](#_Toc410935927)

[5-6使用appendTo()方法向被选元素内插入内容 21](#_Toc410935928)

[5-7使用before()和after()在元素前后插入内容 22](#_Toc410935929)

[5-8使用clone()方法复制元素 23](#_Toc410935930)

[5-9replaceWith()和replaceAll()方法都可以用于替换元素或元素中的内容 24](#_Toc410935931)

[5-10使用wrap()和wrapInner()方法包裹元素和内容 24](#_Toc410935932)

[5-11使用each()方法遍历元素 25](#_Toc410935933)

[5-12使用remove()和empty()方法删除元素 27](#_Toc410935934)

[第6章jQuery事件与应用 27](#_Toc410935935)

[6-1页面加载时触发ready()事件 27](#_Toc410935936)

[6-2使用bind()方法绑定元素的事件 28](#_Toc410935937)

[6-3使用hover()方法切换事件 29](#_Toc410935938)

[6-4使用toggle()方法绑定多个函数 30](#_Toc410935939)

[6-5使用unbind()方法移除元素绑定的事件 31](#_Toc410935940)

[6-6使用one()方法绑定元素的一次性事件 32](#_Toc410935941)

[6-7调用trigger()方法手动触发指定的事件 34](#_Toc410935942)

[6-8文本框的focus和blur事件 34](#_Toc410935943)

[6-9下拉列表框的change事件 35](#_Toc410935944)

[6-10调用live()方法绑定元素的事件 37](#_Toc410935945)

[第7章jQuery动画特效 37](#_Toc410935946)

[7-1调用show()和hide()方法显示和隐藏元素 37](#_Toc410935947)

[7-2动画效果的show()和hide()方法 38](#_Toc410935948)

[7-3调用toggle()方法实现动画切换效果 39](#_Toc410935949)

[7-4使用slideUp()和slideDown()方法的滑动效果 39](#_Toc410935950)

[7-5使用slideToggle()方法实现图片“变脸”效果 41](#_Toc410935951)

[7-6使用fadeIn()与fadeOut()方法实现淡入淡出效果 42](#_Toc410935952)

[7-7使用fadeTo()方法设置淡入淡出效果的不透明度 43](#_Toc410935953)

[7-8调用animate()方法制作简单的动画效果 44](#_Toc410935954)

[7-9调用animate()方法制作移动位置的动画 45](#_Toc410935955)

[7-10调用stop()方法停止当前所有动画效果 46](#_Toc410935956)

[7-11调用delay()方法延时执行动画效果 47](#_Toc410935957)

[第8章jQuery实现Ajax应用 48](#_Toc410935958)

[8-1使用load()方法异步请求数据 48](#_Toc410935959)

[8-2使用getJSON()方法异步加载JSON格式数据 49](#_Toc410935960)

[8-3使用getScript()方法异步加载并执行js文件 52](#_Toc410935961)

[8-4使用get()方法以GET方式从服务器获取数据 53](#_Toc410935962)

[8-5使用post()方法以POST方式从服务器发送数据 55](#_Toc410935963)

[8-6使用serialize()方法序列化表单元素值 57](#_Toc410935964)

[8-7使用ajax()方法加载服务器数据 60](#_Toc410935965)

[第9章 jQuery 常用插件 60](#_Toc410935966)

[9-1表单验证插件——validate 60](#_Toc410935967)

[9-2表单插件——form 61](#_Toc410935968)

[9-3图片灯箱插件——lightBox 62](#_Toc410935969)

[9-4图片放大镜插件——jqzoom 64](#_Toc410935970)

[9-5cookie插件——cookie 66](#_Toc410935971)

[9-6搜索插件——autocomplete 68](#_Toc410935972)

[9-7自定义对象级插件——lifocuscolor插件 70](#_Toc410935973)

[第10章 jQuery UI型插件 72](#_Toc410935974)

[10-1拖曳插件——draggable 72](#_Toc410935975)

[10-2放置插件——droppable 73](#_Toc410935976)

[10-3拖曳排序插件——sortable 76](#_Toc410935977)

[10-4面板折叠插件——accordion 77](#_Toc410935978)

[10-5选项卡插件——tabs 79](#_Toc410935979)

[10-6对话框插件——dialog 80](#_Toc410935980)

[10-7菜单工具插件——menu 82](#_Toc410935981)

[10-8微调按钮插件——spinner 84](#_Toc410935982)

[10-9工具提示插件——tooltip 86](#_Toc410935983)

[第11章 jQuery 工具类函数 89](#_Toc410935984)

$(" div ").html("Hello iMooc!");

$(" p").html("Hello iMooc!");

# 第二章基础选择器

## 2-1#id 选择器(使用身份证号来找人)

过一个id号去查找一个元素，就可以使用如下格式的选择器：

$("#my\_id")

在id号为“default”的元素中显示id号为“divtest”元素的内容。$("#default").html( $("#devtest").html());

## 2-2element 选择器(寻找铅笔)

，jQuery中可以根据元素名查找元素，格式如下：

$(“element”)

其中element就是元素的名称，

如下图所示：在页面中，根据元素名找到了一个元素，并将它显示的字体加粗。

[](http://img.mukewang.com/5286ddff0001239802840097.jpg)

通过元素名称查找按钮，并通过使用attr()方法将它设置为不可用，其中attr()方法的功能是设置或获取元素的某项属性值。

## 2-3.class 选择器(寻找红色铅笔)

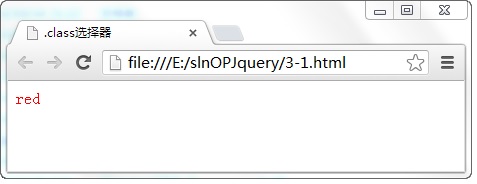
$(“.class”)

其中.class参数表示元素的CSS类别(类选择器)名称。

例如，在页面中，通过class选择器获取某个元素，并显示该元素的class类别名称，如下图所示：

[](http://img.mukewang.com/5286e9f00001a50d03680173.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/5286ea780001d74604770184.jpg)

从图中可以看出，通过class选择器的方式获取元素，并调用元素的attr()方法获取元素的类别名称，并将名称显示在页面中。

## 2-4\* 选择器(取走全部铅笔)

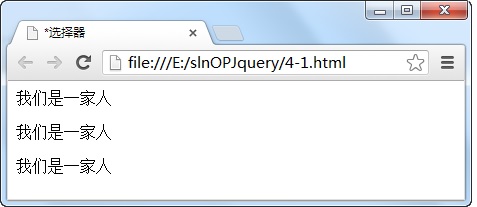
选择器中的参数就一个“\*”，既没有“#”号，也没有“.”号。 由于该选择器的特殊性，它常与其他元素组合使用，表示获取其他元素中的全部子元素。

如下图所示： 使用\*选择器，获取div中的所有子元素并设置三个子元素显示相同的内容。

[](http://img.mukewang.com/5286f46a0001f6dc02890170.jpg)

在浏览器中显示的效果：

由于三个元素都包含在<div>元素中，因此，它们都是<div>元素的子元素，那么，就可以使用$(“div \*”)的方式获取<div>元素中的这三个子元素，并使用html()方法来设置它们显示的内容。

[](http://img.mukewang.com/5286f4a00001331904770207.jpg)

## 2-5sele1,sele2,seleN选择器

有时需要精确的选择任意**多个指定的元素**，类似于从文具盒中挑选出多根自已喜欢的笔，就需要调用sele1,sele2,seleN选择器，它的调用格式如下：

$(“sele1,sele2,seleN”)

其中参数sele1、sele2到seleN为有效选择器，每个选择器之间用“，”号隔开，它们可以是之前提及的各种类型选择器，如$(“#id”)、$(“.class”)、$(“selector”)选择器等。

例如，通过选择器获取其中的任意两个元素，并将它们显示的内容设为相同，如图所示：

[](http://img.mukewang.com/528705b000012a1102890134.jpg)

<script type="text/javascript">

$(".red,.green，.blue").html("hi,我们的样子很美哦!");

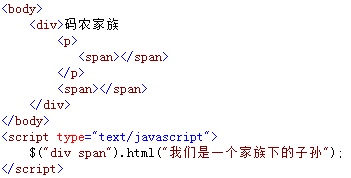
</script>

## 2-6ance desc选择器

ance参数（ancestor祖先的简写）表示父元素；desc参数（descendant后代的简写）表示后代元素，。

比如家族姓氏“div”，家族几代人里，都有名字里带“span”的，就可以用这个ancedesc选择器把这几个人给定位出来。

例如，使用层次选择器，获取<div>元素中的全部<span>元素，并设置它们显示的内容，在如下图所示：

[](http://img.mukewang.com/528b4ecd00013c1503510178.jpg)

## 2-7parent > child选择器

与上一节介绍的ancedesc选择器相比，parent > child选择器的范围要小些，它所选择的目标是子集元素，相当于一个家庭中的子辈们，但不包括孙辈，它的调用格式如下：

$(“parent> child”)

child参数获取的元素都是parent选择器的子元素，它们之间通过“>”符号来表示一种层次关系。

$("div>label").css("border", "solid 5px red");

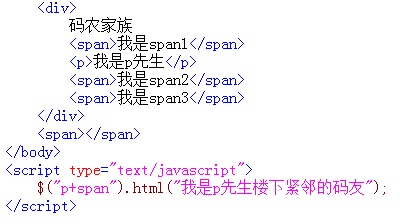
## 2-8prev + next选择器

通过prev + next选择器就可以查找与“prev”元素紧邻的下一个“next”元素，格式如下：

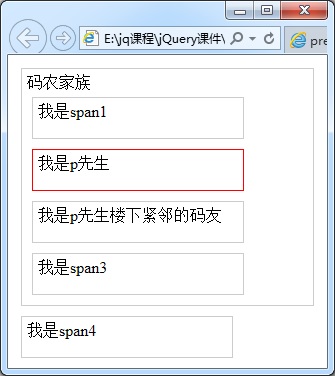
$(“prev + next”)

其中参数prev为任何有效的选择器，参数“next”为另外一个有效选择器，它们之间的“+”表示一种上下的层次关系，也就是说，“prev”元素最紧邻的下一个元素由“next”选择器返回的并且只返回唯的一个元素。

例如，使用prev + next选择器，获取<p>元素最近邻的下一个元素，如下图所示：

[](http://img.mukewang.com/5292e68c000191c204000224.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/5292ebe60001dc2b03350376.jpg)

# **第3章 jQuery过滤性选择器**

## 3-1:first过滤选择器

<script type="text/javascript">

$("li:first").css("background-color", "red");

</script>

## 3-2:eq(index)过滤选择器

如果想从一组标签元素数组中，灵活选择任意的一个标签元素，我们可以使用

:eq(index)

其中参数index表示索引号（即：一个整数），它从0开始，如果index的值为3，表示选择的是第4个元素。例如：

[](http://img.mukewang.com/529c00d40001669503040210.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/529c00f90001098804890294.jpg)

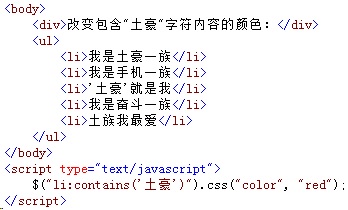
从图中可以看出，通过调用$("li:eq(3)")过滤选择器代码，获取了第4个<li>元素，并使用css()方法设置了该元素在页面中显示的文字样式。

$("li:eq(2)").css("background-color", "#60F");

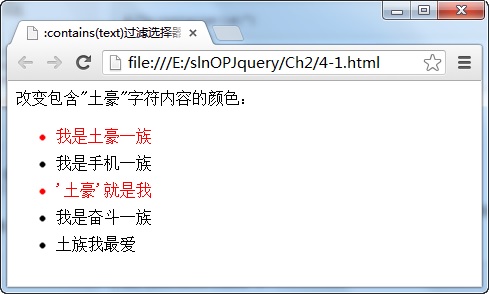
## 3-3:contains(text)过滤选择器

:contains(text)选择器会更加方便，它的功能是选择**包含**指定字符串的全部元素，它通常与其他元素结合使用，获取包含“text”字符串内容的全部元素对象。其中参数text表示页面中的文字。

例如:

[](http://img.mukewang.com/529c067b00014cad03480210.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/529c069b0001879204890294.jpg)

**注意：li:contains('土豪') 土豪为什么必须加单引号呢？因为它是一个字符串，而不是一个变量，所以不加单或双引号的话是会报错的。**

**$("li:contains('jQuery')").css("background", "green");**

## 3-4:has(selector)过滤选择器

除了在上一小节介绍的使用包含的字符串内容过滤元素之外，还可以使用包含的元素名称来过滤，:has(selector)过滤选择器的功能是获取选择器中包含指定元素名称的全部元素，其中selector参数就是包含的元素名称，是被包含元素。

例如：获取指定包含某个元素名的全部<li>元素，并改变它们显示文字的颜色，如下图所示：

[](http://img.mukewang.com/529c1a8300017dd003040210.jpg)

**$("li:has('label')").css("background-color", "blue");**

## 3-5:hidden过滤选择器

:hidden过滤选择器的功能是获取全部不可见的元素，这些不可见的元素中包括type属性值为hidden的元素。

例如，调用:hidden选择器获取不可见的<p>元素，并将该元素的内容显示在<div>元素中

**var $strHTML = $("#hidstr:hidden").val();**

**$("div").html($strHTML);**

**var $strHTML = $("input:hidden").val();**

## 3-6:visible过滤选择器

与上一节的:hidden过滤选择器相反，:visible过滤选择器获取的是全部可见的元素，也就是说，只要不将元素的display属性值设置为“none”，那么，都可以通过该选择器获取。

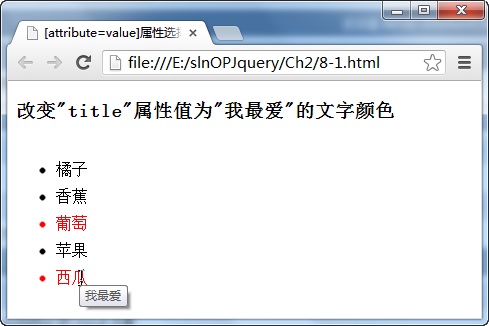
**$("li:visible").css("background-color","blue");**

## 3-7[attribute=value]属性选择器

例如，使用[attribute=value]属性选择器，获取指定属性名和对应值的全部<li>元素，并设置它们显示的文字颜色，如图所示：

[](http://img.mukewang.com/52bcf22e0001409b03620209.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52bcf2560001ad0d04890326.jpg)

从图中可以看出，使用$("li[title='我最爱']")属性选择器代码，获取了2个<li>元素，并调用css()方法设置它们在页面中显示的文字颜色，另外，属性值中的‘’单引号可以不写，由于属性名与属性值是等号，因此，它们之间不是包含关系，而是完全相同。

**$("li[title='蔬菜']").css("background-color", "green");**

## 3-8[attribute!=value]属性选择器

与上一节介绍的[attribute=value]属性选择器正好相反，[attribute!=value]属性选择器的功能是获取不包含属性名，或者与属性名和属性值不相同的全部元素，其中[]是专用于属性选择器的括号符，参数attribute表示属性名称，value参数表示属性值。

**$("li[title!='蔬菜']").css("background-color", "green");**

## 3-9[attribute\*=value]属性选择器

介绍一个功能更为强大的属性选择器[attribute\*=value]，它可以获取属性值中包含指定内容的全部元素，其中[]是专用于属性选择器的括号符，参数attribute表示属性名称，value参数表示对应的属性值。

例如，使用[attribute\*=value]属性选择器，获取属性值中包含某一指定内容的全部<li>元素，并设置它们显示的文字颜色，如下图所示：

$("li[title\*='果']").css("background-color", "green");

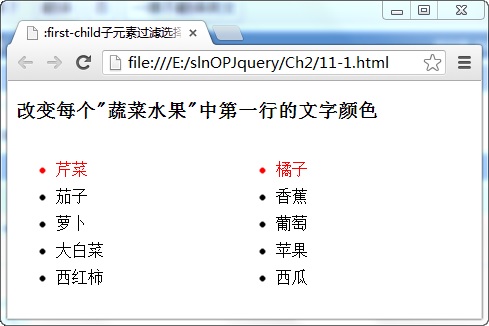
## 3-10 :first-child子元素过滤选择器

通过上面章节的学习，我们知道使用:first过滤选择器可以获取指定父元素中的首个子元素，但该选择器返回的只有一个元素，并不是一个集合，而使用:first-child子元素过滤选择器则可以获取每个父元素中返回的首个子元素，它是一个集合，常用多个集合数据的选择处理。

如下图，如果想把页面中每个ul中的第一个li获取到，并改变其颜色。则可以使用

[](http://img.mukewang.com/529c2a9000014a0e03500322.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/529c2ab600011b5804890326.jpg)

通过$("li:first-child")选择器代码，获取了两个<ul>父元素中的第一个<li>元素，并使用css()方法修改了它们在页面中显示的文字颜色。

**$("li:first-child").css("background-color", "green");**

## 3-11 :last-child子元素过滤选择器

与:first-child子元素过滤选择器功能相反，:last-child子元素过滤选择器的功能是获取每个父元素中返回的最后一个子元素，它也是一个集合，常用多个集合数据的选择处理。

**$("li:last-child").css("background-color", "blue");**

# **第4章jQuery表单选择器 要空格！**

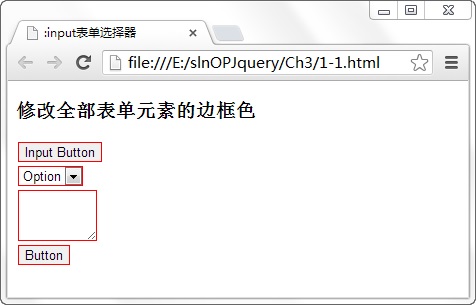
## 4-1:input表单选择器

如何获取表单全部元素？:input表单选择器可以实现，它的功能是返回**全部的表单元素**，不仅包括所有<input>标记的表单元素，而且还包括<textarea>、<select>和<button>标记的表单元素，因此，它选择的表单元素是最广的。

如下图所示，使用:input表单选择器获取表单元素，并向这些元素增加一个CSS样式类别，修改它们在页面中显示的边框颜色。



在浏览器中显示的效果：



可以看出，通过调用$("#frmTest :input")表单选择器代码获取了表单中的全部元素，并使用addClass()方法修改它们在页面中显示的边框颜色。addClass()方法的功能是为元素添加指定的样式类别名称，它的更多使用将会在后续章节中进行详细介绍。

## 4-2:text表单文本选择器（空格）

:text表单文本选择器可以获取表单中全部单行的文本输入框元素，单行的文本输入框就像一个不换行的字条工具，使用非常广泛。

例如，在表单中添加多个元素，使用:text选择器获取单行的文本输入框元素，并修改字的边框颜色，如下图所示：



$("#frmTest :text").addClass("bg\_blue");有空格

## 4-3:password表单密码选择器

如果想要获取密码输入文本框，可以使用:password选择器，它的功能是获取表单中全部的密码输入文本框元素。

例如，在表单中添加多个输入框元素，使用:password获取密码输入文本框元素，并修改它的边框颜色，如下图所示：



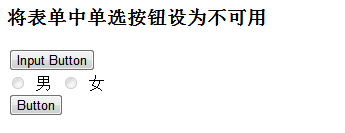
$("form :password").addClass("bg\_red");

$("#frmTest :password").addClass("bg\_red");

## 4-4:radio单选按钮选择器

表单中的单选按钮常用于多项数据中仅选择其一，而使用:radio选择器可轻松获取表单中的**全部单选按钮**元素

$("#frmTest :radio").attr("disabled","true");



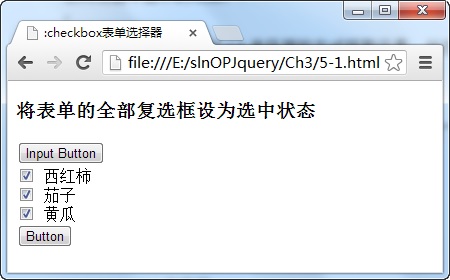
## 4-5:checkbox复选框选择器

表单中的复选框常用于多项数据的选择，使用:checkbox选择器可以快速定位并获取表单中的复选框元素。

例如，在表单中增加多个不同类型的元素，使用:checkbox选择器获取其中的全部复选框元素，并将它们全部设为选中状态，如下图所示：

[](http://img.mukewang.com/52971ac700017fbf03830322.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52971afe0001750304500280.jpg)

$("#frmTest :checkbox").attr("disabled","true");

## 4-6:submit提交按钮选择器

在表单中添加多个不同类型的按钮，使用:submit选择器获取其中的提交按钮，并使用attr()方法修改按钮显示的文本内容，

button>按钮通常也被认为是提交按钮，为更好区分，只要在:submit选择器之前添加“input”符号。

$("#frmTestinput:submit").addClass("bg\_red");

## 4-7:image图像域选择器

当一个<input>元素的“type”属性值设为“image”时，该元素就是一个图像域，使用:image选择器可以快速获取该类全部元素。例如，在表单中添加两种类型的图像元素，使用:image选择器获取其中的一种图像元素，并改变该元素的边框样式，如下图所示：

[](http://img.mukewang.com/52b26da200016f8604270192.jpg)

$("#frmTest :image").addClass("bg\_red");

## 4-8:button表单按钮选择器

表单中包含许多类型的按钮，而使用:button选择器能获取且只能获取“type”属性值为“button”的<input>和<button>这两类普通按钮元素。

$("#frmTest :button").addClass("bg\_blue");

## 4-9:checked选中状态选择器checked选择器与元素之间没有空格

有一些元素存在选中状态，如复选框、单选按钮元素，选中时“checked”属性值为“checked”，调用:checked可以获取处于选中状态的全部元素。

例如，在表单中添加多个复选框和单选按钮，其中有一些元素处于选中状态，使用:checked**获取**并**隐藏**处于选中状态的元素。

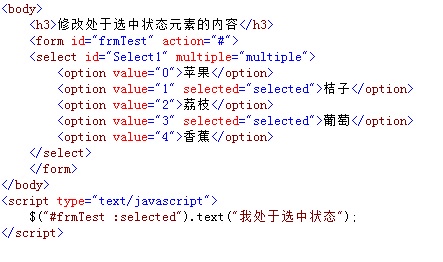
$("#frmTest :checked").attr("disabled", true);空格

只是方便选中已经打勾的项目，然后再加操作，attr（）

## 4-10:selected选中状态选择器

与:checked选择器相比，:selected选择器只能获取<select>下拉列表框中全部处于选中状态的<option>选项元素。

例如，在一个添加多个<option>选项的下拉列表框中，使用:selected选择器修改处于选中状态的内容值，如下图所示：

[](http://img.mukewang.com/52b28d15000198f404430262.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52b28d370001a09004570304.jpg)

var$txtOpt = $("#frmTest:selected").text();

$("#tip").html("选中内容为:" + $txtOpt);

# **第5章jQuery操作DOM元素**

## 5-1使用attr()方法控制元素的属性

attr()方法的作用是设置或者返回元素的属性，其中attr(属性名)格式是获取元素属性名的值，attr(属性名，属性值)格式则是设置元素属性名的值。

<script type="text/javascript">

$("#a1").attr("href", "www.imooc.com");

var $url = $("#a1").attr("href");

$("#tip").html($url);

</script>



## 5-2html()和text()方法操作元素的内容

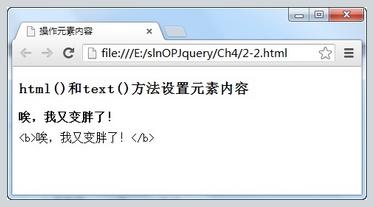
使用html()和text()方法操作元素的内容，当两个方法的参数为空时，表示获取该元素的内容，而如果方法中包含参数，则表示将参数值设置为元素内容。

html()方法可以**获取元素的HTML内容**，因此，原文中的格式代码也被一起获取，而text()方法只是**获取元素中的文本内容**，并不包含HTML格式代码、

var $content = "<b>唉，我又变胖了！</b>";

$("#html").html($content);

$("#text").text($content);



## 5-3addClass()和css()操作元素中的样式

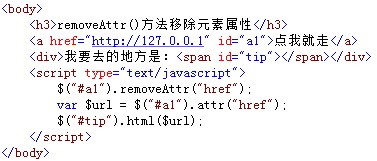


$("#content").css({"background-color":"red","color":"white"});

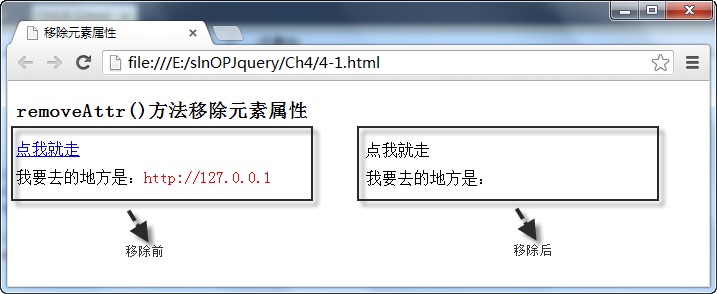
## 5-4removeAttr(name)和removeClass(class)移除元素的属性和样式

前者方法中参数表示移除属性名，后者方法中参数则表示移除的样式名。

例如，使用removeAttr()方法移除<a>元素中的“href”属性，如下图所示：

[](http://img.mukewang.com/52bb960c0001f4c603770161.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52bb962a00017ec807170294.jpg)

从图中可以看出，使用removeAttr()方法移除元素的“href”属性后，再次显示元素的“href”属性值时，则为空值，<a>元素中的文字也丢失可点击的效果。

$("#content").removeClass("blue white");

## 5-5使用append()方法向元素内追加内容

例如，在页面的<body>元素中追加一个具有“id”、“title”属性和显示内容的<div>元素，如下图所示：

[](http://img.mukewang.com/52bb980d0001c85804680122.jpg)

## 5-6使用appendTo()方法向被选元素内插入内容

appendTo()方法也可以向指定的元素内插入内容，它的使用格式是：

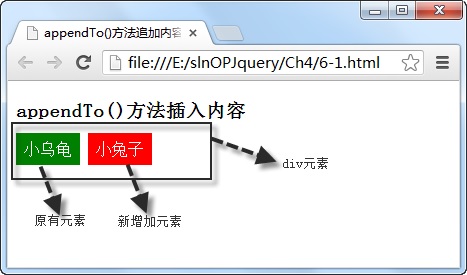
**$(content).appendTo(selector)**

参数content表示需要插入的内容，参数selector表示被选的元素，即把content内容插入selector元素内，默认是在尾部。

例如，使用appendTo()方法，将<div>外的<span>元素插入<div>内，如下图所示：

[](http://img.mukewang.com/52bcd7850001d30303360171.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52bcd7ad0001f82b04670275.jpg)

从图中可以看出，使用appendTo()方法将类别名为“red”的<span>元素插入到<div>元素的尾部，相当于追加的效果。

<script type="text/javascript">

var $html = "<span class='red'>小青蛙</span>"

$($html).appendTo("div");

</script>

## 5-7使用before()和after()在元素前后插入内容

使用before()和after()方法可以在元素的前后插入内容，它们分别表示在整个元素的前面和后面插入指定的元素或内容，调用格式分别为：

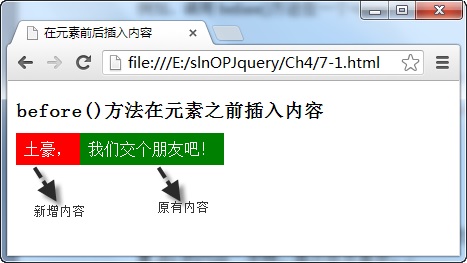
**$(selector).before(content)**和**$(selector).after(content)**

其中参数content表示插入的内容，该内容可以是元素或HTML字符串。

例如，调用before()方法在一个<span>元素插入另一个<span>元素，如下图所示：

[](http://img.mukewang.com/52bcd9cd0001645203820142.jpg)

在浏览器中显示的效果：

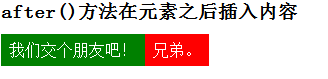
[](http://img.mukewang.com/52bcd9fd0001594d04670263.jpg)

<script type="text/javascript">

var $html = "<span class='red'>兄弟。</span>"

$(".green").after($html);

</script>

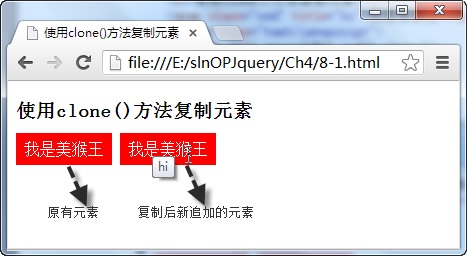


## 5-8使用clone()方法复制元素

例如，使用clone()方法复制页面中的一个<span>元素，并将复制后的元素追加到页面的后面，如下图所示：

[](http://img.mukewang.com/52bb9da80001fede03660123.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52bb9dce0001d15304670256.jpg)

从图中可以看出，使用clone()方法复制元素时，不仅复制了该元素的文本和节点，还将它的“title”属性也一起复制过来了。

$("body").append($(".red").clone());

## 5-9replaceWith()和replaceAll()方法都可以用于替换元素或元素中的内容

，但它们调用时，内容和被替换元素所在的位置不同，分别为如下所示：

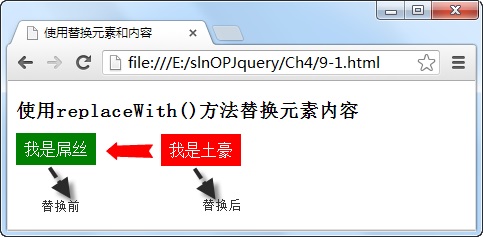
**$(selector).replaceWith(content)**和**$(content).replaceAll(selector)**

参数selector为被替换的元素，content为替换的内容

例如，调用replaceWith()方法将页面中<span>元素替换成一段HTML字符串，如下图所示：

[](http://img.mukewang.com/52bcdc2c0001b69c04750130.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52bcdc4f0001e25f04830237.jpg)

## 5-10使用wrap()和wrapInner()方法包裹元素和内容

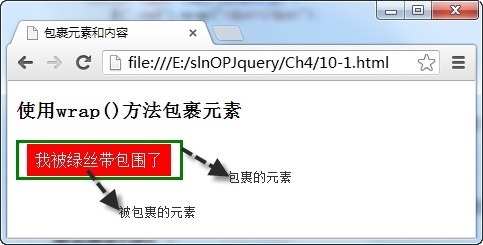
wrap()和wrapInner()方法都可以进行元素的包裹，但前者用于包裹元素本身，后者则用于包裹元素中的内容，它们的调用格式分别为：

**$(selector).wrap(wrapper)**和**$(selector).wrapInner(wrapper)**

例如，调用wrap()方法，将<span>用<div>元素包裹起来，如下图所示：

[](http://img.mukewang.com/52bcdea700016dbf03980120.jpg)

在浏览器中显示的效果：

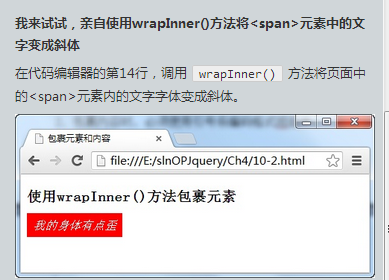
[](http://img.mukewang.com/52bcdf080001301504830245.jpg)

从图中可以看出，红色区域的<span>元素被蓝色边框的<div>元素通过wrap()方法包裹起来。

<script type="text/javascript">

$(".red").wrapInner("<span style='font-style:italic'></span>");

</script>



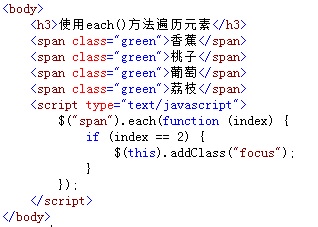
## 5-11使用each()方法遍历元素

使用each()方法可以遍历指定的元素集合，在遍历时，通过回调函数返回遍历元素的序列号，它的调用格式为：

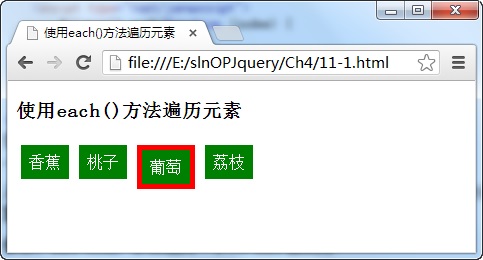
**$(selector).each(function(index))**

参数function为遍历时的回调函数，index为遍历元素的序列号，它从0开始。

例如，遍历页面中的<span>元素，当元素的序列号为2时，添加名为“focus”的样式，如下图所示：

[](http://img.mukewang.com/52bba3720001175203180227.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52bba38e00013f0a04830260.jpg)

<script type="text/javascript">

$("span").each(function (index) {

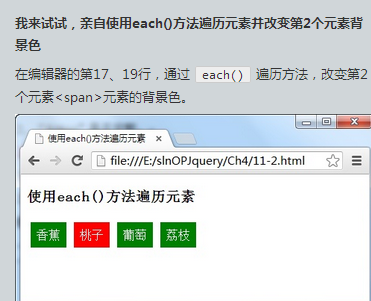
if (index == 1) {

$(this).attr("class", "red");

}

});

</script>



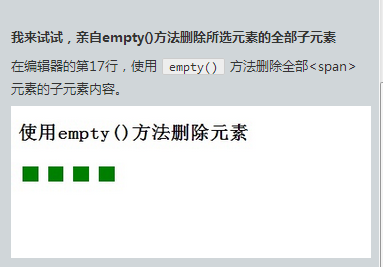
## 5-12使用remove()和empty()方法删除元素

remove()方法删除所选元素本身和子元素，该方法可以通过添加过滤参数指定需要删除的某些元素，而empty()方法则只删除所选元素的子元素。

例如，调用remove()方法删除<span>元素中类别名为“red”的，如下图所示：

[](http://img.mukewang.com/52bceff500018e0702910171.jpg)

**$("span").empty();**

****

**Empty只是清空，框架还在，只剩下框框， 内容都没了 。Remove就是都删了。**

# **第6章jQuery事件与应用**

## 6-1页面加载时触发ready()事件

**$(document).ready(function(){})等价于$(function(){});**

<script type="text/javascript">

document.ready() {

$("#btntest").bind("click", function () {

$("#tip").html("我被点击了！");

});

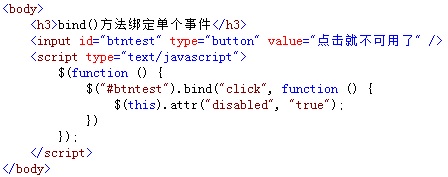
});

</script>

## 6-2使用bind()方法绑定元素的事件

bind()方法绑定元素的事件非常方便，绑定前，需要知道被绑定的元素名，绑定的事件名称。

例如，绑定按钮的单击事件，单击按钮时，该按钮变为不可用。如下图所示：

[](http://img.mukewang.com/52d22aeb0001ee2b04480177.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52d22b0d0001f03204790244.jpg)

可以看出，由于使用bind()方法，绑定了按钮的单击事件，在该事件中将按钮本身的“disabled”属性值设为“true”，表示不可用，当点击时触该事件。

$(function () {

$("#btntest").bind("click mouseout"， function () {

$(this).attr("disabled", "true");}

)

});



## 6-3使用hover()方法切换事件

hover()方法的功能是当鼠标移到所选元素上时，执行方法中的第一个函数，鼠标移出时，执行方法中的第二个函数，实现事件的切实效果，调用格式如下：

**$(selector).hover(over，out);**

**over参数为移到所选元素上触发的函数，out参数为移出元素时触发的函数**。

例如，当鼠标移到<div>元素上时，元素中的字体变成金黄色，如下图所示：

[](http://img.mukewang.com/52d22c9900013dbd03190241.jpg)

<script type="text/javascript">

$(function () {

$("div").hover(

function () {

$(this).addClass("orange");

},

function () {

$(this).removeClass("orange")

})

});

</script>



## 6-4使用toggle()方法绑定多个函数

## 实例

切换 <p> 元素的显示与隐藏状态：

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

$("p").hide();

});

## 定义和用法

toggle() 方法切换元素的可见状态。

如果被选元素可见，则隐藏这些元素，如果被选元素隐藏，则显示这些元素。

### 语法

$(*selector*).toggle(*speed*,*callback*,*switch*)

**toggle()方法可以在元素的click事件中绑定两个或两个以上的函**数，同时，它还可以实现元素的隐藏与显示的切换，绑定多个函数的调用格式如下：

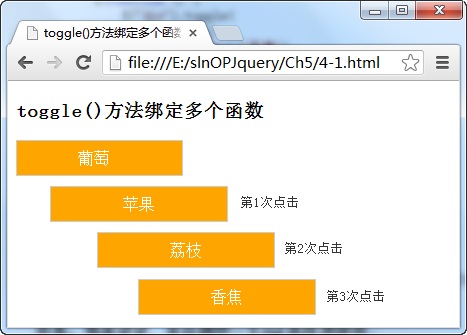
**$(selector).toggle(fun1(),fun2(),funN(),...)**

其中，fun1，fun2就是多个函数的名称

例如，使用toggle()方法，当每次点击<div>元素时，显示不同内容，如下图所示：

[](http://img.mukewang.com/52d22e3a0001c56d02980289.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52d22e570001902a04670335.jpg)

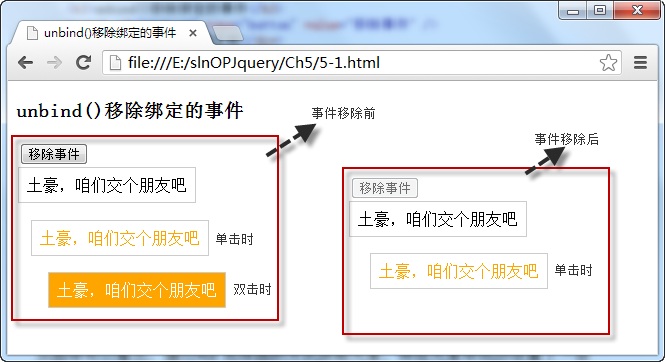
从图中可以看出，**每次点击<div>元素时，都依次执行toggle()方法绑定的函数，当执行到最后一个函数时，再次点击将又返回执行第一个函数。**

## 6-5使用unbind()方法移除元素绑定的事件

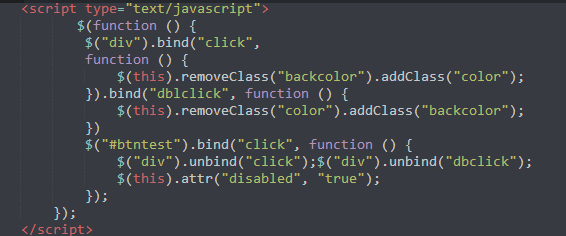
例如，点击按钮时，使用unbind()方法移除<div>元素中已绑定的“dblclick”事件，如下图所示：

[](http://img.mukewang.com/52d22feb0001c9ee04750306.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52d2300c0001ecfe06650362.jpg)

从图中可以看出，当使用unbind()方法移除已绑定的“dblclick”事件时，再次双击<div>元素，样式和文字都没有任何变化，表明移除事件成功。

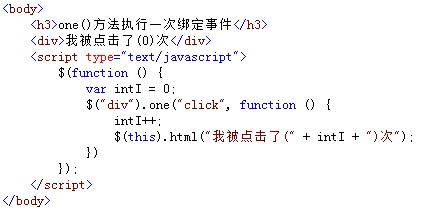
****

**Unbind（）不加参数就是一处所有的绑定事件**

## 6-6使用one()方法绑定元素的一次性事件

one()方法可以绑定元素任何有效的事件，但这种方法绑定的事件只会触发一次例如，

使用one方法绑定<div>元素的单击事件，在事件执行的函数中，累计执行的次数，并将该次数显示在页面中，如下图所示：

[](http://img.mukewang.com/52d231bd000192fb04310219.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52d231dd00011da004890293.jpg)

从图中可以看出，由于使用了one()方法绑定<div>元素的单击事件，因为事件函数只能执行一次，执行完成后，无论如何单击，都不再触发。

**$(function () {**

**varintI = 0;**

**$("div").one("click", function () {**

**intI++;**

**$(this).css("font-size", intI + "px");**

**})**

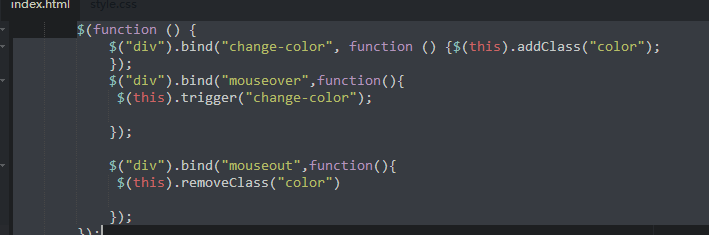
**});**

## 6-7调用trigger()方法手动触发指定的事件

trigger()方法可以直接手动触发元素指定的事件，这些事件可以是元素自带事件，也可以是自定义的事件，总之，该事件必须能执行。

例如，当页面加载时，手动触发文本输入框的“select”事件，使文本框的默认值处于全部被选中的状态，如下图所示：

[](http://img.mukewang.com/52d233a30001df8204290153.jpg)

****

## 6-8文本框的focus和blur事件

focus事件在元素获取焦点时触发，如点击文本框时，触发该事件；而blur事件则在元素丢失焦点时触发，如点击除文本框的任何元素，都会触发该事件。

例如，在触发文本框的“focus”事件时，<div>元素显示提示内容，如下图所示：

[](http://img.mukewang.com/52d235210001b6f203730194.jpg)

**<script type="text/javascript">**

**$(function () {**

**$("input")**

**.bind("focus", function () {**

**$("div").html("请输入您的姓名！");**

**})**

**$("input").bind("blur", function () {**

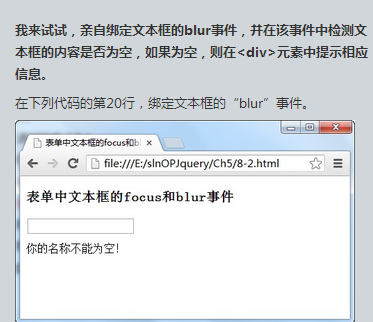
**if ($(this).val().length == 0)**

**$("div").html("你的名称不能为空！");**

**})**

**});**

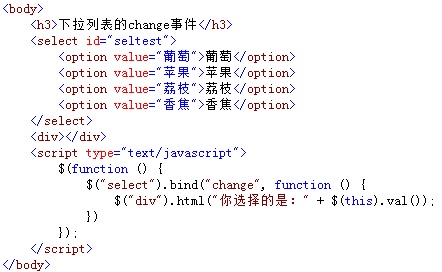
**</script>**

****

## 6-9下拉列表框的change事件

当一个元素的值发生变化时，将会触发change事件，例如在选择下拉列表框中的选项时，就会触change事件。

例如，当在页面选择下拉列表框中的选项时，将在<div>元素中显示所选择的选项内容，如下图所示：

[](http://img.mukewang.com/52d236cc00019b6d04420272.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52d236ea0001022a04890294.jpg)

从图中可以看出，由于使用bind()方法绑定了下拉列表的“change”事件，因此，当选择列表中的选项时，在<div>元素中显示所选择的选项内容。

**<script type="text/javascript">**

**$(function () {**

**$("select").bind("change", function () {**

**if ($(this).val() == "苹果")**

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

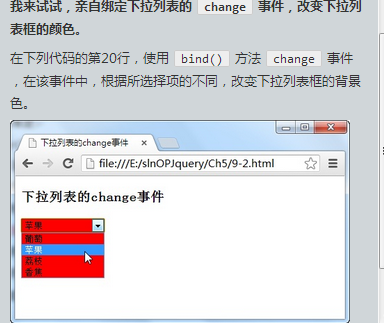
**else**

**$(this).css("background-color", "green");**

**})**

**});**

**</script>**

****

## 6-10调用live()方法绑定元素的事件

与bind()方法相同，live()方法与可以绑定元素的可执行事件，除此相同功能之外，live()方法还可以绑定动态元素，即使用代码添加的元素事件

****

# **第7章jQuery动画特效**

## 7-1调用show()和hide()方法显示和隐藏元素

show()和hide()方法用于显示或隐藏页面中的元素，它的调用格式分别为：

**$(selector).hide(speed,[callback])和$(selector).show(speed,[callback])**

参数speed设置隐藏或显示时的速度值，可为“slow”、“fast”或毫秒数值，可选项参数callback为隐藏或显示动作执行完成后调用的函数名。

例如，在页面中，通过点击“显示”和“隐藏”两个按钮来控制图片元素的显示或隐藏状态，如下图所示：

[](http://img.mukewang.com/52dc9f250001508d03860257.jpg)

$(function () {

$("h4").bind("click", function () {

if ($("#hidval").val() == 0) {

$("ul").hide();

$("#hidval").val(1);

} else {

$("ul").show();

$("#hidval").val(0);

}

});

});

## 7-2动画效果的show()和hide()方法

1. show()和hide()方法中的第一个参数可以设置动画效果的速度，可以是“fast”字符，也可以是以毫秒为单位的时间。
2. show()和hide()方法中的第二个参数是方法执行后的回调函数，该函数只有在方法被成功执行后，才会被调用。

<script type="text/javascript">

$(function () {

$("h4").bind("click", function () {

if ($("#hidval").val() == 0) {

$("ul").show(3000,function(){

$("#hidval").val(1);

})

} else {

$("ul").hide(3000,function(){

$("#hidval").val(0);

})

}

})

});

</script>

## 7-3调用toggle()方法实现动画切换效果

第一节我们学过实现元素的显示与隐藏需要使用hide()与show()，那么有没有更简便的方法来实现同样的动画效果呢？

调用toggle()方法就可以很容易做到，即如果元素处于显示状态，调用该方法则隐藏该元素，反之，则显示该元素

例如，调用toggle()方法以动画的效果显示和隐藏图片元素，如下图所示：

[](http://img.mukewang.com/52dca25300010e0405830241.jpg)

$(function () {

var $spn = $("#spnTip");

$("h4").bind("click", function () {

$("ul").toggle(3000,function(){

$spn.html() == "隐藏" ? $spn.html("显示") : $spn.html("隐藏");

})

});

});

## 7-4使用slideUp()和slideDown()方法的滑动效果

可以使用slideUp()和slideDown()方法在页面中滑动元素，前者用于向上滑动元素，后者用于向下滑动元素，它们的调用方法分别为：

**要注意的是：**slideDown()仅适用于**被隐藏的元素；**slideup()则**相反。**

调用slideUp()和slideDown()方法实现页面中元素的向上和向下的滑动效果，如下图所示：

[](http://img.mukewang.com/52dcb1b20001c7d403880386.jpg)

$(function () {

$("h4").bind("click", function () {

if ($("#hidval").val() == 0) {

$("ul").slideUp(3000,function(){

$("#hidval").val(1);

})

} else {

$("ul").slideDown(3000,function(){

$("#hidval").val(0);

})

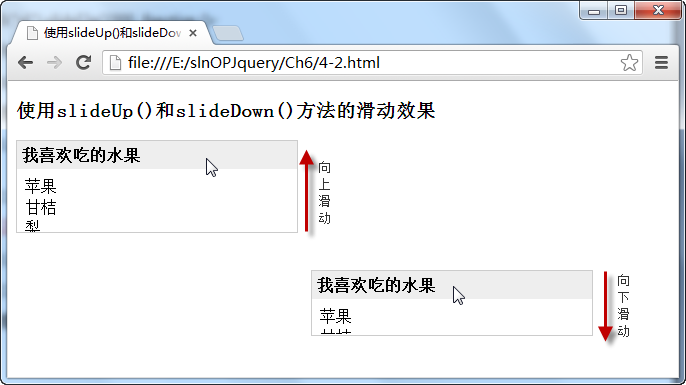
}

})

});

**我来试试，调用slideUp()和slideDown()方法实现“内容”区的滑动显示效果**

在下列代码的第25行和第29行中，分别调用slideUp()和slideDown()方法实现指定“内容”元素的滑动显示效果。

[](http://img.mukewang.com/52dcb2690001f13105070340.jpg)

## 7-5使用slideToggle()方法实现图片“变脸”效果

使用slideToggle()方法可以切换slideUp()和slideDown()，即调用该方法时，如果元素已向上滑动，则元素自动向下滑动，反之，则元素自动向上滑动

例如，在页面中，使用slideToggle()方法实现图片“变脸”效果，如下图所示：

[](http://img.mukewang.com/52dcb5930001bd2603830402.jpg)

(function () {

var $spn = $("#spnTip");

$("h4").bind("click", function () {

$("ul").toggle(2000,function(){

$spn.html() == "向下滑" ? $spn.html("向上滑") : $spn.html("向下滑");

})

})

});

调用slideToggle()方法实现“内容”区的滑动切换。

## 7-6使用fadeIn()与fadeOut()方法实现淡入淡出效果

fadeIn()和fadeOut()方法可以实现元素的淡入淡出效果，前者淡入隐藏的元素，后者可以淡出可见的元素。

例如，分别在页面中以淡入淡出的动画效果显示图片元素，如下图所示：

[](http://img.mukewang.com/549a7819000117a103910414.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/549a782d0001eea505410456.jpg)

$(function () {

$("h4").bind("click", function () {

if ($("#hidval").val() == 0) {

$("ul").fadeIn(1000,function(){

$("#hidval").val(1);

})

} else {

$("ul").fadeOut(1000,function(){

$("#hidval").val(0);

})

}

})

});

## 7-7使用fadeTo()方法设置淡入淡出效果的不透明度

调用fadeTo()方法，可以将所选择元素的不透明度以淡入淡出的效果调整为指定的值，该方法的调用格式为：**$(selector).fadeTo(speed,opacity,[callback])。**

$(function () {

$("span").each(function (index) {

switch (index) {

case 0:

$(this).fadeTo(3000,0.2);

break;

case 1:

$(this).fadeTo(3000,0.6);

break;

case 2:

$(this).fadeTo(3000,0.8);

break;

}

});

});

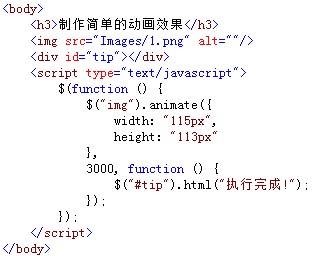


## 7-8调用animate()方法制作简单的动画效果

调用animate()方法可以创建自定义动画效果，它的调用格式为：

**$(selector).animate({params},speed,[callback])**

其中，**params参数为制作动画效果的CSS属性名与值，speed参数为动画的效果的速度，单位为毫秒，可选项callback参数为动画完成时执行的回调函数名。**

[](http://img.mukewang.com/52dcc3b30001fd5803210256.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dcc3ce0001af6604460337.jpg)

<script type="text/javascript">

$(function () {

$("span").animate({

width: "80px",

height: "80px"

},

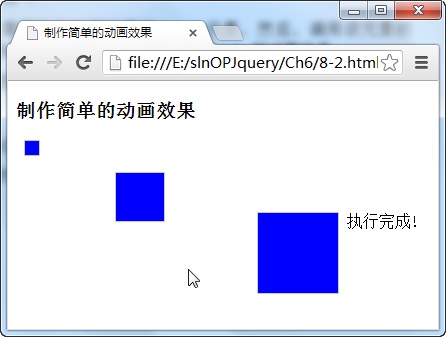
3000, function () {

$("#tip").html("执行完成!");

});

});

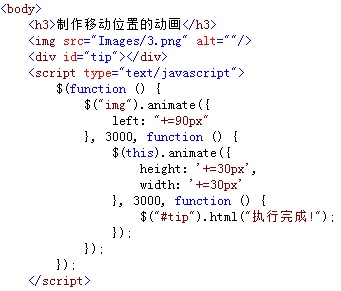
在下列代码的第16行，调用animate()方法制作一个渐渐放大的正方体。

[](http://img.mukewang.com/52dcc41f0001452104460337.jpg)

## 7-9调用animate()方法制作移动位置的动画

调用animate()方法不仅可以制作简单渐渐变大的动画效果，而且还能制作移动位置的动画，在移动位置之前，必须将被移元素的“position”属性值设为“absolute”或“relative”，否则，该元素移动不了。

例如，调用animate()方法先将图片向右移动90px，然后，再将图片宽度与高度分别增加30px，如下图所示：

[](http://img.mukewang.com/52dcc4eb0001c08e03410291.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dcc50b0001ba2205060337.jpg)

从图中可以看出，图片先向右移动了“90px”，然后，移动成功后，再在原来的基础之上以动画的效果增大30px，增加成功后，显示“执行完成！”的字样。

$(function () {

$("span").animate({

left: "+=100px"

}, 3000, function () {

$("span").animate({

height: '+=30px',

width: '+=30px'

}, 3000, function () {

$("#tip").html("执行完成!");

});

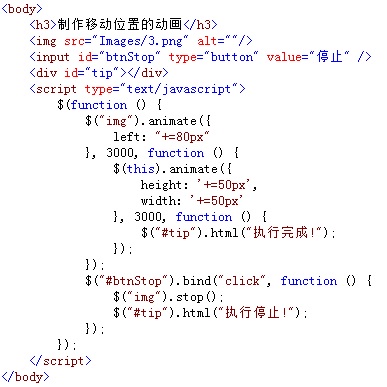
});

## 7-10调用stop()方法停止当前所有动画效果

stop()方法的功能是在动画完成之前，停止当前正在执行的动画效果，这些效果包括滑动、淡入淡出和自定义的动画，它的调用格式为：

**$(selector).stop([clearQueue],[goToEnd])**

其中，例如，在页面中，当图片元素正在执行动画效果时，点击“停止”按钮，中止正在执行的动画效果，如下图所示：

[](http://img.mukewang.com/52dcc63f00019e3203870387.jpg)

## 7-11调用delay()方法延时执行动画效果

delay()方法的功能是设置一个延时值来推迟动画效果的执行，它的调用格式为：

**$(selector).delay(duration)**

其中参数duration为延时值，它的单位是毫秒，当超过延时值时，动画继续执行。

例如，当页面中图片动画正在执行时，点击“延时”按钮调用delay()方法推迟2000毫秒后继续执行，如下图所示：

[](http://img.mukewang.com/52dcc785000162dc03840385.jpg)

# **第8章jQuery实现Ajax应用**

## 8-1使用load()方法异步请求数据

使用load()方法通过Ajax请求加载服务器中的数据，并把返回的数据放置到指定的元素中，它的调用格式为：

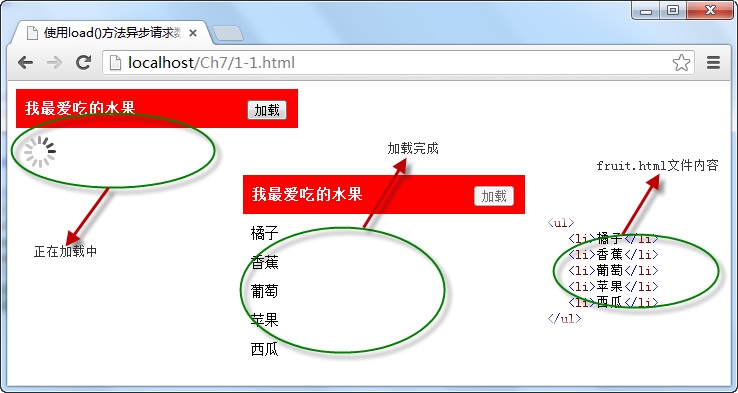
**load(url,[data],[callback])**

参数url为加载服务器地址，可选项data参数为请求时发送的数据，callback参数为数据请求成功后，执行的回调函数。

例如，点击“加载”按钮时，向服务器请求加载一个指定页面的内容，加载成功后，将数据内容显示在<div>元素中，并将加载按钮变为不可用。如下图所示：

[](http://img.mukewang.com/52dccb920001d2d505970337.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dccbaf0001c21507380393.jpg)

从图中可以看出，当点击“加载”按钮时，通过调用load()方法向服务器请求加载fruit.html文件中的内容，当加载成功后，先显示数据，并将按钮变为不可用。

## 8-2使用getJSON()方法异步加载JSON格式数据

使用getJSON()方法可以通过Ajax异步请求的方式，获取服务器中的数组，并对获取的数据进行解析，显示在页面中，它的调用格式为：

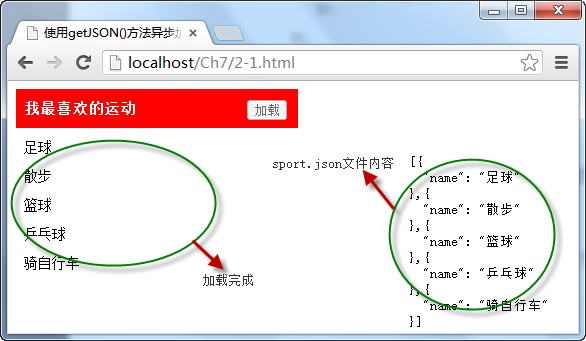
**jQuery.getJSON(url,[data],[callback])或$.getJSON(url,[data],[callback])**

其中，url参数为请求加载json格式文件的服务器地址，可选项data参数为请求时发送的数据，callback参数为数据请求成功后，执行的回调函数。

例如，点击页面中的“加载”按钮，调用getJSON()方法获取服务器中JSON格式文件中的数据，并遍历数据，将指定的字段名内容显示在页面中。如下图所示：

[](http://img.mukewang.com/52dcced70001f67806010370.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dccefe0001c87005860341.jpg)

从图中可以看出，当点击“加载”按钮时，通过getJSON()方法调用服务器中的sport.json文件，获取返回的data文件数据，并遍历该数据对象，以data[“name”]取出数据中指定的内容，显示在页面中。



## 8-3使用getScript()方法异步加载并执行js文件

使用getScript()方法异步请求并执行服务器中的JavaScript格式的文件，它的调用格式如下所示：

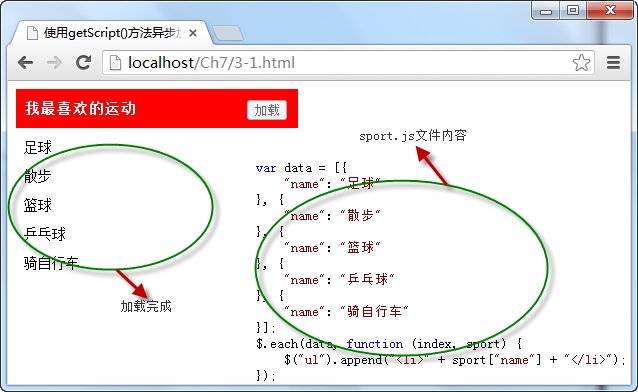
jQuery.getScript(url,[callback])或$.getScript(url,[callback])

参数url为服务器请求地址，可选项callback参数为请求成功后执行的回调函数。

例如，点击“加载”按钮，调用getScript()加载并执行服务器中指定名称的JavaScript格式的文件，并在页面中显示加载后的数据内容，如下图所示：

在浏览器中显示的效果：



[](http://img.mukewang.com/52dcd44e000145de06380392.jpg)

从图中可以看出，当点击“加载”按钮调用getScript()方法加载服务器中的JavaScript格式文件后，自动执行文件代码，将数据内容显示在<ul>元素中。



## 8-4使用get()方法以GET方式从服务器获取数据

使用get()方法时，采用GET方式向服务器请求数据，并通过方法中回调函数的参数返回请求的数据，它的调用格式如下：

$.get(url,[callback])

参数url为服务器请求地址，可选项callback参数为请求成功后执行的回调函数。

例如，当点击“加载”按钮时，调用get()方法向服务器中的一个.php文件以GET方式请求数据，并将返回的数据内容显示在页面中，如下图所示：

[](http://img.mukewang.com/52dcd5b1000161ce05970338.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dcd5d900010aaf06440326.jpg)

从图中可以看出，通过$.get()方法向服务器成功请求数据后，在回调函数中通过data参数传回请求的数据，并以data.name格式访问数据中各项的内容。

****

## 8-5使用post()方法以POST方式从服务器发送数据

1. 调用$.post()方法时，第一个参数为请求服务器的路径，第二个参数为请求进上传的参数，参数格式为key/value方式。
2. 在调用$.post()方法向服务器发送数据时，如果要接收返回的数据，需要添加该方法中的第三个参数，即回调函数，通过该函数中的参数获取返回值。

例如，在输入框中录入一个数字，点击“检测”按钮，调用post()方法向服务器以POST方式发送请求，检测输入值的奇偶性，并显示在页面中，如下图所示：

[](http://img.mukewang.com/52dcd7430001e25004690464.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dcd77200010b4c05450336.jpg)

从图中可以看出，当点击“检测”按钮时，获取输入框中的值，并将该值使用$.post()方法一起发送给服务器，服务器接收该值后并进行处理，最后返回处理结果。

****

## 8-6使用serialize()方法序列化表单元素值

使用serialize()方法可以将表单中有name属性的元素值进行序列化，生成标准URL编码文本字符串，直接可用于ajax请求，它的调用格式如下：

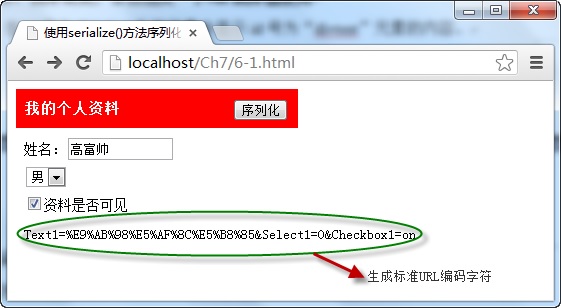
**$(selector).serialize()**

其中selector参数是一个或多个表单中的元素或表单元素本身。

例如，在表单中添加多个元素，点击“序列化”按钮后，调用serialize()方法，将表单元素序列化后的标准URL编码文本字符串显示在页面中，如下图所示：

[](http://img.mukewang.com/52dcd9470001b1e705400481.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52dcd9660001637f05610308.jpg)

从图中可以看出，当点击“序列化”按钮后，调用表单元素本身的serialize()方法，将表单中元素全部序列化，生成标准URL编码，各元素间通过&号相联。

****

****

**返回值 text， select0是男，1是女。Checkbox的状态。**

## 8-7使用ajax()方法加载服务器数据

# 第9章 jQuery 常用插件

## 9-1表单验证插件——validate

[**http://www.cnblogs.com/kissdodog/archive/2012/12/05/2804172.html**](http://www.cnblogs.com/kissdodog/archive/2012/12/05/2804172.html)

****

**$("#frmV").validate(**

**{**

**/\*自定义验证规则\*/**

**rules: {**

**email:{ required: true,**

**email: true,**

**minlength:10,**

**maxlength:30**

**}**

**},**

**/\*错误提示位置\*/**

**errorPlacement: function (error, element) {**

**error.appendTo(".tip");**

**}**

**}**

**);**

## 9-2表单插件——form

通过表单form插件，调用ajaxForm()方法，实现ajax方式向服务器提交表单数据，并通过方法中的options对象获取服务器返回数据，调用格式如下：

**$(form). ajaxForm ({options})**

其中form参数表示表单元素名称；options是一个配置对象，用于在发送ajax请求过程，设置发送时的数据和参数。

例如，在页面中点击“提交”按钮，调用form插件的  
ajaxForm()方法向服务器发送录入的用户名和密码数据，服务器接收后返回并显示在页面中，如下图所示：

[](http://img.mukewang.com/52e4861d000143ad04720449.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4863b0001b62e07360383.jpg)

从图中可以看出，当点击“提交”按钮时，调用form表单插件中的ajaxForm()方法向指定的服务器以ajax方式发送数据，服务器接收后返回并将数据显示。

**发送的是form，指的是整个表单的数据**

## 9-3图片灯箱插件——lightBox

该插件可以用圆角的方式展示选择中的图片，使用按钮查看上下张图片，在加载图片时自带进度条，还能以自动播放的方式浏览图片，调用格式如下：

**$(linkimage).lightBox({options})**

其中linkimage参数为包含图片的<a>元素名称，options为插件方法的配置对象。

例如，以列表的方式在页面中展示全部的图片，当用户单击其中某张图片时，通过引入的图片插件，采用“灯箱”的方式显示所选的图片，如下图所示：

<head>

<title>图片灯箱插件</title>

<link rel="stylesheet" type="text/css" href="http://www.imooc.com/data/jquery.notesforlightbox.css" />

<link href="style.css" rel="stylesheet" type="text/css" />

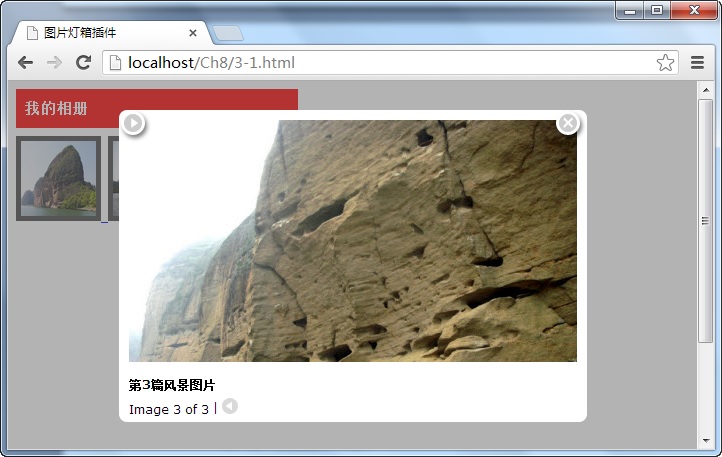
<script type="text/javascript" src="http://www.imooc.com/data/jquery-1.8.2.min.js"></script>

<script type="text/javascript" src="http://www.imooc.com/data/jquery.notesforlightbox.js"></script>

</head>

[](http://img.mukewang.com/52e488760001d36c05070495.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4889600013b1107220457.jpg)

从图中可以看出，当用户点击“我的相册”中某一张图片时，则采用“灯箱”的方式显示选中图片，在显示图片时，还可以切换上下张和自动播放及关闭图片。

## 9-4图片放大镜插件——jqzoom

在调用jqzoom图片放大镜插件时，需要准备一大一小两张一样的图片，在页面中显示小图片，当鼠标在小图片中移动时，调用该插件的jqzoom()方法，显示与小图片相同的大图片区域，从而实现放大镜的效果，调用格式如下：

**$(linkimage).jqzoom({options})**

其中linkimage参数为包含图片的<a>元素名称，options为插件方法的配置对象。

例如，在页面中，添加一个被<a>元素包含的图片元素，当在图片元素中移动鼠标时，在图片的右边，将显示放大后的所选区域效果，如下图所示：

[](http://img.mukewang.com/52e49c2500014a9b04630338.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e49c43000157d205720384.jpg)

从图中可以看出，当在小图片中移动鼠标时，将调用放大镜插件的jqzoom()方法，在图片的右侧显示与小图片所选区域相同的放大区域，实现放大镜的效果。

## 9-5cookie插件——cookie

使用cookie插件后，可以很方便地通过cookie对象保存、读取、删除用户的信息，还能通过cookie插件保存用户的浏览记录，它的调用格式为：

**保存：$.cookie(key，value)；读取：$.cookie(key)，删除：$.cookie(key，null)**

其中参数key为保存cookie对象的名称，value为名称对应的cookie值。

例如，当点击“设置”按钮时，如果是“否保存用户名”的复选框为选中状态时，则使用cookie对象保存用户名，否则，删除保存的cookie用户名，如下图所示：

[](http://img.mukewang.com/52e49d8100016e2c06280481.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e49db900011a7706060340.jpg)

从图中可以看出，由于在点击“设置”按钮时，选择了保存用户名，因此，输入框中的值被cookie保存，下次打开浏览器时，直接获取并显示保存的cookie值。

## 9-6搜索插件——autocomplete

搜索插件的功能是通过插件的autocomplete()方法与文本框相绑定，当文本框输入字符时，绑定后的插件将返回与字符相近的字符串提示选择，调用格式如下：

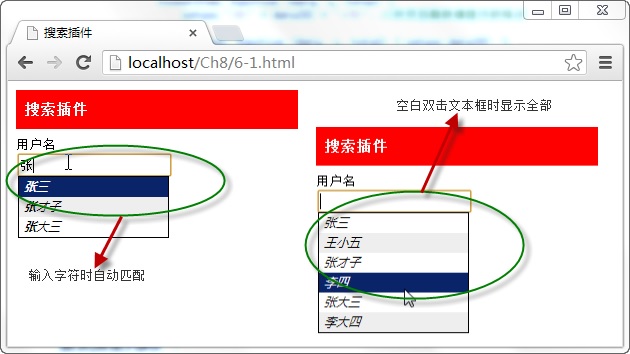
**$(textbox).autocomplete(urlData,[options]);**

其中，textbox参数为文本框元素名称，urlData为插件返回的相近字符串数据，可选项参数options为调用插件方法时的配置对象。

例如，当用户在文本框输入内容时，调用搜索插件的autocomplete()方法返回与输入内容相匹配的字符串数据，显示在文本框下，提示选择，如下图所示：

[](http://img.mukewang.com/52e49eb90001024606410464.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e49ed2000183f806300354.jpg)

从图中可以看出，当文本框与搜索插件相绑定后，输入任意字符时，都将返回与之相匹配的字符串，提示用户选择，文本框在空白双击时，显示全部提示信息。

## 9-7自定义对象级插件——lifocuscolor插件

自定义的lifocuscolor插件可以在<ul>元素中，鼠标在表项<li>元素移动时，自定义其获取焦点时的背景色，即定义<li>元素选中时的背景色，调用格式为：

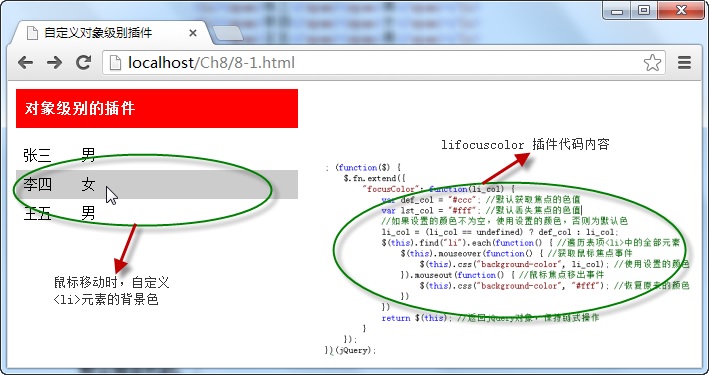
**$(Id).focusColor(color)**

其中，参数Id表示<ul>元素的Id号，color表示<li>元素选中时的背景色。

例如，在页面中，调用自定义的lifocuscolor插件，自定义<li>元素选中时的背景色，如下图所示：

[](http://img.mukewang.com/52e4a100000199ac04200306.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4a11d0001669107090375.jpg)

从图中可以看出，当<ul>元素调用focusColor()方法绑定自定义的插件之后，当鼠标在<li>元素间移动时，显示自定义的背景色。

# 第10章 jQuery UI型插件

## 10-1拖曳插件——draggable

拖曳插件draggable的功能是拖动被绑定的元素，当这个jQuery UI插件与元素绑定后，可以通过调用draggable()方法，实现各种拖曳元素的效果，调用格式如下：

**$(selector). draggable({options})**

options参数为方法调用时的配置对象，根据该对象可以设置各种拖曳效果，如“containment”属性指定拖曳区域，“axis”属性设置拖曳时的坐标方向。

例如，在页面中的<div>元素中添加两个子类<div>，通过与拖曳插件绑定，这两个子类<div>元素只能在外层的父<div>元素中任意拖曳，如下图所示：

[](http://img.mukewang.com/52e4b50c0001545304260177.jpg)

**$(function () {**

**$("#x").draggable({containment:"parent",axis:"x"});**

**$("#y").draggable({containment:"parent",axis:"y"});**

**});**

## 10-2放置插件——droppable

除使用draggable插件拖曳任意元素外，还可以调用droppable UI插件将拖曳后的任意元素放置在指定区域中，类似购物车效果，调用格式如下：

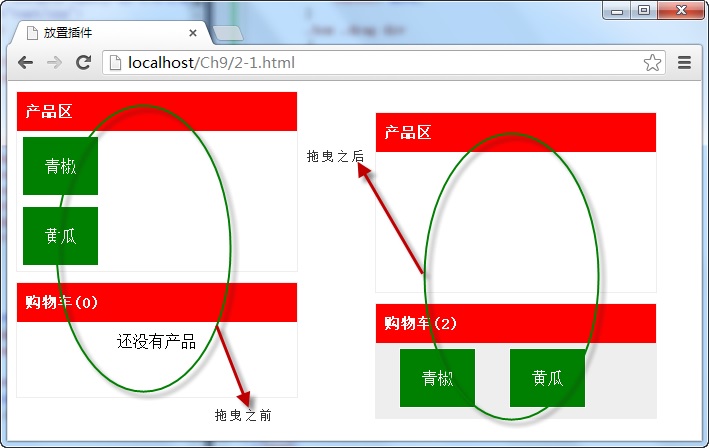
**$(selector).droppable({options})**

selector参数为接收拖曳元素，options为方法的配置对象，在对象中，drop函数表示当被接收的拖曳元素完全进入接收元素的容器时，触发该函数的调用。

例如，在页面中，通过调用droppable插件将“产品区”中的元素拖曳至“购物车”中，同时改变“购物车”的背景色和数量值，如下图所示：

[](http://img.mukewang.com/52e4b7e000012af004800447.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4b7fa0001435707090448.jpg)

**<script type="text/javascript">**

**var intsum = 0;**

**$(function () {**

**$(".drag").draggable();**

**$(".cart").droppable({**

**drop: function () {**

**intsum++;**

**$(this).addClass("focus") .find("#tip").html("");**

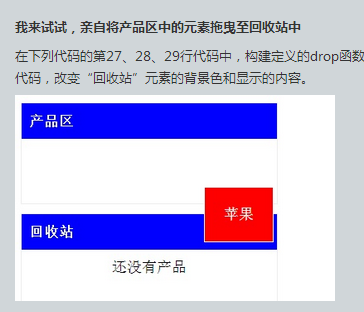
**$("title span").html(intsum);**

**}**

**})**

**});**

**</script>**

****

## 10-3拖曳排序插件——sortable

拖曳排序插件的功能是将序列元素（例如<option>、<li>）按任意位置进行拖曳从而形成一个新的元素序列，实现拖曳排序的功能，它的调用格式为：

**$(selector).sortable({options});**

selector参数为进行拖曳排序的元素，options为调用方法时的配置对象，

例如，在页面中，通过加载sortable插件将<ul>元素中的各个<li>表项实现拖曳排序的功能，如下图所示：

[](http://img.mukewang.com/52e4bb9b0001075303890385.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4bbb60001ae1606600393.jpg)

从图中可以看出，由于使用sortable插件绑定了<ul>元素，并设置了拖曳时的透明度，因此，<ul>中的各个<li>元素则能指定的透明度进行任意的拖曳排序。

**<script type="text/javascript">**

**$(function () {**

**$("ul").sortable({**

**delay:200,**

**opacity:0.35**

**})**

**});**

**</script>**

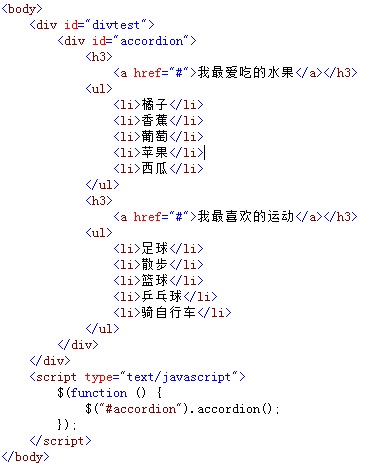
## 10-4面板折叠插件——accordion

面板折叠插件可以实现页面中指定区域类似“手风琴”的折叠效果，即点击标题时展开内容，再点另一标题时，关闭已展开的内容，调用格式如下：

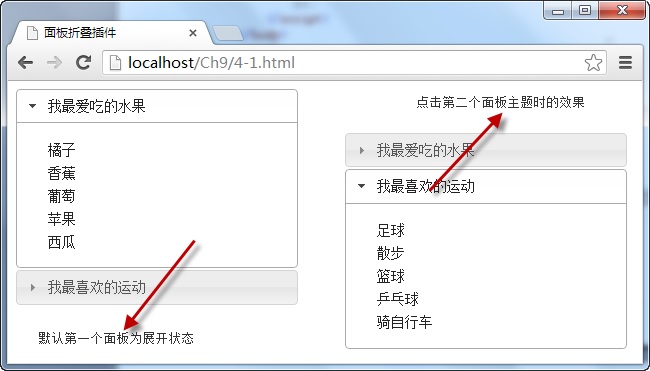
**$(selector).accordion({options});**

其中，参数selector为整个面板元素，options参数为方法对应的配置对象。

例如，通过accordion插件展示几个相同区域面板的折叠效果，如下图所示：

[](http://img.mukewang.com/52e4bd2c0001e58b03680467.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4bd4a0001b4ea06500371.jpg)

从图中可以看出，由于绑定了折叠面板插件，默认为第一个面板的内容为展示状态，点击第二个面板主题时，展示主题对应内容，同时关闭上一个面板内容。

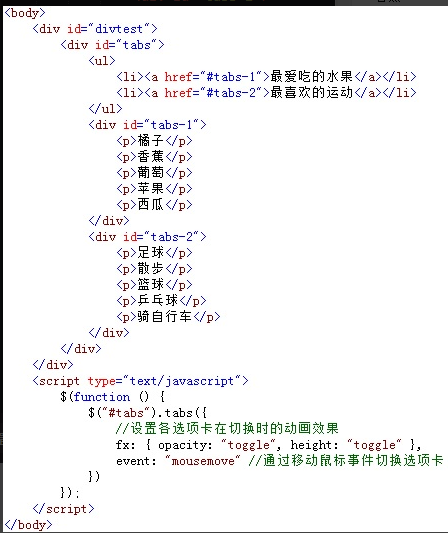
## 10-5选项卡插件——tabs

使用选项卡插件可以将<ul>中的<li>选项定义为选项标题，在标题中，再使用<a>元素的“href”属性设置选项标题对应的内容，它的调用格式如下：

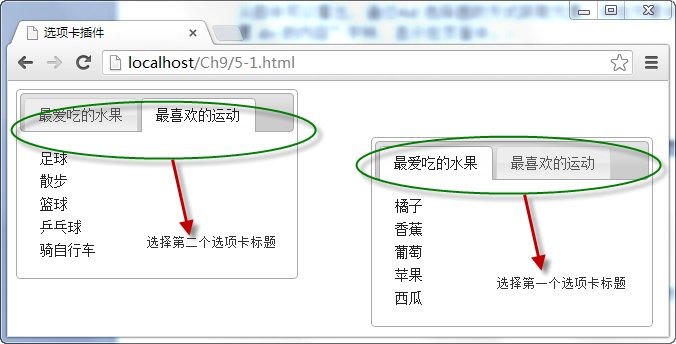
**$(selector).tabs({options});**

selector参数为选项卡整体外围元素，该元素包含选项卡标题与内容，options参数为tabs()方法的配置对象，通过该对象还能以ajax方式加载选项卡的内容。

例如，在页面中，添加选项卡的标题和内容元素，并绑定tabs插件，当点击标题时，以选项卡的方式切内容，如下图所示：



在浏览器中显示的效果：

[](http://img.mukewang.com/52e4bec4000174c706760344.jpg)

从图中可以看出，在tabs()方法的配置对象中，通过“fx”属性设置了选项卡切换时的效果，“event”属性设置鼠标也可以切换选项卡，因此，当鼠标在移动至两个选项卡标题时，对应内容以动画的效果自动切换。

## 10-6对话框插件——dialog

对话框插件可以用动画的效果弹出多种类型的对话框，实现JavaScript代码中alert()和confirm()函数的功能，它的调用格式为：

**$(selector).dialog({options});**

selector参数为显示弹出对话框的元素，通常为<div>，options参数为方法的配置对象，在对象中可以设置对话框类型、“确定”、“取消”按钮执行的代码等。

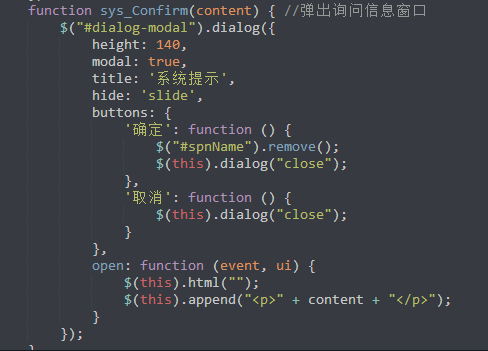
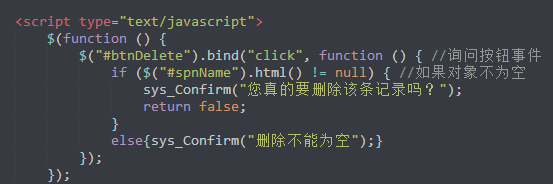
例如，当点击“提交”按钮时，如果文本框中的内容为空，则通过dialog插件弹出提示框，提示输入内容不能为空，如下图所示：

[](http://img.mukewang.com/52e4c1370001b0e805150562.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4c150000111de05350338.jpg)

从图中可以看出，当文本框的内容为空时，调用自定义的sys\_Alert函数，在该函数中再调用dialog插件的dialog()方法，弹出带模式的显示信息对话框。

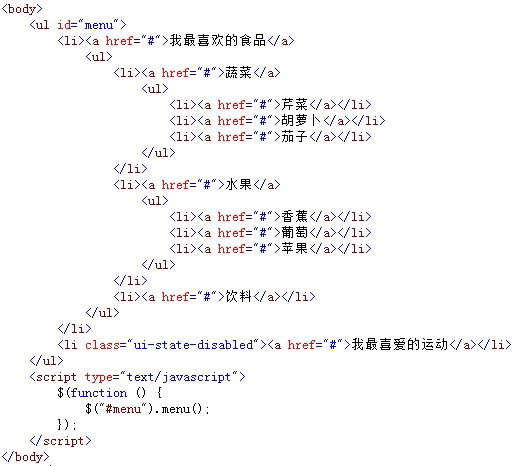
****

## 10-7菜单工具插件——menu

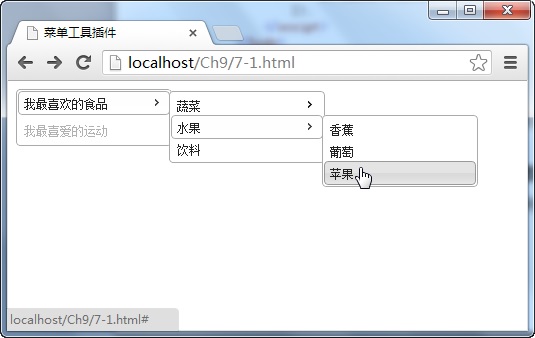
**$(selector).menu({options});**

selector参数为菜单列表中最外层<ul>元素，options为menu()方法的配置对象。

例如，在页面中，通过<ul>元素内联的方式构建一个三层结构的导航菜单，并将最外层<ul>元素通过menu()方法绑定插件，实现导航菜单的功能，如下图所示：

[](http://img.mukewang.com/52e4c2c10001b1a705160466.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4c2dd0001b82505350338.jpg)

从图中可以看出，通过<ul>内嵌的方式，构建一个三层结构的导航菜单，将<li>元素的class属性值设为“ui-state-disabled”，可将菜单选项置为不可用状态。

## 10-8微调按钮插件——spinner

微调按钮插件不仅能在文本框中直接输入数值，还可以通过点击输入框右侧的上下按钮修改输入框的值，还支持键盘的上下方向键改变输入值，调用格式如下：

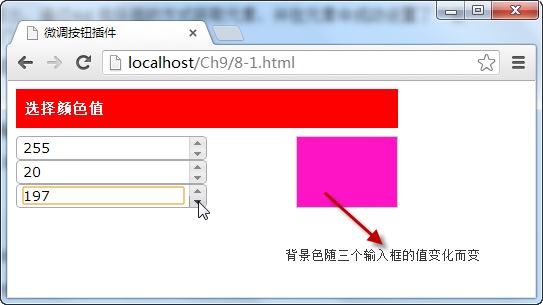
**$(selector).spinner({options});**

selector参数为文本输入框元素，可选项options参数为spinner()方法的配置对象，在该对象中，可以设置输入的最大、最小值，获取改变值和设置对应事件。

例如，将页面中的三个输入文本框都与微调插件相绑定，当改变三个文本框值时，对应的<div>元素的背景色也将随之发生变化，如下图所示：

[](http://img.mukewang.com/52e4c4ad0001d4e007610813.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4c4c70001205f05430305.jpg)

从图中可以看出，由于三个文本框输入元素都绑定微调插件，因此，无论是点击右侧的上下按钮，还是直接在文本框中输入值，都可以改变<div>元素的背景色。

## 10-9工具提示插件——tooltip

工具提示插件可以定制元素的提示外观，提示内容支持变量、Ajax远程获取，还可以自定义提示内容显示的位置，它的调用格式如下：

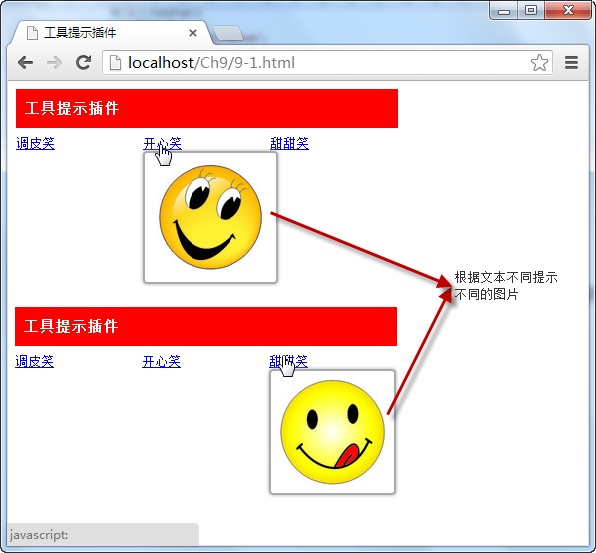
**$(selector).tooltip({options});**

其中selector为需要显示提示信息的元素，可选项参数options为tooltip()方法的配置对象，在该对象中，可以设置提示信息的弹出、隐藏时的效果和所在位置。

例如，将三个<a>元素与工具提示插件相绑定，当把鼠标移动在<a>元素内容时，以动画效果弹出对应的提示图片，移出时，图片自动隐藏，如下图所示：

[](http://img.mukewang.com/52e4c6360001f38506900450.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4c6530001236e05960553.jpg)

从图中可以看出，由于各个<a>元素都绑定了工具提示插件，因此，将在指定的位置并以动画效果展示各个<a>元素中title属性所对应的内容。

**<script type="text/javascript">**

**$(function () {**

**$("#id").toolkit({**

**show: {**

**effect: "slideDown",**

**delay: 350**

**},**

**hide: {**

**effect: "explode",**

**delay: 350**

**},**

**position: {**

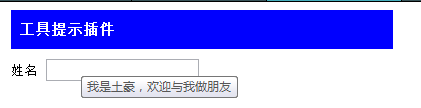
**my: "left top",**

**at: "left bottom"**

**}**

**});**

**});**

****

# ****第11章 jQuery 工具类函数****

## 11-1获取浏览器的名称与版本信息

$(function () {

var strTmp = "您的浏览器名称是：";

if ($.browser.chrome) { //谷歌浏览器

strTmp += "Chrome";

}

if ($.browser.mozilla) { //火狐相关浏览器

strTmp += "Mozilla FireFox";

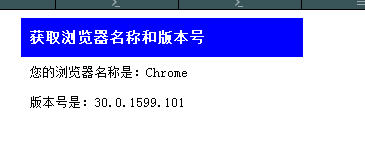
}

strTmp += "<br /><br /> 版本号是：" //获取版本号

+$.browser.version;

$(".content").html(strTmp);

});



## 11-2检测对象是否为空

在jQuery中，可以调用名为$.isEmptyObject的工具函数，检测一个对象的内容是否为空，如果为空，则该函数返回true，否则，返回false值，调用格式如下：

**$.isEmptyObject(obj);**

其中，参数obj表示需要检测的对象名称。

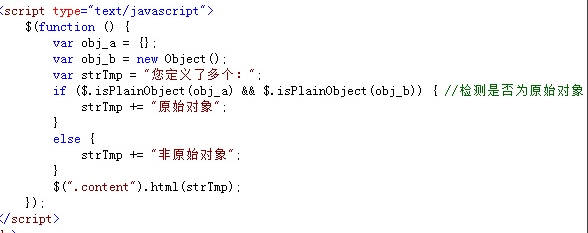
例如，通过$.isEmptyObject()函数，检测某个指定的对象是否为空，并将结果显示在页面中，如下图所示：

[](http://img.mukewang.com/52e4d28b0001b06c03810337.jpg)

## 11-3检测对象是否为原始对象

调用名为$.isPlainObject的工具函数，能检测对象是否为通过{}或new Object()关键字创建的原始对象，如果是，返回true，否则，返回false值，调用格式为：

**$.isPlainObject (obj);**



## 11-4检测两个节点的包含关系

调用名为$.contains的工具函数，能检测在一个DOM节点中是否包含另外一个DOM节点，如果包含，返回true，否则，返回false值，调用格式为：

**$.contains (container, contained);**

参数container表示一个DOM对象节点元素，用于包含其他节点的容器，contained是另一个DOM对象节点元素，用于被其他容器所包含。



## 11-5字符串操作函数

调用名为$.trim的工具函数，能删除字符串中左右两边的空格符，但该函数不能删除字符串中间的空格，调用格式为：

**$.trim (str);**

参数str表示需要删除左右两边空格符的字符串。

例如，通过$.contains()函数，除掉一个两边均有空格符的字符串，并将其执行前后的字符长度都显示在页面中，如下图所示：

[](http://img.mukewang.com/52e4d72d0001a3c005630480.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4d7470001abd205040320.jpg)

从图中可以看出，由于文本框中的字符串前后分别有一个空格字符，因此，它的字符长度为13，调用trim()函数删除字符串前后空格之后，字符串长度则变为11。

## 11-6URL操作函数

例如，通过$.param()函数，对指定的对象进行序列化编码，使其成为可执行传值的URL地址，并将该地址显示在页面中，如下图所示：

[](http://img.mukewang.com/52e4d8a6000185af04750370.jpg)

在浏览器中显示的效果：

[](http://img.mukewang.com/52e4d8c00001965504850276.jpg)

从图中可以看出，通过调用工具函数$.param()可以将一个对象进行序列化并编码成可以在地址栏中直接执行的URL字符串。

## 11-7使用$.extend()扩展工具函数

调用名为$. extend的工具函数，可以对原有的工具函数进行扩展，自定义类级别的jQuery插件，调用格式为：

**$.** **extend ({options});**

参数options表示自定义插件的函数内容。

例如，调用$.extend()函数，自定义一个用于返回两个数中最大值的插件，并在页面中将插件返回的最大值显示在页面中，如下图所示：

<body>

<div id="divtest">

<div class="title">

<span class="fl">自定义工具函数求两值中最小值</span>

<span class="fr">

<input id="btnShow" name="btnShow" type="button" value="计算" />

</span>

</div>

<div class="content">

<div class="tip"></div>

</div>

</div>

<script type="text/javascript">

/\*------------------------------------------------------------/

功能：返回两个数中最小值

参数：数字p1,p2

返回：最小值的一个数

示例：$.MinNum(1,2);

/------------------------------------------------------------\*/

(function ($) {

$.extend({

"MinNum": function (p1, p2) {

return (p1 > p2) ? p2 : p1;

}

});

})(jQuery);

$(function () {

$("#btnShow").bind("click", function () {

$(".tip").html("");

var strTmp = "17与5中最小的数是：";

strTmp +=$.MinNum (17, 5);

//显示在页面中

$(".tip").show().append(strTmp);

});

});

</script>

</body>