续表

选 择 器	语 法	描 述	示 例

ID 选择器	#ID {
	CSS 规则
	}
		以文档元素的惟一标识符 ID 作为选择符	#note{
			font-size:14px;
			width:120px;
			}

类选择器	E.className{
	CSS 规则
	}
		以文档元素的 class 作为选择符	div.note{
			font-size:14px;
			}
			.dream{
			font-size:14px;
			}

群组选择器	E1,E2,E3{
	CSS 规则
	}
		多个选择符应用同样的样式规则	td,p,div.a{
			font-size:14px;
			}

后代选择器	E F{
	CSS 规则
	}
		元素 E 的任意后代元素 F	#links a {
			color:red;
			}

通配选择符	* {
	CSS 规则
	}
		以文档的所有元素作为选择符	*{
			font-size:14px;
			}

几乎所有主流浏览器都支持上面这些常用的选择器。此外 CSS 中还有伪类选择器(E:Pseudo-Elements{ CssRules })、子选择器(E > F{ CssRules })、临近选择器(E + F { CssRules })和属性选择器(E [attr] { CssRules })等。但遗憾的是，主流浏览器并非完全支持所有的 CSS 选择器。

更加详细的介绍可以参考 http://www.w3.org/TR/CSS2/selector.html 网址。

了解这些相关知识后，来看一个有关 CSS 类选择器的简单例子，代码如下：

```
<p style="color:red;font-size:30px;">CSS Demo</p>
```

上面代码的意思是将<p>元素里的文本颜色设置为红色，字体大小设置为 30px。

像上面这样把 CSS 代码和 HTML 代码混杂在一起的做法是非常不妥的，它并不符合表现和内容相分离的设计原则，因此建议使用下面的方法，代码如下：

```
<style>
.demo{                    //给 class 为 demo 的元素添加样式
  color:red;
  font-size:30px;
}
```

```
</style>
<p class="demo">CSS Demo.</p>
```

先把样式写在<style>标签里，然后用 class 属性将元素和样式联系起来，class 作为连接样式和网页结构的纽带。这样的写法不仅容易理解和阅读，而且当需要改变一些样式的时候，只要在<style>标签里改变相关的样式即可。

例如要使所有 class 为 demo 的<p>元素里的字体加粗，可以直接在<style>里编写，而不需要去网页里寻找所有 class 为 demo 的<p>元素再逐个添加样式，代码如下：

```
<style>
.demo{                    //给 class 为 demo 的元素添加样式
   color:red;
   font-size:30px;
   font-weight:bold;      //字体加粗
}
</style>
<p class="demo">CSS Demo.</p>
```

注意 把 CSS 应用到网页中有 3 种方式，即行间样式表、内部样式表和外部样式表。上例中使用的是内部样式表，内部样式表的缺点是不能被多个页面重复使用的。

2. jQuery 选择器

jQuery 中的选择器完全继承了 CSS 的风格。利用 jQuery 选择器，可以非常便捷和快速地找出特定的 DOM 元素，然后为它们添加相应的行为，而无需担心浏览器是否支持这一选择器。学会使用选择器是学习 jQuery 的基础，jQuery 的行为规则都必须在获取到元素后才能生效。

下面来看一个简单的例子，代码如下：

```
<script type="text/javascript">
  function demo(){
      alert('JavaScript demo.');
  }
</script>
<p onclick="demo();">点击我.</p>
```

本段代码的作用是为<p>元素设置一个 onclick 事件，当单击此元素时，会弹出一个对话框，显示效果如图 2-1 所示。

像上面这样把 JavaScript 代码和 HTML 代码混杂在一起的做法同样也非常不妥，因为它并没有将网页内容和行为分离，所以建议使用下面的方法，代码如下：

```
<p class="demo">jQuery Demo</p>
<script type="text/javascript">
    $(".demo").click(function(){          //给 class 为 demo 的元素添加行为
        alert("jQuery demo!");
    })
</script>
```

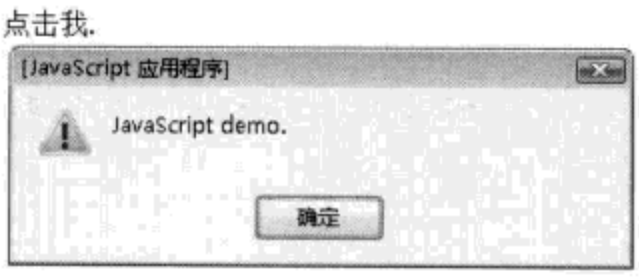


图 2-1　弹出警告框

此时，可以对 CSS 的写法和 jQuery 的写法进行比较。

CSS 获取到元素的代码如下：

```
.demo{                    //给 class 为 demo 的元素添加样式
...
}
```

jQuery 获取到元素的代码如下：

```
$(".demo"{                //给 class 为 demo 的元素添加行为
...
```

jQuery 选择器的写法与 CSS 选择器的写法十分相似，只不过两者的作用效果不同，CSS 选择器找到元素后是添加样式，而 jQuery 选择器找到元素后是添加行为。需要特别说明的是，jQuery 中涉及操作 CSS 样式的部分比单纯的 CSS 功能更为强大，并且拥有跨浏览器的兼容性。

2.2　jQuery 选择器的优势

1. 简洁的写法

$()函数在很多 JavaScript 类库中都被作为一个选择器函数来使用，在 jQuery 中也不例外。其中，$("#ID")用来代替 document.getElementById()函数，即通过 ID 获取元素；$("tagName")用来代替 document.getElementsByTagName()函数，即通过标签名获取 HTML 元素；其他选择器的写法可以参见第 2.3 节。

2. 支持 CSS1 到 CSS3 选择器

jQuery 选择器支持 CSS 1、CSS 2 的全部和 CSS 3 的部分选择器，同时它也有少量独有的选择器，因此对拥有一定 CSS 基础的开发人员来说，学习 jQuery 选择器是件非常容易的事，而对于没有接触过 CSS 技术的开发人员来说，在学习 jQuery 选择器的同时也可以掌握 CSS 选择器的基本规则。

使用 CSS 选择器时，开发人员需要考虑主流浏览器是否支持某些选择器。而在 jQuery 中，开发人员则可以放心地使用 jQuery 选择器而无需考虑浏览器是否支持这些选择器。

> **注意** 为了能有更快的选择器解析速度，从 1.1.3.1 版以后，jQuery 废弃了不常使用的 XPath 选择器，但在引用相关插件后，依然可以支持 XPath 选择器（详见第 2.7.1 小节）。

3. 完善的处理机制

使用 jQuery 选择器不仅比使用传统的 getElementById()和 getElementsByTagName()函数简洁得多，而且还能避免某些错误。看下面这个例子，代码如下：

```
<div>test</div>
<script type="text/javascript">
   document.getElementById("tt").style.color="red";
</script>
```

运行上面的代码，浏览器就会报错，原因是网页中没有 id 为“tt”的元素。

改进后的代码如下：

```
<div>test</div>
<script type="text/javascript">
      if(document.getElementById("tt")){
          document.getElementById("tt").style.color="red";
    }
</script>
```

这样就可以避免浏览器报错，但如果要操作的元素很多，可能对每个元素都要进行一次判断，大量重复的工作会使开发人员感到厌倦，而 jQuery 在这方面问题上的处理是非常不错的，即使用 jQuery 获取网页中不存在的元素也不会报错，看下面的例子，代码如下：

```
<div>test</div>
<script type="text/javascript">
```

```
    $('#tt').css("color","red");      //这里无需判断$('#tt')是否存在
</script>
```

有了这个预防措施，即使以后因为某种原因删除网页上某个以前使用过的元素，也不用担心这个网页的 JavaScript 代码会报错。

需要注意的是，$('#tt')获取的永远是对象，即使网页上没有此元素。因此当要用 jQuery 检查某个元素在网页上是否存在时，不能使用以下代码：

```
if ( $("#tt") ) {
   //do something
}
```

而应该根据获取到元素的长度来判断，代码如下：

```
if ( $("#tt").length > 0 ) {
   //do something
}
```

或者转化成 DOM 对象来判断，代码如下：

```
if ( $("#tt")[0]  ) {
   //do something
}
```

2.3 jQuery 选择器

在正式学习 jQuery 选择器之前，先看几组用传统的 JavaScript 方法获取页面中的元素，然后给元素添加行为事件的例子。

例子 1：给网页中的所有<p>元素添加 onclick 事件。

HTML 代码如下：

```
<p>测试 1</p>
<p>测试 2</p>
```

要做的工作有以下几项。

（1）获取所有的<p>元素。

（2）对<p>元素进行循环（因为获取的是数组对象）。

（3）给每个<p>元素添加行为事件。

JavaScript 代码如下：

```
var items = document.getElementsByTagName("p"); //获取网页中所有的 p 元素
for(var i=0;i < items.length;i++){  //由于获取的是数组对象，因此需要把它循环出来
```

```
        items[i].onclick = function(){              //给每个对象添加 onclick 事件
            //doing something
        }
}
```

例子 2：使一个特定的表格隔行变色。

HTML 代码如下：

```
<table id="tb">
    <tbody>
        <tr><td>第一行</td><td>第一行</td></tr>
        <tr><td>第二行</td><td>第二行</td></tr>
        <tr><td>第三行</td><td>第三行</td></tr>
        <tr><td>第四行</td><td>第四行</td></tr>
        <tr><td>第五行</td><td>第五行</td></tr>
        <tr><td>第六行</td><td>第六行</td></tr>
    </tbody>
</table>
```

要做的工作有以下几项。

（1）根据表格 id 获取表格。

（2）在表格内获取<tbody>元素。

（3）在<tbody>元素下获取<tr>元素。

（4）循环输出获取的<tr>元素。

（5）对<tr>元素的索引值除以 2 并取模，然后根据奇偶设置不同的背景色。

JavaScript 代码如下：

```
var item = document.getElementById("tb");     //获取 id 为 tb 的元素（table）
var tbody = item.getElementsByTagName("tbody")[0]; //获取表格的第 1 个 tbody 元素
var trs = tbody.getElementsByTagName("tr");  //获取 tbody 元素下的所有 tr 元素
for(var i=0;i < trs.length;i++){                    //循环 tr 元素
    if(i%2==0){              //取模（取余数。例如 0%2==01 %2==1 2%2==0 3%2==1）
        trs[i].style.backgroundColor = "#888"; //改变符合条件的 tr 元素的背景色
    }
}
```

例子 3：对多选框进行操作，输出选中的多选框的个数。

HTML 代码如下：

```
<input type="checkbox" value="1" name="check" checked/>
<input type="checkbox" value="2" name="check" />
<input type="checkbox" value="3" name="check" checked/>
<input type="button" value="你选中的个数" id="btn"/>
```

要做的工作有以下几项。

（1）新建一个空数组。

（2）获取所有 name 为“check”的多选框。

（3）循环判断多选框是否被选中，如果被选中则添加到数组里。

（4）获取输出按钮，然后为按钮添加 onclick 事件，输出数组的长度即可。

JavaScript 代码如下：

```
var btn = document.getElementById("btn");     //获取 id 为 btn 的元素(button)
btn.onclick = function(){                      //给元素添加 onclick 事件
          var arrays = new Array();            //创建一个数组对象
          var items = document.getElementsByName("check");
                                               //获取 name 为 check 的一组元素（checkbox）
          for(i=0; i<items.length; i++){       //循环这组数据
              if(items[i].checked){            //判断是否选中
                 arrays.push(items[i].value); //把符合条件的数据添加到数组中
                                  //push()是 JavaScript 数组中的方法
              }
          }
      alert( "选中的个数为: "+arrays.length  )
}
```

上面的几个例子都是用传统的 JavaScript 方法进行操作，中间使用了 getElement ById()、getElementsByTagName()和 getElementsByName()等方法，然后动态地给元素添加行为或者样式。这些虽然都是 JavaScript 中最简单的操作，但不断重复使用 getElementById()和 getElementsByTagName()等冗长而难记的名称，使越来越多的开发人员开始厌倦这种枯燥的写法，并且有时候为了获取网页中的某个元素，需要编写很多的 getElementById()和 getElementsByTagName()方法。然而在 jQuery 中，类似的这些操作则是非常简洁。

下面学习如何使用 jQuery 获取这些元素。

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。在下面的章节中将分别用不同的选择器来查找 HTML 代码中的元素并对其进行简单的操作。为了能更清晰、直观地讲解选择器，首先需要设计一个简单的页面，里面包含各种 元素和元素，然后使用 jQuery 选择器来匹配元素并调整它们的样式。

新建一个空白页面，输入以下 HTML 代码：

```
<div class="one" id="one" >
        id为one,class为one的div
    <div class="mini">class为mini</div>
</div>
<div class="one"  id="two" title="test" >
    id为two,class为one,title为test的div.
    <div class="mini"  title="other">class为mini,title为other</div>
    <div class="mini"  title="test">class为mini,title为test</div>
</div>
<div class="one">
    <div class="mini">class为mini</div>
    <div class="mini">class为mini</div>
        <div class="mini"></div>
    </div>
    <div class="one">
        <div class="mini">class为mini</div>
        <div class="mini">class为mini</div>
        <div class="mini">class为mini</div>
        <div class="mini"  title="tesst">class为mini,title为tesst</div>
    </div>
    <div style="display:none;"  class="none">style的display为"none"的div
</div>
    <div class="hide">class为"hide"的div</div>
    <div>
    包含input的type为"hidden"的div<input type="hidden" size="8"/>
    </div>
    <span id="mover">正在执行动画的span元素.</span>
```

然后用 CSS 对这些元素进行初始化大小和背景颜色的设置，CSS 代码如下：

```
div,span,p {
    width:140px;
    height:140px;
    margin:5px;
    background:#aaa;
    border:#000 1px solid;
    float:left;
    font-size:17px;
    font-family:Verdana;
}
div.mini {
    width:55px;
    height:55px;
    background-color: #aaa;
    font-size:12px;
}
div.hide {
    display:none;
}
```

根据以上 HTML+CSS 代码，可以生成图 2-2 所示的页面效果。

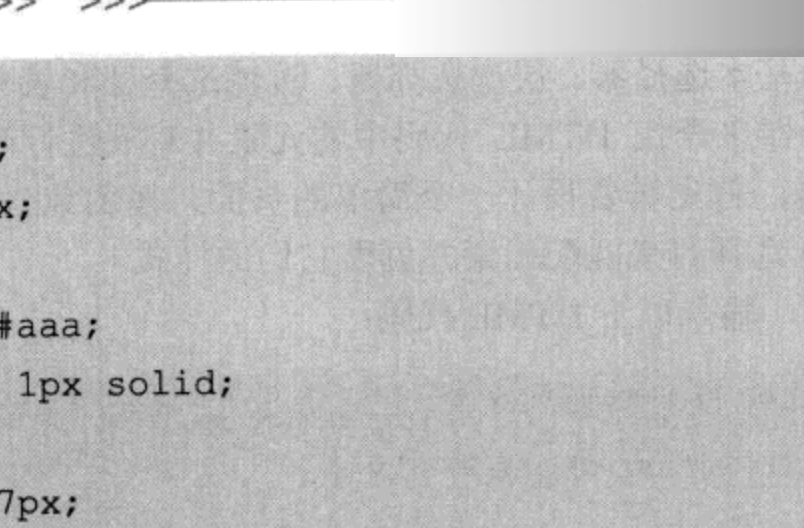


图 2-2　初始状态

2.3.1　基本选择器

基本选择器是 jQuery 中最常用的选择器，也是最简单的选择器，它通过元素 id、class 和标签名等来查找 DOM 元素。在网页中，每个 id 名称只能使用一次，class 允许重复使用。

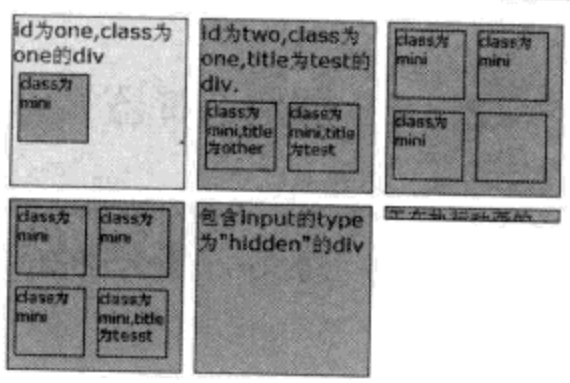
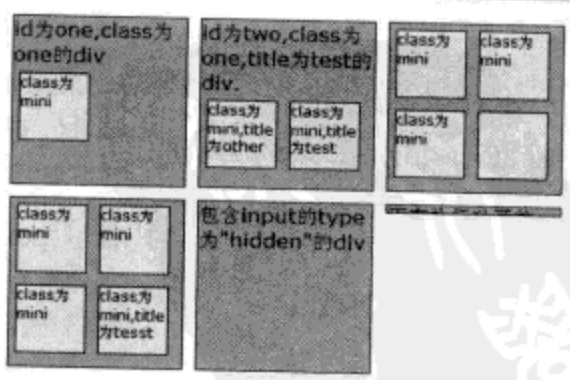
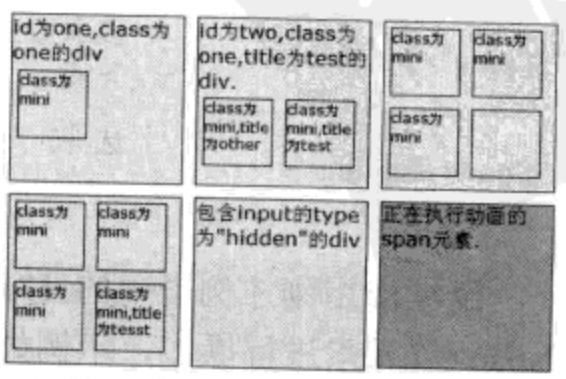
基本选择器的介绍说明如表 2-2 所示。

表 2-2　　基本选择器

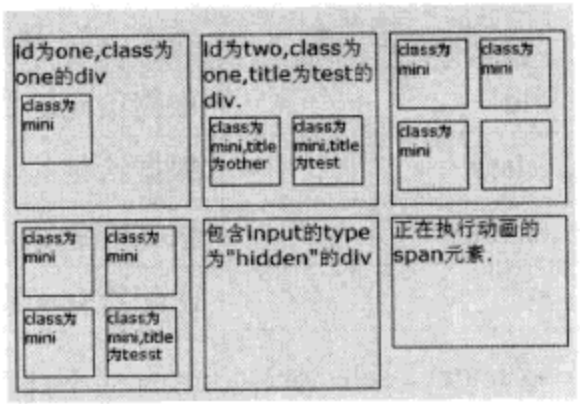
选　择　器	描　　述	返　　回	示　　例
#id	根据给定的 id 匹配一个元素	单个元素	$("#test")选取 id 为 test 的元素
.class	根据给定的类名匹配元素	集合元素	$(".test")选取所有 class 为 test 的元素
element	根据给定的元素名匹配元素	集合元素	$("p")选取所有的<p>元素
*	匹配所有元素	集合元素	$("*")选取所有的元素
selector1，selector2，……，selectorN	将每一个选择器匹配到的元素合并后一起返回	集合元素	$("div,span,p.myClass")选取所有 ，和拥有 class 为 myClass 的<p>标签的一组元素

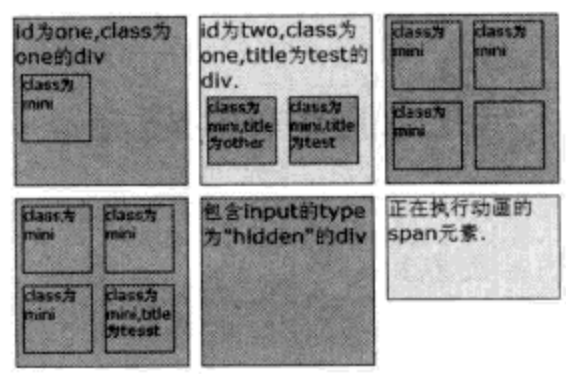
可以使用这些基本选择器来完成绝大多数的工作。下面用它们来匹配刚才 HTML 代码中的 ，等元素并进行操作（改变背景色），示例如表 2-3 所示。

表 2-3　　基本选择器示例

功　　能	代　　码	执　行　后
改变 id 为 one 的元素的背景色	$('#one') .css("background","#bbffaa");	
改变 class 为 mini 的所有元素的背景色	$('.mini') .css("background","#bbffaa");	
改变元素名是 的所有元素的背景色	$('div') .css("background","#bbffaa");	

续表

功　能	代　码	执 行 后
改变所有元素的背景色	$('*') .css("background","#bbffaa");	

改变所有的元素和 id 为 two 的元素的背景色	$('span, #two')
	.css("background","#bbffaa");
		

2.3.2　层次选择器

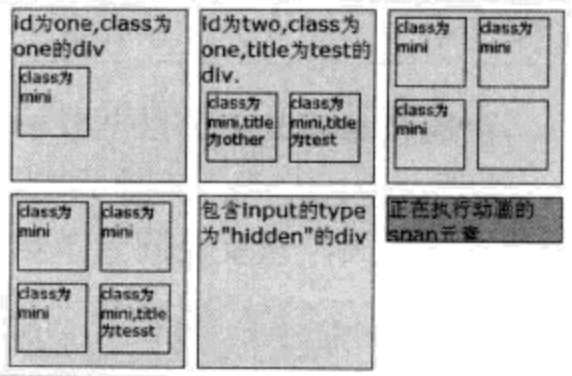
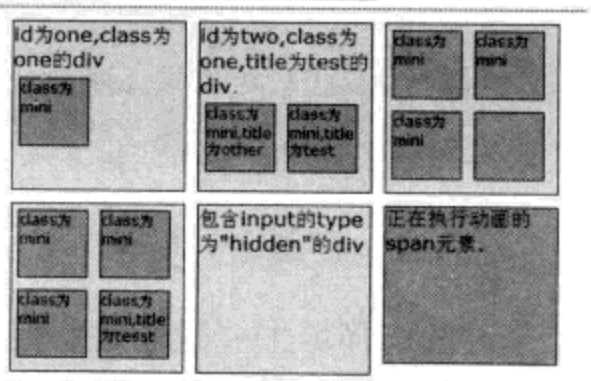
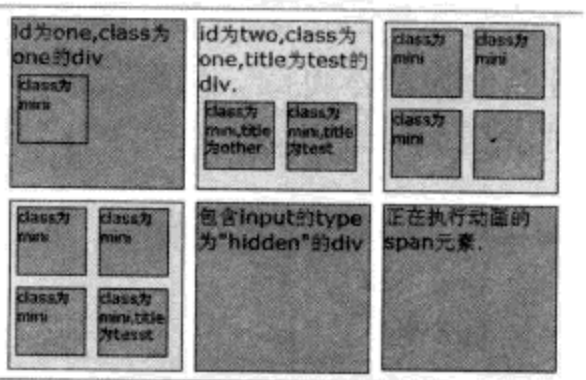
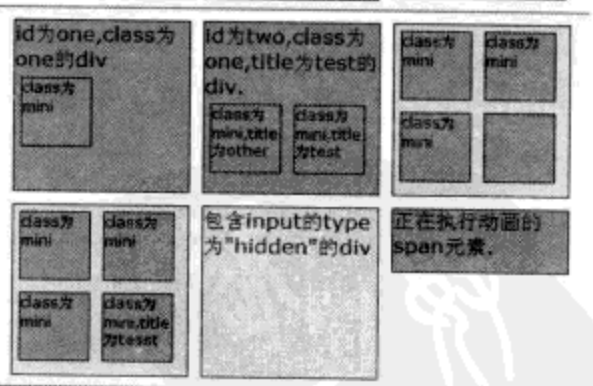
如果想通过 DOM 元素之间的层次关系来获取特定元素，例如后代元素、子元素、相邻元素和兄弟元素等，那么层次选择器是一个非常好的选择。层次选择器的介绍说明如表 2-4 所示。

表 2-4　　层次选择器

选　择　器	描　　述	返回	示　　例
$("ancestor　descendant")	选取 ancestor 元素里的所有 descendant（后代）元素	集合元素	$("div　span")选取 里的所有的元素
$("parent > child")	选取 parent 元素下的 child（子）元素，与$("ancestor descendant")有区别，$("ancestor descendant")选择的是后代元素	集合元素	$("div　>　span")选取 元素下元素名是的子元素
$('prev + next')	选取紧接在 prev 元素后的 next 元素	集合元素	$('.one　+　div')选取 class 为 one 的下一个 元素
$('prev~siblings')	选取 prev 元素之后的所有 siblings 元素	集合元素	$('#two~div')选取 id 为 two 的元素后面的所有 兄弟元素

继续沿用刚才例子中的 HTML 和 CSS 代码，然后用层次选择器来对网页中的 ，等元素进行操作，示例如表 2-5 所示。

表 2-5　　层次选择器示例

功　能	代　码	执 行 后
改变<body>内所有 的背景色	$('body div') .css("background","#bbffaa");	
改变<body>内子 元素的背景色	$('body > div') .css("background","#bbffaa");	
改变 class 为 one 的下一个 元素背景色	$('.one + div') .css("background","#bbffaa");	
改变 id 为 two 的元素后面的所有 兄弟元素的背景色	$('#two ~ div') .css("background","#bbffaa");	

在层次选择器中，第 1 个和第 2 个选择器比较常用，而后面两个因为在 jQuery 里可以用更加简单的方法代替，所以使用的几率相对少些。

可以使用 next()方法来代替$('prev + next') 选择器，如表 2-6 所示。

表 2-6　　$('prev + next')选择器与 next()方法的等价关系

	选　择　器	方　法
等价关系	$(".one + div");	$(". one").next("div");

可以使用 nextAll()方法来代替$('prev~siblings') 选择器，如表 2-7 所示。

表 2-7　　$('prev~siblings')选择器与 nextAll()方法的等价关系

	选 择 器	方 法
等价关系	$("#prev~div");	$("#prev").nextAll ("div");

在此将 siblings()方法与$('prev~siblings')选择器进行比较。

$("#prev~div")选择器只能选择“#prev”元素后面的同辈 元素。而 siblings()方法与前后位置无关，只要是同辈节点就都能匹配。

```
$('#prev ~ div').css("background","#bbffaa"); //选取#prev 之后的所有同辈 div 元素
$('#prev').nextAll("div").css("background","#bbffaa");  //同上
$("#prev").siblings("div").css("background","#bbffaa");
                                    //选取#prev 所有的同辈 div 元素，无论前后位置
```

2.3.3 过滤选择器

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。

1. 基本过滤选择器

表 2-8　　基本过滤选择器

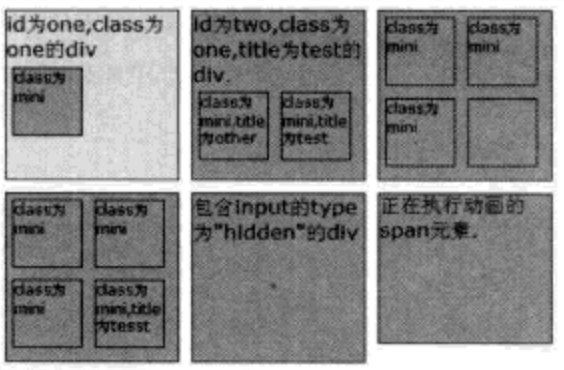
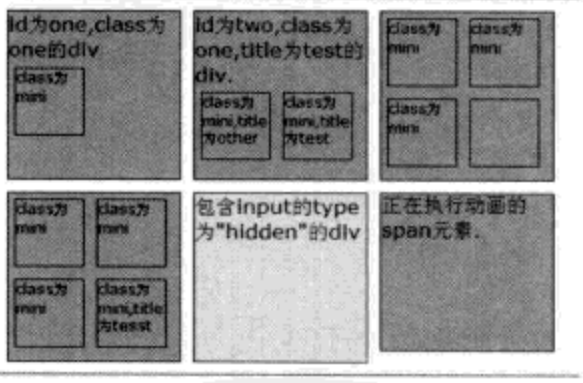
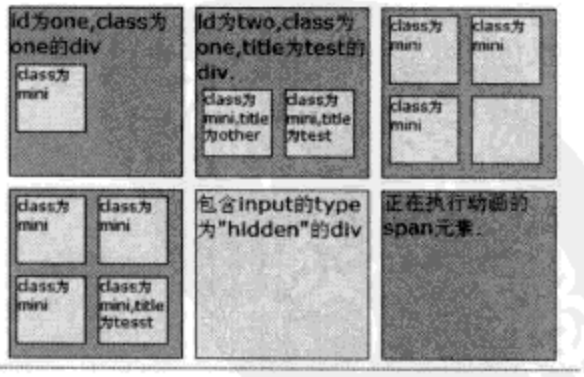
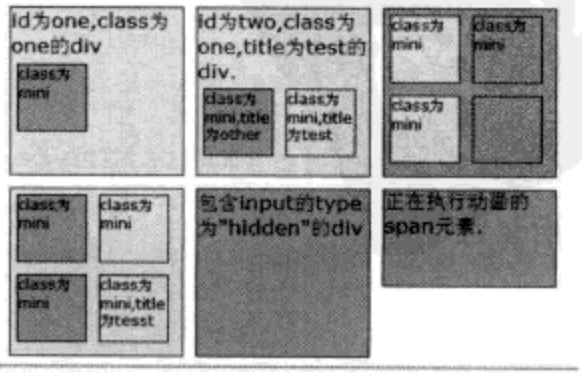
选 择 器	描 述	返 回	示 例
:first	选取第 1 个元素	单个元素	$("div:first")选取所有 元素中第 1 个 元素
:last	选取最后一个元素	单个元素	$("div:last")选取所有 元素中最后一个 元素
:not(selector)	去除所有与给定选择器匹配的元素	集合元素	$("input:not(.myClass)")选取 class 不是 myClass 的<input>元素
:even	选取索引是偶数的所有元素，索引从 0 开始	集合元素	$("input:even")选取索引是偶数的<input>元素
:odd	选取索引是奇数的所有元素，索引从 0 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（index 从 0 开始）	单个元素	$("input:eq(1)")选取索引等于 1 的<input>元素
:gt(index)	选取索引大于 index 的元素（index 从 0 开始）	集合元素	$("input:gt(1)")选取索引大于 1 的<input>元素（注：大于 1，而不包括 1）
:lt(index)	选取索引小于 index 的元素（index 从 0 开始）	集合元素	$("input:lt(1)")选取索引小于 1 的<input>元素（注：小于 1，而不包括 1）

续表

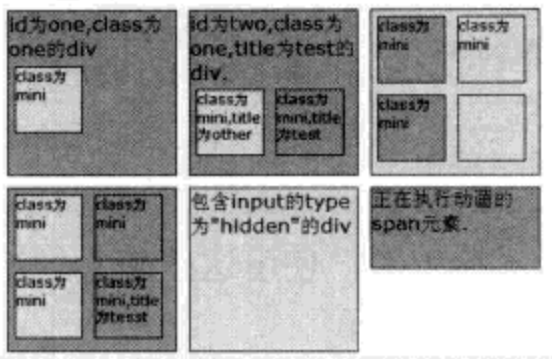
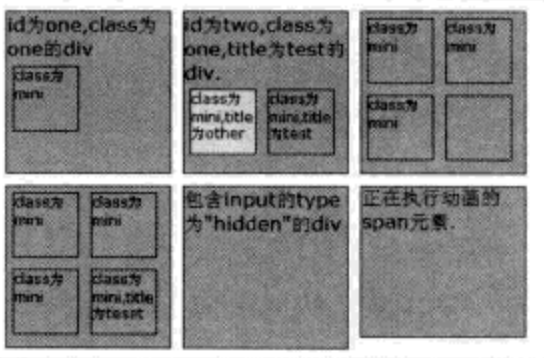
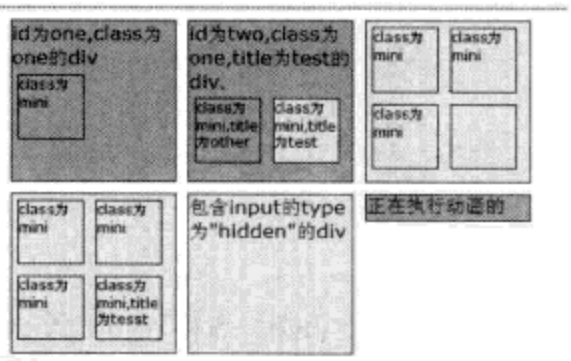
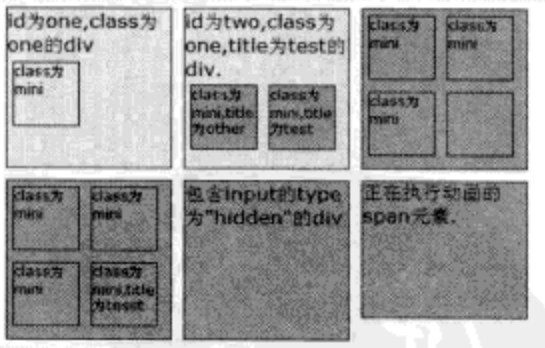
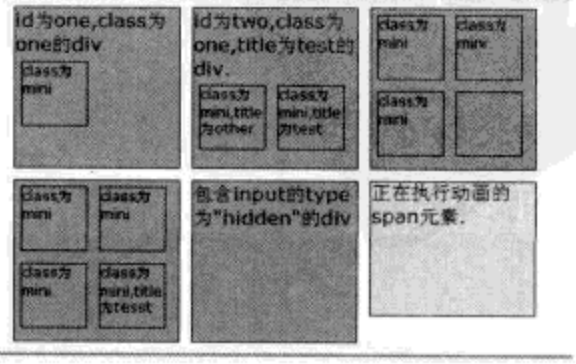
选 择 器	描 述	返 回	示 例
:header	选取所有的标题元素，例如 h1，h2，h3 等等	集合元素	$(":header")选取网页中所有的<h1>，<h2>，<h3>……
:animated	选取当前正在执行动画的所有元素	集合元素	$("div:animated")选取正在执行动画的 元素

接下来，使用这些基本过滤选择器来对网页中的 ，等元素进行操作，示例如表 2-9 所示。

表 2-9　　基本过滤选择器示例

功 能	代 码	执 行 后
改变第 1 个 元素的背景色	`$('div:first')` `.css("background","#bbffaa");`	
改变最后一个 元素的背景色	`$('div:last')` `.css("background","#bbffaa");`	
改变 class 不为 one 的 元素的背景色	`$('div:not(.one)')` `.css("background","#bbffaa");`	
改变索引值为偶数的 元素的背景色	`$('div:even')` `.css("background","#bbffaa");`	

续表

功　能	代　码	执 行 后
改变索引值为奇数的 元素的背景色	$('div:odd') .css("background","#bbffaa");	
改变索引值等于3的 元素的背景色	$('div:eq(3)') .css("background","#bbffaa");	
改变索引值大于3的 元素的背景色	$('div:gt(3)') .css("background","#bbffaa");	
改变索引值小于3的 元素的背景色	$('div:lt(3)') .css("background","#bbffaa");	
改变所有的标题元素，例如<h1>，<h2>，<h3>……这些元素的背景色	$(':header') .css("background","#bbffaa");	基本过滤选择器.
改变当前正在执行动画的元素的背景色	$(':animated') .css("background","#bbffaa");	

2. 内容过滤选择器

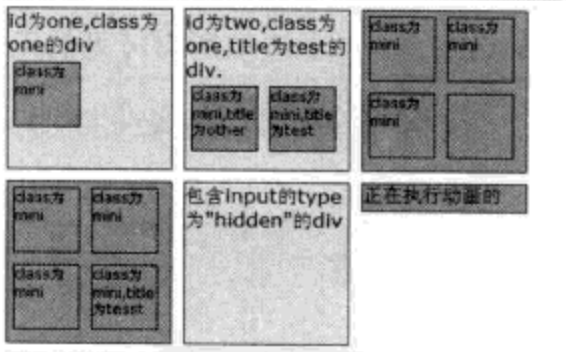
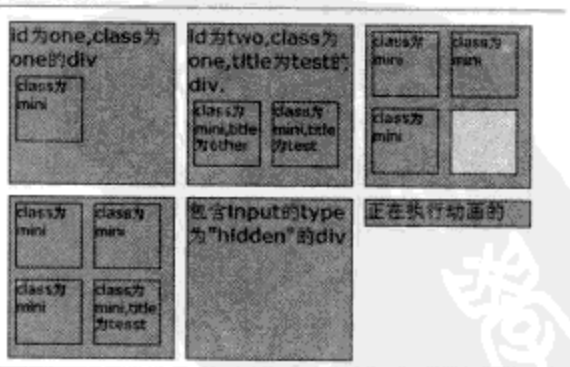
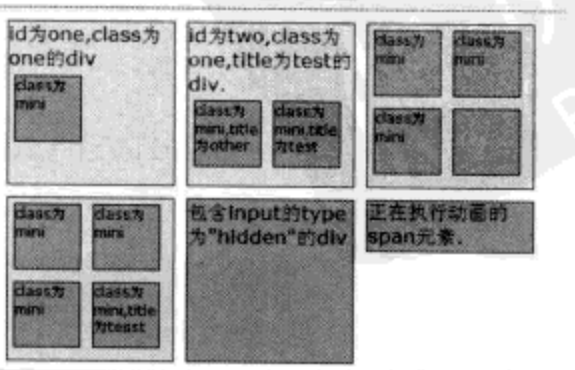
内容过滤选择器的过滤规则主要体现在它所包含的子元素或文本内容上。内容过滤选择器的介绍说明如表 2-10 所示。

表 2-10　　内容过滤选择器

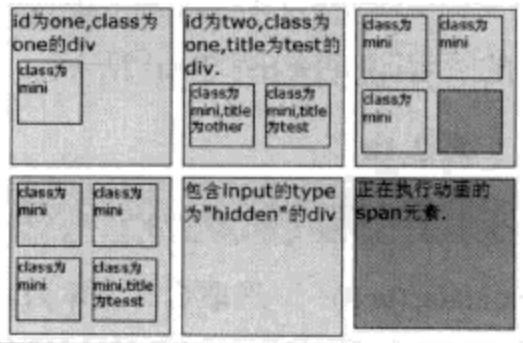
选　择　器	描　　述	返回	示　　例
:contains(text)	选取含有文本内容为“text”的元素	集合元素	$("div:contains('我')")选取含有文本“我”的 元素
:empty	选取不包含子元素或者文本的空元素	集合元素	$("div:empty")选取不包含子元素（包括文本元素）的 空元素
:has(selector)	选取含有选择器所匹配的元素的元素	集合元素	$("div:has(p)")选取含有<p>元素的 元素
:parent	选取含有子元素或者文本的元素	集合元素	$("div:parent")选取拥有子元素（包括文本元素）的 元素

接下来使用内容过滤选择器来操作页面中的元素，示例如表 2-11 所示。

表 2-11　　内容过滤选择器示例

功　　能	代　　码	执　行　后
改变含有文本“di”的 元素的背景色	$('div:contains(di)') .css("background","#bbffaa");	
改变不包含子元素（包括文本元素）的 空元素的背景色	$('div:empty') .css("background","#bbffaa");	
改变含有 class 为 mini 元素的 元素的背景色	$('div:has(mini)') .css("background","#bbffaa");	

续表

功　能	代　码	执　行　后
改变含有子元素（包括文本元素）的 元素的背景色	$ ('div:parent') .css("background","#bbffaa");	

3. 可见性过滤选择器

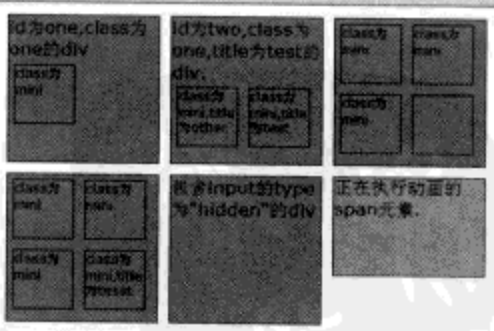
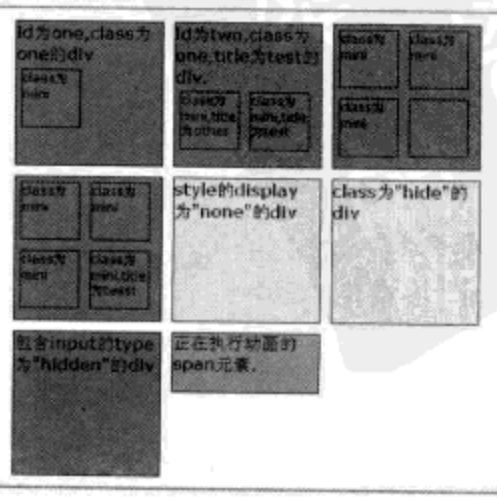
可见性过滤选择器是根据元素的可见和不可见状态来选择相应的元素。可见性过滤选择器的介绍说明如表 2-12 所示。

表 2-12　　可见性过滤选择器

选　择　器	描　述	返回	示　例
:hidden	选取所有不可见的元素	集合元素	$(":hidden"）选取所有不可见的元素。包括<input type="hidden"/>，<div style="display:none;">和<div style="visibility:hidden;">等元素。如果只想选取<input>元素，可以使用$("input:hidden")
:visible	选取所有可见的元素	集合元素	$("div:visible")选取所有可见的 元素

在例子中使用这些选择器来操作 DOM 元素，示例如表 2-13 所示。

表 2-13　　可见性过滤选择器示例

功　能	代　码	执　行　后
改变所有可见的 元素的背景色	$('div:visible') .css("background","#FF6500");	
显示隐藏的 元素	$('div:hidden').show(3000);	

在可见性选择器中，需要注意选择器 :hidden，它不仅包括样式属性 display 为“none”的元素，也包括文本隐藏域(<input type="hidden" />)和 visibility:hidden 之类的元素。

注意　show()是 jQuery 的方法，它的功能是显示元素，3000 是时间，单位是毫秒。

4. 属性过滤选择器

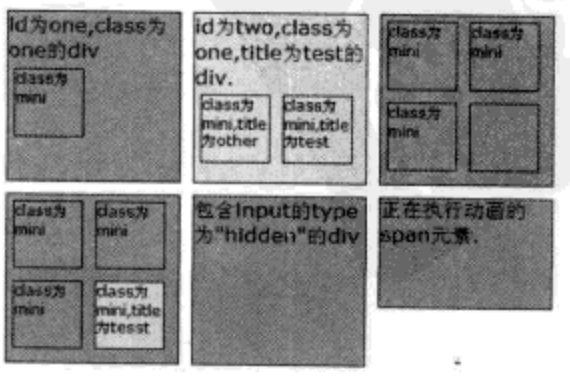
属性过滤选择器的过滤规则是通过元素的属性来获取相应的元素。属性过滤选择器的介绍说明如表 2-14 所示。

表 2-14　　属性过滤选择器

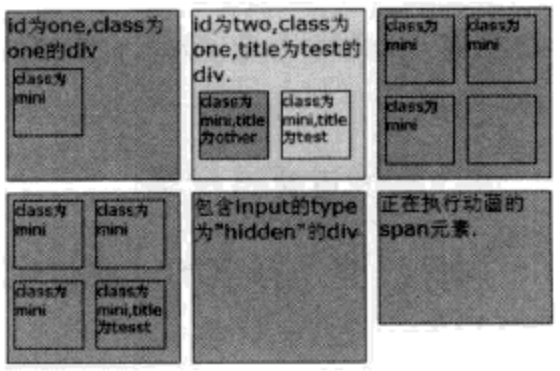
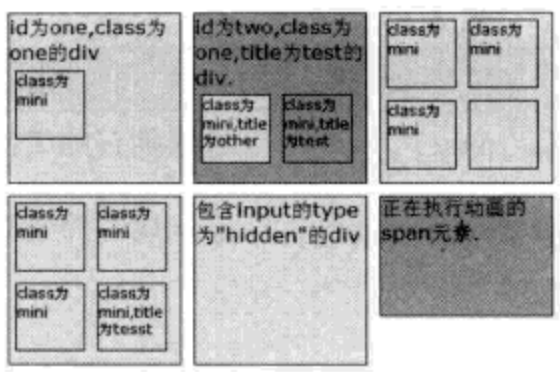
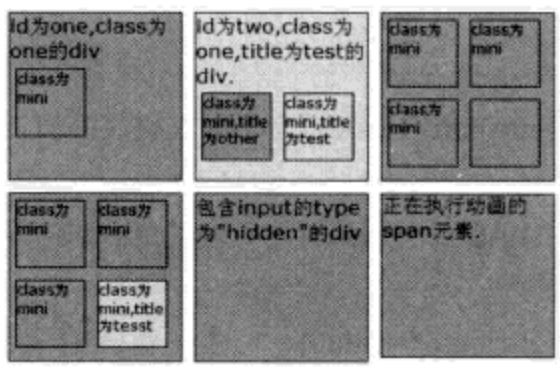
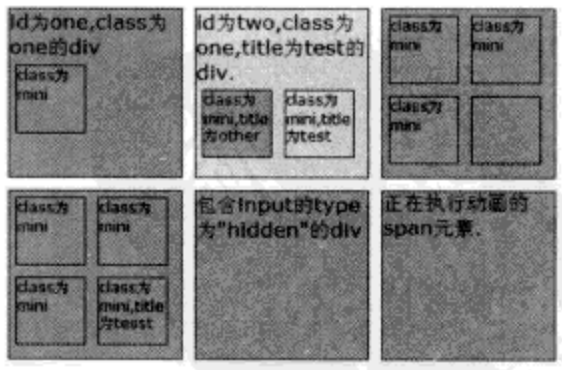
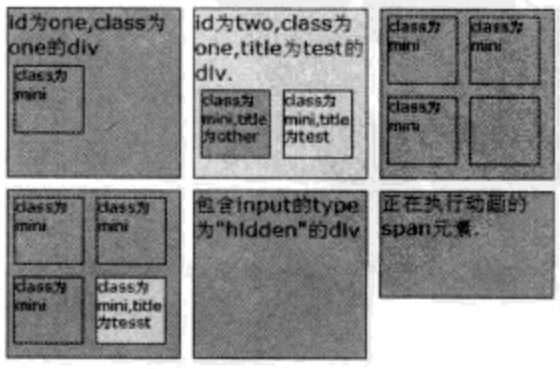
选　择　器	描　　述	返回	示　　例
[attribute]	选取拥有此属性的元素	集合元素	$("div[id]")选取拥有属性 id 的元素
[attribute=value]	选取属性的值为 value 的元素	集合元素	$("div[title=test]")选取属性 title 为“test”的 元素
[attribute!=value]	选取属性的值不等于 value 的元素	集合元素	$("div[title!=test]")选取属性 title 不等于“test”的 元素（注意：没有属性 title 的 元素也会被选取）
[attribute^=value]	选取属性的值以 value 开始的元素	集合元素	$("div[title^=test]")选取属性 title 以“test”开始的 元素
[attribute$=value]	选取属性的值以 value 结束的元素	集合元素	$("div[title$=test]")选取属性 title 以“test”结束的 元素
[attribute*=value]	选取属性的值含有 value 的元素	集合元素	$("div[title*=test]")选取属性 title 含有“test”的< div>元素
[selector1][selector2] [selectorN]	用属性选择器合并成一个复合属性选择器，满足多个条件。每选择一次，缩小一次范围	集合元素	$("div[id][title$='test']")选取拥有属性 id，并且属性 title 以“test”结束的 元素

接下来使用属性过滤选择器来对 和等元素进行操作，示例如表 2-15 所示。

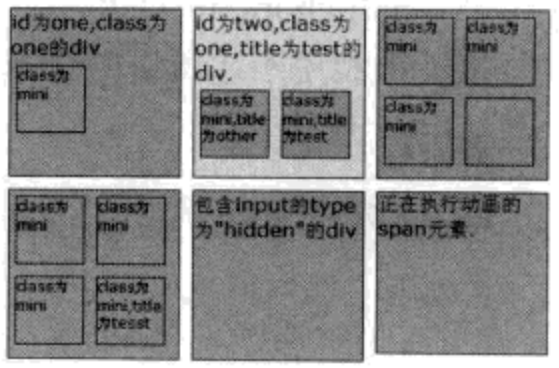
表 2-15　　属性过滤选择器示例

功　　能	代　　码	执　行　后
改变含有属性 title 的 元素的背景色	`$('div[title]')` `.css("background","#bbffaa");`	

续表

功　能	代　码	执 行 后
改变属性title值等于“test”的 元素的背景色	$('div[title=test]') .css("background","#bbffaa");	
改变属性 title 值不等于“test”的 元素的背景色	$('div[title!=test]') .css("background","#bbffaa");	
改变属性title值以“te”开始的 元素的背景色	$('div[title^=te]') .css("background","#bbffaa");	
改变属性 title 值以“est”结束的 元素的背景色	$("div[title$=est]") .css("background","#bbffaa");	
改变属性 title 值含有“es”的 元素的背景色	$("div[title*=es]") .css("background","#bbffaa");	

续表

功　能	代　码	执　行　后
改变含有属性 id，并且属性 title 值含有“es”的 元素的背景色	$("div[id][title*=es]") .css("background","#bbffaa");	

5. 子元素过滤选择器

子元素过滤选择器的过滤规则相对于其它的选择器稍微有些复杂，不过没关系，只要将元素的父元素和子元素区分清楚，那么使用起来也非常简单。另外还要注意它与普通的过滤选择器的区别。

子元素过滤选择器的介绍说明如表 2-16 所示。

表 2-16　　子元素过滤选择器

选　择　器	描　　述	返回	示　　例
: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)	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 算起的
:first-child	选取每个父元素的第 1 个子元素	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$("ul li:first-child"); 选取每个 中第 1 个 元素
:last-child	选取每个父元素的最后一个子元素	集合元素	同样，:last 只返回单个元素，而:last-child 选择符将为每个父元素匹配最后一个子元素。 例如$("ul li:first-child); 选择每个 中最后一个 元素
:only-child	如果某个元素是它父元素中惟一的子元素，那么将会被匹配。如果父元素中含有其他元素，则不会被匹配	集合元素	$("ul li:only-child") 在 中选取是惟一子元素的 元素

:nth-child() 选择器是很常用的子元素过滤选择器，详细功能如下。

（1）:nth-child(even)能选取每个父元素下的索引值是偶数的元素。

（2）:nth-child(odd)能选取每个父元素下的索引值是奇数的元素。

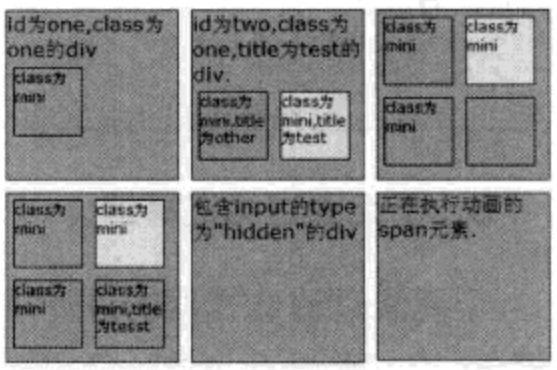
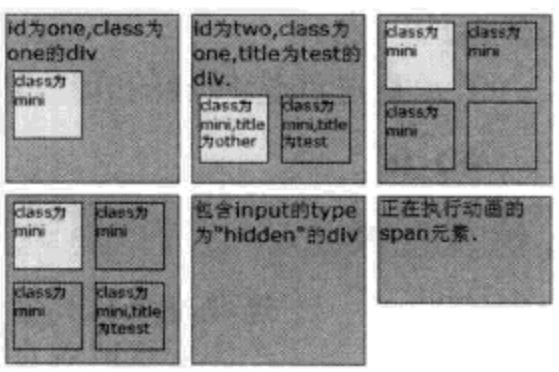
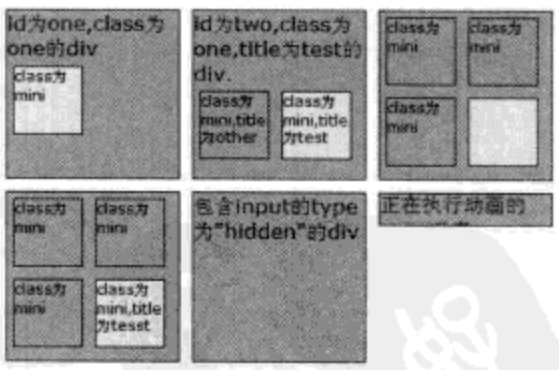
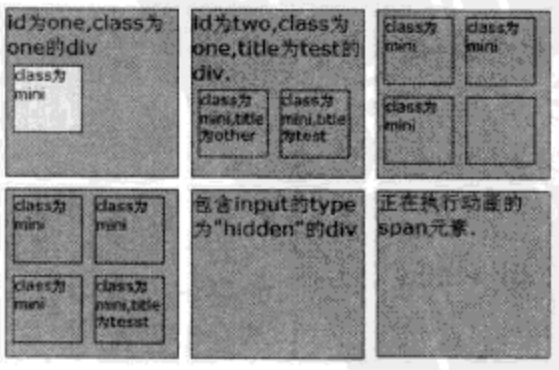
（3）:nth-child(2)能选取每个父元素下的索引值等于 2 的元素。

（4）: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 0 开始）。

（5）: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 0 开始）

接下来利用刚才所讲的选择器来改变 元素的背景色，示例如表 2-17 所示。

表 2-17　　子元素过滤选择器示例

功　能	代　码	执 行 后
改变每个 class 为 one 的 父元素下的第 2 个子元素的背景色	`$('div.one :nth-child(2)')` `.css("background","#bbffaa");`	
改变每个 class 为 one 的 父元素下的第 1 个子元素的背景色	`$('div.one :first-child')` `.css("background","#bbffaa");`	
改变每个 class 为 one 的 父元素下的最后一个子元素的背景色	`$('div.one :last-child')` `.css("background","#bbffaa");`	
如果 class 为 one 的 父元素下只有一个子元素，那么则改变这个子元素的背景色	`$('div.one :only-child')` `.css("background","#bbffaa");`	

注意　eq(index)只匹配一个元素，而:nth-child 将为每一个符合条件的父元素匹配子元素。同时应该注意到 nth-child(index) 的 index 是从 1 开始的，而:eq(index)是从 0 开始的。同理　:first 和:first-child，:last 和:last-child 也类似。

6. 表单对象属性过滤选择器

此选择器主要是对所选择的表单元素进行过滤，例比如选择被选中的下拉框，多选框等等。表单对象属性过滤选择器的介绍说明如表 2-18 所示。

表 2-18　　表单对象属性过滤选择器

选 择 器	描　　述	返回	示　　例
:enabled	选取所有可用元素	集合元素	$("#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	集合元素	$("#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）	集合元素	$("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）	集合元素	$("select :selected");选取所有被选中的选项元素

为了演示这些选择器，需要制作一个包含表单的网页，里面要包含文本框、多选框和下拉列表，HTML 代码如下：

```
    <form id="form1" action="#">
        可用元素: <input name="add" value="可用文本框"/> <br/>
        不可用元素: <input name="email" disabled="disabled" value="不可用文本
框"/><br/>
        可用元素: <input name="che" value="可用文本框" /><br/>
        不可用元素: <input name="name" disabled="disabled" value="不可用文本
框"/><br/>
        <br/>
        多选框: <br/>
        <input type="checkbox" name="newsletter" checked="checked"
value="test1" />test1
        <input type="checkbox" name="newsletter" value="test2" />test2
        <input type="checkbox" name="newsletter" value="test3" />test3
        <input type="checkbox" name="newsletter" checked="checked"
value="test4" />test4
        <input type="checkbox" name="newsletter" value="test5" />test5
        <div></div>

        <br/><br/>
        下拉列表 1: <br/>
```

```
<select name="test" multiple="multiple" style="height:100px">
    <option>浙江</option>
    <option selected="selected">湖南</option>
    <option>北京</option>
    <option selected="selected">天津</option>
    <option>广州</option>
    <option>湖北</option>
</select>

<br/><br/>
下拉列表 2: <br/>
<select name="test2" >
    <option>浙江</option>
    <option>湖南</option>
    <option selected="selected">北京</option>
    <option>天津</option>
    <option>广州</option>
    <option>湖北</option>
</select>
<div></div>
</form>
```

生成的效果图如图 2-3 所示。

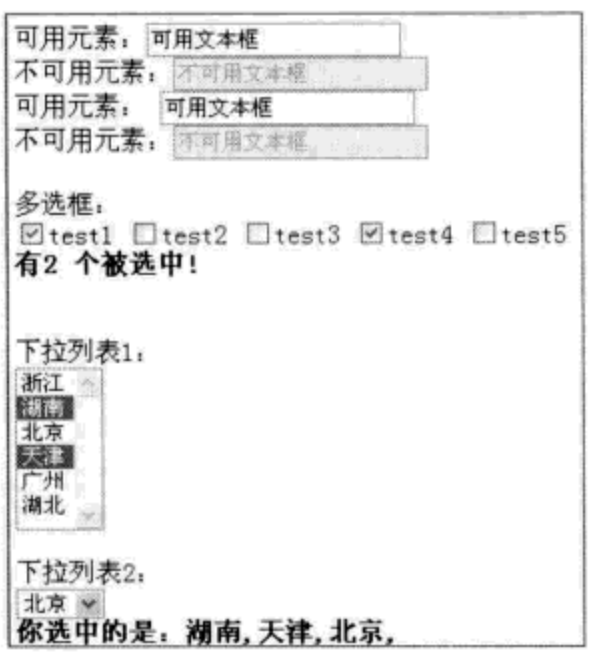


图 2-3　初始状态

现在用 jQuery 的表单过滤选择器来操作它们，示例如表 2-19 所示。

表 2-19 表单对象属性过滤示例

作　用	代　码	执 行 后
改变表单内可用<input>元素的值	$("#form1 input:enabled") .val("这里变化了！");	可用元素：这里变化了！ 不可用元素：不可用文本框 可用元素：这里变化了！ 不可用元素：不可用文本框
改变表单内不可用<input>元素的值	$("#form1 input:disabled") .val("这里变化了！");	可用元素：可用文本框 不可用元素：这里变化了！ 可用元素：可用文本框 不可用元素：这里变化了！
获取多选框选中的个数	$("input:checked").length;	多选框： ☑test1 ☑test2 ☑test3 ☐test4 ☑test5 有4 个被选中！
获取下拉框选中的内容	$("select :selected").text();	下拉列表1： 湖南 北京 天津 广州 湖北 下拉列表2： 浙江 你选中的是：北京,广州,湖北,浙江,

2.3.4 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。

表单选择器的介绍说明如表 2-20 所示。

表 2-20 表单对象属性过滤示例

选　择　器	描　　述	返回	示　　例
:input	选取所有的<input>、<textarea>、<select> 和<button>元素	集合元素	$(":input")选取所有<input>、<textarea>、<select>和<button>元素
:text	选取所有的单行文本框	集合元素	$(":text")选取所有的单行文本框
:password	选取所有的密码框	集合元素	$(":password")选取所有的密码框
:radio	选取所有的单选框	集合元素	$(":radio")选取所有的单选框
:checkbox	选取所有的多选框	集合元素	$(":checkbox")选取所有的复选框
:submit	选取所有的提交按钮	集合元素	$(":submit")选取所有的提交按钮
:image	选取所有的图像按钮	集合元素	$(":image")选取所有的图像按钮
:reset	选取所有的重置按钮	集合元素	$(":reset")选取所有的重置按钮
:button	选取所有的按钮	集合元素	$(":button")选取所有的按钮
:file	选取所有的上传域	集合元素	$(":file")选取所有的上传域
:hidden	选取所有不可见元素	集合元素	$(":hidden")选取所有不可见元素（已经在不可见性过滤选择器中讲解过）

下面把这些表单选择器运用到下面的表单中，对表单进行操作。

表单 HTML 代码如下：

```
<form id="form1" action="#">
    <input type="button" value="Button"/><br/>
    <input type="checkbox" name="c"/>1
    <input type="checkbox" name="c"/>2
    <input type="checkbox" name="c"/>3<br/>
    <input type="file" /><br/>
    <input type="hidden" /><div style="display:none">test</div><br/>
    <input type="image" /><br/>
    <input type="password" /><br/>
    <input type="radio" name="a"/>1<input type="radio" name="a"/>2<br/>
    <input type="reset" /><br/>
    <input type="submit" value="提交"/><br/>
    <input type="text" /><br/>
    <select><option>Option</option></select><br/>
    <textarea></textarea><br/>
    <button>Button</button><br/>
</form>
```

根据以上 HTML 代码，可以生成图 2-4 所示的页面效果。

图 2-4　初始状态

如果想得到表单内表单元素的个数，代码如下：

```
$("#form1 :input").length;     //注意与$("#form1 input")的区别
```

如果想得到表单内单行文本框的个数，代码如下：

```
$("#form1 :text").length;
```

如果想得到表单内密码框的个数，代码如下：

```
$("#form1 :password").length;
```

同理，其他表单选择器的操作与此类似。

2.4 应用 jQuery 改写示例

在本章开头部分，使用传统的 JavaScript 方法编写了 3 个简单的例子。

例子 1：给网页中所有的<p>元素添加 onclick 事件。

例子 2：使一个特定的表格隔行变色。

例子 3：对多选框进行操作，输出选中的多选框的个数。

下面利用刚学会的 jQuery 选择器以及隐式迭代的特性来重写这 3 个例子。

使用 jQuery 选择器重写例子 1，代码如下。

```
$("p").click(function(){ //获取页面中的所有<p>元素，给每一个<p>元素添加 onclick 事件
      //doing something
})
```

使用 jQuery 选择器重写例子 2，代码如下：

```
$('#tb tbody tr:even').css("backgroundColor","#888");
//获取 id 为 tb 的元素，然后寻找它下面的 tbody 标签，再寻找 tbody 下索引值是偶数的 tr 元素
//改变它的背景色
// css("property","value");用来设置 jQuery 对象的样式
```

使用 jQuery 选择器重写例子 3，代码如下：

```
$('#btn').click(function(){
          var length  = $("input[name='check']:checked").length;
          //先使用属性选择器，然后用表单对象属性过滤，最后获取 jQuery 对象的长度
          alert( "选中的个数为: "+ length );
});
```

通过几个简单的 jQuery 选择器就可以将例子改写，而且它们的运行效果与改写前是完全相同的。

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

根据 W3C 的规定，属性值中是不能含有这些特殊字符的，但在实际项目中偶尔会遇到表达式中含有“#”和“.”等特殊字符，如果按照普通的方式去处理出来的话就会出错。解

决此类错误的方法是使用转义符转义。

HTML 代码如下：

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

如果按照普通的方式来获取，例如：

```
$('#id#b');
$('#id[1]');
```

以上代码不能正确获取到元素，正确的写法如下：

```
$('#id\\#b');                    //转义特殊字符“#”
$('#id\\[1\\]');                 //转义特殊字符“[ ]”
```

注意 由于 jQuery.1.3.1 改变了选择器的引擎，在某些特殊情况下，导致转义失效，笔者已经提交了 Bug，也得到 jQuery 作者的回复，预计下一个版本能得到修复。

2. 属性选择器的引号问题

1.3.1 版本彻底放弃了 1.1.0 版本遗留下的@符号，如果你使用 1.3.1 以上的版本，那么你不能在属性前添加@符号，比如：

```
$(' div[@title="test" ');
```

正确的写法是去掉@符号，比如：

```
$(' div[title="test" ');
```

2.5.2 选择器中含有空格的注意事项

选择器中的空格也是不容忽视的，多一个空格或少一个空格也许会得到截然不同的结果。看下面这个例子，它的 HTML 代码如下：

```
<div class="test">
    <div style="display:none;">aa</div>
    <div style="display:none;">bb</div>
    <div style="display:none;">cc</div>
    <div class="test" style="display:none;">dd</div>
</div>
<div class="test" style="display:none;">ee</div>
<div class="test" style="display:none;">ff</div>
```

使用如下的 jQuery 选择器分别获取它们。

```
var $t_a = $('.test :hidden');     //带空格的 JQuery 选择器
var $t_b = $('.test:hidden');      //不带空格的 JQuery 选择器
var len_a = $t_a.length;
var len_b = $t_b.length;
alert("$('.test :hidden') = "+len_a);//输出 4
alert("$('.test:hidden') = "+len_b); /输出  3
```

之所以会出现不同的结果，是因为后代选择器与过滤选择器的不同。

```
var $t_a = $('.test :hidden');     //带空格的
```

以上代码是选取 class 为“test”的元素里面的隐藏元素。
而代码

```
var $t_b = $('.test:hidden');      //不带空格的
```

则是选取隐藏的 class 为“test”的元素。

2.6 案例研究——某网站品牌列表的效果

以下是某网站上的一个品牌列表的展示效果，用户进入该页面时，品牌列表默认是精简显示的（即不完整的品牌列表），如图 2-5 所示。

用户可以单击商品列表下方的“显示全部品牌”按钮来显示全部的品牌。

单击“显示全部品牌”按钮的同时，列表会将推荐的品牌的名字高亮显示，按钮里的文字也换成了“精简显示品牌”，如图 2-6 所示。

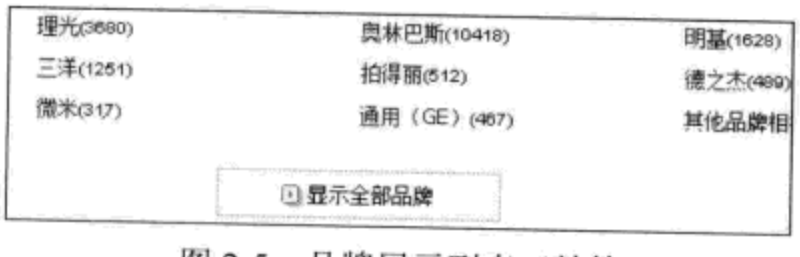


图 2-5　品牌展示列表（精简）

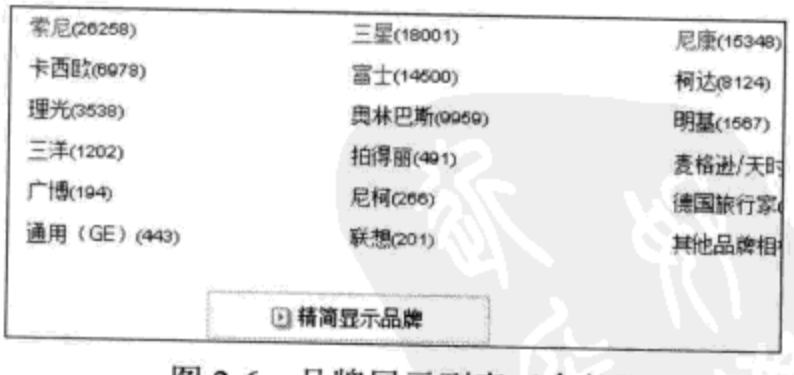


图 2-6　品牌展示列表（全部）

再次单击“精简显示品牌”按钮，即可回到图 2-5 所示的页面。

为了实现这个例子，首先需要设计它的 HTML 结构。HTML 代码如下：

```
<div class="SubCategoryBox">
    <ul>
        <li><a href="#">佳能</a><i>(30440) </i></li>
```

```
            <li><a href="#">索尼</a><i>(27220) </i></li>
            <li><a href="#">三星</a><i>(20808) </i></li>
            <li><a href="#">尼康</a><i>(17821) </i></li>
            <li><a href="#">松下</a><i>(12289) </i></li>
            <li><a href="#">卡西欧</a><i>(8242) </i></li>
            <li><a href="#">富士</a><i>(14894) </i></li>
            <li><a href="#">柯达</a><i>(9520) </i></li>
            <li><a href="#">宾得</a><i>(2195) </i></li>
            <li><a href="#">理光</a><i>(4114) </i></li>
            <li><a href="#">奥林巴斯</a><i>(12205) </i></li>
            <li><a href="#">明基</a><i>(1466) </i></li>
            <li><a href="#">爱国者</a><i>(3091) </i></li>
            <li><a href="#">其他品牌相机</a><i>(7275) </i></li>
        </ul>
        <div class="showmore">
            <a href="more.html"><span>显示全部品牌</span></a>
        </div>
    </div>
```

然后为上面的 HTML 代码添加 CSS 样式。

页面初始化的效果如图 2-7 所示。

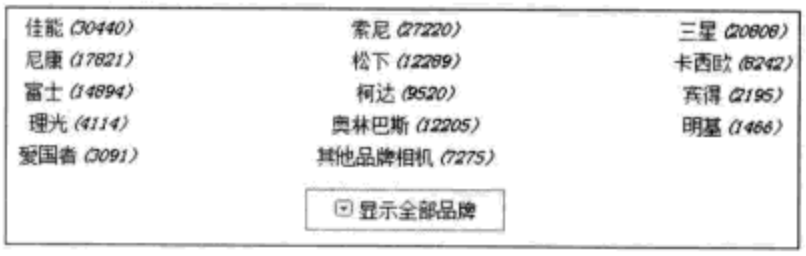


图 2-7　品牌展示列表（精简）

接下来为这个页面添加一些交互效果，要做的工作有以下几项。

（1）从第 7 条开始隐藏后面的品牌（最后一条“其它品牌相机”除外）。

（2）当用户单击“显示全部品牌”按钮时，将执行以下操作。

① 显示隐藏的品牌。

② “显示全部品牌”按钮文本切换成“精简显示品牌”。

③ 高亮推荐品牌。

（3）当用户单击“精简显示品牌”按钮时，将执行以下操作。

① 从第 5 条开始隐藏后面的品牌（最后一条“其它品牌相机”除外）。

② “精简显示品牌”按钮文本切换成“显示全部品牌”。

③ 去掉高亮显示的推荐品牌。

（4）循环进行第（2）步和第（3）步。

下面逐步来完成以上的效果。

（1）从第 5 条开始隐藏后面的品牌（最后一条“其他品牌相机”除外）。

```
var $category = $('ul li:gt(5):not(:last)');
$category.hide();                                    // 隐藏上面获取到的 jQuery 对象
```

$('ul li:gt(5):not(:last)')的意思是先获取<ul>元素下索引值大于 5 的<li>元素的集合元素，然后去掉集合元素中的最后一个元素。这样，即可将从第 7 条开始至倒数第 2 条的所有品牌都获取到。

最后通过 hide()方法隐藏这些元素。

（2）当用户单击“显示全部品牌”按钮时，执行以下操作。

首先获取到按钮，代码如下：

```
var $toggleBtn = $('div.showmore > a');          //获取“显示全部品牌”按钮
```

然后给按钮添加事件，使用 show()方法把隐藏的品牌列表显示出来，代码如下：

```
$toggleBtn.click(function(){
    $category.show();                                //显示全部品牌
    return false;                                    //超链接不跳转
});
```

由于给超链接添加 onclick 事件，因此需要使用“eturn false”语句让浏览器认为用户没有单击该超链接，从而阻止该超链接跳转。

之后，需要将“显示全部品牌”按钮文本切换成“精简显示品牌”，代码如下：

```
$('.showmore a span')
        .css("background","url(img/up.gif) no-repeat 0 0")
        .text("精简显示品牌");                          //这里使用了链式操作
```

这里完成了两步操作，即把按钮的背景图片换成向上的图片，同时也改变了按钮文本内容，将其替换成“精简显示品牌”。

接下来需要高亮推荐品牌，代码如下：

```
$('ul li').filter(":contains('佳能'),:contains('尼康'),:contains('奥林巴斯')")
            .addClass("promoted");;                  //添加高亮样式
```

使用 filter()方法筛选出符合要求的品牌，然后为它们添加 promoted 样式。在这里推荐了 3 个品，即牌佳能、尼康和奥林巴斯。

此时，完成的 jQuery 代码如下：

```
$(function(){                                        //等待 DOM 加载完毕
    var $category = $('ul li:gt(5):not(:last)');
                                    //获得索引值大于 5 的品牌集集合对象(除最后一条)
```

```
        $category.hide();                                    //隐藏上面获取到的 jQuery 对象
        var $toggleBtn = $('div.showmore > a');              //获取“显示全部品牌”按钮
        $toggleBtn.click(function(){
              $category.show();                              //显示$category
              $('.showmore a span')
                    .css("background","url(img/up.gif) no-repeat 0 0")
                    .text("精简显示品牌");                    //改变背景图片和文本
              $('ul li').filter(":contains('佳能'),:contains('尼康
'),:contains('奥林巴斯')")
                    .addClass("promoted");                   //添加高亮样式
              return false;                                  //超链接不跳转
        })
    })
```

运行上面的代码，单击“显示全部品牌”按钮后，显示图 2-8 所示的效果，此时已经能够正常显示全部品牌了。

图 2-8　当按钮被单击后

上面代码中用到的几个 jQuery 方法的意思如下。

- show(): 显示隐藏的匹配元素。
- css(name,value): 给元素设置样式。

注意

- text(string): 设置所有匹配元素的文本内容。
- filter(expr): 筛选出与指定表达式匹配的元素集合，其中 expr 可以是多个选择器的组合。
- addClass(class): 为匹配的元素添加指定的类名。

（3）当用户单击“精简显示品牌”按钮时，将执行以下操作。

由于用户单击的是同一个按钮，因此事件仍然是在刚才的按钮元素上。要将切换两种状态的效果在一个按钮上进行，可以通过判断元素的显示或者隐藏来达到目的，代码结构如下：

```
if(元素显示 ){
    //元素隐藏 ①
}else{
    //元素显示 ②
}
```

代码②就是第（2）步的内容，接下来只需要完成代码①的内容即可。

在 jQuery 中，与 show()方法相反的是 hide()方法，因此可以使用 hide()方法将品牌隐藏起来，代码如下：

```
$category.hide();                                    //隐藏$category
```

然后将“精简显示品牌”按钮文本切换成“显示全部品牌”，同时按钮图片换成向下的图片，这一步与前面类似，只不过是图片路径和文本内容不同而已，代码如下：

```
$('.showmore a span')
      .css("background","url(img/down.gif) no-repeat 0 0")
      .text("显示全部品牌");                          //改变背景图片和文本
```

接下来需要去掉所有品牌的高亮显示状态，此时可以使用 removeClass()方法来完成，代码如下：

```
$('ul li').removeClass("promoted");          //去掉高亮样式
```

它将去掉所有<li>元素上的“promoted”样式，即去掉了品牌的高亮状态。

注意 removeClass(class)的功能和 addClass(class)的功能正好相反。addClass(class)的功能是为匹配的元素添加指定的类，而 removeClass(class)则是从匹配的元素中删除指定的类。

至此完成代码①。

最后通过判断元素是否显示来分别执行代码①和代码②，代码如下：

```
if($category 判断.is(":visible")){          //如果元素显示，则执行对应的代码
```

之后即可将代码①和代码②插入相应的位置。jQuery 代码如下：

```
if($category.is(":visible")){                        //如果元素显示
     $category.hide();                               //隐藏$category
     $('.showmore a span')
          .css("background","url(img/down.gif) no-repeat 0 0")
          .text("显示全部品牌");                      //改变背景图片和文本
     $('ul li').removeClass("promoted"); //去掉高亮样式
}else{
     $category.show();                               //显示$category
     $('.showmore a span')
          .css("background","url(img/up.gif) no-repeat 0 0")
          .text("精简显示品牌");                      //改变背景图片和文本
     $('ul li').filter(":contains('佳能'),:contains('尼康'),:contains('
```

```
奥林巴斯')")
                .addClass("promoted");                    //添加高亮样式
    }
```

至此任务完成，完整的 jQuery 代码如下：

```
    $(function(){                                         //等待 DOM 加载完毕.
            var $category = $('ul li:gt(5):not(:last)');
                                  //获得索引值大于 5 的品牌集合对象(除最后一条)
            $category.hide();                             //隐藏上面获取到的 jQuery 对象
            var $toggleBtn = $('div.showmore > a'); //获取“显示全部品牌”按钮
            $toggleBtn.click(function(){          //给按钮添加 onclick 事件
                if($category.is(":visible")){ //如果元素显示
                        $category.hide();  //隐藏$category
                        $'(.showmore a span')
                              .css("background","url(img/down.gif)no-
repeat 0 0")
                              .text("显示全部品牌"); //改变背景图片和文本
                        $('ul li').removeClass("promoted"); //去掉高亮样式
                }else{
                              $category.show();  //显示$category
                              $('.showmore a span')
                                    .css("background","url(img/up.gif)no-
repeat 0 0")
                                    .text("精简显示品牌"); //改变背景图片和文本
                              $('ul li').filter(":contains('佳能'),:contains('
尼康'),:contains('奥林巴斯')")
                                          .addClass("promoted"); //添加高亮样式
                }
                return false;                                 //超链接不跳转
            })
    })
```

运行代码后，单击按钮，品牌列表会在“全部”和“精简”两种效果之间循环切换，显示效果如图 2-9 和图 2-10 所示。

图 2-9　精简模式

图 2-10　全部模式

在 jQuery 中有一个方法更适合上面的情况，它能给一个按钮添加一组交互事件，而不需要像上例一样去判断，上例的代码如下：

```
toggleBtn.click(function(){
    if($category.is(":visible")){          //如果元素显示
        //元素隐藏        代码①
}else{
        //元素显示        代码②
    }
})
```

如果改成 toggle()方法，代码则可以直接写成以下形式：

```
$toggleBtn.toggle(function(){                    //toggle()方法用来交替一组动作
        //显示元素        代码③
  },function(){
        //隐藏元素        代码④
})
```

当单击按钮后，脚本会对代码③和代码④进行交替处理。

jQuery 还提供了很多简单易用的方法，上面讲解的 toggle()方法只是其中的一种，这些方法将在后面的章节中进行详细介绍。

注意　在本例中，如果用户禁用了 JavaScript 的功能，品牌列表仍然能够完全显示，当用户单击“显示全部品牌”按钮的时候，会跳转到 more.html 页面来显示品牌列表。作为一名专业的开发者，必须要考虑到禁用或者不支持 JavaScript 的浏览器（用户代理）。另外，这点对于搜索引擎优化也特别有帮助，毕竟当前的搜索引擎爬虫基本都不支持 JavaScript。

2.7 其他选择器

2.7.1 jQuery 提供的选择器的扩展

虽然 jQuery 提供了许多实用的选择器，但还是有可能不能满足各种多变的业务需要，不过 jQuery 选择器是可以进一步扩展的。

1. MoreSelectors for jQuery

这是一个 jQuery 的插件，用于增加更多的选择器，例如.color 可以匹配颜色，:colIndex 可以匹配表格中的列，:focus 可以匹配获取焦点的元素等等。

插件地址：http://plugins.jquery.com/project/moreSelectors。

2. Basic XPath

这个插件可以让用户使用基本的 XPath。jQuery 最开始支持 XPath 选择器，但由于使用人数不多，且降低了选择器匹配的效率，因此在 1.2 以后的版本中取消了默认对 XPath 选择器的支持，改为通过插件来实现。

插件地址：http://plugins.jquery.com/project/xpath。

2.7.2 其他使用 CSS 选择器的方法

除了 jQuery 提供了强大的选择器支持外，也有其他一些 JavaScript 脚本也提供了此类纯粹的 CSS 选择器的支持。

1. document.getElementsBySelector()

早在 2003 年，Simon Willison 就编写了该脚本，它的作用是通过选择器来获取文档元素。读者可以通过以下代码获取元素。

```
document.getElementsBySelector('div#main p a.external')
```

该脚本最新版本为 0.4 版，更新日期为 2003 年 3 月 25 日。

发布地址：http://simonwillison.net/2003/Mar/25/getElementsBySelector/。

2. cssQuery()

这是 Dean Edwards 编写的一款利用 CSS 选择器查找元素的脚本。支持所有 CSS1、CSS2 以及部分 CSS3 选择器，jQuery 的选择器其实是源自于此，它支持一些 jQuery 尚不支持的选择器，例如 E:link、E:nth-last-child(n)、E:root、E:lang(fr)、E:target 和 E[foo|="bar"]等。语法结构如下：

```
elements = cssQuery(selector [, from]);
```

该脚本最新版本为 2.0.2 版，更新日期为 2005 年 9 月 10 日。

官方网站：http://dean.edwards.name/my/cssQuery/。

3. querySelectorAll()

这不是一个脚本库，而是 W3C 在 Selectors API 草案中提到的方法，该草案的最新版本是在 2007 年 12 月 21 日发布的。此方法也是用于实现通过 CSS 选择器来获取元素的。IE 8 的 Beta2 中已经率先实现了此方法。相信其他几大浏览器也很快就能实现此方法。

JQuery 的作者 John Resig 也表示将会利用 querySelectorAll()这个浏览器原生的方法来重构 jQuery 的选择器，同时增加一些 jQuery 扩展的选择器，届时 jQuery 选择器的执行效率也将大大提高。

W3C Selectors API：http://www.w3.org/TR/selectors-api/。

2.8 小结

本章详细讲解了 jQuery 中的各种类型的选择器。选择器是行为与文档内容之间连接的纽带，选择器的最终目的就是能够轻松地找到文档中的元素。

在本章的开始列举了 3 个用传统 JavaScript 方法写的简单例子，然后介绍了 jQuery 选择器，并用所学的 jQuery 选择器以及隐式迭代的特性将例子进行改写。此外还讲解了选择器中的一些注意事项，希望能引起初学者的注意。最后以某网站上一个品牌列表作为例子，加深读者对 jQuery 选择器用法的理解。

jQuery

第3章

jQuery 中的 DOM 操作

DOM 是 Document Object Model 的缩写，意思是文档对象模型。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。简单来说，DOM 解决了 Netscape 的 JavaScript 和 Microsoft 的 JScript 之间的冲突，给予了 Web 设计师和开发者一套标准的方法，让他们能够轻松获取和操作网页中的数据、脚本和表现层对象。

3.1 DOM 操作的分类

一般来说，DOM 操作分为 3 个方面，即 DOM Core（核心）、HTML-DOM 和 CSS-DOM。

1. DOM Core

DOM Core 并不专属于 JavaScript，任何一种支持 DOM 的程序设计语言都可以使用它。它的用途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。

JavaScript 中的 getElmentById()、getElementsByTagName()、getAttribute()和 setAttribute()等方法，这些都是 DOM Core 的组成部分。

例如：

- 使用 DOM Core 来获取表单对象的方法：

```
document.getElementsByTagName("form");
```

- 使用 DOM Core 来获取某元素的 src 属性的方法：

```
element.getAttribute("src");
```

2. HTML_DOM

在使用 JavaScript 和 DOM 为 HTML 文件编写脚本时，有许多专属于 HTML-DOM 的属

性。HTML-DOM 的出现甚至比 DOM Core 还要早，它提供了一些更简明的记号来描述各种 HTML 元素的属性。

例如：

- 使用 HTML-DOM 来获取表单对象的方法：

```
document.forms //HTML-DOM 提供了一个 forms 对象
```

- 使用 HTML-DOM 来获取某元素的 src 属性的方法：

```
element.src;
```

通过上面所说的方法，可以发现获取某些对象、属性既可以用 DOM Core 来实现，也可以使用 HTML-DOM 实现。相比较而言 HTML-DOM 的代码通常比较简短，不过它只能用来处理 Web 文档。

3. CSS_DOM

CSS-DOM 是针对 CSS 的操作。在 JavaScript 中，CSS-DOM 技术的主要作用是获取和设置 style 对象的各种属性。通过改变 style 对象的各种属性，可以使网页呈现出各种不同的效果。

例如：设置某元素 style 对象字体颜色的方法：

```
element.style.color = "red";
```

jQuery 作为 JavaScript 库，继承并发扬了 JavaScript 对 DOM 对象的操作的特性，使开发人员能方便地操作 DOM 对象。下面详细介绍 jQuery 中的各种 DOM 操作。

3.2 jQuery 中的 DOM 操作

为了能全面地讲解 DOM 操作，首先需要构建一个网页。因为每一张网页都能用 DOM 表示出来，而每一份 DOM 都可以看作一棵 DOM 树。构建的网页效果如图 3-1 所示。

HTML 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>dom</title>
</head>
<body>
    <p title="选择你最喜欢的水果." >你最喜欢的水果是?</p>
```

```
        <ul>
          <li title='苹果'>苹果</li>
          <li title='橘子'>橘子</li>
          <li title='菠萝'>菠萝</li>
        </ul>
    </body>
    </html>
```

根据上面的网页结构构建出一棵 DOM 树，如图 3-2 所示。

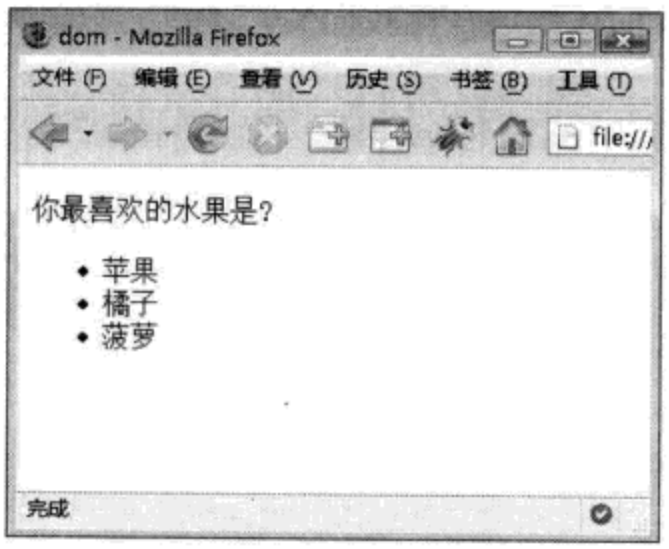


图 3-1 构建的网页

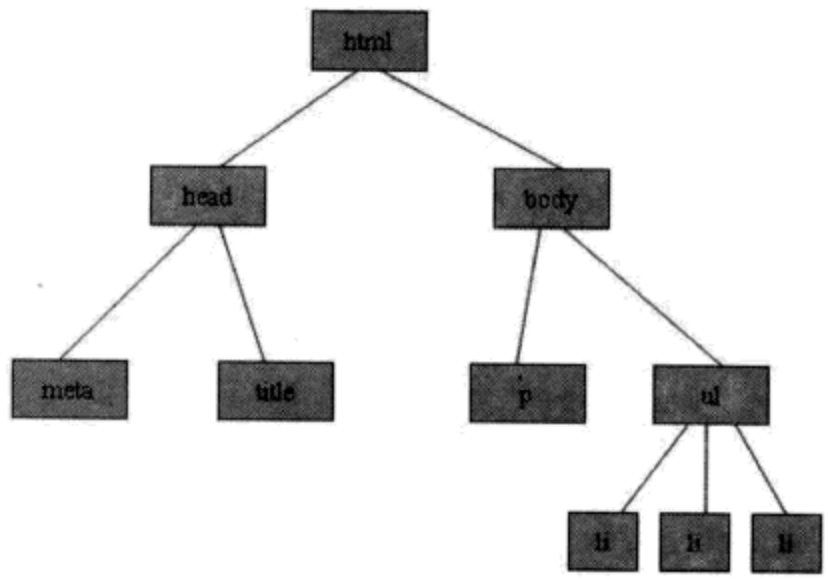


图 3-2 DOM 结构

接下来，对 DOM 的各种操作都将围绕这棵 DOM 树而展开。

3.2.1 查找节点

使用 jQuery 在文档树上查找节点非常容易，可以通过在第 2 章介绍的 jQuery 选择器来完成。

1. 查找元素节点

获取元素节点并打印出它的文本内容，jQuery 代码如下：

```
var $li = $("ul li:eq(1)");          //获取<ul>里第 2 个<li>节点
var li_txt = $li.text();             //获取第 2 个<li>元素节点的文本内容
alert(li_txt);                       //打印文本内容
```

以上代码获取了<ul>元素里第 2 个<li>节点，并将它的文本内容“橘子”打印出来，效果如图 3-3 所示。

2. 查找属性节点

利用 jQuery 选择器查找到需要的元素之后，就可以使用 attr()方法来获取它的各种属性

的值。attr()方法的参数可以是一个，也可以是两个。当参数是一个时，则是要查询的属性的名字，例如：

获取属性节点并打印出它的文本内容，jQuery 代码如下：

```
var $para = $("p");                    //获取<p>节点
var p_txt = $para.attr("title");       //获取<p>元素节点属性 title
alert(p_txt);                          //打印 title 属性值
```

以上代码获取了<p>节点，并将它的 title 属性的值设为“选择你最喜欢的水果.”打印出来的效果如图 3-4 所示。

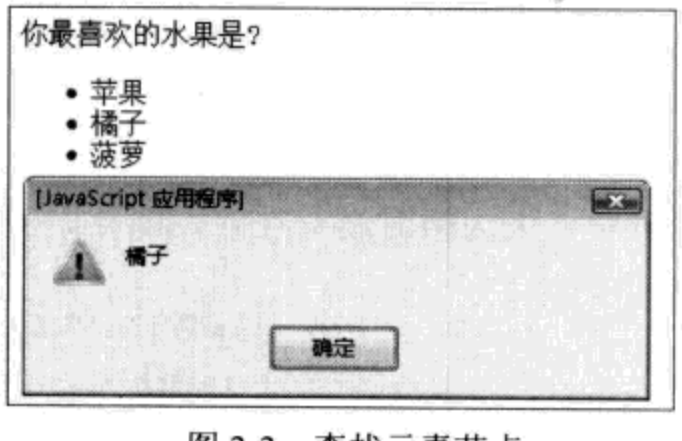


图 3-3　查找元素节点

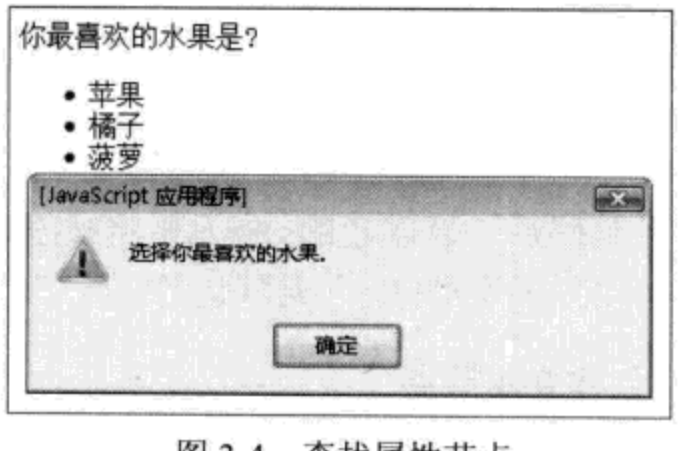


图 3-4　查找属性节点

3.2.2　创建节点

从第 3.2.1 小节可知，用 jQuery 选择器能够快捷而轻松地查找到文档中的某个特定的元素节点，然后可以用 attr()方法来获取元素的各种属性的值。

真正的 DOM 操作并非这么简单。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。

1．创建元素节点

例如要创建两个<li>元素节点，并且要把它们作为<ul>元素节点的子节点添加到 DOM 节点树上。完成这个任务需要两个步骤。

（1）创建两个<li>新元素。

（2）将这两个新元素插入文档中。

第（1）个步骤可以使用 jQuery 的工厂函数$()来完成，格式如下：

```
$( html );
```

$(html)方法会根据传入的 HTML 标记字符串，创建一个 DOM 对象，并将这个 DOM 对象包装成一个 jQuery 对象后返回。

首先创建两个<li>元素，jQuery 代码如下：