从41页开始记录

1。无需判断网页中是否有此元素就可直接对此元素操作，jQuery不会提示错误

2。$('#tt')获取的永远是对象，不能只使用if( $('#tt') ),可以使用if( $(#'tt') ).length > 0来检查此元素是否存在。或转化成DOM对象来判断，if ( $('#tt')[0] )

3。在写隔行换色时，没写function（）函数，导致javascript不起作用，一定要注意

4。jQuery选择器分为基本选择器，层次选择器，过滤选择器和表单选择器

5。获取id 为cr的jQuery对象，var jqobj = $("#cr");

获取DOM对象，var domobj = document.getElementById("cr");

6。由jQuery对象转换成DOM对象，var domobj = jqobj[0]，或domobj = jqobj.get(0)

由DOM对象转换成jQuery对象，var jqobj = $(domobj);

7。在网页中，每个id名称只能使用一次，class允许重复使用

8。基本选择器有

#id。根据id匹配元素，返回单个元素

.class。根据类名匹配元素，返回集合元素

element。根据元素名匹配元素，返回集合元素

\*。匹配所有元素，返回集合元素

selector1,selector2...,selectorN。将每个选择器匹配到的元素合并后一起返回，返回集合元素

如$("#test")选取id为test的元素；$(".test")，选取class为test的元素；$("p")，选取所有的<p>标签；$("\*")，选取所有的元素；$("div,span,p.myClass")选取所有的<div>，<span>及class为myClass的<p>元素

9。子元素与后代元素的区别：如爷爷，爸爸，自己。爸爸/叔叔/伯伯是爷爷的子元素，而自己是爷爷的后代元素而非子元素。

10。层次选择器：如果想通过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，相邻元素和同辈元素等。可用这个。

11。层次选择器有：

$("ancestor descendant")。选取ancestor元素的所有后代元素，返回集合元素

$("parent > child")。选取parent元素下的子元素，返回集合元素

$("prev + next")。选取prev元素后的下一个同辈元素，返回集合元素

$("prev ~ siblings")。选取prev元素后的所有同辈元素，返回集合元素

如$("div span")，选取<div>里的所有的<span>元素；$("div > span")，选取<div>元素下是<span>的子元素；$(".one + div")选取class为one的下一个<div>同辈元素；$("#two ~ div")选取id为two元素后面的所有<div>同辈元素。

前两个比较常用，后两个因为在jQuery里可以用更加简单的方法代替，所有使用的几率相对少些。

12.如使用next()方法代替$("prev + next")选择器。$(".one + div")等价于$(".one").next("div")

13.如使用nextAll()方法代替$("prev ~ siblings")选择器。$("#two ~ div")等价于$("#two").nextAll("div")。

14.相邻元素和同辈元素，结果都不返回自己。基本选择器（除了id）和层次选择器都是返回集合元素。

15.过滤选择器：主要是通过特定的过滤规则来筛选出所需的DOM元素，过滤规则与CSS中的伪类选择器语法相同，即选择器都以一个冒号开头。按照不同的过滤规则，过滤选择器可以分为基本过滤，内容过滤，可见性过滤，属性过滤，子元素过滤和表单对象属性过滤选择器。

15.1基本过滤选择器

:first,选取第一个元素，返回单个元素

:last,选取最后一个元素，返回单个元素

:not(selector),去除所有与给定选择器匹配的元素，返回集合元素

:even,选取索引是偶数的所有元素，索引从0开始，返回集合元素

:odd, 选取索引是奇数的所有元素，索引从0开始，返回集合元素

:eq(index),选取索引等于index的元素（index从0开始），返回单个元素

:gt(index), 选取索引大于index的元素（index从0开始），返回集合元素

:lt(index), 选取索引小于index的元素（index从0开始），返回集合元素

:header,选取所有的标题元素，例如h1，h2，h3等等，返回集合元素

:animated,选取当前正在执行动画的所有元素，返回集合元素

:focus,选取当前获取焦点的元素，返回集合元素

15.2 内容过滤选择器，过滤规则体现在它所包含的子元素或文本内容上

:contains(text),选取含有文本内容为"text"的因素，返回集合元素

:empty,选取不包含子元素或者文本的空元素，返回集合元素

:has(selector),选取含有选择器匹配元素的元素，返回集合元素

:parent,选取含有子元素或者文本的元素，返回集合元素

15.3 可见性过滤选择器，根据元素的可见和不可见状态来选择相应的元素

:hidden,选取所有不可见的元素，返回集合元素。如$(":hidden")选取所有不可见的元素。包括<input type="hidden"/>,<div style="display:none;">和<div style="visibility:hidden;">等元素。如果只想选取<input>元素，可以使用$("input:hidden")

:visible,选取所有可见的元素。

15.4 属性过滤选择器，过滤规则是通过元素的属性来获取相应的元素

[attribute],选取拥有此属性的元素，返回集合元素。如id，title等属性

[attribute=value],选取属性的值为value的元素，返回集合元素

[attribute!=value],选取属性的值不等于value的元素，返回集合元素

[attribute^=value],选取属性的值以value开始的元素，返回集合元素

[attribute$=value],选取属性的值以value结束的元素，返回集合元素

[attribute\*=value],选取属性的值含有value的元素，返回集合元素

[attribute|=value],选取属性等于给定字符串或以该字符串为前缀（该字符串后跟一个连字符“-”）的元素，返回集合元素

[attribute~=value],选取属性用空格分隔的值中包含一个给定值的元素，返回集合元素

[attribute1][attribute2][attributeN],用属性选择器合并成一个符合属性选择器，满足多个条件。每选择一次，缩小一次范围。返回集合元素

如：$("div[id]")选取拥有属性id的元素。$("div[title=test]")选取属性title为”test”的<div>元素。$("div[title!=test]")选取属性title不等于“test”的<div>元素（注意：没有属性title的<div>元素也会被选取）。$("div[title^=test]")选取属性title以“test”开始的<div>元素。$("div[title$=test]")，选取属性title以“test”结束的<div>元素。$("div[title\*=test]")选取属性title含有“test”的<div>元素。$("div[title|='en']")选取属性title等于en或以en为前缀（该字符串后跟一个连字符'-'）的元素。$("div[title~='uk']")选取属性title用空格分隔的值中包含字符uk的元素。$("div[id][title$='test']")选取拥有属性id，并且属性title以"test"结束的<div>元素。

15.5 子元素过滤选择器:过滤规则相对其他的选择器稍微复杂，前提只要将父元素和子元素区分清楚即可。另外要注意它与普通过滤选择器的区别。另，若没有指定子元素的类型，则必须在:nth-child等前必须加空格，如$("div.one :first-child")

:nth-child(index/even/odd/equation),选取每个父元素下的第index个子元素或者奇偶元素(index从1算起），返回集合元素。:eq(index)只匹配一个元素，而:nth-child将为每一个父元素匹配子元素，并且:nth-child(index)的index是从1开始的，而:eq(index)是从0算起。例如：

:nth-child(even)能选取每个父元素下的索引值是偶数的元素。

:nth-child(odd)能选取每个父元素下的索引值是奇数的元素。

:nth-child(2)能选取每个父元素下的索引值等于2的元素。

:nth-child(3n)能选取每个父元素下的索引值是3的倍数的元素，（n从1开始）。

:nth-child(3n+1)能选取每个父元素下的索引值是（3n+1）的元素(n从1开始）

:first-child,选取每个父元素的第1个子元素，返回集合元素。:first只返回单个元素，而:first-child选择符将为每个父元素匹配第1个子元素。例如$("ul li:first-child");选取每个<ul>中第1个<li>元素。

:last-child,选取每个父元素的最后一个子元素，返回集合元素。同样，:last只返回单个元素，而:last-child选择符将为每个父元素匹配最后一个子元素。例如：$("ul li:last-child");选择每个<ul>中最后一个<li>元素。

:only-child,如果某个元素是它父元素中唯一的子元素，那么将会被分配。如果父元素中含有其他元素，则不会被分配，返回集合元素。例如：$("ul li:only-child")在<ul>中选取是唯一子元素的<li>元素。

15.6 表单对象属性过滤选择器，对所选择的表单元素进行过滤，例如选择被选中的下拉框，多选框等元素。

:enabled,选取所有可用元素，返回集合元素。如$("#form1 :enabled"),选取id为form1的表单内的所有可用元素

:disabled,选取所有不可用元素，返回集合元素。如$("#form2 :disabled")选取id为form2的表单内的所有不可用元素

:checked,选取所有被选中的元素（单选框，复选框），返回集合元素。如$("input:checked"),选取所有被选中的<input>元素

:selected,选取所有被选中的选项元素（下拉列表），返回集合运算。如$("select option:selected"),选取所有被选中的选项元素

16.表单选择器：为了使用户能够更加灵活地操作表单，jQuery中专门加入了表单选择器。利用这个选择器，能极其方便地获取到表单的某个或某类型的元素。

:input,选取所有的<input>,<textarea>,<select>和<button>元素，返回集合元素。//注意与$("#form1 input")和$("#form1 :input")的区别。

:text,选取所有的单行文本框，返回集合元素。

:password,选取所有的密码框，返回集合元素。

:radio,选取所有的单选框，返回集合元素。

:checkbox,选取所有的多选框，返回集合元素。

:submit,选取所有的提交按钮，，返回集合元素。

:image,选取所有的图像按钮，返回集合元素。

:reset,选取所有的重置按钮，返回集合元素。

:button,选取所有的按钮，返回集合元素。

:file,选取所有的上传域，返回集合元素。

:hidden,选取所有不可见元素，返回集合元素。

2.5 选择器中的一些注意事项

2.5.1 选择器中含有特殊符号的注意事项

1.选择器中含有"·","#","("或"]"等特殊字符

如<did id="id#b">bb</div> <div id="id[1]">cc</div>

应这样获取：使用转义特殊字符"#",$("#id\\#b");使用转义特殊字符"[]",$("#id\\[1\\]");

2.对于选择时是否使用空格，

$(".test :hidden"),带空格，选取class为test的元素里面的隐藏元素。

$(".test:hidden"),不带空格，选取隐藏的class为test的元素

**第三章 DOM操作的分类**

DOM是Document Object Model的缩写，文档对象模型。根据W3C DOM规范（http://www.w3.org/DOM)，DOM是一种与浏览器/平台/语言无关的接口，使用该接口可以轻松地访问页面中所有的标准组件。

1.DOM操作分为3个方面，即DOM Core（核心），HTML-DOM和CSS-DOM

2.在DOM操作中，常常需要动态创建HTML内容，使文档在浏览器里的呈现效果发生变化，并且达到各种各样的人机交互的目的。

３.动态创建的新元素节点不会被自动添加到文档中，而是需要使用其他方法将其插入文档中，如.append()方法。当创建单个元素时，要注意闭合标签和使用标准的XHTML格式。例如创建一个<p>元素，可以用$("<p/>")或者$("<p></p>")，但不要使用$("<p>")或者大写的$("<P/>").

4.将新创建的节点插入某个文档的方法并非只有append一种，如想把"<p>我想说：</p>"营造成"<p>我想说：<b>你好</b></p>"的效果，可使用以下方法：

append(); $("p").append("<b>你好</b>");

appendTo(); $("<b>你好</b>").appendTo("p");

prepend(); $("<b>你好</b>").prepend("p");

prependTo(); $("p").prependTo("<b>你好</b>");

after(); $("p").after("<b>你好</b>");

insertAfter(); $("<b>你好</b>").insertAfter("p");

before(); $("<b>你好</b>").before("p");

insertBefore(); $("p").insertBefore("<b>你好</b>");

5.删除节点：

remove()方法，从DOM中删除所有匹配的元素，传入的参数用于根据jQuery表达式来筛选元素。当某个节点用remove()方法删除后，该节点所包含的所有后代节点将同时被删除。这个方法的返回值是一个指向已被删除的节点的引用，因此可以在以后再使用这些元素。

detach()方法，detach()和remove()一样，也是从DOM中去掉所有匹配的元素。但需要注意的是，这个方法不会把匹配的元素从jQuery对象中删除，因而可以在将来再使用这些匹配的元素。与remove()不同的是，所有绑定的事件，附加的数据等都会保留下来。

empty()方法，严格来讲，empty()方法并不是删除节点，而是**清空**节点，它能清空元素中的所有后代节点。如用此方法操作<li>元素，则<li>元素里的内容被清空，但<li>元素的框架还在。

6.复制节点

复制节点也是常用的DOM操作之一，使用clone()函数可以复制，但要注意，这样复制的新元素并不具有任何行为:即没有原节点的任何绑定事件；要想复制的节点也有原节点的事件，需要使用clone(true)，即传递一个参数true。

7.替换节点

replaceWith()方法的作用是将所有匹配的因素都替换成指定的HTML或者DOM元素。

replaceAll()方法的作用与replaceWith()方法相同，只是颠倒了replachWith()的参数，如将<p title="Which fruit do you like?">你最喜欢哪种水果?</p>替换成<strong>你最不喜欢的水果是？</strong>:

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

或者 $("<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因素一起消失，需要在新元素上重新绑定事件。

8.包裹节点

如果要将某个节点用其他标记包裹起来，jQuery提供了相应的方法，即wrap()。该方法对于需要在文档中插入额外的结构化标记非常有用，而且它不会破坏原始文档的语义。

如：$("strong").warp("<b></b>");//用<b>标签把<strong>元素包裹起来

得到结果为:<b><strong title="Which fruit do you like?">你最喜欢的水果是？</strong></b>

包裹节点还有两个方法：wrapAll()和wrapInner();

wrapAll():该方法会将所有匹配的元素用一个元素来包裹，它会改变元素的位置。它不同于wrap()方法，wrap()方法是将所有的元素进行单独的包裹

wrapInner()：该方法将每一个匹配的元素的子内容（包括文本节点）用其他结构化的标记包裹起来。

如：$("strong").wrapInner("<b></b>");

结果如下：<strong title="Which fruit do you like?"><b>你最喜欢的水果是？</b></strong>

9.属性操作

jQuery中，用attr()方法来获取和设置元素属性，removeAttr()方法来删除元素属性。

9.1 获取属性和设置属性

如果要获取<p>元素的属性title，那么只需要给attr()方法传递一个参数，即属性名称。如：var txt = $("p").attr("title");

如果要设置<p>元素的属性title的值，也可以使用同一个方法，不同的是，需要传递两个参数，即属性名称和对应的值。如：$("p").attr("title","your title");

如果需要一次性为同一个元素设置多个属性，可以使用下面的代码来实现：

$("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。

注：jQuery中的很多方法都是同一个函数实现获取(getter)和设置(setter)的，例如上面的attr()方法，既能设置元素属性的值，也能获取元素属性的值。类似的还有html(),text(),height(),width(),val()和css()等方法。

9.2 删除属性

在某些情况下，需要删除文档中某个元素的特定属性，可以使用removeAttr()方法来完成该任务。如：$("p").removeAttr("title");//删除p元素的title属性

注：jQuery1.6版本新增了prop()和removeProp()函数,它和attr()和removeAttr()的参数和用法几乎相同，但jQuery中，attribute和property是两个不同的概念。attribute表示HTML文档节点的属性（如常用的id，name，class，title等），property表示JS对象的属性（这里的className和attribute的class等价，因为class是ECMA里的关键字）。prop()函数的设计目标是用于设置或获取指定DOM元素（指的是JS对象，Element类型）上的property；attr()函数的设计目标是用于设置或获取指定DOM元素所对应的文档节点上的attribute。在处理DOM元素的本地属性checked,selected,disabled等时，attr ()认为是初始状态，prop()认为是当前状态。且在jQuery1.6以后，prop()方法返回标准属性：true/false;

10. 样式操作

10.1 获取样式和设置样式

在<p class="myClass" title="Which fruit do you like?">你最喜欢的水果是？</p>这样的代码中，class是<p>元素的属性，因此获取class和设置class都可以使用attr()方法来完成。如：var p\_class = $("p").attr("class");//获取<p>元素的class。

$("p").attr("class","high");//设置<p>元素的class为"high"

10.2 追加样式

上面最后一行代码是将原来的class(myClass)替换为新的class(high)。如果此处需要的是“追加”效果，class属性变为“myClass high”，即myClass和high两种样式的叠加，那么使用addClass()方法。如：$("p").addClass("another");//给p元素追加another类。

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设定了同一样式属性，则后者覆盖前者

10.3 移除样式

删除样式可以用removeAttr("class")来移除全部的class类，也可使用removeClass()来清空class的值：

移除一个：removeClass("myClass");

移除多个：removeClass("myClass").removeClass("another");

或者：removeClass("myClass another");//用空格隔开

10.4 切换样式

toggle方法在1.9以下版本才能使用，之后的版本已经不支持了。但还支持toggleClass()方法。它用例控制样式上的重复切换，如果类名存在则删除它，如果类名不存在则添加它。如：$("p").toggleClass("another");它要么直接删除another，要么在其他class后面追加another

10.5 判断是否含有某个样式

hasClass()可以用来判断元素中是否含有某个class，如果有，则返回true，否则返回false。如：$("p").hasClass("another");

其实此函数来自于is()方法，它跟.is(".another")作用相同。

11. 设置和获取HTML

html()方法类似于JavaScript中的innerHTML属性，可以用来读取或者设置某个元素中的HTML内容。如代码：<p title="your title"><strong>你最喜欢的水果是？</strong></p>

运用html()方法:alert($("p").html());结果是 <strong>你最喜欢的水果是？</strong>

注：html()方法可以用于XHTML文档，但不能用于XML文档。

12.设置和获取文本text

text()方法类似于JS中的innerText属性，可以用来读取或设置某个元素的文本内容。

还用上面的代码，运用text()方法：alert($("p").text());//结果是 你最喜欢的水果是？

与html()方法一样，如果需要为某元素设置文本内容，也需要传递一个参数，如：

$("p").text("改变改变成你最喜欢的水果是？");

注：(1) JS中的innerText属性不能再Firefox浏览器下运行，而jQuery的text()方法支持所有浏览器；(2) text()方法对HTML文档和XML文档都有效。

13. 设置和获取值value

val()方法类似于JS中的value属性，可以用来设置和获取元素的值。无论元素是文本框，下拉列表还是单选框，它都可以返回元素的值。如果元素为多选，则返回一个包含所有选择的值的数组。

所以此方法还可以使单选下拉框，多选下拉框，单选框，多选框中相应的选项被选中。

14.遍历节点

14.1 **children()**方法

该方法用于取得匹配元素的**子元素集合**。不考虑其他后代元素。添加可选参数可通过选择器进行过滤。

14.2 next()方法

该方法用于取得匹配元素后面紧邻的同辈元素。如果提供选择器，则取回匹配该选择器的下一个同辈元素。

14.3 prev()方法

该方法用于取得匹配元素前面相邻的同辈元素，通过选择器进行筛选是可选的。

14.4 siblings()方法

该方法用于取得匹配元素前后所有的同辈元素，通过选择器进行筛选是可选的。

14.5 **closest()**方法

该方法用于取得最近的匹配的**当前或匹配的第一个祖先元素**。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。

14.6 **find()**方法

find()方法获得当前元素集合中每个元素的**后代元素**，通过选择器，jQuery对象或元素来筛选。

14.7 **parent()**方法

parent()方法获得当前匹配元素集合中每个元素的**父**元素，使用选择器进行筛选是可选的。

14.8 **parents()**方法

parents()方法获得当前匹配元素集合中每个元素的**祖先**元素，使用选择器进行筛选是可选的。

14.9 filter()方法

filter()方法将匹配元素集合缩减为匹配指定选择器的元素。

14.10 nextAll()方法

nextAll()获得匹配元素集合中每个元素的所有跟随的同辈元素，由选择器筛选是可选的。

14.11 prevAll()方法

prevAll()获得当前匹配元素集合中每个元素前面的同辈元素，使用选择器进行筛选是可选的。

15. CSS-DOM 操作

CSS-DOM技术简单来说就是读取和设置style对象的各种属性。style属性很有用，但最大不足是无法通过它来提取到通过外部CSS设置的样式信息，然而在jQuery中，这些都是非常的简单。

15.1 直接利用css()方法获取元素的样式属性，如：$("p").css("color");//获取<p>元素的样式颜色。在这里，无论color属性是外部CSS导入，还是直接拼接在HTML元素里（内联），css()方法都可以获取到属性style里的其他属性的值。再如：$("p").css("height");

15.2 直接利用css()方法设置某个元素的单个或多个样式，如：$("p").css("color","red");//设置<p>元素的样式颜色为红色。再如：$("p").css({"fontSize":"30px","backgroundColor":"#888888"});//同时设置字体大小和背景色。再如：$("p").css("opacity","0.5");//将p元素的不透明度设置为半透明。再如：$("p").height(100);//设置<p>元素的高度值为100px;

注：（1）如果值是数字，将会被自动转化为像素值；(2) 在css()方法中，如果属性中带有"-"符号，如font-size和background-color属性，若在设置这些属性的值的时候不带引号，那么就要用驼峰式写法，如：$("p").css({fontSize:"30px",backgroundColor:"#888888"}).即用DOM名称而非CSS名称。

15.3 .css("height")和.height()的区别

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

(2) 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()方法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，并且不带单位。

15.4 offset()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。

15.5 position()方法

position()方法返回匹配元素相对于父元素的位置（偏移）；该方法返回的对象包含两个整形属性：top和left，以像素计；此方法只对可见元素有效。

15.6 scrollTop()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左侧的距离，如：$("p").scrollTop();另外，可以为这两个方法指定一个参数，控制元素的滚动条滚动到指定位置。如：$("p").scrollTop(200).

15.7 超链接提示效果

现在只有在超链接中加入title属性就可看到超链接提示效果。但是这个提示效果的响应速度是非常缓慢的。最好自己做个类似功能的提示：有几个要点1.mouseover时，要抽取title，附加div，定位显示div（不要与光标同坐标）；2.mouseout时，要恢复title，删除div ；3.mousemove时，div的位置也要随之移动。

介绍下js的一个方法三元运算符：它需要三个操作数，语法是 条件？结果1：结果2;这里把条件写在问号(?)的前面，后面跟着用冒号(:)分隔的结果1和结果2.满足条件时结果1否则结果2. 注意：第一个参数必须是布尔值。

介绍下css的alt属性，1.alt属性只能用在img，area和input元素（包括applet元素）；2.使用alt属性是为了给那些不能看到你文档中图像的浏览者提供文字说明。这包括那些使用本来就不支持图像显示或者图像显示被关闭的浏览器的用户，视觉障碍的用户和使用屏幕阅读器的用户。替换文字是用来替代图像而不是提供额外说明文字的。

**第4章 jQuery中的事件和动画**

JavaScript和HTML之间的交互是通过用户和浏览器操作页面时引发的事件来处理的。当文档或者它的某些元素发生某些变化或操作时，浏览器会自动生成一个事件。虽然利用传统的JavaScript事件能够完成这些交互，但jQuery增加并扩展了基本的事件处理机制。jQuery不仅提供了更加优雅的事件处理语法，而且极大地增强了事件处理能力。

一 jQuery中的事件

1. 加载DOM。window.onload和$(document).ready()方法功能类似，后者可以极大地提高Web应用程序的响应速度。比较如下：

1.1 执行时机

window.onload方法是在网页中所有的元素（包括元素的所有关联文件）完全加载到浏览器后才执行，即JavaScript此时才可以访问网页中的任何元素。而通过jQuery中的$(document).ready()方法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，这并不意味着这些元素关联的文件都已经下载完毕。

但也许未加载完的图片的属性（如高度和宽度）此时不一定有效，解决此问题，可使用jQuery中另一个关于页面加载的方法--load()方法。load()方法会在元素的onload事件中绑定一个处理函数。如果处理函数绑定给window对象，则会在所有内容（包括窗口，框架，对象和图像等）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。如$(window).load(function(){

//编写代码

})

它等同于JS中的onload。

1.2 多次使用

JS的window.onload事件一次只能保存对一个函数的引用，它会自动用后面的函数覆盖前面的函数。而jQuery每次调用$(document).ready()方法都会在现有的行为上追加新的行为，这些行为函数会根据注册的顺序依次执行。

1.3 简写方式

$(function(){

//编写代码

})是

$(document).ready(function(){

//编写代码

})的简写

同时，$(document)也可以简写为$()。当$()不带参数时，默认参数就是“document”，因此又可以简写为$().ready(function(){

//编写代码

})

2.事件绑定

在文档装载完成后，若打算为元素绑定事件来完成某些操作，则可以使用bind()方法来匹配元素进行特定事件的绑定，bind()方法的调用格式为：bind(type[,data],fn);

这三个参数：

第1个参数是事件类型，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，当然也可以自定义名称。

第2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。

第3个参数是用来绑定的处理函数。

注：jQuery中的事件绑定类型比普通的JavaScript事件绑定类型少了“on”。例如鼠标单击事件在jQuery中对应的是click()方法，而在JavaScript中对应的是onclick().

2.1 例子：$(".head").bind("click",function(){$(this).next().show();});//在这里，绑定的是class为head元素的点击事件，点击后它后面的同辈元素就显示。this引用的是携带相应行为的DOM元素，为了使该DOM元素能够使用jQuery中的方法，可以使用$(this)将其转换为jQuery对象。

2.2 当发现相同的选择器在代码里反复出现，用变量缓存起来，会优化jQuery性能。

2.3 简写绑定事件。像click，mouseover，mouseout这类事件，在程序中经常会使用到，jQuery提供了一套简写方法。如$(".head").bind("mouseover",function(){

}).bind("mouseout",function(){});

可以简写为：$(".head").mouseover(function(){

}).mouseout(function(){

});

3.合成事件

jQuery有两个合成事件----hover()方法和toggle()方法，类似前面讲过的ready()方法，hover()方法和toggle()方法都属于jQuery自定义的方法。

3.1 hover()方法

hover()方法的语法结构为：hover(enter,leave);

hover()方法用于模拟光标悬停事件。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）。如更改上面的例子。$(".head").hover(function(){

},function(){

})

3.2 mouseover()和mouseout()组合 与 mouseenter()和mouseleave()组合的区别

mouseover()和mouseout()组合，是把父元素与子元素看成一个个的个体，只要有移动就触发。mouseenter()和mouseleave()组合，是把父元素和子元素捆绑在一起，看成一个整体，只有当光标全部移入或移出时才会触发事件

3.3 hover()其实是mouseenter()和mouseleave()的组合。

4. 事件冒泡

之所以称为冒泡，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。如：依次嵌套<body>,<div>,<span>元素，为三个元素都绑定click事件，则当点击<span>时，会出现三个事件。

4.1 事件对象

由于IE-DOM和标准DOM实现事件对象的方法各不相同，导致在不同浏览器中获取事件对象变得比较困难。jQuery进行了必要的扩展和封装，从而使得在任何浏览器中都能很轻松地获取事件对象以及事件对象的一些属性。

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，如：

$("element").bind("click",function(event){ //event:事件对象

//...

});

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

4.1.1 停止事件冒泡

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。

4.1.2 阻止默认行为

网页中的元素有自己默认的行为，例如：单击超链接后会跳转，单击“提交”按钮后表单会提交。有时需要阻止元素的默认行为，如点击“提交”按钮时，验证表单内容，某元素是否是必填字段，长度是否够6位等，当不符合提交条件时，要阻止表单的提交。在jQuery中，提供了preventDefault()方法来阻止元素的默认行为。

注：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。这是对在事件对象上同时调用stopPropagation()方法和preventDefault()方法的一种简写方式。

4.1.3 事件捕获

事件捕获和事件冒泡是刚好相反的两个过程，事件捕获是从最顶端往下开始触发。

遗憾的是，并非所有主流浏览器都支持事件捕获，并且这个缺陷无法通过JavaScript来修复。jQuery不支持事件捕获，若需使用，请直接使用原生的JavaScript。

4.2 事件对象的属性

jQuery在遵循W3C规范的情况下，对事件对象的常用属性进行了封装，使得事件处理在各大浏览器下都可以正常运行而不需要进行浏览器类型判断。

4.2.1 event.type

该属性的作用是可以获取到事件的类型。

4.2.2 event.preventDefault()方法

该方法的作用是阻止默认的事件行为。JavaScript中符合W3C规范的preventDefault()方法在IE浏览器中却无效。jQuery对其进行了封装，使之能兼容各种浏览器。

4.2.3 event.stopPropagation()方法

该方法的作用是阻止事件的冒泡。JavaScript中符合W3C规范的stopPropagation()方法在IE浏览器中却无效。jQuery对其进行了封装，使之能兼容各种浏览器。

4.2.4 event.target

event.target的作用是获取到触发事件的**元素**。jQuery对其封装后，避免了各个浏览器不同标准的差异。

4.2.5 event.relatedTarget

event.relatedTarget属性返回当鼠标移动时哪个元素进入或退出。对于mouseover()和mouseenter()返回鼠标刚刚掠过的元素，对于mouseout()和mouseleave()返回鼠标移向的元素。

4.2.6 event.pageX和event.pageY

该属性的作用是获取到光标相对于页面的x坐标和y坐标。如果没有使用jQuery时，在IE浏览器中是用event.x/event.y，而在Firefox中是用event.pageX/event.pageY。如果页面上有滚动条，则还要加上滚动条的宽度和高度。

4.2.7 event.which

该属性的作用是鼠标单击事件中获取到鼠标的左，中，右键，以1，2，3显示；在键盘事件中获取键盘的按键，以键位序号显示。

4.2.8 event.metaKey

针对不同浏览器对键盘中的<ctrl>按键解释不同，jQuery也进行了封装，并规定event.metaKey为键盘事件中获取<ctrl>按键，返回true或false。

4.3 移除事件

在绑定事件的过程中，不仅可以为同一个元素绑定多个事件，也可以为多个元素绑定同一个事件。

unbind([type],[data]);

第1个参数是事件类型，第2个参数是将要移除的函数，具体说明如下：

(1) 如果没有参数，则删除所有绑定的事件。

(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) 如果把在绑定时传递的处理函数作为第2个参数，则只有这个特定的事件处理函数会被删除。

4.3.1 移除以前注册的事件

unbind();//移除所有事件

unbind("click");//移除所有click事件

4.3.2 移除其中一个事件

unbind("click",myFun2);//删除指定事件，myFun2是函数名。

对于只需要触发一次，随后就要立即解除绑定的情况，jQuery提供了一种简写方法----one()方法。one()方法可以为元素绑定处理函数。当处理函数触发一次后，立即被删除。即在每个对象上，事件处理函数只会被执行一次。

one()方法的结构与bind()方法类似，使用方法也与bind()方法相同，其语法结构如下：

one(type,[data],fn);//如one("click",function(){//函数体;});

5. 模拟操作

在jQuery中，可以使用trigger()方法完成模拟操作。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，以数组形式传递。通常可以通过传递一个参数给回调函数来区别这次事件是代码触发的还是用户触发的。

5.1 常用模拟

以上的例子都是用户必须通过单击按钮，才能触发click事件，但是有时，需要通过模拟用户操作，来达到单击的效果。例如在用户进入页面后，就触发click事件，而不需要用户去主动单击。

如：$("#btn").trigger("click");//页面加载完毕后，即输出想要的效果

也可以直接用简化写法click()，来达到同样的效果：$("#btn").click();

5.2 触发自定义事件

如为元素绑定一个“myClick”事件，代码如下：

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

$("body").append("<p>我的自定义事件</p>");

});

触发此事件，可以用以下代码来实现：

$("#btn").trigger("myClick");

5.3 传递数据

给第2个参数赋值。如:

$("#btn").bind("myClick",function(event,msg1,msg2){

$("body").append("<p>"+msg1+msg2+"</p>");

})

触发此事件，

$("#btn").trigger("myClick",["自定义","传递参数"]);

5.4 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focus事件，而不想执行浏览器默认操作，可以使用jQuery中另一个类似的方法--------triggerHandler("focus");

该方法会触发<input>元素上绑定的特定事件，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，不会得到焦点。

6. bind()的其他用法

前面已经对bind()方法进行了介绍，bind()方法不仅能为元素绑定浏览器支持的具有相同名称的事件，也可以绑定自定义事件。除此之外，bind()方法还能做很多的事情。

6.1 绑定多个事件类型

可以为元素一次性绑定多个事件类型。如：

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

$(this).toggleClass("over");

});

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，这段代码等同于下面的代码：

$("div).mouseover(function(){

$(this).toggleClass("over"):

}).mouseout(funtion(){

$(this).toggleClass("over");

});

很显然，第1种方式能减少代码量，这就是jQuery提倡的“write less,do more"(写得更少，做得更多）理念。

6.2 添加事件命名空间，便于管理

例如可以把为元素绑定的多个事件类型用命名空间规范起来:

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

$("body").append("<p>click事件</p>");

}).bind("mouseover.plugin",function(){

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

}).bind("dblclick",function(){

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

});

然后移除带.plugin的事件

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

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

});

在所绑定的事件类型后面添加命名空间，这样在删除事件时只需要指定命名空间即可。单击<button>元素后，“plugin”的命名空间被删除，而不在“plugin”的命名空间的“dblclick”事件依然存在。

删除多个事件也可以写为以下链式代码：

$("div").unbind("mouseover").unbind("click");

6.3 根据命名空间调用

可以为元素绑定相同的事件类型，然后以命名空间的不同按需调用，如：

//单击div则同时调用两个click事件

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

$("body").append("<p>click事件</p>");

}).bind("click.plugin",function(){

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

});

//单击button，只触发click事件

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

$("div").trigger("click!");//注意！的使用

});

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，则只触发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。

如果需要两者都被触发，则改为：

$("div").trigger("click");//去掉感叹号即可

注：貌似当前jQuery不支持此功能了。

7. jQuery中的动画

动画效果也是jQuery库吸引人的地方。通过jQuery的动画方法，能够轻松地为网页添加非常精彩的视觉效果，给用户一种全新的体验。

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

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

7.1 show()方法和hide()方法

7.1.1 show()方法和hide()方法

show()方法和hide()方法是jQuery中最基本的动画方法，在HTML文档里，为一个元素调用hide()方法，会将该元素的display样式改为"none"。

如以下的两行，效果相同：

$("element").hide();//通过hide()方法隐藏元素。

$("element").css("display","none");//通过css()方法隐藏元素。

当把元素隐藏后，可以使用show()方法将元素的display样式设置为先前的显示状态（“block”或“inline”或其他除了“none”之外的值）。

注意hide()方法在将display属性值设置为“none”之前，会记住原先的display属性值。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。

7.1.2 show()方法和hide()方法让元素动起来

show()方法和hide()方法在不带任何参数的情况下，相当于css("display","none/block/inline"),作用是**立即**隐藏或显示匹配的元素，不会有任何动画。如果希望在调用show()方法时，元素慢慢地显示出来，可以为show()方法指定一个速度参数，例如，指定一个速度关键字"slow"。如：$("element").show("slow");//元素将在600毫秒内慢慢显示出来。其他的速度关键字还有"normal"和"fast"(长度分别是400毫秒和200毫秒）。

不仅如此，还可以为显示速度指定一个数字，单位是毫秒，如：

$("element").show(1000);//元素在1秒钟内显示出来。

从代码执行过程中可以发现，hide(600)方法会同时减少对象的高度/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”。同理，show(600)方法则会从上到下增大对象的高度，从左到右增大对象的宽度，同时增加对象的不透明度，直至完全显示。

7.2 fadeIn()方法和fadeOut()方法

fadeIn()方法和fadeOut()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，直至元素完全消失("display：none").fadeIn()方法则相反。

7.3 slideUp()方法和slideDown()方法

slideUp()方法和slideDown()方法只会改变元素的高度。如果一个元素的display属性值为“none”，当调用slideDown()方法时，这个元素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：jQuery中的任何动画效果，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秒和0.2秒），当不指定时，默认是“normal”。当使用速度关键字时要加引号，例如show("slow")，如果用数字作为时间参数时就不需要加引号，例如show(1000)。

7.4 自定义动画方法animate()

前面已讲3种类型的动画。其中show()方法和hide()方法会同时修改元素的多个样式属性，即高度，宽度和不透明度。fadeIn()方法和fadeOut()方法只会修改元素的不透明度。slideDown()方法和slideUp()方法只会改变元素的高度。

很多情况下，这些方法无法满足用户的各种需求，那么就需要对动画有更多的控制，需要采取一些高级的自定义动画来解决这些问题。在jQuery中，可以使用animate()方法来自定义动画，其语法结构为：

animate(params,[speed],[easing],[callback]);

此方法执行CSS属性集的自定义动画。通过CSS样式将元素从一个状态改变为另一个状态。CSS属性值是逐渐改变的，这样就可以创建动画效果。只有数字值可创建动画（比如margin:"30px")。字符串值无法创建动画（比如background-color:"red")。可以使用"+="或"-="来创建相对动画（relative animations）。

参数说明如下：

(1) params：规定产生动画效果的CSS样式和值，如{height:"300px",width:"100px"}。注：CSS样式使用DOM名称（比如"fontSize")来设置，而非CSS名称(比如"font-size").

(2)speed:可选。规定动画的速度，默认是"normal".

(3)easing:可选。规定在不同的动画点中设置动画速度的easing函数。内置的easing函数有·swing和·linear

(4)callback:可选。animate函数执行完之后，要执行的函数。

7.4.1 自定义简单动画

跟以上方法相比，animate()方法更具灵活性。通过animate()方法，能够实现更加精致新颖的动画效果。如在5秒内向右飘动500px：$("div").animate({left:"500px"},5000);//第一个参数用{}裹住，且值用""号引住。

7.4.2 累加/累减动画

如果在500px之前加上"+="或者"-="符号即表示在当前位置累加或者累减，如：

$("div").animate({left:"+=500px"},5000);

7.4.3 多重动画

(1) 同时执行多个动画

以上例子，只改了left的属性，这是一个很单一的动画。如果需要同时执行多个动画，例如在元素向右滑动的同时，放大元素的高度。如：$("div").animate({left:"+=500px",height:"+=100px"},5000};//{}将第一个参数多个属性包裹起来。

(2) 按顺序执行多个动画

上例两个动画是同时发生的。若想按顺序执行动画，如让<div>元素先向右滑动，然后再放大它的高度，只需把代码拆开，按顺序写就好了：

$("div").animate({left:"+=500px"},5000)

.animate({height:"+=100px"},5000);

像这样，动画效果的执行具有先后顺序，称为"动画队列".

7.4.4 综合动画

接下来将完成更复杂的动画。单击<div>元素后让它向右移动的同时增大它的高度，并将它的不透明度从50%变换到100%，然后再让它从上到下移动，同时它的宽度变大，当完成这些效果后，让它以淡出的方式隐藏。

$("div").animate({left:"+=500px",height:"100px",opacity:"1"},5000)

.animate({top:"200px",width:"200px"},5000)

.fadeOut(5000);

8. 动画回调函数

在上例中，如果想在最后一步切换元素的CSS样式，而不是隐藏要素，即：

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

如果只是按照常规的方式，将fadeOut(5000)改为css("border","5px solid blue");这样并不能得到预期效果。预期的效果是在动画的最后一步改变元素的样式，而实际的效果是，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。即

.animate({top:"200px",width:"200px"},5000,function(){

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

});

这样一来css()方法就加入到动画队列中了。

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

$("div").slideDown("normal",function(){

//在效果完成后做其他的事情

});

这段代码表示div元素将在0.4秒内向下完全展开。当动画完成后，执行回调函数体内的代码。

8. 停止动画和判断是否处于动画状态

8.1 停止元素的动画

需使用stop()方法，即可停止当前正在运行的动画，其语法结构为：

stop([clearQueue],[gotoEnd]);

参数clearQueue和gotoEnd都是可选的参数，都为Boolean值(true或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。gotoEnd必须得在clearQueue已经设置时使用。

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。

8.2 判断元素是否处于动画状态

在使用animate()方法的时候，要避免动画积累而导致的动画与用户的行为不一致。当用户快速在某个元素上执行animate()动画时，就会出现动画积累。解决方法是判断元素是否正处于动画状态，如果元素不处于动画状态，才能为元素添加新的动画，否则不添加。如：if(! $(element).is(":animated")){ //判断元素是否正处于动画状态

//如果当前没有进行动画，则添加新动画

}

8.3 延迟动画

在动画执行过程中，如果想对动画进行延迟操作，那么可以使用delay()方法。

$("div").animate({left:"500px"},1000)

.delay()//若不设置参数，则默认为是normal。设置参数如.delay(2000);

.fadeOut(1000);

9. 其他动画方法

除了上面提到的动画方法，jQuery还有4个专门用于交互的动画方法:

toggle([speed][,callback]);

slideToggle([speed][,easing][,callback]);

fadeTo(speed,opacity[,callback]);//注意，这里speed和opacity都是必填项

fadeToggle([speed][,easing][,callback]);

9.1 toggle()方法

toggle()方法可以切换元素的可见状态。如果元素是可见的，则切换为隐藏的；如果元素是隐藏的，则切换为可见的。如:$("button").click(function(){

$("div").toggle(2000);

});

9.2 slideToggle()方法

slideToggle()方法通过高度变化来切换匹配元素的可见性。这个动画效果只调整元素的高度。如$("div").slideToggle();

9.3 fadeTo()方法

fadeTo()方法可以把元素的不透明度以渐进方式调整到指定的值。这个动画只调整元素的不透明度，即匹配的元素的高度和宽度不会发生变化。$("div").fadeTo(1000,0.2);//注意，speed和opacity都是必填项

9.4 fadeToggle()方法

fadeToggle()方法通过不透明度变化来切换匹配元素的可见性。这个动画效果只调整元素的不透明度。

10. 动画方法概括

10.1 改变样式属性

|  |  |
| --- | --- |
| 方法名 | 说明 |
| hide()和show() | 同时修改多个样式属性即高度/宽度和不透明度 |
| fadeIn()和fadeOut() | 只改变不透明度 |
| slideUp()和slideDown() | 只改变高度 |
| fadeTo() | 只改变不透明度。speed和opacity两个参数必填 |
| toggle() | 用来代替hide()和show()方法，所以会同时修改多个样式属性即高度/宽度和不透明度 |
| slideToggle() | 用来代替slideUp()和slideDown()方法，所以只能改变不透明度 |
| fadeToggle() | 用来代替fadeIn()和fadeOut()方法，所以只能改变不透明度 |
| animate() | params参数为必填。属于自定义动画的方法，以上各种动画方法实质上内部调用了animate()方法。此外，直接使用animate()方法还能自定义其他的样式属性，例如："left","marginLeft","scrollTop"等 |

需要特别注意animate()方法，可以使用它来替代其他所有的动画方法。如，替代.show(400):

.animate({height:"show",width:"show",opacity:"show"},400);

事实上，这些动画就是animate()方法的一种内置了特定样式属性的简写形式。在animate()方法中，这些特定样式的属性值可以为"show","hide"和"toggle",也可以是自定义数字。

10.2 动画队列

（1）一组元素上的动画效果

当在一个animate()方法中应用多个属性时，动画是同时发生的。

当以链式的写法应用动画方法时，动画是按照顺序发生的。

(2) 多组元素上的动画效果

默认情况下，动画都是同时发生的。

当以回调的形式应用动画方式时，动画是按照回调顺序发生的。

另外，在动画方法中，要注意其他非动画方法会插队，例如css()方法要使非动画方法也按照顺序执行，需要把这些方法写在动画方法的回调函数中。

对于此动画实例，因CSS还不是太熟练，故先放一放。

**第5章 jQuery对表单/表格的操作及更多应用**

表单和表格都是HTML的重要组成部分，分别用于采集/提交用户输入的信息和显示列表数据。

一.表单应用

一个表单有3个基本组成部分。

(1) 表单标签：包含处理表单数据所用的服务器端程序URL以及数据提交到服务器的方法。

(2) 表单域：包含文本框，密码框，隐藏域，多行文本框，复选框，单选框，下拉选择框和文件上传框等。

(3) 表单按钮：包括提交按钮，复位按钮和一般按钮，用于将数据传送到服务器上或者取消传送，还可以用来控制其他定义了处理脚本的处理工作。

此处主讲jQuery在表单域中的应用。

1.单行文本框应用

文本框是表单域中最基本的元素，基于文本框的应用有很多，这里讲---获取和失去焦点改变样式。

对于label标签的for属性，是规定label与哪个表单元素绑定，其值等于那个表单元素的id，这是显示的绑定关系。隐式即直接将那个表单元素放入label标签内。

2.多行文本框应用

2.1 高度变化

点击+ -，可以增大或减小文本框高度。但注意要设置最高限度和最低限度。

2.2 滚动条高度变化

点击+ -，可以增大或减小滚动条高度。控制scrollTop属性即可。

3. 复选框的应用

对复选框的最基本应用，就是对复选框进行全选/反选和全不选等操作。复杂的操作需要与选项挂钩，来达到各种级联反应效果

在设置checked属性时，要**用prop()函数**。

在设置级联效应时，可以设置一个标签或比较个数。

4. 下拉列表的应用

主要实现3个功能：

(1) 将选中的选项添加给对方。知识点：对DOM节点的操作

(2) 将全部选项添加给对方。知识点：同上

(3) 双击某个选项将其添加给对方。尚未完成

5. 表单验证

表单作为HTML最重要的一个组成部分，几乎在每个网页上都有体现，例如用户提交信息，用户反馈信息和用户查询信息等。因此它是网站管理者和浏览者之间沟通的桥梁。在表单中，表单验证的作用也是非常重要的，它能使表单更加灵活/美观和丰富。

乱入：

jQuery插件：cookie

它是一个轻量级的cookie插件，可以读取/写入/删除cookie

jQuery.cookie.js的配置：

首先包含jQuery的库文件，在后面包含jQuery.cookie.js的库文件

<script type="text/javascript" src="../../scripts/jquery-1.11.3.js"></script>

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

使用方法：

1.新添加一个会话cookie：

$.cookie("the\_cookie","the\_value");

注：当没有指明cookie有效时间时，所创建的cookie有效期默认到用户关闭浏览器为止所以被称为“会话cookie（session cookie）”。

2.创建一个cookie并设置有效时间为7天：

$.cookie("the\_cookie","the\_value",{expires:7});

注：当指明了cookie有效时间时，所创建的cookie被称为“持久cookie（persistent cookie）”。

3. 创建一个cookie并设置cookie的有效路径：?

$.cookie("the\_cookie","the\_value",{expires:7,path:'/'});

注：在默认情况下，只有设置cookie的网页才能读取该cookie。如果想让一个页面读取另一个页面设置的cookie，必须设置cookie的路径。cookie的路径用于设置能够读取cookie的顶级目录。将这个路径设置为网站的根目录。可以让所有网页都能互相读取cookie（一般不要这样设置，防止出现冲突）。

4. 读取cookie：?

$.cookie("the\_cookie");//cookie存在=>"the\_value"

$.cookie("not\_existing");//cookie不存在=>null

5. 删除cookie，通过传递null作为cookie的值即可：

$.cookie("the\_cookie",null);

-------------相关参数的解释----------------------

1. expires:365

定义cookie的有效时间，值可以是一个数字（从创建cookie时算起，以天为单位）或一个Date对象。如果省略，那么创建的cookie是会话cookie，将在用户退出浏览器时被删除。

2. path:"/"

默认情况下，只有设置cookie的网页才能读取该cookie。

定义cookie的有效路径，默认情况下，该参数的值为创建cookie的网页所在路径（标准浏览器的行为）。如果你想在整个网站中访问这个cookie，需要这样设置有效路径：path:"/"。如果你想删除一个定义了有效路径的cookie，你需要在调用函数时包含这个路径：$.cookie("the\_cookie",null,{path:"/"});。domain:"example.com" 默认值：创建cookie的网页所拥有的域名。

3. secure:true

默认值：false。如果为true，cookie的传输需要使用安全协议（HTTPS）。

4. raw:true

默认值：false。默认情况下，读取和写入cookie的时候自动进行编码和解码（使用encodeURIComponent编码，decodeURIComponent解码）。要关闭这个功能设置raw:true即可。