第五章　jQuery进阶

本章导读

HTML搭起网页的架子，CSS修饰网页的样子，jQuery就如同刀子，可以方便地以多种方式找到网页中的HTML元素，调整该元素的内容，属性以及CSS。jQuery如同美容刀，重塑网页，让网页灵动起来。jQuery之Query的含义就是“查找”，就是方便查找网页元素，j是JavaScript的简写，就是以JavaScript的方式方便地查找网页，并在此基础上操纵网页。

本章首先讲解jQuery的选择参数，并在此基础上讲解查找和筛选元素以及网页内容控制、CSS控制、属性控制、动画等，几乎涵盖了jQuery除事件和AJAX之外的所有内容。

学习目标：

1. 深入理解网页结构；
2. 深入理解$符号与网页结构的关系；
3. 深入理解jQuery的元素选择方法；
4. 熟悉jQuery操纵属性、Class以及CSS、html等；
5. 熟悉jQuery的动画操作指令；

本章目录

---

jQuery是一个基于JavaScript的开源程序库，由John Resig于2006年1月创建。在HTML元素选择、属性读写、事件处理、动画和Ajax等多方面，jQuery都提供简捷高效的处理方法，其独特而优雅的代码风格改变了JavaScript程序员编写程序的设计方式和思路。

在不同浏览器之间，jQuery有良好的兼容性，很好地缓解了Web工程师所面临的这一挑战。自发布以来，快速在全球得到广泛应用，并且开发出各种各样基于jQuery的插件，更是促进了jQuery的推广和应用。图5-1是Google根据全球搜索数据制作出的jQuery与其他JavaScript程序库流行趋势比较图。从图中可以看出自jQuery2006年推出以来，快速取得领先优势。

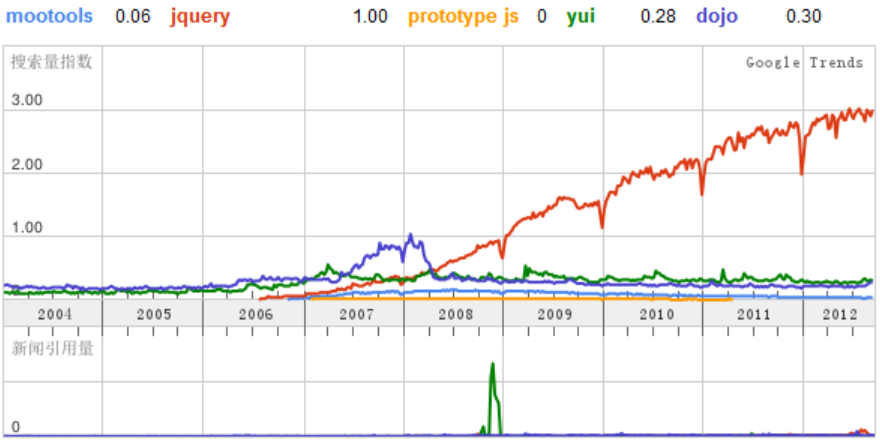


图5-1　2004-2012年JavaScript框架流行趋势图

第一节　理解网页

1、网页结构

图5-2是用于展示“三国人物关系”的网页，图5-3是其网页代码结构图示，从中可以看出，网页代码如同一棵树，也有“根干枝叶果”，其他网页代码也有类似的树形结构。所有网页都是从根Html开始，然后有Head和Body两个主干节点，主干节点下还有不少分支，在例程5-1所示网页中，Body下有三个Div分支，在其之下还有三个更细的Div分支，其中Class名为BigMan的分支下还有5个P分支，也许可以可以称之为“树叶”了，其文本内容可以称之为“果”。“根干枝叶果”没有特别含义，根是确定的，其他所有元素都是HTML下的节点，“根干枝叶果”强调其层次关系。在jQuery中，其操作方法一致。

网页中的“根干枝叶果”用HTML元素表示，每个HTML元素还可以称之为节点(node)。称之为元素，更加强调HTML元素本身，称之为节点更加强调该标记所在位置的上下前后关系。网页中的一个节点可能被包含在其他元素中，也就是说该节点有长辈。长辈不仅限于父节点，还可以向上追溯很多辈，直到根节点为止(Html)，在网页代码中，根就是HTML这个唯一的节点。一个节点也可以有很多晚辈，子子孙孙可以很多层。另外，一个节点除了长辈和晚辈这样的上下关系外，还有兄弟姐妹这样的前后关系，也就是同辈关系。理解网页结构有助于定位元素，操纵网页。

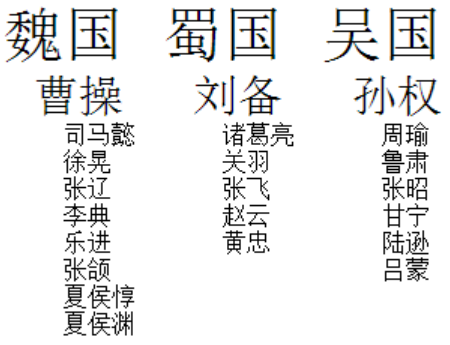


图5-2　三国人物关系示意图

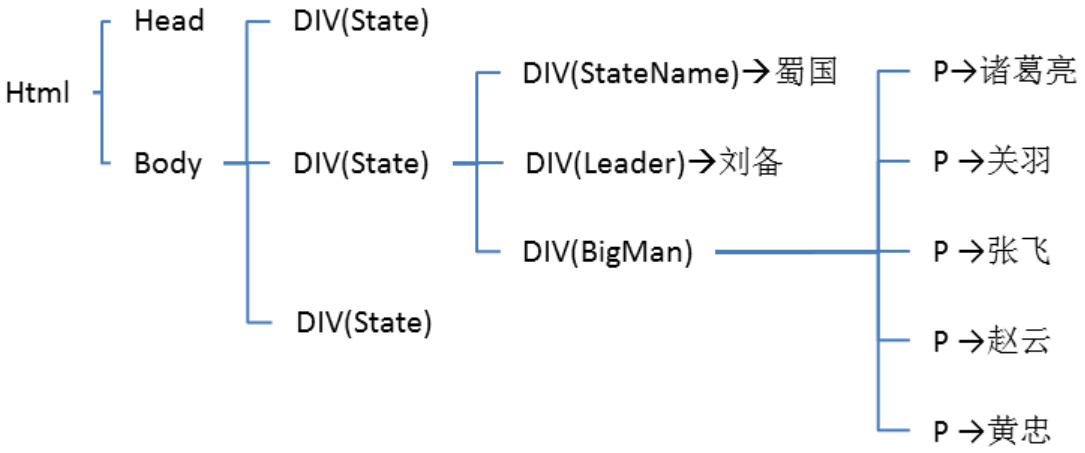


图5-3　网页树形结构示意图

例程5-1

```
<HTML>
    <HEAD>
        <TITLE>网页结构</TITLE>
        <Style type=text/css>
            P{margin:0px}
            .State{margin-left:10px;vertical-align:top;display:inline-block;}
            .StateName{font-size:40px;}
            .Leader{font-size:30px;margin-left:20px}
            .BigMan{font-size:16px;margin-left:40px;}
            .LightBigMan{color:red}
            .LightIt{color:red}
        </Style>
        <SCRIPT LANGUAGE="JavaScript" src="./jQuery-1.9.1.min.js"></SCRIPT>
    </HEAD>
    <BODY>
        <Div class="State">
            <Div class="StateName">魏国</Div>
            <Div class="Leader">曹操</Div>
            <Div class="BigMan">
                <P>司马懿</P>
                <P>徐晃</P>
                <P>张辽</P>
                <P>李典</P>
                <P>乐进</P>
                <P>张颌</P>
                <P>夏侯惇</P>
                <P>夏侯渊</P>
            </Div>
        </Div>
        <Div class="State">
            <Div class="StateName">蜀国</Div>
            <Div class="Leader">刘备</Div>
            <Div class="BigMan">
                <P>诸葛亮</P>
                <P>关羽</P>
                <P>张飞</P>
                <P>赵云</P>
                <P>黄忠</P>
            </Div>
        </Div>
        <Div class="State">
            <Div class="StateName">吴国</Div>
            <Div class="Leader">孙权</Div>
            <Div class="BigMan">
                <P>周瑜</P>
```

```
第46行                    <P>鲁肃</P>
第47行                    <P>张昭</P>
第48行                    <P>甘宁</P>
第49行                    <P>陆逊</P>
第50行                    <P>吕蒙</P>
第51行                </Div>
第52行            </Div>
第53行        </BODY>
第54行    </HTML>
```

2、操纵网页

网页元素在网页中的效果主要以下7个方面决定：元素类型(tag type)、内容(content)、样式(style)、属性(attribute)、事件(event)、时间(time)以及上下文关系(context)等。控制这7个方面，就控制了网页元素的效果。

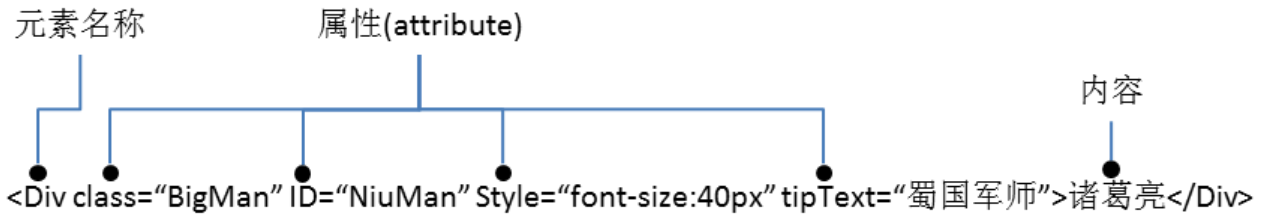


图5-4　网页元素组成图

网页是活的。通过程序控制网页元素的7个方面，网页可以实时呈现效果。正是网页具有“活的”特征才使得网页效果丰富多彩。在控制网页效果时，需要首先选中网页中单个或多个元素，然后再通过各种方法控制网页的方方面面。

在制作网页过程中，首先是选择不同类型的网页元素，不同的网页元素有不同的网页呈现效果，如BR是分行，而IMG用于显示图片，这两个不同类型的元素决定了效果的截然不同。对于某些元素，如P、Div和Span其默认效果差别较大，但是可以通过设置其Style属性予以修改，如默认的display属性、margin属性、line-height属性等。

相同的元素但不同的内容其效果也不相同。$.html()可以读取选中元素的内容，而$.html(内容)则可以修改选中元素的内容。

在元素标记的尖括号内，可以设置属性，包括HTML定义的标准属性，也包括用户自定义的属性，如Img的src用于设置显示图片的位置。读写(Read&Write)属性值，可以改变元素的效果。$.attr(属性名称)可以读取该属性名称的值，$.attr(属性名称,属性值)可以修改(相当于Write)该属性的值。

Style是属性的一种，几乎所有的元素都支持Style属性(BR、Title、Head等是例外)，如Div、Span、P、Img等。Style有不少CSS项目可以设置，以支持不同的效果。由于控制Style对于元素的效果影响很大，所以jQuery提供了多种Style控制方式，如$.css()、$.addClass()以及$.removeClass等。相关内容参见本章“Style控制”。

事件(Event)是指元素对外部发生动作(action)的反应，比如：鼠标移动(mouseover)、鼠标点击(click)和页面载入(load)等。元素能对事件作出反应，该元素必须绑定事件及其相应的处理函数。相关内容参见第六章“事件”。

网页对时间变化作出的反应，让网页自动呈现效果，很多网站的首页图片能自动切换，实际上就网页对时间作出的反应。除此以外，jQuery提供的动画，也是一种时间控制，即在指定时间，网页Style属性CSS条目发生变化，即呈现动画效果。相关内容参见本章“动画效果”。

网页元素内的内容可以控制[通过$.html(内容)]，还可以控制器前后内外的内容，方式是覆盖、插入、替换、复制、包裹、删除等。相关内容参见本章“文档处理”。

第二节　元素选择

操作网页的第一步选择元素，即确定操作对象，可以一次选定，也可以在初次选定元素的基础上，筛选(Filter)出其中一部分，或者以选定元素为基础，根据元素的树形关系查找指定元素。初次选择元素用$(选择参数)，在此基础上的筛选和查找元素是用点运算符(.)和相应的函数。筛选和查找可以执行一次，也可以串联执行多次，如例程5-2所示。

例程5-2

```
第1行    <SCRIPT LANGUAGE="JavaScript">
```

```
第2行          //.addClass("LightIt")前的代码是元素选择代码
第3行
第4行          $(".StateName").addClass("LightIt");//选中魏国、蜀国、吴国
第5行
第6行          //$(".BigMan")选中所有名为BigMan的Div，eq(0)筛选出第一个1个
第7行          //即魏国下的名为BigMan的Div元素
第8行          $(".BigMan").eq(0).addClass("LightIt");
第9行
第10行         //eq(2)后的children("p")的功能是查找P元素，并筛选出奇数部分
第11行         $(".BigMan").eq(2).children("P").filter(":odd").addClass("LightIt");
第12行     </SCRIPT>
```

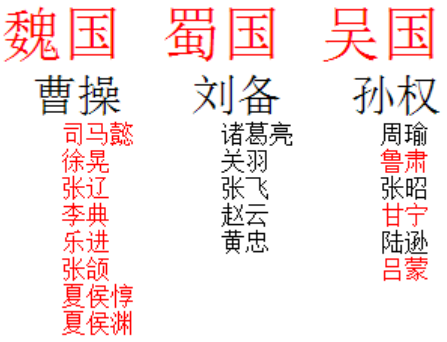


图5-5　不同元素选择效果图

在例程5-2第11行，　$(".BigMan")是初次选择，此时jQuery从整个网页中找到class名为BigMan的全部元素；其后的eq(2)是筛选，即在已经找到的全部元素中找到编号为2的元素(在已有集合中找到满足条件的子集)；eq(2)后面的children("P")是查找，即为eq(2)执行结果(找到的元素)为出发点，找到参数约定条件的元素，此处为P元素；P元素可能很多，filter(":odd")则是筛选出部分元素，此处是筛选出编号为奇数的元素。最后的addClass("LightIt")为选中的P元素附加上名为LightIt的class。在本语句中，执行了初次选择、筛选、查找等定位元素的操作。

虽然JavaScript也提供了选择网页元素的基本命令，但使用不方便较复杂，且各个浏览器之间并兼容性不好，而jQuery提供了方便简单且兼容多种浏览器元素选择方式，使得网页开发效率大大提高。没有基于JavaScript的jQuery程序库的协助，操作网页元素是一个相当繁杂的工作。有了jQuery后，一切都得到了改善。jQuery提供了极为强大的元素定位指令，并以此基础上操控元素的每一部分，除此以外，元素的动画、事件等也都以元素定位为基础。

jQuery提供了丰富的选择参数，这些参数可以用于初选，也可应用在不少筛选和查找函数中。在jQuery选择参数的帮助下，定位网页中的元素变得相当简单。根据选择参数的不同，可以分为：基本参数、层级参数、序号参数、属性参数、子元素参数、内容参数和表单参数等。

1、选择参数

在$(".BigMan")中，$()从本质上讲就是一个函数，其功能是选中网页中的元素，".BigMan"是函数的参数，是字符型数据；eq(2)是筛选函数，这个函数规定其参数为整数数值，不支持其他选择参数，而children("P")查找函数支持选择参数，当省略时则默认选中所有子元素；filter(":odd")是筛选函数，也支持选择参数。在应用各种选择函数时，要注意该函数是否支持选择参数。

⑴　基本参数

基本参数包括：ID选择、Class选择、元素选择以及全部选择和多重选择等。

(a)　ID参数

在网页代码中，绝大多数HTML元素都有ID属性，且其ID属性值在整个网页文档中是唯一的。和CSS一样，ID选择器需要在名称前加上#，且大小写与网页代码中的ID属性值保持一致，例程5-3是ID选择器的一个应用实例，其第24行到第30行之间，是有关ID选择器的应用。当鼠标移动到ID名为ThreeKingdomPoet的Div时，将为其增加一个名为BGRed的Class，该Class的显示效果已经定义在Head-Style中；当鼠标移出时，删除该Class。

例程5-3

```
第1行    <HTML>
第2行        <HEAD>
```

```
第3行                    <TITLE>基本选择器</TITLE>
第4行                    <Style type=text/css>
第5行                        #ThreeKingdomPoet{width:300px;border:2px solid red;text-align:Center;padding:10p
x;}
第6行                        P{text-align:Left;text-indent:2em}
第7行                        .BGRed{background-color:red;}
第8行                        .LightP{color:White}
第9行                        .PoetTitle{font-size:24px;font-weight:bold}
第10行                       .LightPoetTitle{color:White}
第11行                   </Style>
第12行                   <SCRIPT LANGUAGE="JavaScript" src="./jQuery-1.4.2.min.js"></SCRIPT>
第13行           </HEAD>
第14行           <BODY>
第15行                   <Div ID="ThreeKingdomPoet">
第16行                       <Div class="PoetTitle">念奴娇·赤壁怀古</Div>
第17行                       【宋】苏轼
第18行                       <P>大江东去，浪淘尽。千古风流人物。故垒西边，人道是，三国周郎赤壁。
第19行                       乱石崩云，惊涛拍岸，卷起千堆雪。江山如画，一时多少豪杰！</P>
第20行                       <P>遥想公瑾当年，小乔初嫁了，雄姿英发，羽扇纶巾，谈笑间，樯橹灰飞烟灭。
第21行                       故国神游，多情应笑我，早生华发。人间如梦，一樽还酹江月。</P>
第22行                   </Div>
第23行                   <SCRIPT LANGUAGE="JavaScript">
第24行                       //ID选择器------开始
第25行                       $("#ThreeKingdomPoet").mouseover(function(){
第26行                           $(this).addClass("BGRed");
第27行                       });
第28行                       $("#ThreeKingdomPoet").mouseout(function(){
第29行                           $(this).removeClass("BGRed");
第30行                       });//ID选择器------结束
第31行
第32行                       //HTML元素选择器-----开始
第33行                       $("P").mouseover(function(){
第34行                           $(this).addClass("LightP");
第35行                       });
第36行                       $("P").mouseout(function(){
第37行                           $(this).removeClass("LightP");
第38行                       });//HTML元素选择器-----结束
第39行
第40行                       //Class选择器-----开始
第41行                       $(".PoetTitle").mouseover(function(){
第42行                           $(this).addClass("LightPoetTitle");
第43行                       });
第44行                       $(".PoetTitle").mouseout(function(){
第45行                           $(this).removeClass("LightPoetTitle");
第46行                       });//Class选择器-----结束
```

```
第47行                    </SCRIPT>
第48行                </BODY>
第49行            </HTML>
```

(b) Class选择

在网页代码中，绝大多数HTML元素都有Class属性，和ID属性不同，其属性值在整个网页文档中可以多次出现，可使其保持相同或近似的显示风格。 和CSS一样，Class选择器需要在名称前加上英文句点(.)，且大小写与网页代码中的Class属性值保持一致，例程5-3中有Class选择器的应用实例，其第40行到第46行之间，是有关Class选择器的应用。

(c) 元素选择

选择HTML元素，只需标记名称。例程5-3中有HTML标记选择器的应用实例，其第32行到第38行之间，是有关HTML标记选择器的应用。

(d) 全部选择

是jQuery的全部选择器，当执行$("")时，将选中网页中所有元素，这些元素包括Body及其在内的所有元素。

(e) 多重选择

和CSS一样，jQuery选择参数可以分组组合，其格式如下：

```
$("selector1,selector2,…,selectorN")
```

在多重选择中，每个选择之间用英文半角逗号分开。每个选择参数独立应用，互不干扰，选出的元素集合是每个选择参数结果的并集。

例程5-4是在例程5-1的基础增加的JavaScript代码。在第3行和第7行，就一次选中了Class名为StateName和Leader的所有元素。

例程5-4

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        //多重选择器，鼠标移入时效果
第3行        $(".StateName,.Leader").mouseover(function(){
第4行            $(this).addClass("LightIt");
第5行        });
第6行        //多重选择器，鼠标移出时效果
第7行        $(".StateName,.Leader").mouseout(function(){
第8行            $(this).removeClass("LightIt");
第9行        });
第10行   </SCRIPT>
```

(2) 层级选择

层级选择器可根据元素的上下级或者兄弟关系选择指定的元素。如：$(".BigMan>P")表示选择class名为BigMan下子元素P，此时选中的不是BigMan，而是P元素，但并不是所有的P元素，而是位于class名为BigMan之下的P元素。层级选择器包括：后辈选择器、子女选择器、紧随选择器，后随选择器等。

(a) 后辈选择

后辈选择器可以选择祖先元素下的后辈元素(不仅仅是子女元素)。祖先元素和后辈元素都可以用各种选择器表示，两者之间用英文空格分开。格式形如“Para1 Para2”，其中Para1表示祖先的元素，Para2表示后辈元素，选取结果为后辈元素。例程5-5是例程5-1的JavaScript代码，选中例程5-1中State下的所有P元素。

例程5-5

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        alert($(".State P").text());//显示司马懿……夏侯渊,诸葛亮……黄忠,周瑜……吕蒙
第3行    </SCRIPT>
```

(b) 子女选择

子女选择器可以选择父辈元素的子女元素。和后辈选择器相比，子女元素选择器只能选择当前元素的下一级，而后辈元素选择器则可能是多级，隔级等。父辈元素和子女元素都可用各种选择器表示，两者之间用英文大于符号(>)分开。格式如“Para1>Para2”，其中

Para1表示父辈元素，Para2表示子女元素，选择结果为子女元素。例程5-6是例程5-1的JavaScript代码。

例程5-6

```
<SCRIPT language="JavaScript">
        alert($(".BigMan>P").text());//显示司马懿……夏侯渊,诸葛亮……黄忠,周瑜……吕蒙
</SCRIPT>
```

在例程5-6中，将".BigMan>P"修改为".BigMan P"同样有效果，因为P同样是BigMan所代表元素的后辈元素。但是例程5-5中的".State P"修改为".State>P"则不会有预期效果，因为P不是State所代表元素的子女元素，而是孙元素。

(c) 紧随选择

紧随选择器将能选择当前元素后紧邻的同辈元素，其格式为“Para1+Para2”，其中Para1代表位置在前元素，Para2表示紧邻其后的同辈元素，二者皆可用选择器表达，以+联结，选中元素为Para2所代表的元素，例程5-7是例程5-1的JavaScript代码。

例程5-7

```
<SCRIPT LANGUAGE="JavaScript">
        alert($(".StateName+.Leader").text());//显示曹操刘备孙权
</SCRIPT>
```

在例程5-7中，如将$(".StateName+.Leader")修改为$(".StateName+.BigMan")将显示为空，因为紧邻选择器只能选中紧紧跟随StateName的元素，而与BigMan之间，已经相隔了Leader，因此不能选中。

(d) 后随选择

和紧随选择器相似，后随选择器将能选择当前元素后的同辈元素，但不一定紧邻，其格式为“Para1~Para2”，其中Para1代表位置在前的元素，Para2表示其后的同辈元素，二者皆可用选择器表达，以~联结。例程5-8是例程5-1的JavaScript代码。

例程5-8

```
<SCRIPT LANGUAGE="JavaScript">
        alert($(".StateName~.BigMan").text());//显示司马懿……夏侯渊,诸葛亮……黄忠,周瑜……吕蒙
</SCRIPT>
```

在例程5-8中，如将$(".StateName~.BigMan")修改为$(".StateName+.BigMan")将显示为空，因为用加号(+)联结表示紧邻选择器只能选中紧紧跟随StateName的元素，而与BigMan之间，已经相隔了Leader，因此不能选中。

(3) 序号选择

在jQuery中，每个被选中的元素都有一个唯一即时生成的编号，其编号规则是按该元素在网页中出现的先后顺序从0开始，依次增加。jQuery根据编号，可以选中其中部分元素，如first表示选中其中的第一个元素，odd表示选中其中编号为奇数的元素等。类似选择器还有last(最后一个元素)、even(偶数编号元素)、eq(指定编号元素)、gt(大于指定编号的元素)、lt(小于指定编号的元素)。所有的序号选择器前都必须有冒号(:)，如$("P:first")表示选中第1个P元素，在first前面有冒号与P分开，且冒号前后不能有空格以及其他字符。

(a) first(首元素选择)

first用于选中第一个元素。例程5-9是例程5-1的JavaScript代码。在例程5-9中的第2行$("P:first")的含义是选中P元素中的第1个元素，即第一个P元素。

例程5-9

```
1  <SCRIPT LANGUAGE="JavaScript">
2          alert($("P:first").text());//显示司马懿
3          alert($("P:last").text());//显示吕蒙
4
5          alert($(".StateName:odd").text());//显示蜀国
6          alert($(".StateName:even").text());//显示魏国吴国
7
8          alert($("P:eq(15)").text());//显示张昭
9
10         alert($("P:gt(15)").text());//显示甘宁陆逊吕蒙
```

```
第11行                alert($("P:lt(3)").text());//显示司马懿徐晃张辽
第12行            </SCRIPT>
```

(b) last(尾元素选择)

last用于选中最后一个元素。例程5-9是替换例程5-1的JavaScript代码。在例程5-9中的第3行$("P:last")的含义是选中P元素中的最后一个元素，即最后一个P元素。

(c) odd(奇元素选择)

odd用于选中编号为奇数的元素。例程5-9是例程5-1的JavaScript代码，其第5行代码表示选中.StateName中编号为奇数的元素。

(d) even(偶元素选择)

even用于选中编号为偶数的元素。例程5-9是替换例程5-1的JavaScript代码，其第6行代码表示选中.StateName中编号为偶数的元素。奇偶选择器在网页代码中比较常用，如图5-6所示的表格中，利用奇偶选择器很容易实现，其实现代码如例程5-10所示。

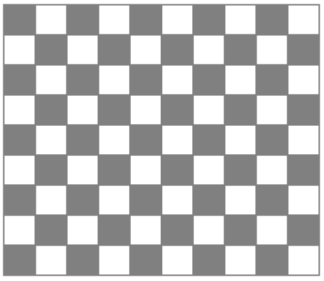


图5-6 奇偶选择器的使用

在例程5-10的代码中，奇数行的奇数编号的单元格被增加了名为BGGray的Class，其定义为背景为灰色；而偶数行的偶数编号的单元格也同样被增加了名为BGGray的Class，就形成了如图5-6的效果。在例程5-10第2行中，$("TR:odd")的含义是选中编号奇数的TR元素，在此基础上找到其子元素TD(编号为奇数)并加上名为BGGray的class。另外，该行代码还可以改写为：
$("TR:odd>TD:odd").addClass("BGGray")。

例程5-10

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行            $("TR:odd").children("TD:odd").addClass("BGGray");//为奇数行的奇数单元格增加名为BGGray的Class
第3行            $("TR:even").children("TD:even").addClass("BGGray");//为偶数行的偶数单元格增加名为BGGray的Class
第4行    </SCRIPT>
```

(e) eq(指定编号选择)

eq选择器可以选定指定编号的元素，如例程5-9第8行$("P:eq(15)")表示选择文档中所有P元素中的第15号元素。编号放在eq后的一对括号中。

(f) gt(大于选择)

gt用于选定大于指定标号的所有元素，如例程5-9第10行$("P:gt(15)")表示选择文档中所有P元素中编号大于15的元素，如例程所示，将显示“甘宁陆逊吕蒙”。和eq选择一样，指定的编号放在其后的一对括号中。

(g) lt(小于选择)

lt选择器用于选定小于指定标号的所有元素，如例程5-9第11行$("P:lt(3)")表示选择文档中所有P元素中小于编号3的元素，如例程所示，将显示“司马懿徐晃张辽”。和eq选择器一样，指定的编号放在其后的一对括号中。

⑷ 属性选择

很多HTML元素都有属性，如常见的Img有src属性，A有Href属性，ID和Class也是HTML元素的属性。根据元素是否有属性以及属性值的不同，也可以选定元素。属性选择器用方括号包围属性的名称和属性值，在例程5-11中第26行到第32行是属性选择的多种方式。

例程5-11

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>jQuery属性选择器</TITLE>
第4行            <SCRIPT LANGUAGE="JavaScript" src="./jQuery-1.4.2.min.js"></SCRIPT>
第5行        </HEAD>
```

```
第6行        <BODY>
第7行            <Div style="width:400px">
第8行                <P style="Font-size:24px">水浒人物杂乱陈</P>
第9行                <P>梁山好汉的绰号可比之以动物，如玉麒麟卢俊义、豹子头林冲等，可观之以体貌，
第10行               如美髯公朱仝、青面兽杨志等。还可以性格如笑面虎朱富，职业如菜园子张青，
第11行               兵器如双鞭呼延灼，特长如浪里白条张顺，以及古英雄如小李广花荣等。</P>
第12行               <P>名号是一个人区别于他人的语言标记，凡人皆有名。生活中，除了比较正统的姓名外，
第13行               往往还有其他。或为打破姓名单调，或为明示人物特点，或交往之初不知姓名指称方便。
第14行               绰号之风，江湖为盛。水浒人物之绰号，虽然廖廖数字，但对人物形象具有重要的提示作
                     用，
第15行               是点睛之笔。说起黑旋风，脑子里马上会浮现出那个皮面黑炭一般抡起板斧呼呼生风的莽汉
                     来。 </P>
第16行               <P class="PersonList" type="StarSky" Sex="male">宋　江·天魁星·及时雨</P>
第17行               <P class="PersonList" type="StarSky" Sex="male">林　冲·天雄星·豹子头</P>
第18行               <P class="PersonList" type="StarLand" Sex="female">扈三娘·地慧星·一丈青</P>
第19行               <P class="PersonList" type="StarSky" Sex="male">武　松·天伤星·行　者</P>
第20行               <P class="PersonList" type="StarLand" Sex="female">孙二娘·地壮星·母夜叉</P>
第21行               <P class="PersonList" Sex="female">潘金莲</P>
第22行               <P class="PersonList" Sex="male">西门庆</P>
第23行               <P class="PersonList" Sex="male">武大郎</P>
第24行           </Div>
第25行           <SCRIPT LANGUAGE="JavaScript">
第26行               alert($("P[Type]").text());//显示宋江林冲扈三娘武松孙二娘的信息
第27行               alert($("P[Sex='female']").text());//显示扈三娘孙二娘潘金莲的信息
第28行               alert($("P[Type!='StarSky']").text());//显示除宋江林冲武松外的所有P元素信息，不仅
                     是LandStar
第29行               alert($("P[Type^='Star']").text());//显示所有type等于StarSky和StarLand的元素
第30行               alert($("P[Sex$='male']").text());//显示所有性别的人物，因为sex等于female和male都
                     以male结尾
第31行               alert($("p[Sex*='em']").text());//显示sex属性值中还有em字母组合的元素，此例中即为
                     female
第32行               alert($("p[Type][Sex='female']").text());//显示扈三娘和孙二娘
第33行           </SCRIPT>
第34行       </BODY>
第35行   </HTML>
```

(a)　属性名称选择

如同例程5-11中第26行，在方括号中仅有属性名称，其含义是选择含有该属性名称的元素。例程5-11中第26行代码$("P[Type]")的含义是选择P中含有Type属性的元素。

(b)　属性值等于选择

如同例程5-11中第27行，在方括号中不但有属性名称，还有该属性的值，其值用引号包围其含义是选择含有该属性名称等于该属性值的元素。例程5-11中第27行代码$("P[Sex='female']")的含义是选择P中含有Sex属性且其值为female的元素。

(c)　属性值不等于选择

如同例程5-11中第28行，在方括号中有属性名称，属性值等号前有叹号(!)表示不等于，其含义是选择含有该属性名称不等于该属性值的元素。例程5-11中第28行代码$("P[Type!='StarSky']")的含义是选择P中Type属性值不等于StarSky的元素。值得注意的是，所有P元素中，除了不能选中Type=StarSky的元素，其他元素都会被选中。

(d)　属性值开始选择

如同例程5-11中第29行，在方括号中有属性名称，属性值等号前有^符号表示开始，其含义是选择含有该属性值中以等号后字符串开始的元素。例程5-11中第29行代码$("P[Type^='Star']")的含义是选择P中Type属性值以Star开始的元素。

(e) 属性值结束选择

如同例程5-11中第30行，在方括号中有属性名称，属性值等号前有$符号表示结束。其含义是选择含有该属性值中以等号后字符串结束的元素。例程5-11中第29行代码$("P[Sex$='male']")的含义是选择P中Sex属性值以male结束的元素。

(f) 属性值含有选择

如同例程5-11中第31行，在方括号中有属性名称，属性值等号前有*符号表示含有等号后的字符组合，含义是选择含有该属性值中以等号后字符串的元素。例程5-11中第31行代码$("P[Sex*='em']")的含义是选择P中Sex属性值中含有em字符组合的元素。

(g) 多重属性选择器

如同例程5-11中第32行，有多个方括号(个数可以很多)，表示满足每个方括号条件的元素。例程5-11中第32行代码$("P[Type][Sex='female']")的含义是选择P中Sex属性等于female且含有Type属性的元素。

⑸ 子元素选择

子元素选择是根据其在父元素中出现的先后顺序，根据子元素指定规则，选中指定编号的元素。子元素选择器有nth-child(序号/odd/even/表达式)、first-child、last-child、only-child。图5-7是子元素选择器的应用效果，其实现代码为例程5-12。

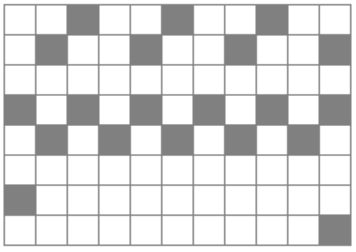


图5-7 子元素选择器应用示例

(a) nth-child

nth-child功能强大，其格式为selector:nth-child(参数)，其中参数可以是具体数值，也可以是even(偶数)、odd(奇数)和表达式(必须是n的表达式)。表达式中n的系数必须为整数，其后可以加上一个整数。

例程5-12

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $("TR:eq(0) TD:nth-child(3n)").addClass("BGGray");//nth-child(3n)每3个一组，第3个被选中
第3行        $("TR:eq(1) TD:nth-child(3n+2)").addClass("BGGray");//nth-child(3n+2)每3个一组，第2个被选中
第4行
第5行        $("TR:eq(3) TD:nth-child(odd)").addClass("BGGray");//nth-child(odd)选中奇数单元格
第6行        $("TR:eq(4) TD:nth-child(even)").addClass("BGGray");//nth-child(even)选中偶数单元格
第7行
第8行        $("TR:eq(6) TD:first-child").addClass("BGGray");//first选中第1个单元格
第9行        $("TR:eq(7) TD:last-child").addClass("BGGray");//last选中最后1个单元格*/
第10行   </SCRIPT>
```

例程5-12第2行的代码$("TR:eq(0) TD:nth-child(3n)").addClass("BGGray")含义是选中表格第1行(通过TR:eq(0)实现)下的TD子元素，并将TD子元素按每3个分组且选中每组第3个，为其增加名为BGGray的Class。该Class的效果是设置背景为灰色。图5-7中第1行是例程5-12第2行执行效果。

例程5-12第3行的代码$("TR:eq(0) TD:nth-child(3n+2)").addClass("BGGray")的执行效果如图5-7第2行。其3n+2的含义是将TD按每3个分组，并将第2个单元格选中为其增加名为BGGray的Class。

nth-child还可以odd和even为参数，分别表示选中其奇数或偶数元素。例程5-12第5、6行是其示例代码，图5-7中第4、5行为其执行效果。nth-child还可以整数为参数，如nth-child(5)表示选中第5个元素，如图5-7中第2排中间$("TR TD:nth-child(3)")。

(b) first-child和last-child

first-child和last-child可以分别选中第一个子元素和最后一个子元素，例程5-12第8、9行是其示例代码，图5-7中最后两行为其执行效果。

子元素选择器和序号选择器有些相似，图5-8是两者的比较。

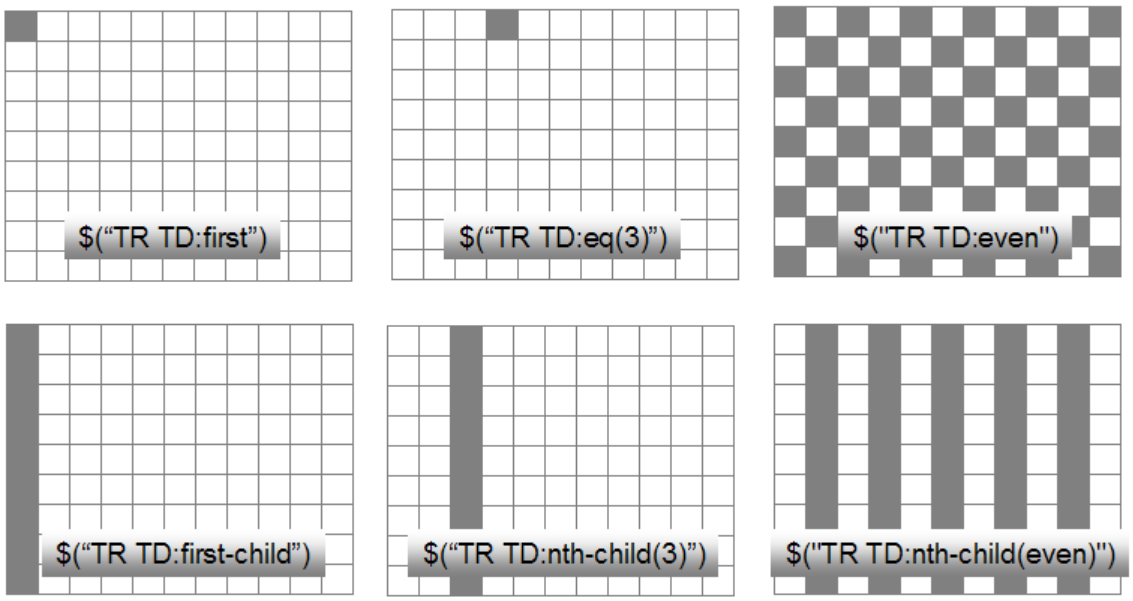


图5-8　序号选择器与子元素选择器对比

序号选择是将所有选中的元素统一从0开始编号，而子元素选择则根据父元素分组，每个组内子元素统一从0开始编号。图5-8中第一排和第二排中间图像，即$("TR TD:eq(3)")和$("TR TD:nth-child(3)")，其中$("TR TD:eq(3)")的含义是选中TR下的所有TD中的编号为3的元素，由于从0开始编号，所以第1排中间的图示为第4个元素被选中；而$("TR TD:nth-child(3)")的含义是分别选中每个TR元素下的TD中第3个元素，虽然同为3，但是其含义不同，一个是编号为3(实际为第4个)，一个是第3个(编号为2)。

同理，$("TR TD:first")是选中所有TD中第1个(仅此一个)，而$("TR TD:first-child")则是选中每行(TR)中第1个TD元素(可能多个)；$("TR TD:even")是所有TD中的偶数被选中(所有TD连续编号)，而$("TR TD:nth-child(even)")则是每行(TR)中的偶数TD单元格被选中(每行TD独立编号)。

(c)　only-child

被匹配元素如果是父元素中唯一的子元素(独生子女)，将会被选中。在例程5-13中，执行代码$("ul li:only-child")将仅会选中“
关汉卿”。

例程5-13

```
<ul>
        <li>李白</li>
        <li>杜甫</li>
        <li>白居易</li>
</ul>
<ul>
        <li>苏东坡</li>
        <li>辛弃疾</li>
        <li>李清照</li>
</ul>
<ul>
        <li>关汉卿</li>
</ul>
```

(6)　内容选择

jQuery选择器可以根据元素所控制的内容选择元素，参数有contains(含有指定内容)、empty(没有内容)、parent(含有内容或子元素)、has(含有指定选择器)。

(a)　contains选择

contains选择用于选择含有指定内容的元素，其格式为selector:contains(text)，其中text为文本，例程5-14是将例程5-1的JavaScript代码替换后的结果。在例程5-14中，text参数为'张'，显示出所有含有张的元素(注意：张字两侧有引号)。

例程5-14

```
<SCRIPT LANGUAGE="JavaScript">
    alert($("P:contains('张')").text());//显示张辽张颌张飞张昭
</SCRIPT>
```

(b) empty选择

empty选择用于选择不含有文本或者子元素的元素。如$("P:empty")就表示选择P中没有内容也没有子元素的P元素。

(c) parent选择

parent选择器用于选择含有文本或者子元素的元素。如$("P:parent")就表示选择P中含有文本或子元素的P标记。对于例程5-1所示代码，所有P元素都将被选中。

(d) has选择

has用于选择含有指定选择的元素，其格式如selector:has(selector)。例程5-15是例程5-1的JavaScript代码。在例程5-15的代码$(".BigMan:has(P:contains('夏侯'))")的含义是选择Class名为BigMan的元素，条件是BigMan中含有P元素且该P元素含有'夏侯'二字。

例程5-15

```
<SCRIPT LANGUAGE="JavaScript">
    alert($(".BigMan:has(P:contains('夏侯'))").text());//司马懿徐晃张辽李典乐进张颌夏侯惇夏侯渊
</SCRIPT>
```

⑺ 可见性选择

网页文档中的元素还可以根据隐藏和显示状态进行选择。

(a) hidden选择

匹配所有不可见元素，如$("div:hidden")将选中所有隐藏的div元素。

(b) visible选择器

匹配所有可见元素，如$("div:visible")将选中所有正在显示的div元素，隐藏的div元素将不被选中。

⑻ 其他

(a) not选择

选中元素中不包含not选择器指定元素的部分，如$("P:not(:contains('张'))")的含义是选中所有P元素，但其内容中不包含“张”字，如例程5-1中的“张辽、张颌、张飞、张昭”等将不包含在内，因为这些P元素中包含有“张”这个字。

(b) header选择

匹配所有标题元素，如H1、H2、H3等，如$(":header")将选中所有的标题元素。

(c) animated选择

匹配所有正在执行动画的元素，$("div:not(:animated)").show("slow")，有关动画的内容将在本章讲解。

除了上述所列选择参数外，还有其他一些选择，如与表单(form)相关的选择参数等，将在“表单”章节讲解。

选择参数不仅仅用在初选，也常用于一些筛选和查找函数中，是jQuery的重要基础。

2、初次选择

$(选择器)即为初选，可以在此基础筛选出部分元素或者以此为基础查找其他元素，如在$(".BigMan").eq(2).children("P").filter(":odd").addClass("LightIt");中，$(".BigMan")是初选，选中所有class名为BigMan的元素，其后eq(2)是筛选，表示仅选中其中编号为2的元素，而children("P")表示查找，以eq(2)执行结果为基础，找到子元素P，而filter(":odd")又是筛选，表示children("p")元素集合中编号为奇数的元素，最后返回的元素集合为编号为奇数的P元素，并用指令addClass给这些元素增加名为LightIt的Class。

筛选和查找都是在初选的基础进行，且筛选和查找的结果仍然可以作为继续筛选和查找的基础。

3、子集筛选

子集筛选是从已有选择结果中选出一部分匹配元素，常用的筛选函数有eq()、filter()、slice()等。

⑴ eq(Index)

从选中元素集合(编号从0开始)中选择编号为Index的元素。例程5-16是例程5-1的JavaScript代码。

例程5-16

```
<SCRIPT LANGUAGE="JavaScript">
    alert($(".Leader").eq(1).text());//显示刘备
</SCRIPT>
```

在例程5-16中，$(".Leader")是初选，选中所有Class名称为Leader的元素，共计3个元素，编号分别是0、1、2。eq(1)是在初选的三个元素中，选择其中编号为1的元素，根据例程5-1的内容，其text()为刘备。

注意：eq()还可以用在选择参数中，如$("P:eq(2)")的含义是直接选中编号为2的P元素，仅选中一个元素；而$("P").eq(2)也能选中编号为2的P元素，不过是在选中全部P元素的基础上筛选出编号为2的元素。一般说来，$("P:eq(2)")的效率要高于$("P").eq(2)。

(2) filter(selector)

filter()是最重要的筛选函数。filter()可以根据其selector参数筛选元素，例程5-16是例程5-1的JavaScript代码。

例程5-17

```
<SCRIPT LANGUAGE="JavaScript">
    alert($("Div").filter(".Leader").text());//显示曹操刘备孙权
</SCRIPT>
```

(3) filter(function(){})

filter根据function返回值保留元素，如果返回值为true，则留下该元素，否则剔除该元素。例程5-18是例程5-1的JavaScript代码。该函数将对每个元素进行计算,且处理函数function()中可以包含各种代码，因此能根据更加复杂的逻辑筛选元素。该函数可以接收一个参数，代表被处理元素的编号。

例程5-18

```
<SCRIPT LANGUAGE="JavaScript">
    alert($(".BigMan>P").filter(function(Index){
        //$(this).text().length含义是字符串长度
        if ($(this).text().length>2){
            return true;
        }else{
            return false;
        }
    }).text());//显示司马懿夏侯惇夏侯渊诸葛亮
</SCRIPT>
```

例程5-18的代码含义是初选结果是所有class为BigMan下的子元素P，function()代码的功能是根据每个元素text的长度选择元素，如果长度大于2，则返回true，该元素保留在筛选集合中，否则返回false，该元素被剔除在结果集合之外。

(4) slice(StartIndex,EndIndex)

slice可以从选中集合截取一个片段，该片段从开始索引值StartIndex开始，到EndIndex为止。例程5-19是slice函数的使用。

例程5-19

```
<SCRIPT LANGUAGE="JavaScript">
    alert($(".BigMan Div").slice(10,15).text());//显示张飞赵云黄忠周瑜鲁肃
</SCRIPT>
```

在例程5-19中，初选函数$(".BigMan Div")将选中BigMan下的所有Div元素，slice(10,15)将会选中其中编号为10到15的元素。省略EndIndex时，将选择从StartIndex开始直到结束的所有元素。当StartIndex为负值，则从尾部开始选起。

(5) has(selector)

保留含有特定后代的元素，如果不含有指定的后代元素，则剔除选中元素集合。例程5-20是例程5-1的JavaScript代码。

例程5-20

```
<SCRIPT LANGUAGE="JavaScript">
    //执行后显示为司马懿徐晃张辽李典乐进张颌夏侯惇夏侯渊
    alert($(".BigMan").has("P:contains('夏侯')").text());
</SCRIPT>
```

在例程5-20中，选中元素为所有class名为BigMan的元素，has筛选出子元素含有“夏侯”两字的元素，过滤掉子元素不含“夏侯”二字的元素，因此仅剩下“司马懿”所在的BigMan。注意此处首先选中的是BigMan，然后根据has()函数中的参数过滤出子元素中含有“夏侯”二字的元素。

⑹ not(selector)

在选定元素集合中剔除与指定selector匹配的元素，与has不同，has是根据子元素限定选定元素集合，而not则是在选定集合中直接剔除与selector匹配的元素。例程5-21是例程5-1的JavaScript代码。

例程5-21

```
<SCRIPT LANGUAGE="JavaScript">
    alert($("P").not(":contains('张')").text());//剔除含有张的元素
</SCRIPT>
```

在例程5-21中，$("P")将选中例程5-1中的所有P元素，但not(":contains('张')")将剔除掉其文本含有“张”字的元素，所以所有含有“张”字的元素都不被显示。

4、定位查找

和筛选相比，查找是以当前已找到元素为出发点，根据Html元素树形结构，查找其他元素，如同辈元素、后辈元素、前辈元素以及父元素和子元素等。

⑴ find(selector)

find搜索所有与查找参数匹配的元素，这个函数是找出正在处理元素的后代元素(后代不仅是子元素)好方法。例程5-22是例程5-1的JavaScript代码。

例程5-22

```
<SCRIPT LANGUAGE="JavaScript">
    alert($("Body").find(".BigMan").eq(2).text());//显示周瑜鲁肃张昭甘宁陆逊吕蒙
</SCRIPT>
```

在5-22中，$("Body")是初选，将选中Body元素，其find(".BigMan")则是查找函数，将以初选元素为基础，选中Body后代元素中其class名为BigMan的元素。

⑵ children(selector)

和find函数一样，children也用于查找后代元素，不过仅限于子元素。children的selector参数可以省略，表示所有子元素。当含有selector参数时，则表示满足selector条件的子元素。例程5-23是例程5-1的JavaScript代码。

例程5-23

```
<SCRIPT LANGUAGE="JavaScript">
    alert($(".State").children().eq(1).text());//显示曹操
    alert($(".State").children(".StateName").eq(1).text());//显示蜀国
</SCRIPT>
```

在例程5-23中，第2行将选中所有class名为State的子元素，包括class分别名为StateName、Leader以及BigMan的Div，并按照出现的顺序分别从0开始编号，因此其eq(1)是曹操，其eq(0)是魏国，eq(2)则是司马懿……夏侯渊等。第3行的children有参数，则只选中class名为StateName的元素，包括魏国、蜀国和吴国，自动为其编号为0、1、2，其eq(1)为蜀国。

⑶ parent(selector)

parent的功能是选择父元素。当省略selector参数时，选中所有父元素，而当含有参数时，则选中符合selector条件的父元素。例程5-24是例程5-1的JavaScript代码。

例程5-24

```
<SCRIPT LANGUAGE="JavaScript">
        alert($("Div").parent().eq(2).text());//显示司马懿徐晃张辽李典乐进张颌夏侯惇夏侯渊
        alert($("Div").parent(".State").eq(2).text());//显示吴国孙权周瑜鲁肃张昭甘宁陆逊吕蒙
</SCRIPT>
```

对于第2行的代码，其含义是选中所有Div元素，通过parent选中其父元素。不同的div其父元素也不同。对于State而言，其父元素为Body，对于StateName、Leader和Bigman而言，其父元素为State，而对于BigMan下的div，其父元素为BigMan。所有这些父元素构成了Div的所有父元素集合，总计7个。

对于第3行代码，同样是选中所有的Div元素，但其父元素仅限于class名为State，因此，被选中的元素肯定是多个Div中其父class名为State。符合这样条件的仅有三个class名为State的Div，其eq(2)的文本为吴国孙权周瑜……吕蒙等。

(4) parents(selector)

和parent()相比，parents()用于查找每个匹配元素的前辈元素(注意：不仅是父元素)，而parent()仅用于查找父元素。当parents()当省略参数时，则找到所有先辈元素，如果有参数，则找到匹配的先辈元素。例程5-25是例程5-1的JavaScript代码。

例程5-25

```
<SCRIPT LANGUAGE="JavaScript">
        alert($("P:contains('司马懿')").parents().size());//显示为4
        alert($("P:contains('司马懿'),P:contains('诸葛亮')").parents().size());//显示为6
        alert($("P:contains('司马懿')").parents("Div").size());//显示为2
        alert($("P:contains('司马懿'),P:contains('诸葛亮')").parents("Div").size());//显示为4
</SCRIPT>
```

第2行代码首先选中“司马懿”所在P元素，其先辈元素包括：class名为BigMan的Div元素、class名为State的Div元素、Body元素、Html元素，总计4个。第3行增加了“诸葛亮”所在的P元素，其先辈元素与“司马懿”相似，总计应该为8个，但有两个元素(Body元素、Html元素)重复，剔除后为6个。第4行是在先辈元素中找到元素名称为Div的元素，其Body元素和Html元素虽然是其先辈元素，但元素名称不是Div，因此被剔除。第5行仍然是找到元素名称为Div的元素，同样剔除Body和Html两个先辈元素，分别留下“司马懿”和“诸葛亮”的两个先辈元素，总计4个(注意：虽然都是State和BigMan，但是不同的元素；不同元素不予合并)。

(5) parentsUnitl(selector)

parentsUnitl()也用于查找全部先辈元素，直到遇到参数所匹配的元素为止，但不包含该匹配元素。当省略参数与parents()相同。

例程5-26

```
<SCRIPT LANGUAGE="JavaScript">
        alert($("P:contains('司马懿')").parentsUntil(".State").size());//显示为1
        alert($("P:contains('司马懿'),P:contains('诸葛亮')").parentsUntil(".State").size());//显示为2
</SCRIPT>
```

第2行是查找“司马懿”的父辈元素，直到遇到class名为State为止，但不包含该元素。“司马懿”的父元素名为BigMan的class，然后名为State的class(但不包含)，因此仅有一个匹配元素。第3行与之类似。

(6) next(selector)及nextAll(selector)

next()以及nextAll()都是找到其后的同辈元素，相当于弟元素，next()是其后紧邻的一个同辈元素，而nextAll()则是其后的所有同辈元素。例程5-27是将例程5-1的JavaScript代码替换后的结果。

例程5-27

```
<SCRIPT LANGUAGE="JavaScript">
        alert($(".StateName").next().text());//显示曹操刘备孙权
        alert($(".StateName").nextAll().text());
        //显示曹操司马懿……夏侯渊，刘备诸葛亮……黄忠，孙权周瑜……吕蒙

        alert($(".StateName").next(".BigMan").text());//显示为空，什么都没有
        alert($(".StateName").nextAll(".BigMan").text());
```

```
第8行          //显示司马懿……夏侯渊，诸葛亮……黄忠，周瑜……吕蒙
第9行      </SCRIPT>
```

next()和nextAll()都可以有selector参数，表示满足selector的后面同辈元素。例程5-27第6行代码中，由于其紧邻元素没有class名为BigMan的元素，因此没有选中任何元素，也就没有任何内容可以被显示；而第7行将next改为nextAll后，由于nextAll是其后所有同辈元素，因此能找到class名为BigMan的元素。

(7) prev(selector)及prevAll(selector)

prev()以及prevAll()都是找到前面的同辈元素，相当于兄元素，prev()是前面紧邻的同辈元素， 而prevAll()则是前面所有的同辈元素。例程5-28是例程5-1的JavaScript代码。

例程5-28

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行          alert($(".BigMan").prev().text());//显示曹操刘备孙权
第3行          alert($(".BigMan").prevAll().text());//显示孙权吴国刘备蜀国曹操魏国
第4行
第5行          alert($(".BigMan").prev(".StateName").text());//显示为空，什么都没有
第6行          alert($(".BigMan").prevAll(".StateName").text());//显示吴国蜀国魏国
第7行    </SCRIPT>
```

prev()和prevAll()都可以有selector参数，表示满足selector的前面同辈元素。例程5-26第5行代码中，由于其紧邻元素没有class名为StateName的元素，因此没有选中任何元素，也就没有任何内容可以被显示；而第6行将prev改为prevAll后，由于prevAll是前面所有同辈元素，因此能找到class名为StateName的元素。

(8) nextUntil()和prevUntil()

nextUntil()和prevUntil()比较相似，两者都是找到同辈元素，一个找哥哥(在前的同辈元素)，一个找弟弟(在后的同辈元素)，直到遇到参数匹配元素为止。当省略参数时，则与nextAll()或prevAll()相似。例程5-29是例程5-1的JavaScript代码，是nextUntil()和prevUntil()的应用。

例程5-29

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行          alert($(".StateName:eq(1)").nextUntil(".BigMan").text());//显示刘备
第3行          alert($(".BigMan:eq(2)").prevUntil(".StateName").text());//显示孙权
第4行          alert($("P:contains('乐进')").prevUntil(":contains('司马懿')").text());//李典张辽徐晃
第5行          alert($("P:contains('乐进')").nextUntil(":contains('周瑜')").text());//张颌夏侯惇夏侯渊
第6行    </SCRIPT>
```

(9) siblings(selector)

siblings()相当于next()和prev()的组合，其功能是取得所有同辈元素的集合，相当于兄弟元素。例程5-30是例程5-1的JavaScript代码。从例程5-30可以看出，当siblings()包含selector时，将选中符合selector条件的同辈元素。

例程5-30

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行          alert($(".BigMan").siblings().text());//显示魏国曹操蜀国刘备吴国孙权
第3行          alert($(".BigMan").siblings(".Leader").text());//显示曹操刘备孙权
第4行    </SCRIPT>
```

第三节　文档处理

文档处理用于处理匹配元素中的内容，前面已经多次使用过的$.html()或$.html(content)即是常用的文档处理函数。文档处理包括：内部插入、外部插入以及包裹等。也还包括清空、删除以及复制等。如图5-9所示，相关代码如例程5-31所示。

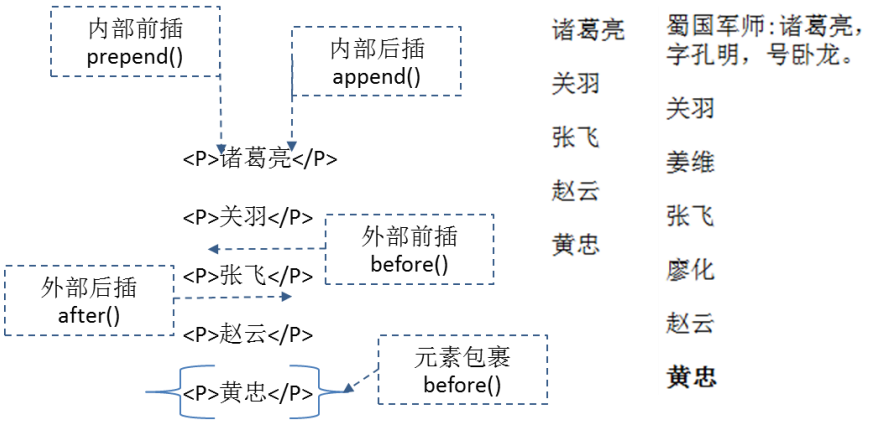


图5-9　文档处理示意图

例程5-31

```
第1行   <Html>
第2行       <Head>
第3行           <Title>文档处理</Title>
第4行           <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行       </Head>
第6行       <Body>
第7行           <P>诸葛亮</P>
第8行           <P>关羽</P>
第9行           <P>张飞</P>
第10行          <P>赵云</P>
第11行          <P>黄忠</P>
第12行          <Script Language="JavaScript">
第13行              $("P:first").prepend("蜀国军师:");//内部前插
第14行              $("P:first").append(", <BR>字孔明，号卧龙。");//内部后插
第15行              var mySelection=$("P:eq(2)");
第16行              mySelection.before("<p>姜维</p>");//外部前插
第17行              mySelection.after("<p>廖化</p>");//外部后插
第18行              $("P:contains('黄忠')").wrap("<div style='font-weight:bold'></div>");
第19行          </Script>
第20行      </Body>
第21行  </Html>
```

内部插入分为：内部前插(prepend，第13行)和内部后插(append，第14行)；外部插入分为：外部前插(before，第16行)和外部后插(after，第17行)；包裹元素常用wrap()函数(第17行)。

值得注意的是：如果将第16、17行代码修改为$("P:eq(2)").before(" 姜维 ");和$("P:eq(2)").after(" 廖化 ");将与现有效果不同。当执行before()之前，“张飞”的编号是2；当执行before()后，“张飞”的编号变成3，而“姜维”编程2。而var mySelection=$("P:eq(2)");则是将“张飞”所在元素保存在变量mySelection中，只要不修改变量内容，其所代表的一直是“张飞”所在元素。

1、内部插入

内部插入用于在匹配元素内插入新内容，插入的内容可以在该元素原有内容之后(append)、也可以在原有内容之前(prepend)，有多种方法支持内部插入，如表1所示。

表1：内部插入操作

操作	描述
$.append(content)	在每个匹配元素原内容之后插入content所指定的新内容。

$.appendTo(selector)	将匹配元素移动并追加到selector参数所匹配的元素内。
$.append(function(index,html))	将function(index,html)返回值插入到匹配元素原有内容之后，其中index代表操作元素编号，html表示内容。
$.prepend(content)	在每个匹配元素原内容之前插入content所指定的新内容。
$.prependTo(selector)	将匹配元素移动并前插到selector参数所匹配的元素内。
$.prepend(function(index,html))	将function(index,html)返回值插入到匹配元素原有内容之前，其中index代表操作元素编号，html表示内容。

例程5-32是部分内部插入功能的应用，图5-10是其执行效果。

例程5-32

```
第1行   <Body>
第2行       <UL>
第3行           <LI>政
第4行               <UL>
第5行                   <LI class="L4">元春</LI>
第6行                   <LI class="L4">珠</LI>
第7行                   <LI class="L4">宝玉</LI>
第8行                   <LI class="L4">环</LI>
第9行                   <LI class="L4">探春</LI>
第10行              </UL>
第11行          </LI>
第12行          <LI>赦</LI>
第13行      </UL>
第14行      <Script Language="JavaScript">
第15行          $("LI").prepend("<b>贾</b>");//在LI内部内容之前插入一个加粗“贾”字
第16行          $("LI").append("[荣国府]");//每个LI内部内容之后插入“[荣国府]”
第17行          $(".L4").prepend(function(index,html){
第18行              return index+".  ";
第19行          });
第20行      </Script>
第21行  </Body>
```

- **贾政**
  - 0. **贾元春** [荣国府]
  - 1. **贾珠** [荣国府]
  - 2. **贾宝玉** [荣国府]
  - 3. **贾环** [荣国府]
  - 4. **贾探春** [荣国府]

  [荣国府]
- **贾赦** [荣国府]

图5-10　例程5-24执行后效果图

注意：“贾赦”后面有“[荣国府]”,而“贾政”后面没有，但“贾探春”后面有两个“[荣国府]”，其原因是例程5-32第16行在每个LI后面都增加了“[荣国府]”，而“贾政”对应的LI还包括第4行到第10行的代码，因此增加在该位置。

第17行到第19行是$.prepend(function(index,html))的一个应用，function(index,html)表示该函数可以输入两个参数，index代表选中的".L4"的序号(索引值)，html代表其对应的内容，其函数返回值插入到其原有内容的前面。$.prepend(function(index,html))很适合用于章节编号，假定某网页中有大量H2标记，其内容前没有编号，使用该功能，能方便地加上编号，其代码如例程5-33所示。

例程5-33

```
<Script Language="JavaScript">
    $("H2").prepend(function(index,html){
        return (index+1)+".";//index从开始编号，index+1则实现从1开始编号
    });
</Script>
```

例程5-34是$.appendTo(content)的应用。appendTo用于将$(参数)匹配元素移动并追加到content选定的元素，prependTo与之类似，不过一个插在原内容之后，一个插在原内容之前。图5-11是例程5-34执行效果，从代码可以看出，P元素原有内容字号不大且没有下划线，移动到Div元素后，font-size和text-decoration都发生了变化，其原因是$("P")所匹配的元素已经被插入到"#Box"所匹配的元素中。

例程5-34

```
<Body>
    <P>世事洞明皆学问，人情练达即文章！</P>
    <Div style="font-size:40px;text-decoration:underline" ID="Box">摘自《红楼梦》 作者：曹雪芹</Div>
    <Script Language="JavaScript">
        $("P").appendTo("#Box");
    </Script>
</Body>
```

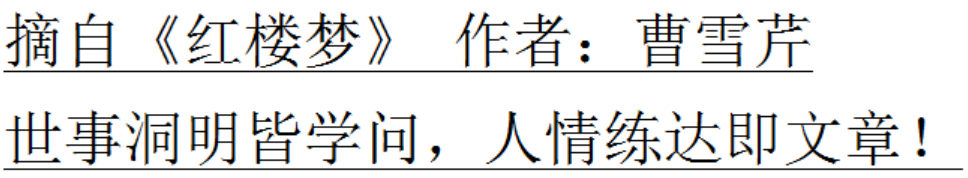


图5-11　例程5-34执行效果图

2、外部插入

内部插入内容在选中元素之内，而外部插入在选中元素之外，新插入的内容与匹配元素成为兄弟姐妹关系，是同辈关系。如选中"李白"元素后，执行$("b").after("杜甫")，则形成内容形式为"李白杜甫"，即在“李白”之后插入“杜甫”。下表是外部插入操作，例程5-35是外部插入的一个应用。

表2：外部插入操作

操作	描述
$.after(content)	在匹配元素后插入content所代表的内容。
$.after(function(idx){})	在匹配元素后插入function(idx)所返回的值，idx代表匹配元素的编号。
$.before(content)	在匹配元素前插入content所代表的内容。
$.before(function(idx){})	在匹配元素前插入function(idx)所返回的值，idx代表匹配元素的编号。
$.insertAfter(selector)	将匹配元素移动并插入到selector所指定的元素之后。
$.insertBefore(selector)	将匹配元素移动并插入到selector所指定的元素之前。

例程5-35

```
<HTML>
    <HEAD>
        <TITLE>外部插入</TITLE>
        <Script language="JavaScript" src="./jquery-1.9.1.min.js"></Script>
        <META http-equiv="Content-type" content="text/html; charset=gb2312">
    </HEAD>

    <BODY>
```

```
第9行                    <Img src="1.gif" title="春意盎然" style="width:220px;">
第10行                   <Script Language="JavaScript">
第11行                       $("Img").after(function(){
第12行                           var imgTitle=$(this).attr("title");
第13行                           $(this).after("<Div>"+imgTitle+"</Div>");
第14行                       });
第15行                   </Script>
第16行               </Body>
第17行               </HTML>
```

3、元素包裹

在“<Div>李白</Div>”代码中称Div元素包裹b元素，在jQuery中，可以给匹配元素包裹一个新元素，如上例的实现代码为$("b").wrap("<Div></Div>")。下表是元素包裹的多种方式，例程5-36是元素包裹的一个应用，图5-12是其执行效果。

表3：包裹操作

操作	描述
$.wrap(html)	将所有匹配元素用html代码包裹。
$.wrap(element)	把所有匹配的元素用DOM元素包裹。
$.wrap(function(Idx){})	把所有匹配的元素用function()函数的返回值包裹。
$.unwrap()	移走匹配元素的父元素。
$.wrapAll(html)	将所有匹配元素用Html元素包裹，形成所有匹配元素的父元素。
$.wrapAll(element)	将所有匹配元素DOM元素包裹，形成所有匹配元素的父元素。
$.wrapInner(html)	将匹配元素的子元素及其内容用Html代码包裹，在匹配元素与原有内容之间插入包裹元素。
$.wrapInner(element)	将匹配元素的子元素及其内容用DOM元素包裹，在匹配元素与原有内容之间插入包裹元素。
$.wrapInner(function(Idx){})	将匹配元素的子元素及其内容用function()返回的Html代码包裹，在匹配元素与原有内容之间插入包裹元素。
element是DOM对象，有关DOM对象将后续章节中讲解。	

例程5-36

```
第1行   <Html>
第2行       <Head>
第3行           <Title>文档处理</Title>
第4行           <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行           <Style type=text/css>
第6行               #Box{width:200px;text-align:Center}
第7行               P{margin:0px;}
第8行           </Style>
第9行       </Head>
第10行      <Body style="width:200px">
第11行          <Div id="Box">
第12行              <P>诸葛亮</P>
第13行              <P>关羽</P>
第14行              <P>张飞</P>
第15行              <P>赵云</P>
第16行              <P>黄忠</P>
第17行              <P>周瑜</P>
```

```
第18行                    <P>鲁肃</P>
第19行                    <P>张昭</P>
第20行                    <P>甘宁</P>
第21行                    <P>陆逊</P>
第22行                    <P>吕蒙</P>
第23行                </Div>
第24行                <Script Language="JavaScript">
第25行                    $("P").slice(0,5).wrap("<div style='border:1px solid gray'></Div>");
第26行                    $("P").slice(6).wrapAll("<div style='border:1px solid gray'></Div>");
第27行                    $("P:gt(8)").wrapInner("<b></b>");//相当于<P><b>陆逊</b></P><P><b>吕蒙</b></P>
第28行                </Script>
第29行            </Body>
第30行        </Html>
```

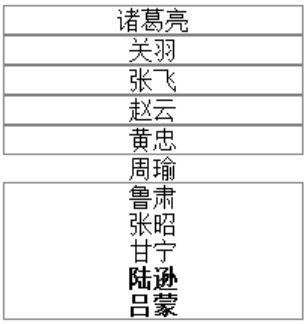


图5-12　不同包裹模式效果图

例程5-36第25行wrap()函数是给每个选中的元素外包裹一个带边框的Div元素，所以从“诸葛亮……黄忠”每个元素都有边框(共计5个)；而第26行wrapAll()则是在所有匹配元素外增加一个带边框的Div元素(总计1个)；第27行的wrapInner()则是将每个匹配元素内的内容包括在B元素内。

例程5-37是wrap()函数的一个具体应用，图5-12是其执行效果。

例程5-37

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>元素包裹</TITLE>
第4行            <Script language="JavaScript" src="./jquery-1.9.1.min.js"></Script>
第5行            <META http-equiv="Content-type" content="text/html; charset=gb2312">
第6行            <Style type="text/css">
第7行                .imgBox{width:250px;text-align:Center;background-color:#E6E6E6;padding-top:20px}
第8行                .imgBox Div{font-weight:bold;margin-top:10px;margin-bottom:10px}
第9行            </Style>
第10行       </HEAD>
第11行
第12行       <BODY>
第13行           <Img src="1.gif" title="春意盎然" style="width:220px;">
第14行           <Script Language="JavaScript">
第15行               $("Img").wrap("<Div class='imgBox'></Div>");//元素包裹
第16行               $("Img").after(function(){
第17行                   var imgTitle=$(this).attr("title");
第18行                   $(this).after("<Div>"+imgTitle+"</Div>");
```

```
第19行                    }); //元素外部插入
第20行                </Script>
第21行    </Body>
第22行    </HTML>
```

图5-13　元素包裹

4、替换、删除和克隆

一般说来，内部插入、外部插入以及包裹都不会改变被选中的元素，而替换将使匹配元素被新的元素或内容所替代，匹配元素将不存在，如例程5-38所示。删除将影响到匹配元素，$.empty()将删除匹配元素内的子元素及其内容；$.detach()和$.remove()则将删除匹配元素及其内的内容，二者之间稍有区别，请看表4。$.clone()则可以克隆元素，如例程5-39。

表4：替换、删除和克隆

操作	描述
$.replaceWith(content)	用content所代表的内容替换匹配元素。
$.replaceAll(selector)	用$(参数)的内容替换selector匹配所匹配的元素，如例程5-38第5行所示。
$.empty()	清空匹配元素包括子元素在内的内容，但该元素还存在。
$.remove(content)	删除所有匹配元素，被删除元素被保存在jQuery中，但事件等被删除。
$.detach(content)	删除所有匹配元素，元素被保留在jQuery中，包括事件等。
$.clone()	克隆并选中匹配元素。
$.clone(true)	克隆元素及其事件并且选中匹配元素

例程5-38

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        //<P>张飞</P>被替换为<B>张飞</B>，注意不是<P><b>张飞</b></P>
第3行        $("P:contains('张飞')").replaceWith("<b>张翼德</b>");
第4行        //<b>周公瑾</b>替换<P>周瑜</P>
第5行        $("<b>周公瑾</b>").replaceAll("P:contains('周瑜')");
第6行        alert($(".State:eq(2)").html());
第7行    </SCRIPT>
```

例程5-39

```
第1行    <HTML>
第2行        <HEAD>
第3行            <TITLE>元素clone</TITLE>
```

```
第4行                    <Script language="JavaScript" src="./jquery-1.9.1.min.js"></Script>
第5行                    <META http-equiv="Content-type" content="text/html; charset=gb2312">
第6行            </HEAD>
第7行
第8行            <BODY>
第9行                    <Div><img src="1.gif" style="width:120px"></Div>
第10行                   <Script Language="JavaScript">
第11行                           $("div").append($("img").clone()).append($("img").clone());
第12行                   </Script>
第13行           </Body>
第14行   </HTML>
```

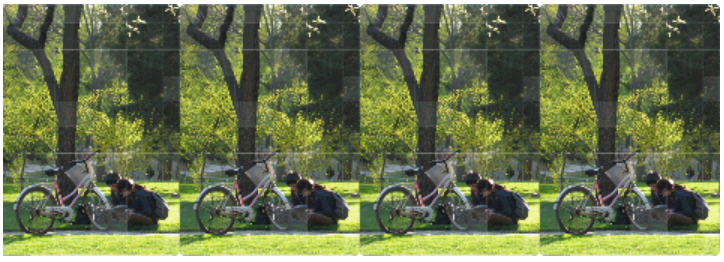


图5-14　元素clone

第四节　class操控

几乎每个Html元素都可以有一个或多个class属性，且可以多个元素有相同的class属性。在Head-Style和File-Style中，可以定义class的CSS属性，以使相关元素呈现不同效果。jQuery能方便地为选中元素增加【$.addClass()】或减少class【$.removeClass()】，从而操纵元素的显示效果。

$.addClass()和$.removeClass()经常配对使用，如鼠标移动到(mouseover)某个元素时增加class，当鼠标离开该元素时(mouseout)则删除增加的class。jQuery提供的$.toggleClass()则能实现如果某元素存在某个class，则移走该class，否则增加该class，一个函数实现$.addClass()和$.removeClass()两个函数的功能。

jQuery提供$.hasClass(className)能判断选中元素是否存在指定className，如果存在则返回true，否则返回false。

$.addClass()、$.removeClass()和$.toggleClass()都有多种形式的变体，如下表所示。

表5：class操控

操作	描述
$.addClass(className)	为匹配元素增加class。className为class的名称，前面不能有半角句点。注：可以增加多个class，如$.addClass("Class1 Class2")，每个class名称之间用空格分隔。
$.addClass(function(index,class){})	为匹配元素增加class。增加class名称由function()函数返回确定(如例程5-40)。该函数有两个参数，其中index为匹配元素的编号，class匹配元素现有class值。
$.removeClass(className)	从匹配元素中删除className对应的class。
$.removeClass(function(index,class){})	从匹配元素中删除function()返回的class。function()函数可以返回一个或多个由空格分隔的class名称。index参数为匹配元素的编号，class参数为元素的class属性值。
$.toggleClass(className)	如果className已存在则删除，不存在则增加。例如：在click事件中应用该函数，则单击增加className，再单击则删除className，或与之相反。
$.toggleClass(className,switch)	当switch为true时，将为匹配元素增加名为className的class，相当于addClass()；当switch为false时，将删除匹配元素中名为className的class。
$.toggleClass(function(index,class){},[switch])	根据function()函数的返回值确定操作class的名称。当switch为true时增加class，当switch为false时删除class。省略switch时则已有function()返回class则删除，否则则增加。参数index为匹配元素的编号；class为匹配元素现有class名称。
$.hasClass(className)	判断匹配元素是否含有className所指定的class，如有则返回true，否则返回false。

$.addClass()、$.removeClass()以及$.toggleClass()都支持function()，其含义是根据function()的返回值决定操作。比如：$.addClass()根据返回确定添加class的名称，如例程5-40所示。$.removeClass()与$.addClass()类似，只不过$.removeClass()是删除class。

例程5-40

```
<Script LANGUAGE="JavaScript">
        $("P").addClass(function(Index,Class){
                if(Index<8){
                        return "Wei";
                }else{
                        if(Index>12){
                                return "Wu";
                        }else{
                                return "Shu";
                        }
                }
        });
</Script>
```

第五节　属性操作

不少元素含有属性，如Img含有src等属性，table含有cellpadding等属性。jQuery可以方便地操作属性，$.attr(attrName)获取匹配元素的指定属性(attrName)的值(内容)，而$.attr(attrName,attrValue)则可以设置匹配元素指定属性(attrName)的值(attrValue)，$.removeAttr(attrName)则可以从选中元素移走(删除)指定属性(attrName)。更多操作如下表所示。

表6：属性操作

操作	描述
$.attr(attrName)	获取第一个匹配元素的属性值，如该元素没有指定属性(attrName)，则返回undefined。
$.attr(attrName/attrValue值对)	以“名/值”形式设置所有匹配元素的属性。
$.attr(attrName,attrValue)	为匹配元素指定属性(attrName)并设置属性值(attrValue)。
$.attr(attrName,function(){})	为匹配元素指定属性(attrName)设置由函数function()返回的属性值。
$.removeAttr(attrName)	删除所有匹配元素中指定的属性(attrName)。

例程5-11、例程5-42(效果如图5-15所示)是jQuery操作attr的应用。

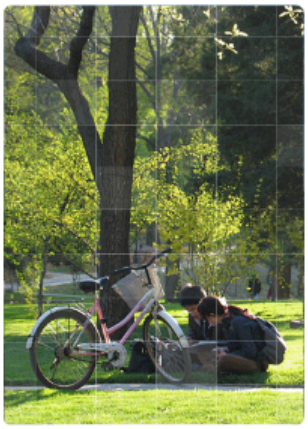
例程5-41

```
<Script Language="JavaScript">
        //读取img的src属性作为每个img元素的title属性值
        $("img").attr("title",function(){
                return $(this).attr("src");
        })
</Script>
```

例程5-42

```
<HTML>
    <HEAD>
        <TITLE>jQuery操作attr</TITLE>
        <Script language="JavaScript" src="./jquery-1.9.1.min.js"></Script>
        <META http-equiv="Content-type" content="text/html; charset=gb2312">
```

```
第6行                    <Style type="text/css">
第7行                        .imgBox{display:inline-block;width:220px;}
第8行                        .imgBox img{width:200px}
第9行                        #imgTitle{text-align:Center;font-weight:bold}
第10行                       #controlArea span{display:inline-block;width:30px;height:30px;border:1px solid red}
第11行                       .selectedSpan{background-color:red}
第12行                   </Style>
第13行           </HEAD>
第14行
第15行   <Body>
第16行           <Div class="imgBox">
第17行                   <img src="1.gif" ID="imgShow">
第18行                   <Div ID="imgTitle">春意</Div>
第19行                   <Div ID="controlArea">
第20行                       <span imgName="1.gif" imgTitle="春意" class="selectedSpan">□</span>
第21行                       <span imgName="2.gif" imgTitle="钟亭">□</span>
第22行                       <span imgName="3.gif" imgTitle="入画">□</span>
第23行                       <span imgName="4.gif" imgTitle="淡定">□</span>
第24行                       <span imgName="5.gif" imgTitle="绿塘">□</span>
第25行                   </Div>
第26行           </Div>
第27行
第28行           <Script Language="JavaScript">
第29行                   $("#controlArea>span").click(function(){
第30行
第31行                       $(this).addClass("selectedSpan");//为选定元素addClass
第32行                       //基于当前选定查找同辈class名为selectedSpan元素，并removeClass
第33行                       $(this).siblings(".selectedSpan").removeClass("selectedSpan");
第34行
第35行                       var thisImgTitle=$(this).attr("imgTitle");//获取当前元素的imgTitle属性
第36行                       var thisImgName=$(this).attr("imgName");//获取当前元素的imgName属性
第37行
第38行                       $("#imgTitle").html(thisImgTitle);//改变ID为imgTitle之html内容
第39行                       $("#imgShow").attr("src",thisImgName);//修改ID为imgShow的src属性值
第40行                   });
第41行           </Script>
第42行   </Body>
第43行   </HTML>
```



春意

图5-15　例程5-42执行效果图

第六节　读写CSS属性值

CSS是网页的重要组成部分，是网页美观的基础。jQuery提供了多种读取CSS属性的函数(方法)。$.addClass()和$.removeClass()能改变匹配元素的Class名称，而Class可以设定多种CSS效果。但是这种方法是将Class作为一个整体进行控制，而不能控制CSS每个项目。即$.addClass()和$.removeClass()的操作单位是class，而读取CSS属性则是具体到每个CSS项目。读取粒度有很大不同。

表7：读写CSS属性值

操作	描述
$.css(cssItemName)	读取第一个匹配元素的CSS属性的值。
$.css(cssItemName,value)	设置所有匹配元素指定样式(cssItemName)的属性值。注意：数字将被转换为像素值。
$.css(cssItemName,function(index,value){})	设置所有匹配元素指定样式(cssItemName)的属性值(由function()返回值确定)。数值将自动转换为像素值。index为元素的索引序号，value为原cssItemName的值，返回值为设置值。如例程5-43所示，执行效果如图5-16所示。
$.css(cssItemName/Value值对)	把一个“cssItemName/Value值对”设置为所有匹配元素的样式属性，适用于在所有匹配的元素上设置多属性。例程5-43第15行是其应用示例。
$.offset()	读取匹配元素相对于文档左上角的相对偏移，返回值为一个对象，属性有top和left，分别对应于垂直偏移和水平偏移。例程5-44第15行、16行是其应用示例。
$.offset(坐标值对)	设置匹配元素相对于文档偏移坐标，坐标值对形如{top:10,left:30}。例程5-44第35行是其应用示例。
$.position()	读取匹配元素相对于父元素的偏移，返回值对象与$.offset()相同。
$.scrollTop()	获得匹配元素相对于垂直滚动条顶部的偏移。
$.scrollTop(value)	设置匹配元素相对于垂直滚动条顶部的偏移。
$.scrollLeft()	获得匹配元素相对于水平滚动条左侧的偏移；
$.scrollLeft(value)	设置匹配元素相对于水平滚动条左侧的偏移；
$.height()	获得第一个匹配元素的高度值(单位为px)；
$.height(val)	设置匹配元素的高度值，如未指明单位，则默认为px；
$.width()	获得第一个匹配元素的宽度值(单位为px)；
$.width(val)	设置匹配元素的宽度值，如未指明单位，则默认为px；
$.innerHeight()	获取第一个匹配元素内部高度(包括补白、不包括边框)，对可见和隐藏元素均有效
$.innerWidth()	获取第一个匹配元素内部宽度(包括补白、不包括边框)，对可见和隐藏元素均有效
	获取第一个匹配元素外部高度(默认包括补白、边框)，对可见和隐藏元素均有效。如

$.outerHeight(switch)	switch值为false，则不包括补白和边框。
$.outerWidth(switch)	获取第一个匹配元素外部宽度(默认包括补白、边框)，对可见和隐藏元素均有效。如switch值为false，则不包括补白和边框。

例程5-43

```
第1行  <Html>
第2行      <Head>
第3行          <Title>CSS属性设置</Title>
第4行          <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行      </Head>
第6行      <Body>
第7行          <Div class="BigMan">
第8行              <P>诸葛亮</P>
第9行              <P>关羽</P>
第10行             <P>张飞</P>
第11行             <P>赵云</P>
第12行             <P>黄忠</P>
第13行         </Div>
第14行         <Script LANGUAGE="JavaScript">
第15行             $("P").css({"font-size":"12px","margin":"0px"});//用key/Value设置CSS属性
第16行             //用户function()函数的返回值设置属性
第17行             $("P").css("font-size",function(index,cssValue){
第18行                 //注意:font-size的属性值含有px字符，形如:19px。
第19行                 //cssValue为当前元素font-size的CSS属性值
第20行                 //相当于$(this).css("font-size");
第21行                 var currentFontSize=cssValue;
第22行                 currentFontSize=currentFontSize.replace("px","");//替换px为空白，相当于删除px
第23行                 return currentFontSize*(index+1)+"px";//返回值用于设置属性
第24行             });
第25行         </Script>
第26行     </Body>
第27行 </Html>
```

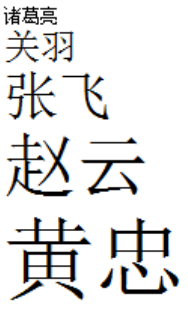


图5-16　例程5-43执行效果

例程5-44

```
第1行  <Html>
第2行      <Head>
```

```
第3行                    <Title>元素横向弹动</Title>
第4行                    <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行            </Head>
第6行            <Body>
第7行                    <Div id="Box" style="width:300px;height:100px;border:1px solid gray;padding-top:30px">
第8行                            <span style="color:red">●</span>
第9行                    </Div>
第10行                   <Script LANGUAGE="JavaScript">
第11行
第12行                           setTimeout("walkingDiv(true)",10);
第13行
第14行                           function walkingDiv(Direction){
第15行                                   var BoxPosition=$("#Box").offset();//取得id名为Box元素的相对于文档左上角
的偏移
第16行                                   var BoxLeft=BoxPosition.left;//左边的值
第17行                                   var BoxRight=BoxLeft+$("#Box").width();//到右边框的值
第18行                                   var myDiv=$("#Box>span");//取得span元素;
第19行                                   var nowPosition=myDiv.offset();//span元素相对于文档左上角的偏移量
第20行                                   if(Direction==true){//根据Direction的值确定左移动还是右移动
第21行                                           var newLeft=nowPosition.left+2;//右移
第22行                                   }
第23行                                   else{
第24行                                           var newLeft=nowPosition.left-2;//左移
第25行                                   }
第26行                                   if(newLeft>=BoxRight-BoxLeft){//到达右边界处理
第27行                                           newLeft=newLeft-2;//左移
第28行                                           Direction=false;//设定移动方向
第29行                                   }
第30行                                   if(newLeft<=BoxLeft){//到达左边界处理
第31行                                           newLeft=newLeft+2;//右移
第32行                                           Direction=true;//设定移动方向
第33行                                   }
第34行                                   var newTop=nowPosition.top;//顶坐标不改变
第35行                                   myDiv.offset({left:newLeft,top:newTop});
第36行
第37行                                   //启动下一次移动
第38行                                   setTimeout("walkingDiv("+Direction+")",10);
第39行                           }
第40行                   </Script>
第41行           </Body>
第42行   </Html>
```

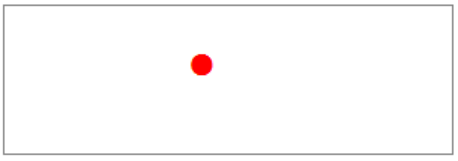


图5-17　小球横向弹动效果

第七节　动画效果

观察图5-18，有些像电影胶片。假定这些图片都出现在第一个图片框中，可以想象随着时间推移，图片在从右到左移动进场，其实现代码如例程5-45所示。在例程中，“setInterval("slidePic()",40);”的功能是每个40毫秒执行一次slidePic()这个函数；而slidePic()的核心功能是每次减小图片left值20个像素(Left是一个CSS属性)。综合其描述，其功能就是每隔40毫秒，图片的left值减小20个像素，视觉效果就是图片从右到左运动进场。

图5-18　动画效果示意图

jQuery提供了动画函数，如例程5-45所示，一行代码即可实现。“$("#Box>img").animate({"left": '0px'}, "slow");”的功能是选中id为Box下的img元素，并执行动画函数，修改其left属性，直到其值变化为0px为止，其动画速度为slow(慢)。凡元素CSS数值属性都可以用animate函数予以控制，实现动画效果，如:Left、Top、Height、Width、Padding、Margin、Font-Size等等。jQuery动画的本质就是以指定的速度(单位时间内改变值，如40毫秒减少10个像素)改变CSS属性，其速度可通过计算按照某种规律变化，不一定匀速改变。

例程5-45

```
第1行  <Html>
第2行      <Head>
第3行          <Title>动画</Title>
第4行          <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行          <Style type=text/css>
第6行              #Box{width:152px;height:214px;overflow:hidden;background-color:#EBEBEB;display:in
line-block}
第7行              #Box Img{position:relative;width:152px;left:152px}
第8行          </Style>
第9行      </Head>
第10行     <Body>
第11行         <Div id="Box"><Img src="3.gif"></Div>
第12行         <Script LANGUAGE="JavaScript">
第13行             setInterval("slidePic()",40);
第14行             function slidePic(){
第15行                 var myImg=$("#Box>img");
第16行                 var position=myImg.position();
第17行                 var positionLeft=position.left;
第18行                 positionLeft-=20;
第19行                 myImg.css("left",positionLeft+"px");
第20行             }
第21行         </Script>
第22行     </Body>
第23行 </Html>
```

例程5-46

```
第1行    <Script LANGUAGE="JavaScript">
第2行        $("#Box>img").animate({"left": '0px'}, "slow");
第3行    </Script>
```

$.animate()功能强大，jQuery还提供其他简捷的动画函数(方法)，如表8所示。

表8：动画方法

操作	描述
$.show()	显示隐藏的匹配元素。
$.show(speed[,function(){}])	以动画方式显示所有匹配元素，并可在动画完成后触发并执行函数function()(可选)。显示时根据指定的速度动态地改变每个匹配元素的高度、宽度以及不透明度等。
$.hide()	隐藏匹配元素。
$.hide(speed[,function(){}])	以动画方式隐藏所有匹配元素，并可在动画完成后触发并执行函数function()(可选)。隐藏时根据指定的速度动态地改变每个匹配元素的高度、宽度以及不透明度等。
$.toggle()	切换元素的可见状态，如可见，则切换为隐藏，如隐藏，则切换为可见。相当于$.show()和$.hide()的组合。
$.toggle(switch)	switch为true或false。如为true，则相当于$.show()，如为false，则相当于$.hide()。
$.toggle(speed[,function(){}])	切换元素可见状态(隐藏或显示)。相当于$.show()和$.hide()的组合。
$.slideDown(speed[,function(){}])	通过高度变化(向下增大)动态显示所有匹配元素，在动画完成后触发并执行可选函数function()。
$.slideUp(speed[,function(){}])	通过高度变化(向上减小)动态显示所有匹配元素，在动画完成后触发并执行可选函数function()。
$.slideToggle(speed[,function(){}])	通过高度变化来切换所有匹配元素，相当于$.slideDown()和$.slideUp()的组合。
$.fadeIn(speed[,function(){}])	通过透明度变化实现所有匹配元素的淡入效果(最终显示)，并在动画完成后触发并执行可选函数function()。
$.fadeOut(speed[,function(){}])	通过透明度变化实现所有匹配元素的淡出效果(最终隐藏)，并在动画完成后触发并执行可选函数function()。
$.fadeTo(speed,opacity[,function(){}])	通过透明度变化实现所有匹配元素的渐变到指定透明度opacity，并在动画完成后触发并执行可选函数function()。
$.animate(properties[,duration[,easing[,function(){}]]])	$.animate()函数的简单形式，如例程5-49所示。properties用于设置CSS属性及其属性值(动画完成时的终值)，以对象形式申明，如例程5-49第10行所示。duration用于设置动画速度，单位为毫秒，也可以使用slow、normal和fast。easing用于设置easing插件，easing的使用前提是正确应用easing插件，如如例程5-49所示第4行所示，easing使用如第10行之“easeOutBounce”。function()为动画执行完毕后触发并执行的函数，与前述动画函数相同。
$.animate(properties,options)	$.animate()函数的完整形式，如例程5-50所示。有关options的相关参数，请参见表9所示。
$.stop(stop([clearQueue],[gotoEnd]))	停止在匹配元素上正在运行的动画。
$.delay(duration[,queueName])	适用于动画的时间延迟，duration为时间，单位为毫秒；queueName为动画对象名称，省略为系统默认动画，也可设置为自定义动画名称。
方括号表示可选参数。speed可设置为slow、normal和fast，也可设置为具体数值，单位毫秒。	

表9：自定义动画options参数选项

操作	描述

duration	取值String或Number，默认值为400毫秒。duration用于设置动画速度，单位为毫秒(数值)，也可以使用slow、normal和fast。
easing	取值String，默认值为swing。easing用于设置easing插件，easing的使用前提是正确应用easing插件。
queue	取值String或Boolean，默认值为true。queue用于设置动画是否进入动画序列。如果取值为false，则后续动画立即启动，否则动画先后启动。如例程5-50单击“按钮1”则是3个动画同时启动，而“按钮2”则是3个动画先后启动。
specialEasing	取值自定义对象。用于设置多个CSS属性以及对应变化模式(easing选项)，如例程5-51第14行。
step	每个动画每个CSS属性每一步执行时触发并执行的函数。
progress	每个动画被执行后所触发并执行的函数。
complete	成功执行完毕后触发并执行的函数。
done	动画结束时触发并执行的函数。
fail	动画执行失败所触发并执行的函数。
always	当动画完成或停止不成功所触发并执行的函数。

不少动画方法在动画完成都可以触发并执行一个函数，例程5-47是用动画函数slideDown()将一个隐藏的图片显示，并在显示完毕后执行一个函数显示“动画结束”4个汉字。例程5-48是对例程5-47改进，让动画完成后执行的函数能将图片的标题以动画的形式显示在图片底部(执行效果如图5-19所示)。

例程5-47

```
第1行  <Html>
第2行      <Head>
第3行          <Title>动画</Title>
第4行          <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行      </Head>
第6行      <Body>
第7行          <Img src="3.gif" style="display:none" alt="湖入画 画如湖">
第8行          <Script LANGUAGE="JavaScript">
第9行              $("img").slideDown("slow",function(){
第10行                 alert("动画结束");
第11行             });
第12行         </Script>
第13行     </Body>
第14行 </Html>
```

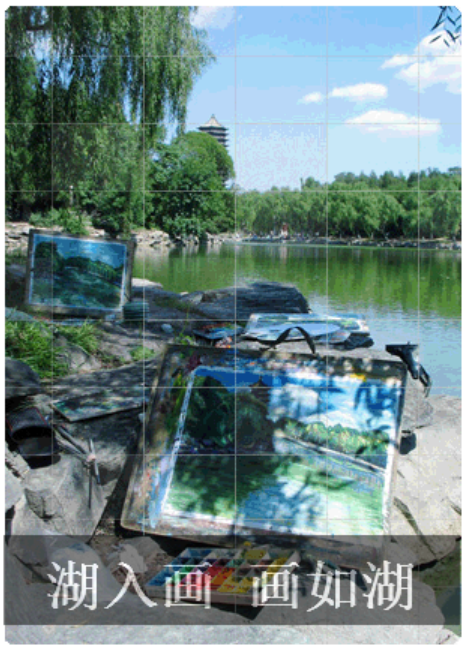


图5-19　例程5-48执行效果图

例程5-48

```
<Html>
    <Head>
        <Title>动画</Title>
        <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
        <Style type=text/css>
            #imgTitle{display:none;position:absolute;color:white;padding-top:15px;
                font-weight:bold;text-align:center;font-size:2em}
        </Style>
    </Head>
    <Body>
        <Img src="3.gif" style="display:none" alt="湖入画 画如湖">
        <Div id="imgTitle"></Div>
        <Script LANGUAGE="JavaScript">
            $("img").slideDown("slow",function(){//图片显示完毕后执行的函数
                var imgPosition=$(this).offset();//取得图片位置
                var imgHeight=$(this).height();//取得图片高度
                var imgWidth=$(this).width();//取得图片宽度
                var imgTitle=$(this).attr("alt");//取得图片标题
                $("#imgTitle")
                    //设置DivCSS属性
                    .css({"width":imgWidth,"height":"60px","background-color":"black",opacity:0.5})
                    .offset({left:imgPosition.left-3,top:imgPosition.top+imgHeight-75})//设置图片位置
                    .html(imgTitle)//设置元素内容
                    .slideDown("slow");//动画显示该元素
            });
        </Script>
```

```
第27行          </Body>
第28行    </Html>
```

$.animate()都支持一个叫easing的插件，能让动画更加自然，例程5-49是一种弹簧弹动的效果(运行代码能更好地感知)。使用easing功能需要下载插件，其引用方式与jQuery类似，如第5行所示。

例程5-49

```
第1行    <Html>
第2行        <Head>
第3行            <Title>动画插件的应用</Title>
第4行            <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行            <Script Language="JavaScript" src="./jquery.easing.1.3.min.js"></Script>
第6行        </Head>
第7行        <Body>
第8行            <Img src="3.gif" width="20">
第9行            <Script LANGUAGE="JavaScript">
第10行                $("img").animate({width:"302px"},"slow","easeOutBounce");
第11行           </Script>
第12行       </Body>
第13行   </Html>
```

例程5-50是$.animate()完整形式的应用。当单击id名为btnGo1的按钮时，3个animate()函数同时执行，而单击id名为btnGo2的按钮时，3个animate()则是先后执行，即执行完毕一个函数后，再执行另外的animate()函数。区别就是queue参数的使用，如果不使用queue参数，默认值为true。

例程5-50

```
第1行    <Html>
第2行        <Head>
第3行            <Title>自定义动画$.animate()的应用</Title>
第4行            <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行            <Style type="text/css">
第6行                #txtDiv1,#txtDiv2{border:1px solid red;width:80px}
第7行            </Style>
第8行        </Head>
第9行        <Body>
第10行           <button id="btnGo1">按钮1</button>
第11行           <button id="btnGo2">按钮2</button>
第12行           <div id="txtDiv1">程序设计</div><BR>
第13行           <div id="txtDiv2">思维体操</div>
第14行           <Script LANGUAGE="JavaScript">
第15行               $("#btnGo1").click(function(){
第16行                $("#txtDiv1").animate( { width: "400px"},{queue: false, duration: 1000 })
第17行                        .animate( { fontSize: '5em' } ,{queue: false, duration: 1000 })
第18行                        .animate( { borderLeftWidth:15 }, 1000);
第19行               });
第20行
第21行               $("#btnGo2").click(function(){
第22行                $("#txtDiv2").animate( { width: "400px"}, 1000 )
```

```
第23行                                  .animate( { fontSize: '5em' } , 1000 )
第24行                                  .animate( { borderRightWidth:15 }, 1000);
第25行                              });
第26行                      </Script>
第27行              </Body>
第28行      </Html>
```

图5-20　例程5-50执行效果图

例程5-51是specialEasing的应用实例，其width是线性变化，而height是弹簧模式。使用specialEasing的前提是正确引用Easing插件，如第4行所示。

例程5-51

```
第1行    <Html>
第2行            <Head>
第3行                    <Title>自定义动画之specialEasing应用</Title>
第4行                    <Script Language="JavaScript" src="./jQuery-1.9.1.min.js"></Script>
第5行                    <Script Language="JavaScript" src="./jquery.easing.1.3.min.js"></Script>
第6行            </Head>
第7行            <Body>
第8行                    <Img src="3.gif" width="20">
第9行                    <Script LANGUAGE="JavaScript">
第10行                          $("img").animate(
第11行                                  {width:"304px",height:"426px"},
第12行                                  {
第13行                                          duration:5000,
第14行                                          specialEasing:{width: 'linear',height: 'easeOutBounce'}
第15行                                  }
第16行                          );
第17行                   </Script>
第18行           </Body>
第19行   </Html>
```

第八节　其他

1、$.is(selector)

is用于检查选中的集合中是否有selector指定的元素，如果有则返回true，否则false，例程5-？是其应用示例。

例程5-52

```
<SCRIPT LANGUAGE="JavaScript">
        var isBigMan=$("P").parent().is(".BigMan");//isBigMan的值为true
        var isPContain=$("Div").children("P").is("p:contains('陆逊')");//isPContain的值为true
</SCRIPT>
```

2、$.map(function(){})

将选中的元素转换成数组，并可在转换过程中进行处理，如例程5-？所示。

例程5-53

```
<SCRIPT LANGUAGE="JavaScript">
        var arrLeader=$(".Leader").map(function(){
                return $(this).text();
        }).get();
        alert(arrLeader);//显示为曹操、刘备、孙权
</SCRIPT>
```

$.map有两种形式：$(selector).map(function(){})和$.map(array,function(){})，前一种形式是将选中的元素或对象变成数组，后一种形式是对数组的处理，例程5-？是其应用。在第7行，$.map是对已有数组arrLeader的处理，并形成新的数组arrNew。

例程5-54

```
<SCRIPT LANGUAGE="JavaScript">
        var arrLeader=$(".Leader").map(function(){
                return $(this).text();
        }).get();
        alert(arrLeader);//显示为曹操、刘备、孙权

        var arrNew=$.map(arrLeader,function(element,index){
                return "编号"+index+":"+element;
        });
        alert(arrNew);//编号1:曹操,编号2:刘备,编号3:孙权
</SCRIPT>
```

第九节　更多了解

1、jQuery串联操作

第十节　小结

1、本章总结

2、学习方法