第三章　jQuery初步

本章导读

HTML和CSS搭档，网页仅能呈现静态效果。Javascript是一种运行在浏览器中基于事件面向对象的脚本语言，能响应用户操作，呈现动态效果。jQuery基于JavaScript，让查找和操纵网页元素以及数据访问等变得相当简单，且能更好地支持多种浏览器。

学习目标：

1. 了解JavaScript、jQuery书写习惯；
2. 掌握jQuery之$符号的功能；
3. 掌握jQuery之事件格式；
4. 掌握jQuery之class和attr操作；

本章目录

第一节　快速入门

例程3-1是一个简单的JavaScript程序，和已学过的网页相比，BODY内增加了Script元素，其属性Language说明其内的代码编写语言为JavaScript，代码从第9行开始到第18行为止。

例程3-1

```
第1行  <Html>
第2行      <Head>
第3行          <Title>第一个JavaScript程序</Title>
第4行          <Meta http-equiv="Content-type" content="text/html; charset=gb2312">
第5行      </Head>
第6行      <Body>
第7行          <Div>求三角形斜边长</Div>
第8行          <Script Language="JavaScript">
第9行              /*已知直角三角形两个直角边长，求斜边长*/
第10行             var sideA=3;
第11行             var sideB=4;
第12行             var sideC;
第13行
第14行             sideC=Math.sqrt(sideA*sideA+sideB*sideB);
```

```
第15行                    alert(sideC);
第16行
第17行                    sideC=Math.sqrt(3*3+4*4);
第18行                    alert(sideC);
第19行                </Script>
第20行            </Body>
第21行    </Html>
```

第10行的含义是让sideA的值为3(相当于数学中“令直角边a等于3”)，在程序设计中通常称之为赋值语句，即将等号右边的3赋值给等号左边的sideA，sideA被称之为变量，也称引用量，即使用其代表的值。首次使用变量时，名称前用var(variable的简写)予以申明其为一个新的变量。在程序设计中，变量通常遵循“先申明后使用”。变量可以仅申明然后供其他语句使用，如第12行所示，第14行使用第12行申明的变量sideC。

第14行的Math.sqrt()的含义是求平方根，Math是JavaScript的数学对象，包含有不少数学函数和常量等。对象与其函数之间用句点连接，表示sqrt()这个功能在Math对象中，不能离开Math单独使用sqrt()函数。Math对象必须写成Math，不能写成其他，如math、MATH都不能表示Math对象。被求平方根的内容须放在sqrt后面的括号内，即求括号内计算结果的平方根。括号内的星号(*)代表乘号，除法用斜杠(/)表示。

第15行的alert()也是函数，是JavaScript自带函数，可以称之为系统函数，其功能是显示括号中的结果。在本例中显示sideC所代表的内容，显示效果如图3-1所示。

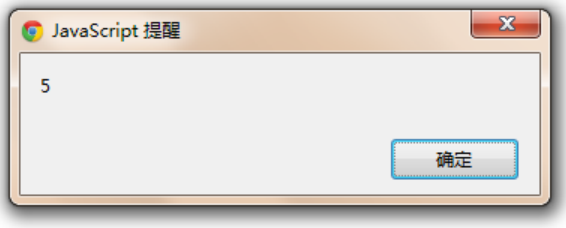

图3-1　alert函数弹出的对话框

第9行是注释语句。注释语句对程序执行结果没有任何影响，但有利于开发人员管理和维护程序，好的习惯是多写注释。和HTML标记相似，开始标记是/*，结束标记是*/。另外，如果是单行注释，还可以用//撰写注释。如第10行可以写成“//已知直角三角形两个直角边长，求斜边长”。

第17行也是求sideC的值，与第14行相比，相当于sideA代表3，sideB代表4，能感觉到第17行更为简洁易懂，为什么要采用变量呢？请看例程3-2。在例程中，利用两个直角边求斜边长、周长、面积等，修改sideA和sideB的值即可，其他部分会自动随之变化。虽然可以将sideA和sideB换成具体的数，但是如果要多次修改，则修改量会大大增加，并且随着修改位置增多，出现错误的概率也随之增大。

例程3-2

```
第1行    <Html>
第2行        <Head>
第3行            <Title>初学JavaScript程序</Title>
第4行            <Meta http-equiv="Content-type" content="text/html; charset=gb2312">
第5行        </Head>
第6行        <Body>
第7行            <Div>求三角形相关数据</Div>
第8行            <Script Language="JavaScript">
第9行                /*已知直角三角形两个直角边长，求斜边长*/
第10行               var sideA=3;
第11行               var sideB=4;
第12行               var sideC,triArea,triPeri;//分别代表边长C，三角形面积和周长
第13行
第14行               sideC=Math.sqrt(sideA*sideA+sideB*sideB);
```

```
第15行
第16行                          triArea=sideA*sideB/2;//面积，两个直角边乘积的二分之一
第17行                          triPeri=sideA+sideB+sideC;//周长，三条边之和
第18行
第19行                          alert("斜边长="+sideC+"  周长="+triPeri+"  面积="+triArea);
第20行                      </Script>
第21行              </Body>
第22行      </Html>
```

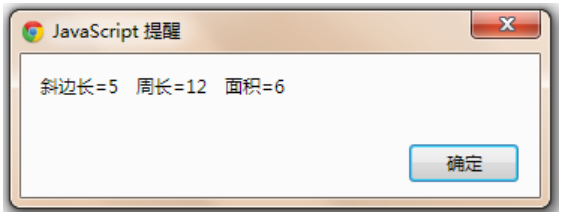

图3-2　例程3-2执行效果

和例程3-2相比，例程3-3没有了sideC变量，其执行效果相同。通过比较可以看出，sideC被Math.sqrt(sideA*sideA+sideB*sideB)代替，但是同样代码出现在多个位置且代码较长，如果用变量代替，代码将简洁易懂。如果sideC的计算方法有调整，在例程3-2调整第14行即可且计算一次，但例程3-3则要调整两处，其每次执行其代码都需要重复计算。因此，用变量代替计算结果，不但代码简洁易读，便于代码维护且减少计算量。

例程3-3

```
第1行   <Html>
第2行           <Head>
第3行                   <Title>初学JavaScript程序</Title>
第4行                   <Meta http-equiv="Content-type" content="text/html; charset=gb2312">
第5行           </Head>
第6行           <Body>
第7行                   <Div>求三角形相关数据</Div>
第8行                   <Script Language="JavaScript">
第9行                           /*已知直角三角形两个直角边长，求斜边长*/
第10行                          var sideA=3;
第11行                          var sideB=4;
第12行                          var triArea,triPeri;//三角形面积和周长
第13行
第14行                          triArea=sideA*sideB/2;//面积，两个直角边乘积的二分之一
第15行                          triPeri=sideA+sideB+Math.sqrt(sideA*sideA+sideB*sideB);//周长，三条边之和
第16行
第17行                          alert("斜边长="+Math.sqrt(sideA*sideA+sideB*sideB)+"  周长="+triPeri+"  面积="+tr
iArea);
第18行                  </Script>
第19行          </Body>
第20行  </Html>
```

例程3-2第19行和例程3-3第17行都有配对英文引号及其内部文字，这样的内容被称之为字符串(String)，称var sideA=5中的5为数值(Number)。数值型数据和字符串型数据是程序设计语言中两种重要的数据类型。变量可以存数值型数据，也可以存字符型数据，如例程3-4所示。strFirst、strSecond、strThird和strForth都是变量，都代表字符型数据。

例程3-4

```
第1行           <Script Language="JavaScript">
```

```
第2行          var strFirst="Hello!";
第3行          var strSecond="World!";
第4行          var strThird=strFirst+strSecond;
第5行          alert(strThird);//显示为Hello!World!
第6行
第7行          //注意引号的使用
第8行          var strForth="strFirst"+"strSecond";
第9行          alert(strForth);//显示为strFirststrSecond
第10行       </Script>
```

注意例程3-4第4行中的加号(+)，其功能是将两个字符串拼接在一起。字符型数据不支持减号(-)、乘号(*)或者除号(/)运算符。第8行是将两个变量两侧加上引号，其含义完全发生变化，不再是两个变量所代表的值进行拼接，而是两个字符串直接拼接，与变量所代表的值没有任何关系。在程序设计语言中，将带引号的字符串称之为字符串直接量，将变量称之为引用量。直接量和引用量都有数据类型，比如：字符型直接量、数值型直接量、字符型引用量、数值型引用量。对于字符型直接量必须加上引号，数值型直接量不能加上引号(如例程3-2和例程3-3中的3、4等)，引用量(变量)都不能加上引号。注意：只有字符型直接量需要加上引号，并注意引号必须配对使用。

一般说来，只有相同的数据类型，才能在一起运算，当不同的数据类型在一起运算时，JavaScript常能自动进行数据类型转换，如例程3-5所示。在第17行，和数学一样，优先执行内部括号，然后逐层展开。首先执行(-dataB)，虽然dataB是字符型数据，但不支持负号运算符，所以被自动强制转换成数值型数据，即-2，然后执行其外层括号，得到5，然后再与其他变量或直接量做运算。

例程3-5

```
第1行    <Script Language="JavaScript">
第2行          var firstData="钢铁侠";
第3行          var secondData=2;
第4行          var thirdData=firstData+secondData;//数值2自动转换为字符型数据运算
第5行          alert(thirdData);//钢铁侠2
第6行
第7行          var dataA="3";
第8行          var dataB="2";
第9行          var dataC=dataA+dataB;//字符串拼接运算
第10行         alert(dataC);//显示为32
第11行
第12行         dataD=dataA-dataB;//字符串不能不做减法运算，自动转换为数值型数据运算
第13行         alert(dataD);//显示为1
第14行
第15行         alert(dataA+"-"+dataB+"="+dataA+dataB);//显示为3-2=32，全部是字符型运算
第16行         alert(dataA+"-"+dataB+"="+(dataA-dataB));//显示为3-2=1，优先做内部括号运算
第17行         alert(dataA+"+"+dataB+"="+(dataA-(-dataB)));//显示为3+2=5，注意括号运算
第18行   </Script>
```

自动转换如同潜规则总是不如显规则更容易理解。parseInt()可以将括号内的参数转换为整数；parseFloat()可以将括号内的参数转换为浮点数(带小数点的数)；Number()将括号内的参数转换为数值型数据；String()将括号内的参数转换为字符型数据。例程3-6是数据转换的应用示例。

例程3-6

```
第1行    <Script Language="JavaScript">
第2行          //要注意括号的配对，如同数学中一样，
第3行          //没有大括号之类等，只有圆括号
第4行          var X1="3";
第5行          var X2="2";
```

```
第6行            var X3=168;
第7行            var X4=parseInt(X1)+parseInt(X2);//X4值为5
第8行            var X5=X1+X2;//X5值为字符型数据：32
第9行            alert(X1+"-"+X2+"="+(parseInt(X1)-parseInt(X2)));//显示为3-2=1
第10行           alert(X1+"+"+X2+"="+(parseInt(X1)+parseInt(X2)));//显示为3+2=5
第11行           alert(X1+"-"+X2+"="+String((parseInt(X1)-parseInt(X2))));//显示为3-2=1
第12行
第13行           var X6="吉祥数字"+String(X3);//X6为字符型数据：吉祥数字168
第14行       </Script>
```

前面例程都是固定值，不能计算由用户输入的数据。如例程3-1根据直角三角形直角边计算斜边，如果能输入直角边的值，就更具有通用性。利用JavaScript提供的prompt()函数就可以实现用户数据的输入，如例程3-7所示。

例程3-7

```
第1行    <Script Language="JavaScript">
第2行        /*已知直角三角形两个直角边长，求斜边长*/
第3行        var sideA=prompt("请输入第一直角边长",3);//输入值默认为字符型
第4行        var sideB=prompt("请输入第二直角边长",4);//输入值默认为字符型
第5行
第6行        sideA=Number(sideA);//转换为数值型
第7行        sideB=Number(sideB);//转换为数值型
第8行
第9行        var sideC,triArea,triPeri;//分别代表边长C，三角形面积和周长
第10行
第11行       sideC=Math.sqrt(sideA*sideA+sideB*sideB);
第12行
第13行       triArea=sideA*sideB/2;//面积，两个直角边乘积的二分之一
第14行       triPeri=sideA+sideB+sideC;//周长，三条边之和，如果没有转换则得到字符串拼接，如345
第15行
第16行       alert("斜边长="+sideC+"  周长="+triPeri+"  面积="+triArea);
第17行   </Script>
```

注意prompt()函数有两个参数，即prompt(Para1,Para2)，其中Para1表示提示信息，Para2表示默认输入值(该参数可以省略)。prompt()函数执行结果即为输入值，其数据类型为字符型。如果不进行数据类型转换，sideA+sideB+sideC将成为字符型，如sideA为8，sideB为6，则sideC为数值型10，sideA+sideB+sideC将为字符型8610，如果将sideA和sideB转换为数值型，当然sideC自然也为数值型10，sideA+sideB+sideC则为24，而不是8610。

注意事项：

1. 大小写敏感。JavaScript是大小写敏感的语言，不同的大小写组合，含义不同或者错误。如math.sqrt()是错误的，正确的写法是Math.sqrt()，将alert()写成Alert()也是错误的。
2. 分号不能忘。JavaScript每个语句后都必须有分号(;)，分号是语句结束的标志。JavaScript允许多个语句写在同一行，语句和语句之间用分号分开，但不是好习惯。
3. 多写注释。注释对程序执行效果没有影响，但能提高代码可读性、让开发者本人和相关人员更快速正确地理解程序。
4. 规范命名。变量名称最好语义清楚，让开发者及其相关人员一眼即明白其功能。为了阅读方便，最好能遵守相应命名规范，如骆驼命名法(除第一个单词外，各个单词首字母大写，或者采用下划线连接每个单词，如side_A等)。另外，变量名称第1个字符必须是字母(大小写均可)、下划线(_)或美元符 ($)，后续字符可以是字母、数字、下划线或美元符。

前面的例程没有与HTML元素等发生联系，而例程3-8通过jQuery的特征符号$，可以方便地操作HTML标记，图3-3是其执行效果。和例程3-1相比，例程3-8第4行是新增内容，第23行到第25行是JavaScript代码。第4行的作用是引入jQuery库，否则不能使用jQuery功能。初学时，暂可认为该行是使用jQuery约定(这种模式使用jQuery需要开发者电脑连入互联网。不联网同样可以使用jQuery，请参考本章后面的进一步阅读)。

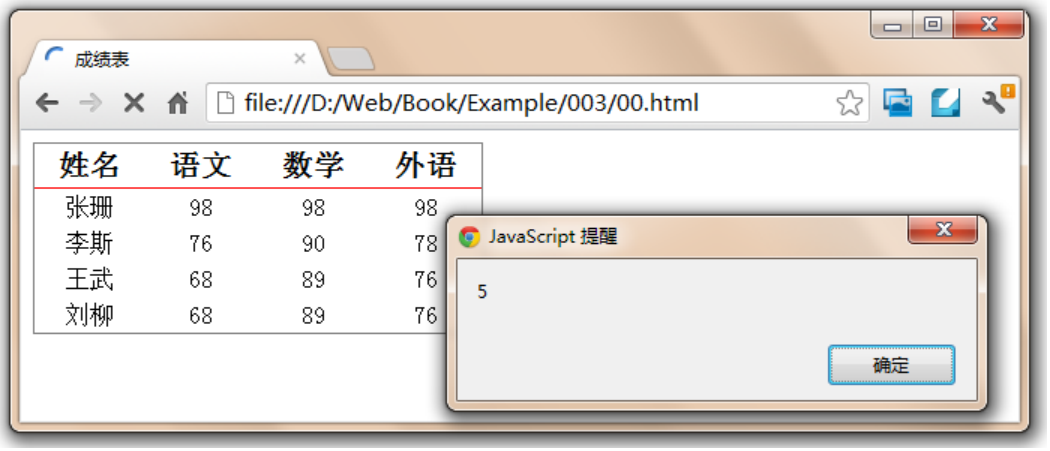

图3-3　显示成绩表行数量

例程3-8

```
第1行   <HTML>
第2行       <HEAD>
第3行           <TITLE>成绩表 </TITLE>
第4行           <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.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               TH{border-bottom:1px solid red;font-size:20px;Height:30px}
第8行               TD{text-align:Center;height:24px;font-size:16px}
第9行               .TR_Odd{background-color:#EAEAEA}
第10行              .HighLight{background-color:black;color:White;}
第11行          </Style>
第12行      </HEAD>
第13行
第14行      <BODY>
第15行          <Table style="width:300px;border:1px solid gray" cellpadding="0" cellspacing="0">
第16行              <TR><TH>姓名</TH><TH>语文</TH><TH>数学</TH><TH>外语</TH></TR>
第17行              <TR><TD>张珊</TD><TD>98</TD><TD>98</TD><TD>98</TD></TR>
第18行              <TR><TD>李斯</TD><TD>76</TD><TD>90</TD><TD>78</TD></TR>
第19行              <TR><TD>王武</TD><TD>68</TD><TD>89</TD><TD>76</TD></TR>
第20行              <TR><TD>刘柳</TD><TD>68</TD><TD>89</TD><TD>76</TD></TR>
第21行          </Table>
第22行          <SCRIPT LANGUAGE="JavaScript">
第23行              //trCount：表格行的数量
第24行              var trCount=$("TR").size();
第25行              alert(trCount);
第26行          </SCRIPT>
第27行      </BODY>
第28行  </HTML>
```

例程3-8第24行$("TR").size()的功能是取得网页中TR的数量，并将该数量赋值给trCount，第25行是通过alert()函数显示该数值并弹出相应对话框，如图3-3所示。

$符号是jQuery最重要的操作，是jQuery操作的基础，其最基本功能是选中对象，如$("TR")表示选中网页中所有TR标记。基本的选中方式如下：

1. $("#IDName")：选中网页中指定的ID，ID名称放在#号后，用以替代IDName；
2. $(".ClassName")：选中网页中指定的Class，Class名称放在.号后，用以替代ClassName；
3. $("HTMLTag")：选中网页中指定的HTML标记，标记前没有任何符号如井号和句点等，用相应标记替代HTMLTag即可，如TR，TD，DIV，SPAN等；

和在Head-Style和File-Style中定义CSS相比，jQuery选择元素方式与之非常相似。在Style片段中，定义元素的效果类似“P{font-size:12px}”，而定义class的效果则类似“.hideIt{display:none}”，定义id效果则类似“#myID{width:300px;}”。即id前面加上#，class前面加上英文句点，而元素则直接使用名称。

jQuery基本使用模式由两部分组成，即“选中+操作”，例程3-8第24行是选中TR，操作是size()。jQuery功能强大，如图3-4所示的效果(隔行灰色显示各行一目了然)在第25行后增加一行代码即可简单实现，如例程3-9所示。

姓名	语文	数学	外语
张珊	98	98	98
李斯	76	90	78
王武	68	89	76
刘柳	68	89	76

图3-4　奇数行特别显示效果图

例程3-9

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        //trCount：表格行的数量
第3行        var trCount=$("TR").size();
第4行        alert(trCount);
第5行        $("TR:odd").addClass("TR_Odd");
第6行    </SCRIPT>
```

用表格显示数据是比较常用的方式，为了一目了然，通常隔行略有区别，如图3-4所示。当数据行比较少时，可以手工在需要的行增加相应class，使奇数行或者偶数行别样显示。但是当行数很多，或者增减行数、调整顺序时，手工操作就会变得比较繁琐。通过jQuery语句，可以很容易实现，如例程3-9所示。

第二节　操作Class

用表格显示数据是比较常用的方式，为了一目了然，通常隔行略有区别，如图3-4所示。当数据行比较少时，可以手工在需要的行增加相应class，使奇数行或者偶数行别样显示。但是当行数很多，或者增减行数、调整顺序时，手工操作就会变得比较繁琐。通过jQuery语句，可以很容易实现，如例程3-10所示。

例程3-10

```
第1行  <HTML>
第2行      <HEAD>
第3行          <TITLE>成绩表 </TITLE>
第4行          <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.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              TH{border-bottom:1px solid red;font-size:20px;Height:30px}
第8行              TD{text-align:Center;height:24px;font-size:16px}
第9行              .TR_Odd{background-color:#EAEAEA}
第10行             .HighLight{background-color:black;color:White;}
第11行         </Style>
第12行     </HEAD>
第13行
第14行     <BODY>
```

```
第15行                    <Table style="width:300px;border:1px solid gray" cellpadding="0" cellspacing="0">
第16行                        <TR>
第17行                            <TH>姓名</TH><TH>语文</TH><TH>数学</TH><TH>外语</TH>
第18行                        </TR>
第19行                        <TR>
第20行                            <TD>张珊</TD><TD>98</TD><TD>98</TD><TD>98</TD>
第21行                        </TR>
第22行                        <TR>
第23行                            <TD>李斯</TD><TD>76</TD><TD>90</TD><TD>78</TD>
第24行                        </TR>
第25行                        <TR>
第26行                            <TD>王武</TD><TD>68</TD><TD>89</TD><TD>76</TD>
第27行                        </TR>
第28行                        <TR>
第29行                            <TD>刘柳</TD><TD>68</TD><TD>89</TD><TD>76</TD>
第30行                        </TR>
第31行                    </Table>
第32行                    <SCRIPT LANGUAGE="JavaScript">
第33行                        $("TR:odd").addClass("TR_Odd");
第34行                    </SCRIPT>
第35行            </BODY>
第36行    </HTML>
```

在第9行，定义了名为“.TR_Odd”的class，在表格中，没有一个TR引用该class，其效果通过第32到34行实现。在网页代码中，JavaScript相关代码必须写在HTML的SCRIPT标记对之间，该标记大小写不敏感，但是其间的代码大小写敏感。另外，每句JavaScript语句结束后，须用半角分号明确标记其结束，否则将导致错误。

在例程3-10中，第33行代码的功能是“选中奇数行TR并给其增加名为TR_Odd的class”，而该class已经定义了相应显示方式，从而实现了图3-4中的显示效果。

可以将第33行代码分为两个部分，即“选中+操作”，$("TR:odd")是选中对象，addClass("TR_Odd")是其操作。实际上Windows上的很多软件的操作方式大抵如此。如在资源管理器，可以选中一个文件，然后选择“删除”或者其他操作。在Office Word中，可以选中一段文字，然后设置其字体。只不过一个是可视化鼠标操作，一个是代码实现。实际上，这些软件的内部代码实现，也大抵如此。

上述选择方式与Style中定义CSS方式类似，ID前面必须有#(井号)，Class前面必须.(句点)，而HTML标记前则没有任何符号。另外，千万不要忘了英文双引号。

例程3-10的$("TR:odd")中，除了TR这个HTML标记外，还有odd，其含义是选中所有编号为奇数的TR。类似odd功能还有：

1. even：编号为偶数的对象；
2. first：第一个对象；
3. last：最后一个对象；
4. eq(index)：编号为index的对象；
5. lt(index)：编号小于index的对象；
6. gt(index)：编号大于index的对象；

如：$("TR:gt(0)")则表示编号大于0的所有对象，0替换gt(index)中的index。对于例程3-10而言，即是指非首行对象，首行对象的TR编号为0。

在jQuery中，会将所有选中的对象进行编号，编号顺序以网页中出现的先后为序，第一个出现的编为0(在很多程序设计语言中，编号从0开始)。在例程3-10中，第一个出现的TR为姓名所在的行，该TR被编为0。

图3-5是将例程3-10的表格内容替换为空白表格，然后将$("TR:odd").addClass("TR_Odd");替换为$("TD:odd").addClass("TR_Odd");即可达到该效果。对于TR_Odd这个class，是否要改变TD_Odd呢？实际上TR_Odd只是一个class的名

称，完全没有改变。如果当初针对TR时该名字为GrayBackground，也许就没有这样的疑惑，其功能的实现不是用字面名字确定的，而是与其定义以及该定义的引用有关。如果改变为TD_Odd且没有定义将毫无效果。

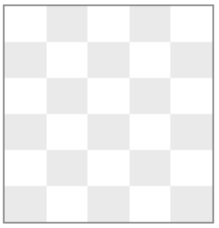

图3-5　单元格灰白交错显示

addClass()的功能是为匹配的元素增加指定的class，如果该class定义了显示效果，网页将立即按指定的效果显示。除了addClass()外，还有removeClass()，其功能与addClass()相反，即从匹配元素中移走class，如果该class定义了显示效果，则网页中匹配元素将立即没有该效果。如例程3-11所示，执行第4行代码将移走TR_Odd，原有奇数行的效果都将被移走。

例程3-11

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $("TR:odd").addClass("TR_Odd");
第3行        alert("用removeClass()移走TR_Odd，注意观察效果!");
第4行        $("TR:odd").removeClass("TR_Odd");
第5行    </SCRIPT>
```

第三节　CSS操作

CSS对网页美观至关重要。jQuery也提供了操作CSS的函数，如例程3-12所示。

例程3-12

```
第1行    <Script Language="JavaScript">
第2行        $("body").css("font-size","12px");
第3行    </Script>
```

例程3-12第2行代码就是通过$.css()函数设置CSS属性，在本例中，是选中Body元素，并设置其字号为12px。当然，可以选中元素的方法同样适用于其他元素、id或class，可以设置的CSS属性也可以是其他，如：background-color、width等CSS支持的属性。其CSS属性的写法与CSS标准一致，其取值也保持一致。

$.css()不仅可以设置CSS属性，还可以取得当前属性的值，如例程3-13所示。当仅有一个参数时，即获得指定属性的值。

例程3-13

```
第1行    <Script Language="JavaScript">
第2行        var nowSize=$("body").css("font-size");
第3行        alert(nowSize);//显示形如19px等，各个浏览器因为设置不同而有所变化，但形势保持一致
第4行    </Script>
```

$.css()函数不仅可以设置单个CSS属性，还可以一次设置多个CSS属性，使得使用更加方便，如例程3-14所示。从中可以看出，多个CSS属性设置类似网页中CSS定义。在设置多个CSS属性的$.css()函数中，设置条目必须放在一对花括号中“{}”，每个条目采用“键/值”模式，即一个条目一个值，条目和值之间用冒号(:)分隔，多个“键/值”之间用逗号(,)分隔。另外，$.css()函数可以取得CSS属性值，但每次仅能取得一个属性值，不支持一次取得多个属性值。

例程3-14

```
第1行    <Script Language="JavaScript">
第2行        $("body").css({"font-size":"12px","background-color":"gray"});
第3行    </Script>
```

第四节　事件处理

不少网站的数据表格，还提供鼠标跟随功能，即当鼠标移动到某行时，该行高亮显示，醒目地提醒用户鼠标所指向的行，如图3-6所示，其“王武”所在行用黑底白字方式醒目显示。该功能实现代码如例程3-15所示，将该段代码置于例程3-10第33行后即可。

姓名	语文	数学	外语
张珊	98	98	98
李斯	76	90	78
王武	68	89	76
刘柳	68	89	76

图3-6　鼠标跟随显示

例程3-15

```
第1行    $("TR").mouseover(function(){
第2行        $(this).addClass("HighLight");
第3行    });
第4行    $("TR").mouseout(function(){
第5行        $(this).removeClass("HighLight");
第6行    });
```

例程3-15程序的第1行、第4行中都有$("TR")，其含义是选中网页中所有TR标记，其后的mouseover(鼠标进入)和mouseout(鼠标离开)被称之为事件(event)。常用事件还有click(单击)、dblclick(双击)以及keydown(键盘键按下)、keyup(键盘键松开)等。虽然事件不同，但使用方法相同。当事件发生时，将执行的指定代码，其基本格式如下，将要执行的代码写在function()后的花括弧中。

```
对象.事件名称(function(){});
```

对于例程3-15第1-3行代码可以改造为$("TR").mouseover(function(){$(this).addClass("HighLight");});即将3行融为1行，其中的$(this).addClass("HighLight");为mouseover事件发生将要执行的代码。但这种书写模式密密麻麻不利于阅读，一般都从“function(){”后分行，然后书写代码，结尾的“});”放在最后独立一行，便利于阅读和调试。尤其是当事件发生后要执行的代码不止一行时，更是如此。

由于括号较多，如果发生括号不匹配，将导致错误。建议先写$("TR").mouseover();然后将光标移动到括弧内，写下function(){}，即$("TR").mouseover(function(){});，在此基础上，将光标移动到花括号内，回车换行，然后写下相关代码。这样的写法基本能规避括号不匹配问题。当然选中对象和事件名称可以根据需要调整。

例程3-15代码中的第2行和第5行有些相似。从字面意思可以看出，addClass是向被选中对象增加指定Class，removeClass是移走被选中对象的Class。第1-3行的功能是当鼠标移动到选中对象的某行时，向改行所对应的TR增加名为HighLight的Class，该Class能将背景设置为黑色(black)，字体颜色设置为白色(white)。当鼠标移出该行时，移走改行所对应的名为HighLight的Class。

例程3-15的第2、5行中，有$(this)，其功能仍然是选中对象，只不过选中的是mouseover或者mouseout事件发生时的对象，而不是所有TR对象。如果将this改为"TR"，则将当鼠标指向某行时，所有行都会被增加名为HighLight的Class。当鼠标移走时，所有行都将被移走该Class。注意：this不能加上引号，加上引号即意味着找到一个名为this的HTML元素，而该标记是不存在的。

在例程3-15中应用到了mouseover和mouseout属性，常用的属性还有click(鼠标单击)、dblclick(鼠标双击)等，更多的事件将在相关章节中讲述。例程3-16是click事件的应用。从代码中可以看出mouseover和mouseout被替换成了click，addClass()和removeClass()被替换成toggleClass()。toggleClass()的功能是当被选中元素有名为HighLight的class时，移走该class，相当于removeClass()；如果没有则增加该class，相当于addClass()。

例程3-16

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $("TR").click(function(){
第3行            $(this).toggleClass("HighLight");
第4行        });
第5行    </SCRIPT>
```

例程3-15有一个明显的不足，即当鼠标移动到标题行时会变成黑底白字，而标题行通常不需要这样的效果。可将例程3-2中的$("TR")部分修改成$("TR:gt(0)")即仅限于编号大于0的TR行。标题行是第一个出现的TR，编号为0，通过该语句可以规避标题行被选中。

元素能对事件作出反应，首先必须将事件绑定(bind)到元素之上，然后触发(trigger)事件。$("TR").click(function(){})的功能是将click事件(其他事件也一样)绑定到选中的元素(绑定时元素已经存在)，此处为TR元素。一旦将事件绑定到元素，则一旦事件发生，则可执行绑定时定义在function(){}中的代码。

事件依赖于元素存在，并不依赖id或class。jQuery通过id、class或者元素名称找到元素，并将事件最终是绑定在元素上。例程3-17第12行到第14行绑定click事件到class名为testP的元素，此处为P元素。当单击按钮时，将执行第16、17行，即移走名为testP的class。在单击按钮前，单击P元素，将显示“你点我啦!”；当单击按钮并移走名为testP的class后，再单击P元素，仍然显示“你点我啦!”。由此可见，事件依赖元素而存在，而不是依赖id或class，虽然可以通过id或class找到元素并绑定事件。

例程3-17

```
第1行  <Html>
第2行      <Head>
第3行          <Title>第一个JavaScript程序</Title>
第4行          <Script type="Text/JavaScript" 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      <Body>
第8行          <P class="testP">第一段</P>
第9行          <P class="testP">第二段</P>
第10行         <button>移走名为testP的class</button>
第11行         <Script Language="JavaScript">
第12行             $(".testP").click(function(){
第13行                 alert("你点我啦!");
第14行             });
第15行             $("button").click(function(){
第16行                 $(".testP").removeClass("testP");
第17行                 alert("testP被移走啦!");
第18行             });
第19行         </Script>
第20行     </Body>
第21行 </Html>
```

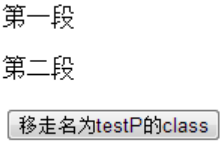

图3-7　例程3-17执行效果

第五节　属性操作

如图3-8所示，当单击下方的小图片时，自动显示相应大图片，在实际应用中，还能间隔一定时间自动切换图片(这部分功能将在后续章节实现)。为了实现单击小图显示大图，必须能改变大图IMG标记中的SRC属性，其属性值与小图片相同。即将小图片IMG标记的SRC属性内容取出，放到大图IMG标记的SRC属性中。例程3-18是其实现代码。

图3-8　图片切换

例程3-18

```
第1行   <HTML>
第2行       <HEAD>
第3行           <TITLE>单击鼠标切换图片</TITLE>
第4行           <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.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               .smallPic{width:51px}
第8行           </Style>
第9行       </HEAD>
第10行
第11行      <BODY>
第12行          <IMG src="1.jpg" ID="bigPic"><BR><BR>
第13行          <IMG src="1.jpg" class="smallPic">
第14行          <IMG src="2.jpg" class="smallPic">
第15行          <IMG src="3.jpg" class="smallPic">
第16行          <IMG src="4.jpg" class="smallPic">
第17行          <IMG src="5.jpg" class="smallPic">
第18行          <IMG src="6.jpg" class="smallPic">
第19行
第20行          <SCRIPT LANGUAGE="JavaScript">
第21行              $(".smallPic").click(function(){
第22行                  var clickPic=$(this).attr("src");
第23行                  $("#bigPic").attr("src",clickPic);
第24行              });
第25行          </SCRIPT>
第26行      </BODY>
第27行  </HTML>
```

在例程3-18中，大图片有唯一名为bigPic的ID，每个小的图片都有名为smallPic的Class。第21行的功能是当click名为smallPic的class时，执行第22行和第23行代码。

在第22行代码中，attr("src")的含义是取得指定对象src属性的值。在本行代码中，$(this)代表被click的对象。比如：当点击第二个小图片时，其src属性值为2.jpg。取得的内容放入到clickPic中。clickPic称为变量，其前的var的功能是表明该语句将申明一个新的变量。变量是一种引用量，在使用时引用其代表的内容。在申明变量时，可以暂不指定变量的内容。如可将例程3-18中第21行到第24行调

整如例程3-19所示。变量所代表的内容可以发生变化。比如：当单击第2个图片时代表的是“2.jpg”，当单击第4个图片时，代表的是"4.jpg"。注意：在使用变量时，两侧不能有双引号。如将$("#bigPic").attr("src",clickPic)修改为$("#bigPic").attr("src","clickPic")将发生错误。

例程3-19

```
第1行    $(".smallPic").click(fprogAttr01unction(){
第2行        var clickPic;
第3行        clickPic=$(this).attr("src");
第4行        $("#bigPic").attr("src",clickPic);
第5行    });
```

例程3-18中第23行的功能是选中ID名为bigPic的对象，并将其src属性变化为clickPic所代表的内容。和第22行的attr("src")相比，attr("src",clickPic)多了逗号和变量clickPic，其功能则变为设置指定对象的src属性，即将clickPic的内容传输给src属性。在程序设计中，一般将attr()称之为函数，括号中的内容称之为参数，当参数多余1个时，每个参数之间用逗号分开。当attr()函数仅有一个参数时是取得属性的内容，当有两个参数时，是改变属性的内容。为什么函数具有这样的功能呢？是开发者定义该函数时就已经约定。在JavaScript相关章节中，将学习相关内容。此处，可以暂时认为就应该如此，然后逐步解开谜团。

元素有固定的属性，如Img有src属性，也可以自定义属性，如例程3-20所示，第14行执行后给id为ctrlImg的元素增加自定义属性nowImgNo，并设定属性值为1。第17行则取得该自定义属性的属性值，并随后将其值增1。根据新的imgNo值，计算出图片文件的名称。第23行则用该图片文件的名称修改属性src的值，从而达到更换图片。这个程序存在问题，当图片编号大于9，则找不到对应的图片，留待后续章节解决。

例程3-20

```
第1行  <Html>
第2行      <Head>
第3行          <Title>第一个JavaScript程序</Title>
第4行          <Script type="Text/JavaScript" 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      <Body>
第8行          <img src="1.jpg" id="showImg"><BR><BR>
第9行          <button id="ctrlImg">下一张</button>
第10行         <Script Language="JavaScript">
第11行             //假定存在1.jpg、2.jpg、……、9.jpg
第12行
第13行             //id为ctrlImg对应的button增加自定义属性nowImgNo，其属性值为1
第14行             $("#ctrlImg").attr("nowImgNo",1);
第15行
第16行             $("button").click(function(){
第17行                 var imgNo=$("#ctrlImg").attr("nowImgNo");//取得属性nowImgNo的值
第18行                 imgNo=parseInt(imgNo)+1;
第19行
第20行                 $("#ctrlImg").attr("nowImgNo",imgNo);//改变nowImgNo属性的值
第21行
第22行                 var imgFilename=imgNo+".jpg";
第23行                 $("#showImg").attr("src",imgFilename);
第24行             });
第25行         </Script>
```

```
第26行          </Body>
第27行  </Html>
```

下一张

图3-9　例程3-20执行效果

第六节　文档操作

$.html()可以取得选中元素内的网页代码；$.html(网页内容)可以更改选中元素内的网页代码。当$.html()的括号中不包含任何内容时，取得选中元素的内容；如果括号中包含内容，即用该内容修改选中元素的内容。例程3-21是$.html()的应用。

例程3-21

```
第1行   <HTML>
第2行           <HEAD>
第3行                   <TITLE>$.html()的应用</TITLE>
第4行                   <Script type="Text/JavaScript" language="JavaScript" src="./jquery-1.4.2.min.js"></Script
    >
第5行                   <META http-equiv="Content-type" content="text/html; charset=gb2312">
第6行           </HEAD>
第7行           <BODY>
第8行                   <UL>
第9行                           <LI>李白</LI>
第10行                          <LI>杜甫</LI>
第11行                          <LI>白居易</LI>
第12行                          <LI>杜牧</LI>
第13行                  </UL>
第14行                  <Div ID="clickedPoet">当点击的诗人名称时，内容随之变化</Div>
第15行                  <SCRIPT LANGUAGE="JavaScript">
第16行                          $("LI").click(function(){
第17行                                  //将当前被点击元素的内容放入clickedContent中
第18行                                  var clickedContent=$(this).html();
第19行
第20行                          //显示变量clickedContent内容
第21行                          alert(clickedContent);
第22行
第23行                          //将选中诗人的名称显示在ID为clickedPoet的Div中
第24行                          $("#clickedPoet").html(clickedContent);
第25行                          });
第26行                  </SCRIPT>
```

```
第27行          </BODY>
第28行  </HTML>
```

在例程3-21中，第18行var clickedContent=$(this).html();中的$(this).html()将取得被click的LI元素中的内容，然后将该内容赋值给clickedContent。

在例程3-21中，第24行$("#clickedPoet").html(clickedContent);中的clickedContent将替换ID为clickedPoet中的内容，从而实现内容跟随点击而变化。

第七节　元素选择

在jQuery中，元素选择非常重要，几乎所有的操作都是“选中元素然后执行针对该元素的操作”。如果不能选中元素，则不能执行针对元素的操作。本部分内容以例程3-22为例，图3-10是其初始执行效果图。

例程3-22

```
第1行   <Html>
第2行           <Head>
第3行                   <Title>元素选择</Title>
第4行                   <Script type="Text/JavaScript" 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                           .highLight{font-size:16px;font-weight:bold;}
第8行                           #Wei,#Shu,#Wu{width:150px;display:inline-block}
第9行                   </Style>
第10行          </Head>
第11行          <Body>
第12行                  <Div id="Wei">
第13行                          <Div class="Leader">曹操</Div>
第14行                          <Div class="ideaMan">司马懿</Div>
第15行                          <UL class="powerMan">
第16行                                  <Li>曹仁</Li>
第17行                                  <Li>曹洪</Li>
第18行                                  <Li>夏侯惇</Li>
第19行                                  <Li>夏侯渊</Li>
第20行                          </UL>
第21行                  </Div>
第22行                  <Div id="Shu">
第23行                          <Div class="Leader">刘备</Div>
第24行                          <Div class="ideaMan">诸葛亮</Div>
第25行                          <UL class="powerMan">
第26行                                  <Li>关羽</Li>
第27行                                  <Li>张飞</Li>
第28行                                  <Li>赵云</Li>
第29行                                  <Li>姜维</Li>
第30行                          </UL>
第31行                  </Div>
第32行                  <Div id="Wu">
第33行                          <Div class="Leader">孙权</Div>
```

```
第34行                    <Div class="ideaMan">周瑜</Div>
第35行                    <UL class="powerMan">
第36行                        <Li>陆逊</Li>
第37行                        <Li>甘宁</Li>
第38行                        <Li>黄盖</Li>
第39行                        <Li>凌统</Li>
第40行                    </UL>
第41行                </Div>
第42行                <SCRIPT LANGUAGE="JavaScript">
第43行                    $("Li").addClass("highLight");//为所有Li元素增加名为highLight的class。
第44行                </SCRIPT>
第45行            </Body>
第46行    </Html>
```

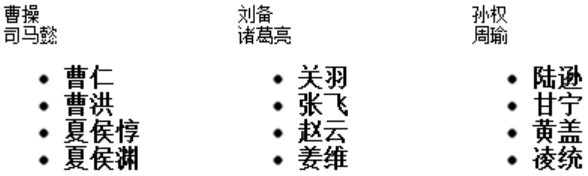

图3-10　元素选择示例

jQuery提供了丰富的元素选择方式，可以以元素方式选择(如例程3-22第43行)，也可以用id(如例程3-23第5行所示)、class(如例程3-23第2行所示)方式选择。注意id所代表的名称前必须加上井号(#)，class所代表的名称前必须加上句点(.)。

例程3-23

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $(".Leader").addClass("highLight");//曹操、刘备、孙权被加上highLight
第3行
第4行        //id为Wu元素被加上highLight，显示效果是Wu即吴被显示为highLight效果
第5行        $("#Wu").addClass("highLight");
第6行    </SCRIPT>
```

除了上述基本的选择方式外，还有其他选择元素的方式，前面已提到的$("TR:odd")含义是选中tr元素中编号为奇数的tr元素。注意，被选中的元素内部有编号，编号从0开始，以该元素在网页中出现的顺序为序。odd为定义，相当于选择满足编号为奇数的tr元素。与odd对应的是even，即偶数编号的元素。其他类似定语如例程3-23所示。

例程3-24

```
第1行    <Script Language="JavaScript">
第2行        //eq(N)表示选中编号为N的元素
第3行        $(".ideaMan:eq(1)").addClass("highLight");//诸葛亮被选中并添加名为highLight的class。
第4行
第5行        //gt(N)表示选中编号大于N的元素
第6行        $("Li:gt(9)").addClass("highLight");//黄盖、凌统被选中。所有li被选中其中编号大于9的部分
第7行        //lt(N)表示选中编号小于N的元素
第8行        $("Li:lt(3)").addClass("highLight");//曹仁、曹洪、夏侯惇被选中。所有li被选中其中编号小于3的部分
第9行
第10行       //first表示选中第一个元素
第11行       $(".Leader:first").addClass("highLight");//曹操被选中，class名为Leader的所有元素中第一个。
第12行       //last表示选中最后一个元素
```

```
第13行          $(".Leader:last").addClass("highLight");//孙权被选中，class名为Leader的所有元素中最后。
第14行      </Script>
```

根据元素的树形关系也可以查找元素。在例程3-25第2行，即选中Wei下面的li元素，这种方式能选中Wei下面的后辈元素(不一定是子元素)；而第4行则是选中Wu下面子元素li，而Wu下面没有li子元素，所以一个元素都没有被选中。在选中元素时，空格表示选中后辈元素，大于符号(>)则是选中子元素。第5行是选中子元素的一个应用。

例程3-25

```
第1行   <Script Language="JavaScript">
第2行           $("#Wei Li").addClass("highLight");//选中id为Wei的元素下的Li元素，选中曹操下面的所有Li元素
第3行
第4行           $("#Wu>Li").addClass("highLight");//没有选中任何元素
第5行           $(".powerMan:eq(2)>Li").addClass("highLigth");//选中powerMan中编号为2的元素下的li，即Wu下面的li元素
第6行   </Script>
```

在选中元素的基础上，可以筛选出其中一部分作为操作元素，如例程3-26所示。第3行的eq(2)是找到编号为2的元素，在该例中是找到所有Div中编号为2的元素。注意eq(N)可以用在$()的参数中，其含义是在初始查找时即找到编号为N的元素，每次找到一个(如果编号超过范围，则一个也找不到)；而$().eq(N)则是找到的元素中查找编号为N的元素。

例程3-26

```
第1行   <Script Language="JavaScript">
第2行           //eq(N)找到编号为N的元素
第3行           $("Div").eq(2).addClass("highLight");//所有div中编号为2的元素，此处为司马懿
第4行           //filter()用于过滤出满足条件的元素，括号中参数与$()参数相同
第5行           $("Div").filter(".ideaMan").addClass("highLight");//从所有Div中过滤出其class名为ideaMan的元素
第6行
第7行           $("#Wu").children("Div").addClass("highLight");//查找id为Wu元素下名为Div的元素。孙权、周瑜被选中
第8行
第9行           //li:eq(5)是张飞，张飞所在li的父元素是ul，其父元素是id为Shu的Div
第10行          $("Li:eq(5)").parent().parent().addClass("highLight");
第11行          //find()查找满足参数的元素，其参数与$()的参数取值相同
第12行          $("#Wu").find("Li").addClass("highLight");//查找id为Wu的元素，并找到后辈元素Li
第13行          $("#Wu").find(".Leader").addClass("highLight");//查找id为Wu的元素，并找到后辈中class名为Leader的元素
第14行  </Script>
```

filter()用于在找到元素的范围内过滤元素。在例程3-26第5行是在找到的所有Div元素中找到class名为Leader的元素。注意filter()的参数中可以是id，class或者元素等。

children()、parent()、find()则是以查找到的元素为基础，查找子元素、父元素以及后辈元素。children()的参数与$()的参数类似，可以是id，class或者元素等；parent()也可以接受参数，不过由于父元素常常只有一个，因此，通常省略参数即查找指定元素的父元素；find()则可以查找后辈元素，注意不仅仅是子元素，还包括子子孙孙等元素。children()是查找子元素，而find()是查找后辈元素，也包括查找子元素。

有了多种形式的元素查找方法，就可以针对选中元素执行各种操作。

第八节　常用操作

1、元素同辈编号：$.index()

$.index()有多种用法，当index()函数没有参数时，其功能是获取指定元素在同辈元素中的编号(同辈元素从0开始编号)，如例程3-

27所示(替换例程3-22的JavaScript代码)。当被选择元素是多个时，则返回第一个匹配元素在同辈元素中的位置，如第3行所示，匹配元素实际上是3个，但只查找第一个匹配元素即“司马懿”所在的div，其同辈元素包括：Div(曹操)、Div(司马懿)和UL元素(包括曹仁等4人)，“司马懿”所在Div在其同辈中编号为1。

例程3-27

```
第1行   <Script Language="JavaScript">
第2行           var indexNo1=$("#Shu").index();//1。id为Shu的Div元素在同辈元素中的位置，
第3行           var indexNo2=$(".ideaMan").index();//1。class为ideaMan的第一个匹配元素在同辈中的位置。
第4行           var indexNo3=$("Li:eq(10)").index();//2。编号为10的Li在同辈位置，即黄盖在所在组的编号。
第5行           $("Li").click(function(){
第6行                   var clickNo=$(this).index();//被单击元素在同辈中位置
第7行                   alert(clickNo);
第8行           });
第9行   </Script>
```

例程3-27第6行等号左边的$(this)代表当事件发生时所单击的li在同辈元素中的编号并把编号值赋值给clickNo。即单击任何一个Li时，都将显示出该Li在同辈元素中的编号。

2、元素隐藏显示：$.hide()与$.show()

$.hide()能将被选中元素隐藏，而$.show()则能将选中元素予以显示，如例程3-28所示。当网页被打开时，首先执行$("#imgBox img").eq(0).show();，此时id名为imgBox下的img元素中第一个img元素被显示(在本程序中所有img元素因为第8行代码的缘故被默认为隐藏)，然后给id名为ctrlImg下的span元素绑定click事件。当click事件发生时，首先取得被单击span在同辈元素中的编号，然后将本程序中所有img元素隐藏，并将对应编号的img元素显示。

图3-11　$.show()与$.hide()的应用

例程3-28

```
第1行   <Html>
第2行           <Head>
第3行                   <Title>元素选择</Title>
第4行                   <Script type="Text/JavaScript" 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:380px;background-color:#E7E7E7;padding:10px 10px 5px 10px}
第8行                           #imgBox img{display:none}
第9行                           #ctrlImg{text-align:Center;margin-top:5px}
第10行                  </Style>
第11行          </Head>
第12行          <Body>
```

```
第13行                    <Div id="imgBox">
第14行                        <Img src="1.jpg"><Img src="2.jpg"><Img src="3.jpg"><Img src="4.jpg"><Img src="5.j
pg">
第15行                        <Div id="ctrlImg">
第16行                            <span>■</span><span>■</span><span>■</span><span>■</span><span>■</spa
n>
第17行                        </Div>
第18行                    </Div>
第19行                    <SCRIPT LANGUAGE="JavaScript">
第20行                        $("#imgBox img").eq(0).show();
第21行                        $("#ctrlImg span").click(function(){
第22行                            var clickNo=$(this).index();//clickNo保存被点击的编号
第23行                            $("#imgBox Img").hide();
第24行                            $("#imgBox Img").eq(clickNo).show();
第25行                        });
第26行                    </SCRIPT>
第27行            </Body>
第28行    </Html>
```

3、前后内部插入：$.prepend()与$.append()

从例程3-29(效果如图3-12所示)可以看出HTML代码部分应该仅仅显示“元迎探惜”四个字的列表。由于执行了第15行代码，则在每个条目前增加一个“贾”；执行第16行后，则增加了一个“春”字，并且这个“春”字带有黑体格式。$.prepend()和$.append()都是在选中元素内部插入内容(都可以包含HTML格式信息)，$.prepend()在原有内容最前面增加信息，而$.append()则是在原有内容最后面增加信息。

例程3-29

```
第1行    <Html>
第2行            <Head>
第3行                    <Title>内部插入</Title>
第4行                    <Script type="Text/JavaScript" 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            <Body>
第8行                    <OL>
第9行                        <LI>元</LI>
第10行                       <LI>迎</LI>
第11行                       <LI>探</LI>
第12行                       <LI>惜</LI>
第13行                   <OL>
第14行                   <Script Language="JavaScript">
第15行                       $("Li").prepend("贾");
第16行                       $("Li").append("<span style='font-weight:bold'>春</span>");
第17行                   </Script>
第18行           </Body>
第19行   </Html>
```

1. 贾元春
2. 贾迎春
3. 贾探春
4. 贾惜春

图3-12　元素内部插入

$.html(内容)也可以改变选中元素的内容，但是$.html(内容)是用html()参数中的内容覆盖选中元素原有内容。$.prepend()和$.append()则会保留原内容并在其前或后增加新的内容。

4、前后外部插入：$.after()与$.before()

与$.prepend()和$.append()不同，$.after()和$.before()是在元素外部插入，$.after()是选中元素之后插入，$.before()则是在选中元素之前插入。例程3-30第12行执行后，“贾宝玉”条目之前将有“林黛玉”条目，当执行第13行时，因$("li")将选中“贾宝玉”和新增的“林黛玉”两个li元素，所以在每个元素之后插入“薛宝钗”这个li元素。

例程3-30

```
第1行   <Html>
第2行       <Head>
第3行           <Title>外部插入</Title>
第4行           <Script type="Text/JavaScript" 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       <Body>
第8行           <UL>
第9行               <Li>贾宝玉</Li>
第10行          </UL>
第11行          <Script Language="JavaScript">
第12行              $("Li").before("<Li>林黛玉</Li>");
第13行              $("Li").after("<Li><span style='font-weight:bold'>薛宝钗</span></Li>");
第14行          </Script>
第15行      </Body>
第16行  </Html>
```

- 林黛玉
- **薛宝钗**
- 贾宝玉
- **薛宝钗**

图3-13　外部插入

5、输入元素读写：$.val()

不同输入元素，其读写元素值的方法也不同，这里仅讲述读写文本框的值。$.val()的使用方法与$.html()相似。当$.val()没有参数时，读取选中元素的值；当$.val(内容)有参数时，则是向选中元素写入值(输入值)。在例程3-31中，其HTML代码的input元素中没有任何(文本框为空)，当执行第11行后，其文本框将有“请输入值”，实现代码是$("input").val("请输入值");。当单击按钮时，将执行第13行代码，其$("input").val()将取得input元素的值(根据文本框中值而变化)并将文本内容复制给变量inputVal。第14行代码是显示变量inputVal的值。注意：从文本框中取得的值为字符串形式的数据。

请输入值 OK

图3-14　文本框的值

例程3-31

```
第1行   <Html>
第2行       <Head>
第3行           <Title>文本框的值</Title>
```

```
第4行                    <Script type="Text/JavaScript" 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            <Body>
第8行                    <input type=text>
第9行                    <button>OK</button>
第10行                   <Script Language="JavaScript">
第11行                           $("input").val("请输入值");
第12行                           $("button").click(function(){
第13行                                   var inputVal=$("input").val();
第14行                                   alert(inputVal);
第15行                           });
第16行                   </Script>
第17行           </Body>
第18行   </Html>
```

第九节　更多了解

1、字符串：由静至动

如例程3-32和图3-15所示的多行多列的表格中，已选中第2行3列单元格。如果要求输入行的编号、列的编号并将该单元格类似图3-15方式标记，该如何实现呢？此时行号、列号成为变量，而不是一个固定值，相当于第21行代码中1和2两个数字用变量代替。假定如例程3-33所示的写法，会有什么效果呢？

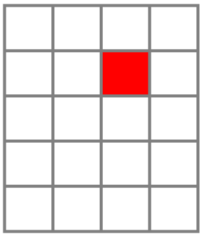

图3-15　例程3-32效果图

例程3-32

```
第1行    <Html>
第2行            <Head>
第3行                    <Title>字符串：由静至动</Title>
第4行                    <Script type="Text/JavaScript" 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                            table{border:1px solid gray}
第8行                            td{width:30px;height:30px;border:1px solid gray}
第9行                            .bgRed{background-color:red}
第10行                   </Style>
第11行           </Head>
第12行           <Body>
第13行                   <Table cellpadding="0" cellspacing="0">
第14行                           <Tr><td> </td><td> </td><td> </td><td> </td></Tr>
```

```
第15行                    <Tr><td> </td><td> </td><td> </td><td> </td></Tr>
第16行                    <Tr><td> </td><td> </td><td> </td><td> </td></Tr>
第17行                    <Tr><td> </td><td> </td><td> </td><td> </td></Tr>
第18行                    <Tr><td> </td><td> </td><td> </td><td> </td></Tr>
第19行                </Table>
第20行                <Script Language="JavaScript">
第21行                    $("tr:eq(1)>td:eq(2)").addClass("bgRed");
第22行                </Script>
第23行            </Body>
第24行        </Html>
```

例程3-33

```
第1行            <Script Language="JavaScript">
第2行                var rowNo=prompt("请输入行号",1);
第3行                var colNo=prompt("请输入列号",2);
第4行                $("tr:eq(rowNo)>td:eq(colNo)").addClass("bgRed");
第5行            </Script>
```

任何单元格都不会被选中！放在引号中内容是直接量，是常量，是不会变化的量，虽然其中有rowNo和colNo，只不过是和变量字母相同的几个字母而已，不会发生变化。引用量或者变量，不能存在于直接量中。修改如例程3-34即可实现。

例程3-34

```
第1行            <Script Language="JavaScript">
第2行                var rowNo=prompt("请输入行号",1);
第3行                var colNo=prompt("请输入列号",2);
第4行                $("tr:eq("+rowNo+")>td:eq("+colNo+")").addClass("bgRed");
第5行            </Script>
```

粗看例程3-34第4行代码，会发现rowNo左右两侧有加号同时还有双引号，这是不是就是直接量或常量不会变化呢？注意，引号是从左到右配对。即第一个双引号出现后，会自动查找到下一个配对引号为止。在第4行代码中，"tr:eq("是第一对配对双引号。")>td:eq("是第二对配对双引号，")"是第三对配对双引号。

$("tr:eq(1)>td:eq(2)").addClass("bgRed");这行代码不复杂，还算易于理解，但写成例程3-34第4行容易犯晕，并且初学者难以正确书写。怎么办呢？可以从简单入手。$("tr:eq(1)>td:eq(2)").addClass("bgRed");实现了选中行号为1，列号为2的单元格，只要让行号和列号变化即可达到目的。如例程3-35所示，逐步改变即可。

例程3-35

```
第1行        <Script Language="JavaScript">
第2行            $("tr:eq(1)>td:eq(2)").addClass("bgRed");
第3行            //在上行1和2两侧加上两对双引号变化为
第4行            //$("tr:eq(""1"")>td:eq(""2"")").addClass("bgRed");
第5行            //在两个紧邻的双引号之间插入加号，加号的功能是拼接字符串
第6行            //变化后如下行代码所示，完全等价于$("tr:eq(1)>td:eq(2)").addClass("bgRed");
第7行            $("tr:eq("+"1"+")>td:eq("+"2"+")").addClass("bgRed");
第8行
第9行            //将"1"和"2"用变量代替
第10行           var rowNo="1";
第11行           var colNo="2";
第12行           //变化后
第13行           $("tr:eq("+rowNo+")>td:eq("+colNo+")").addClass("bgRed");
```

```
第14行
第15行            //上述代码测试通过后，rowNo和colNo的值替换为来源于prompt()即可。
第16行        </Script>
```

2、变量名：名副其实

不管是CSS(class或id命名)还是JavaScript语言都涉及到命名(变量命名等)，在计算机其他编程语言中也同样会涉及到命名。如同父母为子女命名一样，命名要遵守一些规则。

命名规则：

1. 命名首字母必须为字母(a-zA-Z)，下划线(_)，或者美元符号($)开始；其后也只能是字母(a-zA-Z)、数字(0-9)和下划线(_)的组合，并且其间不能包含空格；
2. 不能使用保留字，比如：JavaScript中不能让变量名为var，因为var已经有固定含义，用于申明变量，每种语言保留字不尽相同；
3. 命名需要有意义，能通过命名大致了解该名称的用途；
4. 命名不仅让编程者自己明白，还能让可能阅读程序的其他人轻松明白；
5. 命名不能太长比如50个字符，或者太短仅仅一个字符。例如在JavaScript中有多个变量且都是一个字母，会让其他阅读人痛苦不堪；
6. 采用统一命名规范，比如都用汉语拼音，或者英文命名，最好不要拼音或者英文混用；
7. 自己短时间明白，不意味长时间记忆深刻；也许会重用代码，不遵守规范，也会让自己承受折磨；

在遵守上述规则的前提下，还有一些经典的命名规范供参考。

- 帕斯卡命名法：每个名字由有含义的单词组成，并且首字母大写，如UserName、GoodsPrice等；
- 骆驼命名法：与帕斯卡命名法相似，但第一个单词首字母须小写，如userName、goodsPrice等。
- 匈牙利命名法：与帕斯卡或骆驼命名法不同的是，匈牙利命名法还要求用首字母表示数据类型，如sUserName其中s表示该变量用于保存字符型数据(String的首字母简写)、iGoodsPrice中的首字母表示整数(Integer的首字母简写)。

在JavaScript中，建议采用骆驼命名法。