1、 用$("#id")代替document.getElementById("id")

用$("tagName")代替document.getElementByTagName("tagName")

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

1-1、jQuery对象转化成DOM对象

jQuery对象不能使用DOM对象的方法，但是如果转化成DOM对象就可以使用其中的方法。转化方法如下：

(1)jQuery对象是一个数组对象，可以通过[index]来获取相应的DOM对象。eg.

var $cr=$(".class"); //jQuery对象

var cr=$cr[0];或者var cr=$cr.get(0); //DOM对象

2、jQuery选择器：

①基本选择器

最基本最常用的选择器，通过id,class和标签名等来查找DOM元素。

#id 根据给定的id匹配一个元素 单个元素 $("#test")

.class 根据给定的类名匹配元素 集合元素 $(".test")选取所有class为test的元素

element 根据给定的元素名匹配元素 集合元素 $("p")选取所有的<p>元素

\* 匹配所有元素 集合元素 $("\*")选取所有元素

selector1,

selector2, 将每一个选择器匹配到的元素 集合元素 $("div,span,p.myClass")选取所有<div>,<span>和拥有class

...... 合并后一起返回 为myClass的<p>标签的一组元素

selectorN

②层次选择器

通过DOM元素之间的层次关系来获取特定元素，例如后代元素、子元素、相邻元素和兄弟元素

div的后代元素指的是div里的所有DOM元素，子元素指的是div里最外层范围最大的元素

$("div span") 选取<div>里的所有<span>（后代）元素 集合元素

$("div>span") 选取div元素下的span（子）元素 集合元素

$("prev+next") 选取紧接在prev元素之后的next元素，

=$("prev").next("next") prev可以是.class等 集合元素

$("prev~siblings") 选取prev之后的所有siblings兄弟元素 集合元素

=$("prev").nextAll("siblings")

$("prev").siblings("div")选择prev所有的同辈div元素，无论前后位置

③过滤选择器

1、基本过滤选择器

:first 选取第一个元素 单个元素 $("div:first")选取所有div元素中第一个div元素

:last 选取最后一个元素 单个元素 $("div:last")选取所有div元素中最后一个div元素

:not(selector) 去除所有与给定选择器

匹配的元素 集合元素 $("input:not(.myClass)")选取class不是myClass的input元素

:header 选取所有的标题元素，

例如h1,h2,h3等等 集合元素 $(":header")选取网页中的<h1>,<h2>,<h3>......

:even 选取索引是偶数的所有 集合元素 $("input:even")选取索引是偶数的<input>元素

元素，索引从0开始

:odd 选取索引是奇数的所有

元素，索引从0开始 集合元素 $("input:odd")选取索引是奇数的<input>元素

:eq(index) 选取索引等于index的元素 单个元素 $("input:eq(1)")选取索引等于1的<input>元素

(index从0开始)

:gt(index) 选取索引大于index的元素 单个元素 $("input:eq(1)")选取索引大于1的<input>元素

(index从0开始)

:lt(index) 选取索引小于index的元素 单个元素 $("input:eq(1)")选取索引小于1的<input>元素

(index从0开始)

2、内容过滤选择器

:contains(text) 选取含有文本内容为"text"的元素 集合元素 $("div:contains('我')")选取含有文本“我”的div元素

:empty 选取不包含子元素或内容的元素 集合元素 $("div:empty")选取不包含子元素或内容的<div>元素

:has(selector) 选取含有选择器所匹配的元素的 集合元素 $("div:has(p)")选取含有p元素的div元素

元素

:parent 选取含有文本或子元素的元素 集合元素 $("div:parent")选取含有子元素或文本的div元素

3、可见性过滤选择器

:hidden 选取所有不可见元素 集合元素 $(":hidden")选取所有不可见元素，包括<input type="hidden"/>

<div style="display:none">和<div style="visibility:hidden">

如果只想选取input元素，可以用$("input:hidden")

:visible 选取所有可见元素 集合元素 $("div:visible")选取所有可见div元素

4、属性过滤选择器

[ ibute] 选取拥有此属性的元素 $("div[id]")选取拥有属性id的元素

[attribute=value] 选取属性的值为value的元素 $("div[title='test']")选取属性title为test的<div>元素

见《锋利的JQuery》P43

5、

④表单选择器

1、本选择器能使用户更加灵活的操作表单

:input 选取所有的input,select,textarea,button等元素 $(":input")选取所有的input,select,textarea,button等元素

:text 选取所有单行文本框 $(":text")选取所有单行文本框

:password 选取所有密码框 $(":password")选取所有密码框

:radio

:checkbox

:submit

:image

:reset

:button

:file

:hidden

注意事项：当属性中含有".","#","()","[]"的特殊字符是要用转义字符\\转义，例如：$("input\\[1\\]")

第3章

1、jQuery中的DOM分类：DOM core,HTML\_DOM,CSS\_DOM

DOM core并非仅限于处理网页，也可以用来处理任何一种使用标记语言编写出来的文档，例如XML

2、HTML\_DOM(仅限于处理网页)

获取表单对象的方法：document.forms//

获取某元素的src属性的方法：elements.src;

3、CSS\_DOM

主要获取和设置style对象的各种属性。例如设置某元素字体颜色的方法：element.style.color="red"

4、attr()方法

参数为一个时，获取属性的名称；

var $para=$("p");

var p\_txt=$para.attr("title");

参数为两个时，改变属性的值。

$para.attr("title","我的世界");将题目改成“我的世界”

3.2.2 创建节点

用$("html")

创建两个<li>新元素

var $li\_1=$("<li></li>");

var $li\_2=$("<li></li>");

添加节点

将上述两个节点添加到ul标签下

$("ul").append($li\_1);

$("ul").append($li\_2);

创建文本节点

var $li\_1=$("<li>香蕉</li>");

创建属性节点

var $li\_1=$("<li title='香蕉'>香蕉</li>");

注意：无论$(html)中的代码多么复杂，都要使用上述的方式进行创建

3.2.3 插入节点

插入节点的方法

|  |  |  |
| --- | --- | --- |
| 方法 | 描述 | 示例 |
| append() | 向每个匹配的元素内部追加内容 | <p>我想说：</p>  $(“p”).append(“<b>你好</b>”);  <p>我想说：<b>你好</b></p> |
| appendTo() | $(A).appendTo(B)将A添加到B中 | <p>我想说：</p>  $(“<b>你好</b>”).appendTo(“p”);  <p>我想说：<b>你好</b></p> |
| prepend() | 插入到匹配元素内容前面 | <p>我想说：</p>  $(“p”).prepend(“<b>你好</b>”);  <p><b>你好</b>我想说：</p> |
| prependTo() | $(A).prependTo(B)将A添加到B前面 |  |
| after() | 在每个匹配的元素之后插入内容 | <p>我想说：</p>  $(“p”).after(“<b>你好</b>”);  <p>我想说：</p><b>你好</b> |
| insertAfter() | $(A).insertAfter(B)将A插入到B的后面 |  |
| before() | 在每个匹配的元素之前插入内容 | <p>我想说：</p>  $(“p”).before(“<b>你好</b>”);  <b>你好</b><p>我想说：</p> |
| insertBefore() | $(A).insertBefore(B)将A插入到B的前面 |  |

利用上述方法移动节点：

var $one\_li=$("ul li:eq(1)");

var $two\_li=$("ul li:eq(2)");

$two\_li.insertBefore($one\_li);//将第三个节点插入到第二个节点的前面

3.2.4 删除节点

方法remove()和empty()

1、remove()方法：

$(“ul li:eq(1)”).remove();//获取第二个元素后将它从网页中删除

注意：①这个方法的返回值是一个指向删除节点的引用，因此可以在以后再使用这个节点。②当某个节点用remove()方法删除后，该方法包含的所有后代节点将同时被删除。

另外remove()方法也可以通过传递参数来选择性地删除元素，代码如下：

$(“ul li”).remove(“li[title!=’菠萝’]”);

2、empty()方法

严格来讲，此方法并不是删除节点，而是清空节点，清空元素中的所有后代节点。

$(“ul li:eq(1)”).empty()清空第二个节点的内容，只剩下<li>标签默认的符号”·”

3.2.5 复制节点

方法clone()

$(“ul li”).click(function(){$(this).clone().appendTo(“ul”)});//点击<li>元素，复制当前节点，并将其添加到<ul>元素节点下。

注意：这种复制方法，被复制的新元素并不具有任何行为，如果需要新元素也具有复制功能(本例中的单机事件)，则要使用下列方法：

$(this).clone(true).appendTo(“ul”);//参数true表示，复制元素的同时也复制元素中绑定的事件。

3.2.6 替换节点

方法replaceWith()和replaceAll()，两种的方法作用相同，只不过前后替换顺序不一样

将<p>用<strong>你最不喜欢的水果是？</strong>替换

$(“p”).replaceWith(“<strong>你最不喜欢的水果是？</strong>”);

$(“<strong>你最不喜欢的水果是？</strong>”).replaceAll(“p”);

注意：如果在替换之前，已经为元素绑定事件，替换后原先绑定的事件将会与被替换的元素一起消失，需要在新元素上重新绑定事件。

3.2.7 包裹节点

方法wrap()、wrapAll()和wrapInner()

用于在文档中插入额外的结构化标记，而且不会破坏原始文档语义。

①wrap()

$(“strong”).wrap(“<b></b>”);//用<b>标签把每一个<strong>元素分别包裹起来。结果如下：

<b><strong title=”选择你喜欢的水果。”>你最喜欢的水果是？</strong></b>

②wrapAll()

该方法会将所有匹配的元素用一个元素来包裹。它不同于wrap()方法，后者是将所有元素分别进行包裹。

$(“strong”).wrapAll(“<b></b>”);结果如下：

<b>

<strong title=”选择你喜欢的水果。”>你最喜欢的水果是？</strong>

<strong title=”选择你喜欢的水果。”>你最喜欢的水果是？</strong>

</b>

③wrapInner()方法

该方法是将匹配元素的内容用其他结构化的标记包裹起来。

例如包裹<strong>标签的子内容

$(“strong”).wrapInner(“<b></b>”);结果如下：

<strong title=”你最喜欢的水果。”><b>你最喜欢的水果是什么？</b></strong>

3.2.8 属性操作

①attr()方法

获取p元素节点属性title

var p\_txt=$(“p”).attr(“title”);

如果要设置title属性的值，则要传递两个参数即属性名和对应的值。

$(“p”).attr(“title”,”your title”);//设置单个属性值

如果要一次性设置多个属性，可以使用下列方法：

$(“p”).attr(“title”:”your title”,”name”:”test”);//将一个“名/值”形式的对象设置为匹配元素的属性

②删除属性removeAttr()

删除p元素的title属性，代码如下：

$(“p”).removeAttr(“title”);//删除p元素的属性title

3.2.9 样式操作

①追加样式addClass()

$(“p”).addClass(“another”);结果如下：<p class=”myclass another”></p>,此时<p>元素同时拥有两个class值。

注意：在CSS中有以下两条规定，

1. 如果给一个元素添加了多个class值，那么就相当于合并了它们的样式。
2. 如果有不同的class设置了同一样式属性，则后者覆盖前者。

例如：

font-weight:bold;

color:red;

font-style:italic;

color:blue;

在以上样式中存在两个color属性，根据第二条，后面的color属性会覆盖前面的color属性，所以最终的color属性值为blue，而不是red。

样式的最终结果为：

font-weight:bold;

font-style:italic;

color:blue;

②移除样式

$(“p”).removeClass(“another”);//移除another样式

$(“p”).removeClass(“another high”);//移除两个样式，中间用空格隔开

③判断是否含有某个样式

hasClass()方法

如果有则返回true,没有则返回false,例如，$(“p”).hasClass(“another”);

3.2.10

设置和获取html、文本和值

1. html()方法

此方法类似于js中的innerHTML属性，可以用来读取或设置某个元素中的html内容。

例如，<p>标签的html代码如下：<p><strong>你最喜欢的水果是？</strong></p>

获取<p>标签的html代码：var p\_html=$(“p”).html();alert(p\_html);//输出内容为<strong>你最喜欢的水果是？</strong>

要设置某元素的html代码，也要用该方法，但要将设置内容作为参数传入其中。

1. text()方法

此方法类似于js中的innerText属性，可以用来读取或设置某个元素中的文本内容。

<p><strong>你最喜欢的水果是？</strong></p>

var p\_text=$(“p”).text();alert(p\_text);//输出结果为：你最喜欢的水果是？

要为某元素设置内容，也需要向其中传递参数。例如：

$(“p”).text(“你最喜欢的水果是？”);//设置<p>元素的文本内容

1. val()方法

有如下html代码：

<select id="single">  
 <option>选择1号</option>  
 <option>选择2号</option>  
 <option>选择3号</option>  
</select>  
<select id="multiple" multiple="multiple" style="height: 120px">  
 <option selected="selected">选择1号</option>  
 <option>选择2号</option>  
 <option>选择3号</option>  
 <option>选择4号</option>  
 <option selected="selected">选择5号</option>  
</select>

要使第一个下拉框的第二项被选中，可以使用以下代码：

$("#single").val("选择2号");

要想使下拉列表框的第二项和第三项被选中，可以用以下代码：

$(“#multiple”).val([“选择2号”,”选择3号”]);//以数组形式赋值

注意：在jQuery中，val()方法是从最后一个选项往前读取，如果选项的value或者text中任意一项符合就会被选中。

3.2.11 遍历节点

①children()方法

该方法用于取得匹配元素的子元素集合。

var $body=$(“body”).children();

②next()方法

该方法用于取得匹配元素后面紧邻的同辈元素，例如，<p>元素下一个同辈元素是<ul>，因此可以通过next()方法来获取<ul>元素。代码如下：

var $p1=$(“p”).next();

③prev()方法

该方法用于取得匹配元素前面紧邻的同辈元素。

从DOM树的结构中可以知道<ul>元素的上一个同辈节点是<p>，因此可以通过prev()方法来获取<p>元素。代码如下：

var $ul=$(“ul”).prev();

得到的结果是<p title=”选择你最喜欢的水果。”>你最喜欢的水果是？</p>

④siblings()方法

该方法用于取得匹配元素前后所有的同辈元素。

⑤closest()方法

用来取得最近的匹配元素。首先检查当前元素是否匹配，如果匹配则直接返回元素本身。如果不匹配，则向上查找父元素，逐级向上直到找到匹配选择器的元素。如果什么都没找到，则返回一个空的jQuery对象。

例如，给点击的目标元素的最近的li元素添加颜色，代码如下：

$(document).bind(“click”,function(e){$(e.target).closest(“li”).css(“color”,”red”);});

除此之外，在jQuery中还有很多遍历节点的方法，例如find()，filter()、nextAll(),prevAll(),parent()和parent()等。

3.2.12 CSS-DOM操作

CSS-DOM操作简单来说就是读取和设置style对象的各种属性。

可以直接利用css()方法获取元素的样式属性，jQuery代码如下：

$(“p”).css(“color”);//获取<p>元素的样式颜色。

无论color属性是外部css导入，还是直接拼接在html元素里(内联)，css()方法都可以获取到属性style里的其他属性的值。

也可以直接利用css()方法设置某个元素的单个样式，例如：

$(“p”).css(“color”,”red”);

与attr()方法一样，css()方法也可以同时设置多个样式属性，代码如下：

$(“p”).css({“fontSize”:”30px”,”backgroundColor”:”#888888”});//同时设置字体大小和背景颜色

--------------------------------------------------------------------------------------------------------------------------

注意：(1)如果值是数字，将会被自动转换为像素值。

(2)在css()方法中，如果属性中带有”-”符号，例如font-size和background-color属性，如果在设置这些属性的时候不带引号，那么就要用驼峰式写法，例如：$(“p”).css({“fontSize”:”30px”,”backgroundColor”:”#888888”});如果带上来引号，既可以写成”font-size”，也可以写成”fontSize”。

--------------------------------------------------------------------------------------------------------------------------

对透明度的设置，可以直接使用opacity属性，jQuery已经处理好了兼容性问题，如下代码所示，将<p>元素的透明度设置为半透明：

$(“p”).css(“opacity”,”0.5”);

如果需要获取某个元素的height属性，则可以通过如下jQuery代码实现：

$(element).css(“height”);

在jQuery还有另外一种方法也可以获取元素的高度，即height()。代码如下：

$(“p”).height();//获取<p>元素高度值。

height()也能用来设置元素的高度，如果传递的值是一个数字，则默认的单位是px，如果要用其他单位(例如em)，则必须传递一个字符串。代码如下：

$(“p”).height(“10em”);

注意：(1)在jQuery1.2版本以后的height()方法可以用来获取window和document的高度。

1. 两者的区别是：css()方法获取的高度值与样式的设置有关，可能会得到”auto”，也可能得到”10px”之类的字符串；而height()获取的高度值则是元素在页面中的实际高度，与样式的设置无关，而且不带单位。

与height()方法对应的还有一个width()方法，它可以取得匹配元素的宽度值(px)，也可以设置匹配元素的宽度。

还有以下几个常用的方法。

①offset()方法

它的作用是获取元素在当前视窗的相对偏移，其中返回的对象包含两个属性，即top和left，它只对可见元素有效。例如用它来获取<p>元素的偏移量。代码如下：

var offset=$(“p”).offset();//获取<p>元素的offset

var left=offset.left;//获取左偏移

var top=offset.top//获取顶部偏移

②position()方法

它的作用是获取元素相对于最近的一个position样式属性设置为relative或者absolute的祖父节点的相对偏移，与offset()一样，他返回的对象也包括两个属性，即top和left。代码如下：

var position=$(“p”).position();

var left=position.left;

var top=position.top;

③scrollTop()方法和scrollLeft()方法

这两个方法的作用分别是获取元素的滚动条距顶端的距离和距左侧的距离。例如使用下面的代码获取<p>元素滚动条距离：

var $p=$(“p”);

var scrollTop=$p.scrollTop();//获取元素滚动条距顶端的距离

var scrollLeft=$p.scrollLeft();//获取元素滚动条距左侧的距离

也可以为两个方法传入参数，来设置滚动条的位置。代码如下：

$(“textarea”).scrollTop(300);//设置元素的垂直滚动条的位置

$(“textarea”).scrollLeft(300);//设置水平滚动条的位置

# 第四章 jQuery中的动画和事件

## 4.1 jQuery中的事件

### 4.1.1 加载DOM

以浏览器加载文档为例，在页面加载完毕以后，浏览器会通过JavaScript为DOM元素添加事件。在js中会使用window.onload方法，在jQuery中会使用$(document).ready()方法。后者可以极大地提高web应用程序的响应速度。下面阐述两者的区别：

1. 执行的时机

window.onload方法是在网页中的所有元素(包括元素的所有关联文件)完全加载到浏览器后才执行，即js此时才可以访问网页中的任何元素。而通过$(document).ready()方法所注册的事件处理程序，在DOM完全就绪时就可以被调用。此时网页的所有元素对jQuery而言都是可以访问的，但是所下载的文件不一定都下载完毕。

另外，为了解决文件或者图片还没有下载下来对其中的属性进行操作不一定有效这样的问题，可以使用jQuery中的load()方法。load()方法会在元素的onload事件中绑定一个处理函数，如果处理函数绑定在window对象上，会在所有内容(包括窗口、框架、对象和图片等)加载完毕后触发，如果处理函数绑定在元素上，则会在元素的内容加载完毕后触发。代码如下：$(window).load(function(){//编写代码});等价于JavaScript中的window.onload=function(){//编写代码}

②多次使用

假设网页中有两个函数：

function one(){alert(“one”);}

function two(){alert(“two”)}

当网页加载完毕后，通过如下JavaScript代码来分别调用这两个函数：

window.onload=one;

window.onload=two;

然而运行代码后，发现只弹出字符串”two”对话框

原因是JavaScript的onload事件一次只能保存对一个函数的引用，他会自动用后面的函数覆盖前面的函数，因此不能在现有的行为上添加新的行为。为了达到两个函数顺序触发的效果，只能再创建一个新的JavaScript方法来实现，JavaScript代码如下：

window.onload=function(){one();two();}

而jQuery就能很好的处理这种情况：

$(document).ready(function(){one();});

$(document).ready(function(){two();});

1. 简写方式

$(document).ready(function(){//编写代码});的简写形式有如下两种：

$(function(){//编写代码});

另外$(document)也可以简写成$()。不带参数时默认参数就是document，

$().ready(function(){//编写代码});

### 4.1.2事件绑定

可以使用bind()方法对匹配元素进行特定事件的绑定。bind()的调用格式为：

bind(type[,data],function);

总共用三个参数，说明如下：

第一个是事件类型，类型包括：blur,focus,load,resize,scroll,unload,click,dblclick,mousedown,mouseup,mousemove,mouseover,mouseout,mouseenter,mouseleave,change,select,submit,keydown,keypress,keyup和error等，当然也可以是自定义名称。

第二个参数为可选参数，作为event.data属性值传递给事件对象的额外数据对象。

注意：jQuery的事件绑定类型比JavaScript的少了”on”

①基本效果

绑定单机事件：

$(function(){$(“#panel h5.head”).bind(“click”,function(){$(this).next(“div.content”,show());});});

简写形式如下：

$(“#panel h5.head”).mouseover(function(){$(this).next(“div.content”).show();});

如果内容是显示的则隐藏，如果内容是隐藏的则显示。

在绑定事件的函数里添加如下代码：

If($content.is(“:visible”)){

$content.hide();

}else{

$content.show();

}

这里使用了is()方法来进行判断。

4.1.3 合成事件

hover()方法和toggle()方法

1. hover()方法

语法结构为hover(enter,leave)

该方法用于模拟鼠标悬停事件。当光标移动到元素上时，会触发指定的第1个函数(enter);当光标移出这个元素时，会触发指定的第2个函数(leave)。

代码如下：

$(**function**(){  
 $("#panel h5.head").hover(**function**(){  
 $(**this**).next("div.content").show();  
 },**function**(){  
 $(**this**).next("div.content").show();  
 });  
});

1. toggle()方法

语法结构为toggle(fn1,fn2,fnN);

该方法用于模拟鼠标点击事件。第一次点击元素触发第一个函数(fn1)，当再次点击触发第二个函数(fn2)，如果有更多函数则依次触发，直到最后一个。随后的每次点击都重复对这几个函数的轮番调用。

该方法还有另外一个作用：切换元素的可见状态。如果元素是可见的，单击切换后则变为隐藏；如果是隐藏的，点击切换后则变为可见。

### 4.1.4 事件冒泡

1、什么是冒泡

<div id="content">

外层div元素

<span>内层span元素</span>

外层div元素

</div>

<div id="msg"></div>

对于上述代码，分别为#content、span和body绑定click事件。

当单击span元素时，其他两个也会响应事件，这就是由事件冒泡引起的。之所以称之为冒泡，是因为事件会按照DOM的层次结构像水泡一样不断向上直至顶端。

2、事件对象

在程序中使用事件对象非常简单，只需要为函数添加一个参数，jQuery代码如下：

$(“element”).bind(“click”,function(event){//事件对象});

单击”element”元素时，事件对象被创建。这个事件对象只有事件处理函数才能访问。事件处理函数执行完毕后，事件对象就被销毁。

1. 停止事件冒泡

可以阻止事件中其他对象的事件处理函数被执行。在jQuery中提供了stopPropagation()方法来停止事件冒泡。

$("span").bind("click",function(event){

var txt=$("#msg").html()+"<p>内层span元素被单击。</p>";

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

event.stopPropagation();

});

1. 阻止默认行为

网页中的元素有自己的默认行为，例如，单击超链接后会跳转、单击“提交”后会提交表单，有时需要阻止元素的默认行为。

preventDefault()方法来阻止元素的默认行为。

例如，在验证表单中，如果验证不通过则要阻止提交按钮的提交行为(默认行为)。

如果想同时对事件对象停止冒泡和默认行为，可以在事件处理函数中返回false。这是在事件对象上同时调用stopPropagation()和preventDefault()的一种简写方式。即

event.stopPropagation()-----🡪return false

event.preventDefault-------🡪return false

③事件捕获

事件捕获和事件冒泡的过程正好相反，从最顶端往下开始触发。

注意：jQuery不支持事件捕获，要想使用事件捕获请直接使用原生的JavaScript。

### 4.1.5 事件对象的属性

①event.type()方法

该方法可以捕获到事件的类型。

$("a").click(function(event){

alert(event.type);

return false; //阻止链接跳转

});

输出结果为click。

②event.stopPropagation()方法

1. event.preventDefault()方法
2. event.target()方法

该方法的作用是获取到触发事件的元素。

$("a[href=http://google.com]").click(function(event){

alert(event.target.href);

return false; //阻止链接跳转

});

输出结果是”<http://google.com>”

⑥event.pageX()方法event.pageY()方法

$("a").click(function(event){

alert("Current mouse position:"+event.pageX+","+event.pageY);

//获取鼠标当前相对于页面的位置

return false; //阻止链接跳转

});

⑦event.which()方法

该方法的作用是在鼠标单击事件中获取鼠标的左中右键；在键盘事件中获取键盘的按键。

$(function(){

$("body").mousedown(function(e){

alert(e.which);//1=左键；2=中键；3=右键

});

});

⑧event.metaKey()方法

该方法为键盘事件中获取<ctrl>按键

⑨event.originalEvent()方法

该方法的作用是指向原始的事件对象。

### 4.1.6 移除事件

①移除按钮元素上以前注册的事件

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

$("#btn").unbind(“click”);或者$(“#btn”).unbind();

});

unbind()的语法结构：unbind([type][,data])

第一个参数是事件类型，第二个参数是将要移除的参数类型。

1. 如果没有参数，则删除所有绑定事件。
2. 如果提供了事件类型作为参数，则只删除该类型的绑定事件。
3. 如果还传递了事件函数作为第2个参数，则只删除特定的事件处理函数。
4. 移除<button>元素其中一个事件

首先为这些匿名处理函数指定一个变量。

$(function(){

$("#btn").bind("click",myFun1=function(){

$("#test").append("<p>我的绑定函数1</p>");

}).bind("click",myFun2=function(){

$("#test").append("<p>我的绑定函数2</p>");

}).bind("click",myFun3=function(){

$("#test").append("<p>我的绑定函数3</p>");

});

});

删除绑定函数2，代码如下：

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

$("#btn").unbind("click",myFun2);

});

one()方法用于只需要触发一次，随后就立即解除绑定。

语法结构：one(type,[data],fn);

### 4.1.7 模拟操作

1、常用模拟

例如，用户进入页面后就触发click事件，而不需要用户去主动单击。

用trigger()方法来实现。触发id为btn的按钮的click事件：$(“#btn”).trigger(“click”);也可以使用简化的写法click()：$(“#btn”).click();

2、触发自定义事件

例如为元素绑定一个”myClick”事件，代码如下：

$(“#btn”).bind(“myClick”,function(){

$(“#test”).append(“<p>我的自定义事件。</p>”);

});

想要触发这个事件，可以使用以下代码来实现：

$(“#btn”).trigger(“myClick”);

3、传递参数

trigger(type[,data])方法有两个参数，第1个参数是要触发的事件类型，第2个参数是要传递给事件处理函数的附加数据，一数组形式传递。

通常可以传递一个参数给回调函数来区分这次事件是代码触发的还是用户触发的。

例子如下：

$("#btn").bind("myClick",function(event,message1,message2){

$("#test").append("<p>"+message1+message2+"</p>");

});

$("#btn").trigger("myClick",["我的自定义","事件"]);

4、执行默认操作

trigger()方法触发事件后，会执行浏览器默认操作。例如：

$(“input”).trigger(“focus”);

以上代码不仅会触发为<input>元素绑定的focus事件，也会使<input>元素本身得到焦点(这是浏览器的默认操作)。

如果只想触发绑定的事件，而不想执行浏览器的默认操作，可以使用triggerHandler()方法。

$(“input”).triggerHandler(“focus”)方法

该方法只会触发<input>上绑定的特定事件，同时取消浏览器对此事件的默认操作，即文本框只触发绑定的focus事件，而不会获得焦点。

### 4.1.8 其他方法

bind()方法

1. 绑定多个事件类型

$(“div”).bind(“mouseover mouseout”,function(){});//当光标滑入<div>元素时，该元素的class切换为”over”;当光标滑出<div>元素时，class切换为先前的值。

1. 添加事件命名空间，便于管理

$(function(){

$("div").bind("click.plugin",function(){

$("body").append("<p>click事件</p>");

});

$("div").bind("mouseover.plugin",function(){

$("body").append("<p>mouseover事件</p>");

});

$("div").bind("dbclick",function(){

$("body").append("<p>dbclick事件</p>");

});

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

$("div").unbind(".plugin");

});

});

这样在删除事件时只需要指定命名空间即可。未指定”plugin”的命名空间的”dbclick”依然存在。

删除多个代码也可以使用链式代码：$(“div”).unbind(“click”).unbind(“mouseover”);

1. 相同事件名称，不同命名空间的执行方法

$(function(){

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

$("body").append("<p>click事件</p>");

});

$("div").bind("click.plugin",function(){

$("body").append("<p>click.plugin事件</p>");

});

$("button").trigger("click!");

});

当单击<div>元素后，会同时触发click事件和click.plugin事件。如果只是单击<button>元素，只触发click事件，而不触发click.plugin事件。

注意：trigger(“click!”)后面的感叹号的作用是匹配所有不包含在命名空间中的click方法。

如果两者都要触发，改为如下代码即可：

$(“div”).trigger(“click”);//去掉感叹号

## 4.2 jQuery中的动画

### 4.2.1 show()方法和hide()方法

1、show()方法和hide()方法

调用hide()方法，会将该元素的display样式改为”none”。

隐藏element元素：$(“element”).hide();//通过hide()方法隐藏元素

这段的代码的功能与用css()方法设置display属性效果相同：

element.css(“display”,”none”);//通过css()方法隐藏元素

可以使用show()方法将隐藏的元素的display属性改为block或者inline或者其他除了none之外的值。

$(“element”).show();

注意：display()在将”内容”的display属性设置为”none”之前，会记住原先display属性值(“block”或”inline”或其他除了”none”之外的值)。当调用show()方法时，就会根据hide()方法记住的display属性值来显示元素。

注意：用jQuery做动画效果要求要在标准模式下，否则可能会引起动画抖动。标准模式即要求文件头部包含如下的DTD定义：

2、show()方法和hide()方法让元素动起来

show()和hide()在不带任何参数的情况下，相当于css(“display”,”none/block/inline”)，作用是立即隐藏和显示匹配的元素，不会有任何动画。如果希望在调用show()方法时，让元素慢慢的显示出来，可以为其指定一个速度参数，例如，制定一个速度关键字slow，代码如下：

$(“element”).show(“slow”);//

元素会在600ms内缓缓显示出来。其他的还有normal和fast(分别为400ms和200ms)，还可以自定义事件，单位为ms，例如$(“a”).show(1000)，超链接将在1s内显示出来。

hide(600)方法会同时减少“内容”的高度、宽度和不透明度，直至这3个属性的值都为0，最后设置该元素的CSS规则为”display:none”。

### 4.2.2 fadeIn()方法和fadeOut()方法

这两种方法只改变元素的不透明度。

fadeOut()方法会在指定的一段时间内降低元素的不透明度，直到元素完全消失(“display:none”)。fadeIn()方法则相反。

### 4.2.3 slideUp()方法和slideDown()方法

这两个方法只会改变元素的高度。如果元素的display属性值为none，当调用slideDown()方法时，这个方法将由上至下延伸显示。slideUp()方法正好相反，元素将由下到上缩短隐藏。

注意：jQuery中的所有动画效果都可以指定3种速度参数，即slow，normal和fast(时间长度分别是0.6秒，0.4秒，0.2秒)。当使用速度关键字是要加引号，例如show(“fast”)

### 4.2.4 自定义动画方法animate()

animate(params,speed,callback);

参数说明如下：

1. params：一个包含样式属性及值的映射，比如{property1:”value1”,property2:”value2”,property3:”value3”,……}。
2. speed:速度参数，可选
3. callback:在动画完成时执行的函数，可选。
4. 自定义简单动画

使一个带有样式的<div>元素，通过单击的方式在3000ms内向右移动500像素。代码如下：

$(function(){

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

$(this).animate({left:"500px"},3000);

});

});

注意：在使用animate()方法之前，为了能影响该元素的”top”、”left”、”bottom”和”right”样式属性，必须先把元素的position属性设置为”relative”或者”absolute”。

1. 累加、累减动画

如果在上述”500px”之前加上”+=”或者”-=”符号即表示在当前位置累加或者累减。代码如下：

$(“this”).animate({left:”+=500px”},3000);

1. 多重动画
2. 同时执行多个动画

改动上诉例子，在元素移动的同时放大的元素的高度。代码如下：

$(this).animate({left:”500px”,height:”200px”},3000);

1. 按顺序执行多个动画

上述中，两个动画是同时发生的，如果想要先向右移动，然后再增大高度，只需要把代码拆开，然后按照顺序写就可以了。

$(this).animate({left:”500px”},3000);

$(this).animate({height:”200px”},3000);

因为animate()方法都是对同一个jQuery对象进行的操作，所以也可以改为链式的写法，

代码如下：

$(this).animate({left:”500px”},3000).animate({height:”200px”},3000);

像这样动画的执行具有先后顺序，成为”动画队列”。

1. 综合动画

单击<div>元素后让它从左到右移动的同时增大它的高度，并将他的不透明度从50%增加到100%，然后让他从上到下移动，同时他的宽度变大，当完成这些操作后，让它以淡出的方式隐藏。代码如下：

### 4.2.5 动画回调函数

在上例中要在最后改变元素的css样式，css(“border”,”5px solid blue”);

如果只是用常规的方法将fadeOut(“slow”)改为css(“border”,”5px solid blue”),这样不能达到预期的效果，实际的效果是，刚开始执行动画的时候，css()方法就被执行了。

出现这个问题的原因是,css()方法并不会加入到动画队列中，而是立即执行。可以使用回调函数(callback)对非动画方法实现排队。

只要把css()方法写在最后一个动画的回调函数里即可。代码如下：

$(this).animate({left:"500px",height:"200px",opacity:"1"},3000).animate({top:"200px",width:"200px"},3000,function(){

$(this).css("border","5px solid blue");

});

callback回调函数适用于jQuery所有动画效果方法，例如slideDown()方法的回调函数：

$(“#element”).slideDown(“normal”,function(){//在效果完成后做其他的事情});

### 4.2.6 停止动画和判断是否处于动画状态

1、停止元素的动画

对于上述例子，如果需要在某处停止动画，需要使用stop()方法。语法结构如下：

stop([clearQueue][,gotoEnd]);

两个参数都是可选参数，为Boolean值(true或者false)。clearQueue代表是否要清空未执行完的动画队列，gotoEnd代表是否直接将正在执行的动画跳转到末状态。

如果直接使用stop()方法，则会立即停止当前正在进行的动画，如果接下来还有动画等待继续进行，则以当前状态开始接下来的动画。代码如下：

$("#panel").hover(function(){

$(this).stop().animate({height:"150px",width:"300px"},200);

},function(){

$(this).stop().animate({height:"22",width:"60"},300);

});

例如先移入后移出，在移出时就调用stop()方法结束上面的移入动画。

如果遇到组合动画，例如：

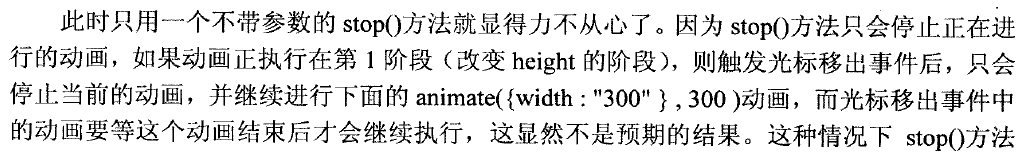
$("#panel").hover(function(){

$(this).stop().animate({height:"150px",width:"300px"},200).animate({width:"300"},300);

},function(){

$(this).stop().animate({height:"22",width:"60"},300).animate({width:"60"},300);

});

这种情况下，就把stop()方法的第一个参数(clearQueue)设置为true，此时程序就把当前程序接下来尚未执行完的动画队列都清空。

第二个参数(gotoEnd)可以让正在执行的动画直接到达结束时刻的状态，通常用于后一个动画需要基于前一个动画的末状态的情况。

注意：jQuery只能设置正在执行的动画的最终状态，而没有提供直接到达未执行动画队列最终状态的方法。

2、判断元素是否处于动画状态

if(!$(element).is(“:animated”)){

//如果当前没有进行动画，则添加新动画

}

### 4.2.7 其他动画方法

1、toggle()方法

可以切换可见状态。如果是可见的，则隐藏；如果是隐藏的，则切换为可见。

2、slideToggle()方法

通过改变高度来使内容在可见和隐藏之间切换。

1. fadeTo()方法

可以把元素的不透明度以渐进的方式调整到指定的值。

$(this).next(“div.content”).fadeTo(600,0.2);

# 第5章 jQuery对表单、表格的操作及更多应用

## 5.1 表单应用

### 5.1.1 单行文本框应用

获取和失去焦点改变样式

$(":input").focus(function(){

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

}).blur(function(){

$(this).removeClass("focus");

});

### 5.1.2 多行文本框应用

1、高度变化

2、滚动条高度变化

### 5.1.3 复选框应用

# 第6章 jQuery与Ajax的应用

## 6.5 jQuery中的Ajax

$.ajax()方法属于最底层方法，第2层是load()、$.get()和$.post()方法，第3层是$.getScript()和$.getJSON()方法。

### 6.5.1 load()方法

1、载入html文档

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

|  |  |  |
| --- | --- | --- |
| 参数名称 | 类型 | 说明 |
| url | String | 请求html页面的url地址 |
| data(可选) | Object | 发送至服务器的key/value数据 |
| callback(可选) | Function | 请求完成时的回调函数，无论请求成功或失败 |

代码如下:

<div id=”resText”></div>

$(function(){

$(“#send”).click(function(){

$(“#resText”).load(“text.html”);

});

});

2、可以从要加载的html文档中选择性的加载一些内容，load()方法的URL参数的语法结构为：”url selector”。注意URL和选择器之间有一个空格。

$(“#resText”).load(“text.html .para”);//选择text.html中的类为para的内容。

1. 传递方式

load()如果没有参数传递，则采用GET方式传递；反之，则会自动转换为POST方式。

无参数方式load(“text.php”,function(){});

有参数方式load(“text.php”,{name:”rain”,age:”22”},function(){});

1. 回调参数

回调函数有3个参数，分别代表请求返回的内容、请求状态和XMLHttpRequest对象。代码如下：

$(“#resText”).load(“test.php”,function(responseText,textStatus,XMLHttpRequest){

//responseText:请求返回的内容

//textStatus:请求状态：success、error、notmodified、timeout4中

//XMLHttpRequest:XMLHttpRequest对象

});

注意：在load()中，无论Ajax请求是否成功，只要当请求完成后(complete)后，回调函数(callback)就被触发。对应下面将介绍的$.ajax()方法中的complete回调函数。

### 6.5.2 $.get()方法和$.post()方法

load()方法通常用来从Web服务器上获取静态数据文件。在项目中，如果需要传递一些参数给服务器页面，那么可以使用$.get()或者$.post()方法(或者是$.ajax()方法)。

注意：$.get()和$.post()方法时jQuery中的全局函数。

1、$.get()方法

$.get()方法使用GET方式来进行异步请求。

结构为：$.get(url[,data][,callback][,type])

①使用参数

|  |  |  |
| --- | --- | --- |
| 参数名称 | 类型 | 说明 |
| url | String | 请求的HTML的url地址 |
| data | Object | 发送到服务器的key/value数据会作为QueryString附加到请求的url中 |
| callback | Function | 载入成功是回调函数(只有当response返回状态是success才调用该方法)自动将请求结果和状态传递给该方法 |
| type | String | 服务器端返回内容的格式，包括xml、html、script、json、text和\_default |

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

$.get("test.php",{

username:$("#username").val(),

content:$("#content").val()

},回调函数);

});

回调函数只有两个参数：

function(data,textStatus){

//data： 返回的内容，可以是xml文档，json文件，html片段等等

//textStatus：请求状态:success,error,notmodified,timeout4种

}

回调函数只有在成功返回success后才才被调用，这点与load()方法不一样。

1. 数据格式
2. html片段

并不需要对html片段进行处理就可以将数据加到主页中。

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

$.get("test.php",{

username:$("#username").val(),

content:$("#content").val()

},function(data,textStatus){

$("#resText").html(data);

});

});

1. xml文档

这里对数据要进行处理，但是处理方法和html文档一样。也可以使用常规的attr(),find(),filter()和其他方法。

1. JSON文件

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

$.get("test.php",{

username:$("#username").val(),

content:$("#content").val()

},function(data,textStatus){

var username=data.username;

var content=data.content;

var txtHTML="<div class='comment'><h6>"

+username+"</h6><p class='para'>"

+content+"</p></div>";

$("#resText").html(txtHTML);

},"json");

});

第4个参数type设置为json来代表期待服务器端返回的数据格式。

A、在不需要与其他应用程序共享数据的时候，使用html片段来返回数据一般是最简单的。

B、如果数据文件需要重用，那么JSON文件是不错的选择，他在性能和文件大小方面具有优势。

C、当远程应用程序未知时，xml文件是最好的选择，他是web服务领域的”世界语”。

2、$.post()方法

他与$.get()方法的区别：略

### 6.5.3 $.getScript()方法和$.getJson()方法

1、$.getScript()

本方法可以直接加载js文件，跟加载html文件一样方便，而且不需要对JavaScript文件进行处理，JavaScript文件会自动执行。

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

$.getScript("test.js");

});

本方法也有回调函数，他会在JavaScript文件成功载入后运行，代码如下：

$.getScript("H:\jquery.color.js",function(){

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

$(".block").animate({backgroundColor:'red'},1000).animate({backgroundColor:'blue'},1000);

});

});

2、$.getJSON()方法

本方法用于加载json文件，与$.getScript()方法语法相同，且同样有回调函数。

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

$.getJSON("test.json",function(data){

//data:返回的数据

});

});

jQuery提供了一个通用的遍历方法$.each()，可以用于遍历对象和数组。第1个参数是对象或者数组，第2个参数是回调函数。回调函数拥有两个参数，第1个为对象的成员或者数组的索引，第2个为对应变量或内容。代码如下：

### 6.5.4 $.ajax()方法

该方法是jQuery最底层ajax实现，结构为$.ajax(options)，该方法只有一个参数，但在这个对象里包含了$.ajax()方法所需要的请求设置以及回调函数等信息，参数以key/value的形式存在，所有参数都是可选的。代码如下：

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

$.ajax({

type:"GET",

url:"test.js",

dataType:"script"

});

});

## 6.6 序列化元素

1、serialize()方法

该方法能将DOM元素内容序列化为字符串，例如将表单的内容序列化，代码如下：

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

$.get("test.php",$("#form1").serialize(),function(data,textStatus){$(“#resText”).html(data);});

});

即使在表单中再增加字段，脚本仍然能够使用，而且不需要做其他多余工作。

2、serializeArray()方法

该方法不是返回字符串，而是将DOM元素序列化后，返回JSON形式的数据

3、$.param()方法

用来对一个数组或对象进行key/value形式的序列化。比如将一个普通的对象序列化。

var obj={a:1,b:2,c:3};

var k=$.param(obj);

alert(k);//输出a=1&b=2&c=3

## 6.7 jQuery中的ajax全局事件

Ajax开始时会触发ajaxStart()方法的回调函数，当请求结束时，会触发ajaxStop()方法的回调函数。