返回目录

第六章　事件(Event)

本章导读

在网页，事件(Event)是由浏览器(Browser)、窗口(Window)、文档(Document)或元素(Element or HTML Tag)触发的动作(action)，该动作可被捕获(Capture)并编写相应的处理代码。事件种类较多，如：键盘事件(keypress、keydown、keyup等)、鼠标事件(mouseover、mouseout、mousedown、mouseup等)。

前述章节初涉事件(如：click、mouseover、mouseout等)，在本章还将讲解到事件绑定(bind)、事件触发(trigger)、事件定义、事件参数、事件模型等重要内容。同时还将讲述传统JavaScript的事件处理与jQuery事件处理的区别和联系。

事件是Web页面设计的重要内容，是实现人机交互的重要手段和途径。

学习目标：

1. 事件模型；
2. 了解传统JavaScript事件处理与jQuery区别与联系；
3. 掌握事件绑定及其相关操作；
4. 掌握各种事件类型；
5. 掌握事件参数；

本章目录

第一节　事件原理

Windows、Mac OS、iOS以及Android等操作系统上的应用程序都是基于事件驱动，从某种意义上讲，基于GUI(Graphical User Interface的简写，即图形用户接口)的现代应用程序几乎都是事件驱动(Event Driven)，Web应用程序也不例外。所有事件驱动的应用程序几乎都采用相同的工作模式：建立事件处理机制、等待相关事件发生(比如：鼠标单击click)、对事件作出反应(Response)。除了常见的鼠标事件、键盘事件外，还有其他事件，如：页面加载事件、浏览器事件等。近年来在手机、平板电脑上流行的滑动手势(swipe gesture)操作等也属于事件范畴。另外，事件驱动是面向对象程序设计(Oriented Object，简称OO)的核心。

事件是Web应用程序的基础，如果不使用事件，JavaScript的使用将受到严重制约。在jQuery中，把事件分为：鼠标事件、键盘事件、文档事件、浏览器事件等。同时还有事件绑定(Band)和解除(Unband)、事件触发(Trigger)以及事件对象(Event Object)等。

以例程6-1为例说明事件相关概念，类似这段例程的代码在前面相关章节已多次使用。

例程6-1

```
第1行    <HTML>
```

```
第2行        <HEAD>
第3行            <TITLE>事件模型</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
第8行        <BODY>
第9行            <Div>
第10行               <P>click Me(点点我看看！)</P>
第11行           </Div>
第12行           <SCRIPT LANGUAGE="JavaScript">
第13行               $("P").click(function(){
第14行                   alert("我在P!");
第15行               });
第16行               $("Div").click(function(){
第17行                   alert("我在Div!");
第18行               });
第19行               $("Body").click(function(){
第20行                   alert("我在Body!");
第21行               });
第22行           </SCRIPT>
第23行       </BODY>
第24行   </HTML>
```

在第13-15行中，$("P").click()的含义是选中P元素，并“绑定(bind)”事件click。注意，这里仅仅是绑定，即告诉浏览器系统，元素P将对click事件作出反应，但该反应只在事件发生才作出，此处仅是绑定，不作出反应。

$("P").click()内的function()及其后花括号内的“alert("我在P!");”是click事件发生时要执行的代码，即事件发生时要作何反应。在此处是弹出对话框，显示“我在P!”。需求不同，花括号内的代码也不尽相同。

当在P元素上单击时，将触发click事件并调用其内的代码。例程6-1第13-15行、第16-18行、第19-21行其功能相似。

可以这样理解，首先是在元素上绑定事件及其处理代码，当事件发生时(action)，调用编写的处理代码即对事件做出响应(response)，Web页面的“绑定-动作-响应”(bind-action-response)就是典型的事件驱动模式(Event Driven)。

仅仅在P元素上单击一次，将首先显示“我在P!”，不过其后将依次显示“我在Div!”、“我在Body!”，如图6-1所示。为什么一次事件，3个元素都做出了反应呢？这就是事件传播模型。当事件在一个元素上发生时，如果该元素绑定了该事件及其处理代码，则执行相关处理后向其父元素传播，否则直接将该事件向其父元素传播，如果父元素绑定该事件，则处理后向更上级传递，否则直接向更上级元素传递。就像一个气泡，逐步向上运动。在事件驱动模型中，这种模式被称之为“自底向上冒泡模型”，也是绝大部分浏览器默认支持的模式。例程6-1中，P元素接受click事件并处理后，该事件被继续传递到Div元素并执行相关代码，然后继续传递Body元素并执行相关代码。在例程中，Div是P元素的父元素、Body是Div的父元素。

图6-1　依①→②→③次序触发click事件

大部分浏览器都默认支持自底向上冒泡模型传播事件，如图6-1、图6-2所示。在浏览器中，Body元素的父元素是Html、然后是document和window对象。document和window对象在网页中并不表现为网页代码，而是在幕后起作用，将在其他章节予以讲解。

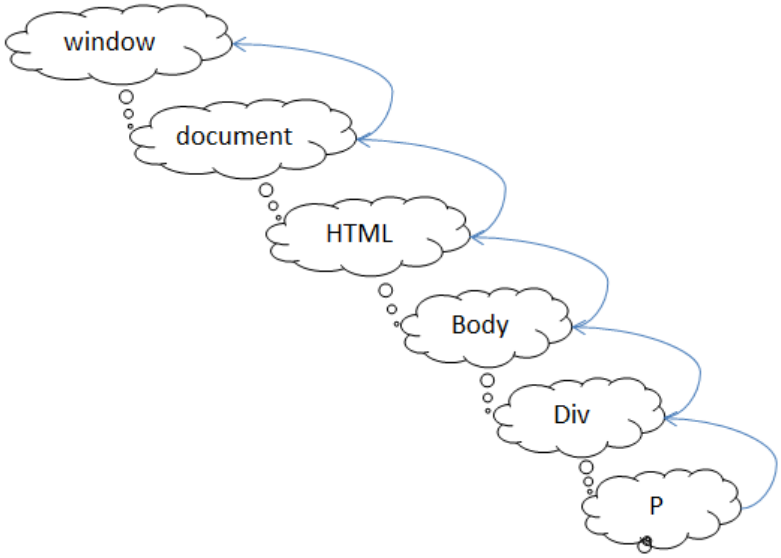

图6-2　浏览器事件冒泡传播模型

浏览器采用冒泡模型具有天然合理性，如图6-3所示，假定内层方框相当于一个元素，其外层方框相当于其父元素，越往外，辈分越高。当内层元素发生事件时，也相当于在外层元素管辖范围内发生事件。现实生活中，下级分管范围内发生的事情，需要向上汇报，也与之类似。浏览器内发生的事件，绝大部分都有冒泡传播的特征，只有极少事件不具有该特征，当讲述该事件，将特别指出。

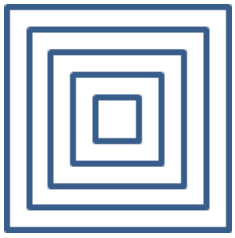

图6-3　浏览器事件冒泡传播模型

虽然事件具有冒泡特征，但只有该元素及其相关长辈绑定事件及其处理代码后才会对事件作出反应，如果没有绑定事件及其处理代码，事件也仅仅是路过。

事件依赖元素而存在。事件一旦绑定到元素，元素未删除且事件没有被删除或移走(解除绑定)，则事件一直存在。元素属性，包括但不限于id、class、style等调整，不影响元素的事件，如例程6-2所示。

例程6-2

```
第1行   <Html>
第2行       <Head>
第3行           <Title>事件测试</Title>
第4行           <Style type="text/css">
第5行               *{font-size:14px;}
第6行               .normalIt{font-size:1em;}
第7行               .lightIt{font-size:2em;font-weight:Bold;color:red}
第8行           </Style>
第9行       </Head>
第10行      <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第11行      <Body>
第12行          <P class="normalIt">元春</P><P class="normalIt">迎春</P>
第13行          <P class="normalIt">探春</P><P class="normalIt">惜春</P>
```

```
第14行                <Script LANGUAGE="JavaScript">
第15行                    $(".normalIt").mouseover(function(){
第16行                        $(this).addClass("lightIt");
第17行                    });
第18行                    $(".normalIt").mouseout(function(){
第19行                        $(this).removeClass("lightIt");
第20行                    });
第21行                    $(".normalIt").removeClass("normalIt");//移走P元素上的class。
第22行                </Script>
第23行            </Body>
第24行        </Html>
```

在例程6-2第15到第20行，分别绑定mouseover()和mouseout()事件到Class名为normalIt的P元素，而第21行则是删除名为normalIt的Class，即P元素没有了Class属性。当打开网页时，首先是根据Class名称绑定事件到P元素，然后删除该Class属性，但是移动鼠标，会发现P元素仍然能执行addClass()和removeClass()。因为事件的存在依赖于元素，当事件成功绑定到元素后，只要元素存在，该事件没有被删除，则该事件一直起作用。属性等是帮助jQuery找到元素，匹配元素绑定事件后，删除定位参数，不影响元素事件的绑定。

如果将第21行代码移动到第15行前，第14行后会发生什么呢？mouseover()和mouseout()事件还会起作用吗？实际上，当执行removeClass("normalIt")后(删除名为normalIt对应的Class)，原第15、第18行将找不到元素，当然也不能绑定事件，结果自然也就没有预期效果。

在$("P").click(function(){})的function()可以改写为function(e)，其中e为形式参数，可以改写为其他变量名。e是事件对象，一旦事件发生，该对象相关属性就获得相应属性值，且可执行该对象相关方法，改变事件默认行为。如例程6-3所示。

例程6-3

```
第1行    <Html>
第2行        <Head>
第3行            <Title>事件测试</Title>
第4行        </Head>
第5行        <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第6行        <Body>
第7行            <Ul>
第8行                <Li>贾</Li>
第9行                <Li>史</Li>
第10行               <Li>王</Li>
第11行               <Li>薛</Li>
第12行           </Ul>
第13行           <Script LANGUAGE="JavaScript">
第14行               $("Body,Ul,Li").click(function(e){
第15行                   var nowElement=e.currentTarget.tagName;
第16行                   alert(nowElement);//依次显示LI、UL、BODY
第17行               })
第18行           </Script>
第19行       </Body>
第20行   </Html>
```

在第15行，e.currentTarget的含义是事件对象e的当前气泡元素，或者说是正在接受事件的元素。tagName是元素的名称。因此，该语句的含义是获得正在接受事件的元素名称。由于事件具有气泡特征，当在Li上触发元素时，将能依次显示LI、UL、BODY。

事件对象还有诸多属性，如event.which能获得键盘按键、鼠标按键(左中右)，如例程6-4所示。更多“事件对象”相关信息请参看本章其他小节。

例程6-4

```
第1行  <Html>
第2行      <Head>
第3行          <Title>事件测试</Title>
第4行      </Head>
第5行      <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第6行      <Body>
第7行          学号<input type=text>(仅限于数字)
第8行          <Script LANGUAGE="JavaScript">
第9行              $("input").keyup(function(e){
第10行                 //0的ASCII的值为48，1是49，……57
第11行                 //如果输入的内容不在该范围，就表示输入错误。
第12行                 var inputKeyCode=e.which;//此处，e.which为输入键的ASCII值
第13行                 if(inputKeyCode<48 || inputKeyCode>57){//小于48或者大于57则输入错误
第14行                     alert("输入错误!");
第15行                     return false;//当返回false时，本次事件后续行为被取消
第16行                 }
第17行             });
第18行         </Script>
第19行     </Body>
第20行 </Html>
```

例程6-4第12行的e.which用于获取事件发生时输入的值，此处是键盘事件，相当于获得键盘输入的值(ASCII值)。由于学号仅限于数字，而0-9的ASCII的值介于48-57之间，范围内合法，范围外违规，以此判断。另外，其return false(第15行)相当于终止函数执行，后续行为被取消，比如：将输入值显示在文本框，向其上级元素冒泡事件等。

第二节　事件绑定

将Html元素等与事件及其处理代码联系起来的方法称之为绑定(bind)事件，将事件及其代码与Html元素脱钩的方法称之为注销事件(unbind)。例程6-5提供了JavaScript传统的直接静态绑定法，jQuery提供了多种形式更加友好的事件绑定方法。

例程6-5

```
第1行  <HTML>
第2行      <HEAD>
第3行          <TITLE>事件模型</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
第8行      <BODY onclick="alert('我在Body!');">
第9行          <Div onclick="alert('我在Div!');">
第10行             <P onclick="alert('我在P!');">click Me(点点我看看！)</P>
第11行         </Div>
第12行     </BODY>
第13行 </HTML>
```

在例程6-5的P、Div、Body中都有onclick及其代码，当click时，将弹出对应对话框，其效果与例程6-1相同。采用这种方式，将可执行的JavaScript代码与网页代码融合在一起，不利于代码管理和维护，从某种意义上讲，是一种侵扰式编程。在实际工作中，页面显示效果通常由平面或Web设计师完成，而JavaScript代码通常由Web工程师完成，将两者代码融合在一起，不利于双方工作的开展。将JavaScript代码与网页效果代码分离，是良好的工程习惯。jQuery提供了不错的解决方案，如例程6-1所示。

jQuery绑定事件的方法较多，可以根据需要选择不同的绑定方式。

1、简捷绑定法

前述章节应用过的事件绑定方法，基本都属于事件简捷绑定方法，其格式为$.事件名称(fn)，其中$的功能是选择元素，事件名称如click等，fn代表该事件处理代码，总体功能是为匹配元素绑定事件其及其处理代码，如例程6-6所示。

例程6-6

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $("Body").click(function(){
第3行            alert("页面被点击!");
第4行        });
第5行    </SCRIPT>
```

要实现例程6-5的效果，可采用例程6-7所示的更加简短代码予以实现。

例程6-7

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $("body,div,p").click(function(){
第3行            var clickedElementName=this.tagName;
第4行            alert("我在"+clickedElementName+"元素");
第5行        });
第6行    </SCRIPT>
```

简捷绑定法仅能对匹配成功的元素绑定事件，如例程6-8，其第2行移走了名为normalIt的class，可是第3行希望匹配名为normalIt的元素，并绑定click事件。由于没能成功匹配元素，当然也就不能绑定事件。

例程6-8

```
第1行    <Script LANGUAGE="JavaScript">
第2行        $(".normalIt").removeClass("normalIt");//移走名为normalIt的class。
第3行        $(".normalIt").click(function(){
第4行            $(this).addClass("lightIt");
第5行        });
第6行    </Script>
```

在jQuery中，fn即事件处理函数可以含有参数Event对象，其格式为fn(e)，其中e为Event对象，包含有事件的各种信息，如页面位置数据等，如例程6-9所示。有关Event的更多信息参见本章第3节。

例程6-9

```
第1行    <SCRIPT LANGUAGE="JavaScript">
第2行        $("Body").click(function(e){
第3行            alert(e.pageX);//pageX表示鼠标点击位置离页面左边界的距离
第4行        });
第5行    </SCRIPT>
```

2、通用绑定法

$.bind是jQuery提供的通用事件绑定方法，其格式如下：

$.bind绑定事件

单事件绑定：$.bind(type[,data],fn)

多事件绑定：$.bind({type:f1,type:f2,……,type:fn})

type：必须参数，事件类型(如click等)；data：可选参数，向事件传递数据；fn：事件处理函数

例程6-10是单事件绑定，Body被绑定click事件及其处理程序，省略data参数，其功能是当用鼠标单击页面区域时，用alert弹出对话框。$.bind可以绑定各种事件，也包括自定义事件。有关自定义事件的使用将本章“自定义事件”讲述。

例程6-10

```
<SCRIPT LANGUAGE="JavaScript">
    $("Body").bind("click",function(){
        alert("页面被点击!");
    });
</SCRIPT>
```

$.bind在绑定单事件时，还可以传递相关数据，其数据可通过Event.data访问，如例程6-11所示，其new Date().toLocaleString()的含义是点击鼠标时将时间传递给Event.data，并通过参数e传递给处理函数，并通过e.data访问传递的数据。

例程6-11

```
<SCRIPT LANGUAGE="JavaScript">
    $("Body").bind("click",new Date().toLocaleString(),function(e){
        alert(e.data);//e.data表示数据
    });
</SCRIPT>
```

向事件传递的数据还可以是各种复合数据，如数组、自定义对象等，如例程6-12所示。

例程6-12

```
<Html>
    <Head><Title>事件测试</Title></Head>
    <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
    <Body>
        <Ul><li>李白</li><li>杜甫</li></Ul>
        <Script Language="JavaScript">
            $("li:eq(0)").bind("click",[12,45,43],function(e){
                alert(e.data[1]);//显示45
            })
            $("li:eq(1)").bind("click",{left:100,top:20},function(e){
                alert(e.data.left);//显示100
            })
        </Script>
    </Body>
</Html>
```

$.bind可以绑定多个事件及其对应的处理函数，如例程6-13所示。当鼠标离开或进入Body时，提示不同的信息。注意：花括号的应用以及事件之间的逗号。

例程6-13

```
<SCRIPT LANGUAGE="JavaScript">
    //注意花括号的配对
    $("P").bind({
        mouseover:function(){
            alert("鼠标进入");
```

```
第6行                    },
第7行                    mouseout:function(){
第8行                            alert("鼠标离开!");
第9行                    }
第10行           });
第11行     </SCRIPT>
```

如果是多个事件对应一个处理程序，可以采用例程6-14所示的方法，其第2行代码绑定两个事件分别是mouseover和mouseout，当两个事件被触发时，将执行相同的处理代码，其效果与例程6-15和例程6-16相当。很明显，例程6-14会更加简洁。

例程6-14

```
第1行     <Script LANGUAGE="JavaScript">
第2行           $("P").bind("mouseover mouseout",function(){
第3行                   $(this).toggleClass("lightIt");
第4行           });
第5行     </Script>
```

例程6-15

```
第1行     <Script LANGUAGE="JavaScript">
第2行           $("P").bind("mouseover",function(){
第3行                   $(this).toggleClass("lightIt");
第4行           });
第5行           $("P").bind("mouseout",function(){
第6行                   $(this).toggleClass("lightIt");
第7行           });
第8行     </Script>
```

例程6-16

```
第1行     <Script Language="JavaScript">
第2行           $("P").bind({
第3行                   mouseover:function(){
第4行                           $(this).toggleClass("lightIt");
第5行                   },
第6行                   mouseout:function(){
第7行                           $(this).toggleClass("lightIt");
第8行                   }
第9行           });
第10行    </Script>
```

3、一次绑定法

和其他绑定事件的方法不同，用一次绑定法绑定的事件仅能被执行一次，然后会自动解除在触发事件元素上的绑定，而其他绑定方法则可以多次被执行，直到删除元素或者被解除绑定为止。一次绑定法的使用格式如下：

$.one绑定事件

用法1：$.one(type[,data],fn)

用法2：$.one(type[,selector][,data],fn)

用法3：$.one({type:f1,type:f2,type:fn}[,selector][,data])

type：必须参数，事件类型(如click等)；data：可选参数，向事件传递数据；fn：事件处理函数

例程6-17在Body元素上绑定click事件，当单击鼠标后再次单击鼠标，将不能弹出对话框，即第一次单击将弹出对话框，以后则对该事件没有反应。

例程6-17

```
<SCRIPT LANGUAGE="JavaScript">
    $("Body").one("click",function(){
        alert("单击页面多次，看看弹出对话框几次？");
    });
</SCRIPT>
```

例程6-18第7-10行是将click一次性绑定到第一个li元素上，在绑定的同时传递自定义对象数据到event对象。第11-14行则是将click一次性绑定到Body元素上，但仅仅在单击Ul下li元素时被触发(即由参数"Ul>li"匹配的元素)。另外，单击“白居易”和“杜甫”、“李白”效果一样。虽然在Body元素上一次性绑定click事件时，“白居易”对应的li元素还不存在，但是在click时，该事件会自动向上冒泡，传递到Body时，发现已绑定事件click，且来源于Ul>li匹配的元素，因此执行一次相关代码。另外，单击任何一个li后，再单击其他li元素，同样不会显示该诗人名字，因为仅仅在Body上绑定一次，执行一次后即解除绑定。

例程6-18

```
<Html>
    <Head><Title>事件测试</Title></Head>
    <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
    <Body>
        <Ul><li>李白</li><li>杜甫</li></Ul>
        <Script Language="JavaScript">
            $("li:eq(0)").one("click",{birthYear:701,deathYear:762},function(e){
                var poet=$(this).text();
                alert(poet+", 生于"+e.data.birthYear+", 卒于"+e.data.deathYear);
            })
            $("Body").one("click","UL>li",function(){
                var poet=$(this).text();
                alert(poet);
            })
            $("UL").append("<li>白居易</li>");
        </Script>
    </Body>
</Html>
```

4、委托绑定法

$.on()能对已经存在的元素或将存在的元素绑定事件，而$.bind、$.one以及简捷式事件绑定则只能对已经存在的元素绑定事件，其格式如下。注意观察会发现，$.on()与$.one()的用法2、用法3非常相似。

$.on()绑定事件

用法1：$.on(type[,selector][,data],fn)

用法2：$.on({type:f1,type:f2,type:fn}[,selector][,data])

type：必须参数，事件类型(如click等)；data：可选参数，向事件传递数据；fn：事件处理函数

$.on()的使用分为两种情况，当省略selector时，相当于$.bind()即直接绑定事件；当含有selector参数时，则是$.on()的典型用法即委派式绑定，其功能是当匹配元素的晚辈元素(有selector指定)发生事件时，调用fn对应代码进行处理，如例程6-19所示。

例程6-19

```
<Html>
    <Head><Title>事件测试</Title></Head>
```

```
第3行          <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行          <Body>
第5行                  <Ul><li>李白</li><li>杜甫</li></Ul>
第6行                  <Script Language="JavaScript">
第7行                          $("UL").on("click","li",function(){
第8行                                  var clickedPoet=$(this).text();
第9行                                  alert(clickedPoet);
第10行                         });
第11行                         $("UL").append("<li>白居易</li>");
第12行                 </Script>
第13行         </Body>
第14行   </Html>
```

在例程6-19中，第7-10行的本意是为li元素绑定事件，但li元素可能动态发生变化，此时可委托其长辈元素UL代为管理，当后备元素发生指定事件时，则调用相关处理代码。例程6-20、例程6-21是$.on()的另外用法，其功能相同。

例程6-20

```
第1行    <Html>
第2行          <Head><Title>事件测试</Title></Head>
第3行          <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行          <Style type="text/css">
第5行                  .lightIt{font-weight:bold;color:red}
第6行          </Style>
第7行          <Body>
第8行                  <Ul><li>李白</li><li>杜甫</li></Ul>
第9行                  <Script Language="JavaScript">
第10行                         $(document).on(
第11行                                 {
第12行                                         mouseover:function(){
第13行                                                 $(this).toggleClass("lightIt");
第14行                                         },
第15行                                         mouseout:function(){
第16行                                                 $(this).toggleClass("lightIt");
第17行                                         }
第18行                                 },"Ul>li"
第19行                         );
第20行                         $("UL").append("<li>白居易</li>");
第21行                 </Script>
第22行         </Body>
第23行   </Html>
```

例程6-21

```
第1行    <Html>
第2行          <Head><Title>事件测试</Title></Head>
第3行          <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行          <Style type="text/css">
```

```
第5行                    .lightIt{font-weight:bold;color:red}
第6行            </Style>
第7行            <Body>
第8行                    <Ul><li>李白</li><li>杜甫</li></Ul>
第9行                    <Script Language="JavaScript">
第10行                           $("Body").on("mouseover mouseout","li",function(){
第11行                                   $(this).toggleClass("lightIt");
第12行                           });
第13行                           $("UL").append("<li>白居易</li>");
第14行                   </Script>
第15行           </Body>
第16行   </Html>
```

一般说来，已有元素和新增元素都是Body元素的子元素，因此只要selector参数能匹配的元素，都能触发指定的事件；而doucment对象更是Body元素的长辈元素，因此更能匹配其后辈元素。

5、悬停事件绑定

$.hover方法模拟链接的hover，即当鼠标进入匹配元素时执行一个函数，当鼠标离开匹配元素时执行另外一个函数，其格式为：hover(over_fn,out_fn)，over_fn代表鼠标进入时执行的函数，out_fn代表鼠标离开时执行的函数。例程6-22是其应用示例。$.hover可以用$.bind方法绑定mouseover、mouseout事件实现。

例程6-22

```
第1行   <HTML>
第2行           <HEAD>
第3行                   <TITLE>$.hover方法</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                           .highLight{font-weight:bold}
第8行                   </Style>
第9行           </HEAD>
第10行
第11行          <BODY>
第12行                  <Div>
第13行                          <P>唐诗</P>
第14行                          <P>宋词</P>
第15行                  </Div>
第16行                  <SCRIPT LANGUAGE="JavaScript">
第17行                          $("P").hover(
第18行                                  function(){
第19行                                          $(this).addClass("highLight");
第20行                                  },
第21行                                  function(){
第22行                                          $(this).removeClass("highLight");
第23行                                  }
第24行                          );
```

```
第25行                    </SCRIPT>
第26行            </BODY>
第27行    </HTML>
```

6、切换事件绑定

$.toggle方法的格式是：$.toggle(f1,f2,……,fn) ，其中f1,……,fn代表多个函数。当单击匹配对象时，顺序依次执行指定函数，当执行到最后一个函数后，从头开始循环并依次执行。$.toggle内部绑定click事件，可以用unbind解除click事件使得$.toggle失效。例程6-23是$.toggle的应用示例。

例程6-23

```
第1行                    <SCRIPT LANGUAGE="JavaScript">
第2行                            $("Body").toggle(
第3行                            function(){
第4行                                    alert("第1次点击");
第5行                            },function(){
第6行                                    alert("第2次点击");
第7行                            },function(){
第8行                                    alert("第3次点击");
第9行                            });
第10行                   </SCRIPT>
```

当在Body区域即页面区域单击鼠标时，将依次显示“第1次点击→第2次点击→第3次点击→第1次点击→第2次点击……”。

第三节　事件注销B

所谓事件注销是让已绑定事件的元素不再对事件作出相应反应。在jQuery中事件注销方法主要有$.unbind()和$.off()两种，分别与$.bind()和$.on()对应，其格式如下：

jQuery事件注销

方法1：$.unbind([type][,fn])

方法2：$.off(type[,selector][,fn])

type代表事件名称，fn事件处理函数。

对于$.unbind()，如果省略全部参数，则注销所有通过$.bind()绑定在该元素上的事件，如果指定type即事件，则删除所有该事件及其对应函数，如果同时指定type和fn，则删除该事件对应的函数。对于$.off()，不能省略type参数，当省略selector时，则注销所有type事件及其对应的处理函数，如果指定selector则删除对应元素的事件及其函数，如果参数fn未省略，则删除相应函数。如例程6-24所示。注意：一个type可以对应多个fn，$.unbind()或$.off()可以仅注销某一个fn。

例程6-24

```
第1行    <Html>
第2行            <Head><Title>事件测试</Title></Head>
第3行            <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行            <Style type="text/css">
第5行                    .lightIt{font-weight:bold;color:red}
第6行            </Style>
第7行            <Body>
第8行                    <Ul><li>李白</li><li>杜甫</li></Ul>
第9行                    <Script Language="JavaScript">
第10行                           function Fa(){
第11行                                   var lenText=$(this).text().length;
第12行                                   alert("Fa()执行，事件文本长度="+lenText);
```

```
第13行                          }
第14行                          function Fb(){
第15行                                  var eventText=$(this).text();
第16行                                  alert("Fb()执行，事件文本内容="+eventText);
第17行                          }
第18行                          function Fc(){
第19行                                  var eventIndex=$(this).index();
第20行                                  alert("Fc()执行，事件元素编号="+eventIndex);
第21行                          }
第22行                          function Fd(){
第23行                                  var eventTime=new Date().getTime();
第24行                                  alert("Fd()执行，事件发生时间="+eventTime);
第25行                          }
第26行                          $("li").bind("click",Fa).bind("click",Fb);
第27行                          $("UL").on("click","li:eq(0)",Fc).on("click","li:eq(0)",Fd);
第28行                          $("UL").on("click","li:eq(1)",Fc).on("click","li:eq(1)",Fd);
第29行                          $("UL").on("click","li:eq(2)",Fa).on("click","li:eq(2)",Fb);//此时元素尚不存在
第30行                          $("UL").on("click","li:eq(3)",Fc).on("click","li:eq(3)",Fd);//此时元素尚不存在
第31行                          $("UL").append("<li>白居易</li>").append("<li>李商隐</li>");//增添新元素
第32行
第33行                          $("li:eq(0)").unbind();//删除两个click事件及其对应函数Fa()、Fb()；
第34行                          $("li:eq(1)").unbind("click",Fb);//仅删除一个click事件及其对应的Fb()；
第35行                          $("UL").off("click","li:eq(2)",Fa);//仅删除由$.on()绑定的1个事件及其对应函数
第36行                          $("UL").off("click","li:eq(3)");//删除由$.on()绑定的全部事件
第37行                  </Script>
第38行          </Body>
第39行  </Html>
```

在例程6-24中有Fa，Fb，Fc，Fd四个函数，当执行时能显示函数名称等信息。第26行，为已有的li元素绑定click事件及其处理函数Fa，Fb，这种绑定方法只能对已经存在的元素绑定事件，不能为第31行新增的li元素绑定事件。第27到第30行用$.onf()分别为4个不同的li元素绑定click事件及其处理函数。值得注意的是第29、3行绑定事件时，该li元素尚不存在，但$.on()可以为已存在和将存在的元素捆绑事件。

第33行采用$.unbind()解除绑定，解除绑定在第一个li元素上的所有事件及其函数。第34行仅仅解除绑定的click事件中的Fb函数，其余函数不受影响，如Fa以及Fc，Fd。第35行解除由“li:eq(2)”发起的click事件及其对应的Fa函数，Fb不受影响。第36行则删除由“li:eq(3)”发起的click事件及其全部对应函数，包括Fc和Fd。注意：$.off()解除时选定元素的形式必须$.on()保持一致。

第四节　事件触发

当事件绑定到匹配元素后，一旦事件发生，将自动触发该事件相应处理程序，如元素被绑定click事件，当在元素上单击时，将自动启动相应处理处理程序，这种形式的触发可以称之为应急触发或系统触发。除此以外，事件还可以通过程序触发，如例程6-25所示，其第7-10行的功能是为li元素绑定事件，但单击某个li元素时，将显示该条目文本。第12行在所有事件之外，当网页打开时将被执行，其功能是触发绑定在第1个li元素上的click事件，注意该事件的发生时，没有单击鼠标即自动发生，和用鼠标单击第1个元素的效果一致(显示“李白”)。第15行是另外一种形式的触发事件，即匹配元素调用没有参数的事件名称，即没有参数的事件方法，绝大部分事件都支持该模式。

例程6-25

```
第1行   <Html>
第2行           <Head><Title>事件测试</Title></Head>
第3行           <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
```

```
第4行          <Body>
第5行              <Ul><li>李白</li><li>杜甫</li></Ul>
第6行              <Script Language="JavaScript">
第7行                  $("li").click(function(){
第8行                      var clickedPoet=$(this).text();//获得诗人名称
第9行                      alert(clickedPoet);
第10行                 });
第11行
第12行                 $("li:eq(0)").trigger("click");//触发第1个li元素的click事件
第13行
第14行                 $("body").dblclick(function(){
第15行                     $("li:eq(1)").click();//触发第2个li元素的click事件
第16行                 });
第17行                 $("body").keydown(function(){
第18行                     $("li").trigger("click");//触发全部li元素的click事件
第19行                 });
第20行             </Script>
第21行         </Body>
第22行   </Html>
```

当元素的某种事件被触发后，可能会有默认行为，如文本框获得焦点后，其内有光标在闪烁，其光标闪烁就是默认行为。其他元素也有类似情况，如按钮被点击后，四周会有边框。例程6-26第22行通过$.focus()让文本框获得焦点，其与用鼠标单击该文本框或者键盘Tab键让其获得焦点效果一致，都有光标在闪烁。

例程6-26

```
第1行   <Html>
第2行       <Head><Title>事件测试</Title></Head>
第3行       <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行       <Style type="text/css">
第5行           .lightIt{background-color:red}
第6行       </Style>
第7行       <Body>
第8行           学号<input type=text id="studId"><BR>
第9行           学校<input type=text id="studSchool">
第10行          <Script Language="JavaScript">
第11行              $("#studId").focus(function(){
第12行                  $(this).addClass("lightIt");
第13行              });
第14行
第15行              $("#studId").focus();//网页一打开就让匹配元素获得焦点
第16行
第17行              //失去焦点时触发
第18行              $("#studId").blur(function(){
第19行                  var inputText=$(this).val();
第20行                  if(inputText.length==0){
第21行                      alert("请输入学号!");
第22行                      $(this).focus();//触发当前元素获得焦点，不填入学号不能离开输入框
```

```
第23行                                    //确保学号必须录入
第24行                                }
第25行                            });
第26行                    </Script>
第27行            </Body>
第28行    </Html>
```

图6-4　例程6-26执行效果

在jQuery中，可以通过类似$("li").click(fn)方式给P元素绑定click事件及其处理代码fn，如果省略fn则是触发匹配元素指定事件。如$("li").click()则是在所有li元素上触发click行为；$("input").focus()则是让input元素发生获得焦点行为。即$.click(fn)绑定click事件，$.click()是触发click事件，其中click可以替换为绝大部分其他事件 ，如dblclick，keydown、focus、resize等，具体请参考本章“事件罗列”。

事件触发还有其他形式，如例程6-25、例程6-26所示的$.trigger(enentName)，其中eventName代表事件名称，如click、dblclick、keydown、focus等。在事件触发的同时，还可以传递事件数据(eventData)，如例程6-27所示，其格式为$.trigger(enentName,eventData) 。

例程6-27

```
第1行    <BODY>
第2行        <Button>单击触发Body的click事件!</Button>
第3行        <SCRIPT LANGUAGE="JavaScript">
第4行            $("Body").bind("click",function(event,a,b){
第5行                alert("a="+a+" b="+b);
第6行            });
第7行            $("Button").click(function(){
第8行                $("Body").trigger("click",["李白","杜甫"]);
第9行            });
第10行       </SCRIPT>
第11行   </BODY>
```

当单击Button时，触发Body的click事件，并通过数组传递数据，虽然没有点击Body但仍显示“a=李白 b=杜甫”，然后显示“a=undefined b=undefined”。这是因为事件冒泡，从Body内的Button传递到Body的click事件，而此时a，b没有任何数据，因此显示undefined。同理，当在Body区域单击时，也同样会显示“a=undefined b=undefined”。

触发事件还可以用riggerHandler()方法，其格式为：$.triggerHandler(eventName,eventData) ，其eventName为事件名称，eventName也是可选参数，数组对象。和$.trigger()相比，$.triggerHandler()不触发浏览器默认动作，也不产生事件冒泡，即$.triggerHandler()仅仅触发该事件相应处理代码，不触发事件的模型行为。另外，当匹配元素超过1个时，仅第1个元素的事件被触发。在例程6-28中，虽然$("P")匹配两个P元素，但仅仅第一个P元素的click事件被触发。如果将$.triggerHandler()变更为$.trigger()则两个P元素的click事件都会被触发。

例程6-28

```
第1行    <BODY>
第2行        <P>李白</P>
第3行        <P>杜甫</P>
第4行        <Button>在P上触发click事件</Button>
第5行        <SCRIPT LANGUAGE="JavaScript">
第6行            $("P").click(function(){
第7行                alert($(this).html());
第8行            });
```

```
第9行                    $("Button").click(function(){
第10行                        $("P").triggerHandler("click");
第11行                    });
第12行            </SCRIPT>
第13行        </BODY>
```

第五节　事件对象

当事件发生时，必然会有诸多相关信息，如：事件发生的时间、匹配元素、相对网页位置以及键盘事件是由哪个按键产生的事件、鼠标事件是由左中右哪个按键触发等，有些像When、Who、Where、What等，具体事件处理代码类似How。事件有冒泡传播特性，该事件是否允许冒泡，有些事件有默认行为，事件是否允许默认行为发生等。上述都可以通过事件对象(Event)予以处理，或者说事件对象就是用于保存事件发生时的相关信息以及事件本身的处理。

事件处理函数的第一个参数为Event对象，虽然该参数常被省略。如$("P").click(function(e){}); 中，e通常被省略，如需处理事件的相关信息，可以增加该参数。另外，e相当于形式参数，可以用其他名字代替，如myEvent等。e所代表的Event对象有多种属性和方法，如：鼠标位置、键盘按键等，这些信息有助于编写各种类型的程序。例程6-29利用鼠标位置实现单击图片右部分，则图片编号增大，单击图片左部分则图片编号减小，利用event.pageX(事件发生时鼠标离页面左边界的位置，即click事件发生在页面的位置)、event.pageY(事件发生时鼠标离页面顶边界的位置)。

例程6-29

```
第1行   <Html>
第2行       <Head><Title>事件测试</Title></Head>
第3行       <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行       <Body>
第5行       <img src="1.jpg">
第6行       <Script Language="JavaScript">
第7行           $("Img").click(function(e){
第8行               var pageX=e.pageX,pageY=e.pageY;//取得事件发生时鼠标所在页面位置
第9行               var leftX=$(this).offset().left,topY=$(this).offset().top;//取得图片所在页面位置
第10行              var imgWidth=$(this).width();//取得图片宽度
第11行              var imgSrc=$(this).attr("src");//图片src属性的值
第12行
第13行              if(pageX>(leftX+imgWidth/2))//单击图片右侧时
第14行                  imgChange(1,imgSrc);//增大图片编号，根据函数定义
第15行              else
第16行                  imgChange(0,imgSrc);//减小图片编号，根据函数定义
第17行          });
第18行          function imgChange(Mode,imgSRC){
第19行              //去掉.jpg后为图片编号，也即当前图片编号
第20行              var currentImgNo=parseInt(imgSRC.replace(".jpg"));//此处用正则表达式处理更好
第21行
第22行              switch(Mode){
第23行                  case 1://增大图片编号，最大图片编号为8
第24行                      var imgName=(currentImgNo==8?1:currentImgNo+1)+".jpg";
第25行                      $("Img").attr("src",imgName);
第26行                      break;
第27行                  case 0://减小图片编号
第28行                      var imgName=(currentImgNo==1?8:currentImgNo-1)+".jpg";
```

```
第29行                                    $("Img").attr("src",imgName);
第30行                                    break;
第31行                                }
第32行                            }
第33行                    </Script>
第34行                    </Body>
第35行            </Html>
```

上述程序还可以增加键盘控制，即向右的箭头图片编号增大，向左的箭头图片编号减小。该部分绑定在Body元素上，如例程6-30所示。在实际应用中，很多网络小说就采用类似的方式控制翻页。

例程6-30

```
第1行        $("Body").bind("keydown",function(e){
第2行                alert(e.which);//显示按键编号，运行时刻删除或注释改行
第3行                switch(e.which){
第4行                        case 39:
第5行                                imgChange(1);
第6行                                break;
第7行                        case 37:
第8行                                imgChange(0);
第9行                                break;
第10行               }
第11行       });
```

在例程6-30中，function(e)中的e代表触发事件时的event，其e.which代表事件发生时按键(鼠标或键盘按键)。可以用alert(e.which)显示按键值。下表列出了event对象的属性和方法。

表1：jQuery事件对象(Event)的方法和属性

属性名称	描述
event.stopPropagation()	阻止事件冒泡，长辈节点(元素)无法处理该事件并获得事件通知。如例程6-31所示。
event.stopImmediatePropagation()	立即阻止事件冒泡，当前元素的后续事件及其长辈节点(元素)都无法处理该事件并获得事件通知。如例程6-31所示。
event.preventDefault()	阻止事件默认动作，如例程6-32所示，单击链接将打开新的网址，这次其链接的默认行为，但执行本指令，将不再打开新网页。
event.target	初始触发事件的元素。
event.currentTarget	正在起泡阶段的元素，与this等同。
event.relatedTarget	由事件触发的相关元素，仅适用于mouseout和mouseover事件。mouseover中的relatedTarget指向鼠标来自元素，而mouseout中的relatedTarget指向鼠标去向元素，如例程6-31所示。
event.delegateTarget	事件委托元素，如例程6-33所示。
event.data	事件绑定时传递给事件处理函数的数据，参考例程6-35和例程6-36。
event.pageX	鼠标离文档左边边界位置，参考例程6-29。
event.pageY	鼠标离文档顶部边界位置，参考例程6-29。
event.which	按键值，仅对鼠标或键盘事件有效，参考例程6-30。
event.type	触发事件的事件名称，如例程6-34。
event.result	由事件触发的处理函数返回的最后一个值，如例程6-34。
event.timeStamp	时间戳，事件触发时与1970-1-1日的时间差，单位毫秒，如例程6-34。

事件能冒泡传播，其传播可以通过事件对象的方法予以阻止，如例程6-31第12行e.stopImmediatePropagation()所示，其中e为事件对象。阻止事件冒泡有两种方法，分别是：event.stopPropagation()和event.stopImmediatePropagation()，这两种方法都能阻止事件冒泡，其长辈元素将不能捕获该事件，但当一个元素的同一事件有多个处理时，后一种方法能阻止后续事件触发，而前一种方法不阻止后续事件的触发。在本例中，li元素绑定3个click事件，其中第2个click事件中应用了e.stopImmediatePropagation()，因此，其第3个click事件将不会被执行，如修改成e.stopPropagation()则将被执行。当然，采用这两种方法，单击li元素时，Body的click事件都不会触发，因为这两种方法，都将阻止事件冒泡，长辈元素将不能捕获该事件。

例程6-31

```
第1行   <Html>
第2行       <Head><Title>事件测试</Title></Head>
第3行       <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行       <Body>
第5行           <Ul><li>李白</li><li>杜甫</li></Ul>
第6行           <Script Language="JavaScript">
第7行               $("li").click(function(e){
第8行                   var clickedPoet=$(this).text();//取得点击诗人名称
第9行                   alert(clickedPoet);
第10行              });
第11行              $("li").click(function(e){
第12行                  e.stopImmediatePropagation();//立即终止后续事件执行
第13行
第14行                  var clickedElement=this.tagName;//取得元素名称
第15行                  alert(clickedElement);
第16行              });
第17行              $("li").click(function(e){
第18行                  var liIndex=$(this).index();//取得点击元素的序号
第19行                  alert(liIndex);
第20行              });
第21行              $("Body").click(function(e){
第22行                  var liIndex=$(e.target).index();//触发事件元素的序号
第23行                  alert(liIndex);
第24行              });
第25行          </Script>
第26行      </Body>
第27行  </Html>
```

一些元素某种事件有默认行为，如单击链接，浏览器将默认打开网页，文本框获得焦点将有闪烁光标等。事件的默认行为可以取消。如例程6-32即取消了单击链接打开网页的行为。

例程6-32

```
第1行   <Html>
第2行       <Head><Title>事件测试</Title></Head>
第3行       <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行       <Body>
第5行           <a href="http://www.pku.edu.cn">北京大学</a>
第6行           <Script Language="JavaScript">
第7行               //如无下述代码，单击链接将打开该网址
第8行               $("a").click(function(e){
```

```
第9行                                   e.preventDefault();//不再默认打开网址
第10行                                  var clickedLink=$(this).attr("href");//获取href属性值
第11行                                  alert(clickedLink);//显示链接网址
第12行                              });
第13行                      </Script>
第14行          </Body>
第15行  </Html>
```

事件可以采用委托绑定法，当事件触发时，this指向触发事件的元素，如果需要知道委托元素，可通过event.delegateTarget，如例程6-33第8行所示。

例程6-33

```
第1行   <Html>
第2行           <Head><Title>事件测试</Title></Head>
第3行           <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行           <Body>
第5行                   <Ul><li>李白</li><li>杜甫</li></Ul>
第6行                   <Script Language="JavaScript">
第7行                           $("ul").on("click","li",function(e){
第8行                                   var content=$(e.delegateTarget).html();//取得委托元素的内容
第9行                                   alert(content);//显示<li>李白</li><li>杜甫</li>
第10行                          });
第11行                  </Script>
第12行          </Body>
第13行  </Html>
```

例程6-34是有关event.timeStamp(事件触发时间戳记)、event.type(事件类型)以及event.result的使用。event.result一般涉及到同一个事件多次触发(顺序执行)，其保存事件最近一次的返回值，如第17行返回“ABC”，第20行通过e.result获得最近一次返回值并加上“XYZ”作为该事件处理函数的返回值，第23行获得最近一次事件的返回值，所以结果为“ABCXYZ”。event.result的使用，有助于在多次事件之间传递数据。

例程6-34

```
第1行   <Html>
第2行           <Head><Title>事件测试</Title></Head>
第3行           <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第4行           <Body>
第5行                   <Ul><li>李白</li><li>杜甫</li></Ul>
第6行                   <Div id="eventInfo"></Div>
第7行                   <Script Language="JavaScript">
第8行                           $("li").bind("mouseenter mouseleave",function(myE){
第9行                                   var eventType=myE.type;//事件类型
第10行                                  var eventTime=myE.timeStamp;//事件发生时间
第11行                                  //显示内容到id名为eventInfo的div中
第12行                                  $("#eventInfo").html("元素内容:"+$(this).text()+
第13行                                          " 事件类型:"+eventType+" 事件时间:"+eventTime);
第14行                          });
第15行
第16行                          $("li").click(function(e){
```

```
第17行                    return "ABC";//返回字符ABC
第18行                });
第19行                $("li").click(function(e){
第20行                    return e.result+"XYZ";//返回字符XYZ
第21行                });
第22行                $("li").click(function(e){
第23行                    alert(e.result);//显示为ABCXYZ
第24行                });
第25行
第26行            </Script>
第27行        </Body>
第28行    </Html>
```

event.data可以在绑定事件时向事件处理函数传递的数据，如例程6-35，第3行将message所代表的“apple”传递给li:eq(0)元素的click事件，第7行是将message所代表的“orange”传递给li:eq(1)元素的click事件，当单击这两个元素时，将分别显示apple或orange。

例程6-35

```
第1行    <Script Language="JavaScript">
第2行        var message = 'apple';
第3行        $('li:eq(0)').bind('click',message,function(e){
第4行         alert(e.data);
第5行        });
第6行        message = 'orange';
第7行        $('li:eq(1)').bind('click', message,function(e) {
第8行         alert(e.data);
第9行        });
第10行       //点击li:eq(0)李白或者li:eq(1)杜甫，分别显示apple和orange
第11行   </Script>
```

例程6-36与例程6-35相似，也是显示message内容，不过均显示为“orange”，其功能是显示message内容。当事件绑定时，并没有执行函数中的代码，当事件触发时执行，一般说来，事件触发时上述代码都已经执行完毕，此时message的内容为orange，因此alert(message)将都显示为orange。

例程6-36

```
第1行    <Html>
第2行        <Head>
第3行            <Title>事件测试</Title>
第4行        </Head>
第5行        <Script language="javascript" src="./jquery-1.9.1.min.js"></Script>
第6行        <Body>
第7行            <Ul><li>李白</li><li>杜甫</li></Ul>
第8行            <Script Language="JavaScript">
第9行                var message = 'apple';
第10行               $('li:eq(0)').bind('click', function(){
第11行                alert(message);
第12行               });
第13行               message = 'orange';
```

```
第14行                    $('li:eq(1)').bind('click', function() {
第15行                     alert(message);
第16行                    });
第17行                    //点击li:eq(0)李白或者li:eq(1)杜甫，都将显示orange
第18行                </Script>
第19行          </Body>
第20行    </Html>
```

第六节　自定义事件

一说起事件，很多人常想起鼠标事件、键盘事件等，除此以外，还可以自定义事件，并可触发事件执行相应的代码，如例程6-21所示。在第7行为P元素绑定一个名为myEvent的自定义事件，并定义了相应的处理函数。该事件被定义后，和其他事件一样，可以通过$.trigger触发，如第11行所示。在定义myEvent事件时，同样可以传递event.data数据，函数也同样支持event参数等。$.trigger也同样可以触发事件并传递参数。与jQuery原有定义保持一致。和键盘和鼠标事件不一样的是自定义事件不会在键盘和鼠标事件发生时自动触发，须通过指定代码触发。

例程6-37

```
第1行    <BODY>
第2行          <Style type=text/css>
第3行                .highLight{background-color:black;color:white;font-weight:bold}
第4行          </Style>
第5行          <P>自定义事件，在页面双击即可触发。</P>
第6行          <SCRIPT LANGUAGE="JavaScript">
第7行                $("P").bind("myEvent",function(){
第8行                      $(this).addClass("highLight");
第9行                });
第10行               $("Body").dblclick(function(){
第11行                     $("P").trigger("myEvent");
第12行               });
第13行         </SCRIPT>
第14行   </BODY>
```

第七节　事件罗列

jQuery支持的事件较多，大致可以分为：鼠标事件(mouse event)、键盘事件(keyboard event)、文档事件(document event)、表单事件(form event)、浏览器事件(browser event)。

1、鼠标事件

鼠标事件是由鼠标操作所触发的事件，jQuery支持的鼠标事件如表2所示，其中mouseenter和mouseleave非JavaScript标准事件。

表2：鼠标事件

事件名称	JavaScript事件	描述	备注
click()	onclick	当单击鼠标时触发该事件；或在指定元素触发该事件；	mouseover与mousemove的区别是：mouseover仅在鼠标进入元素时触发事件，在元素内移动不触发事件，而mousemove则当鼠标在元素内移动时连续触发事件。mouseout与mouseleave的区别是当鼠标离开被选元素及其子元素时，都将触发mouseout
dblclick()	ondblclick	当双击鼠标时触发该事件；或在指定元素触发该事件；	
mousedown()	onmousedown	当鼠标按下时触发该事件；或在指定元素触发该事件；	
mouseenter()	------	当鼠标进入某元素时触发该事件；或在指定元素触发该事件；	
mouseleave()	------	当鼠标离开某元素时触发该事件；或在指定元素触发该事件；	
hover(over,out)	------	鼠标悬停事件，相当于mouseenter()和mouseleave()的综合。	

toggle(fn,fn)	------	鼠标单击切换事件，依次调用toggle()函数的多个函数。	事件，而mouseleave仅在离开被选元素触发该事件，与子元素没有关系。mouseover与mouseenter与之类似。当触发dblclick与mousedown事件时，click事件同样被激活。
mousemove()	onmousemove	当鼠标在某元素移动触发该事件；或在指定元素触发该事件；	
mouseout()	onmouseout	当鼠标离开某元素时触发该事件；或在指定元素触发该事件；	
mouseover()	onmouseover	当鼠标进入某元素时触发该事件；或在指定元素触发该事件；	
mouseup()	onmouseup	当鼠标松开时触发该事件；或在指定元素触发该事件；	

jQuery事件对象(Event)的方法和属性

事件分类	事件名称	描述	备注
表单事件	blur()	当元素失去焦点触发该事件；或在指定元素触发该事件；	主要用于带有输入性质的元素，如input等；
	select()	文本被选择时触发该事件；或在指定元素触发该事件；	
	submit()	当提交表单时触发该事件；或在指定表单触发该事件；	仅适用于Form元素
	change()	当元素内容发生改变时触发该事件；或在指定表单触发该事件；	该事件仅适用于文本域、textarea和select 元素。
	focus()	当元素获得焦点时触发该事件；或在指定元素触发该事件；	鼠标点击元素或tab键定位到元素时，该元素获得焦点。
键盘事件	keydown()	当键按下时触发该事件；或在指定元素触发该事件；	keypress只能捕获单个可显示字符，区分大小写；keydown和keyup能捕获出PrtSc之外的所有字符，包括特殊键如：方向键、ctrl状态、alt状态等。
	keypress()	当键按下时触发该事件；或在指定元素触发该事件；	
	keyup()	当键松开时触发该事件；或在指定元素触发该事件；	
窗口/文档事件	load()	当元素及其子元素加载时触发该事件；	load与unload在不同浏览器存在兼容性问题。ready()仅适用于文档，有三种写法：$(document).ready(fn)、$().ready(fn)、$(fn)，其中fn代表函数(function)。
	ready()	当网页文档就绪时触发该事件，多数事件都应在该事件出发时开始工作，以防止网页内容不完整导致问题。	
	unload()	仅适用于window对象，即进入新网页或离开，以及关闭窗口时触发unload 事件	
浏览器事件	resize()	浏览器窗口尺寸改变时触发该事件；	
	scroll()	当浏览器滚动条时触发事件及其代码；	

2、键盘事件

第八节　$.ready()

第九节　更多了解

1、事件驱动

事件驱动不仅仅在信息技术领域得到广泛的应用，在实际工作生活中，也经常看见事件驱动的身影。所谓事件驱动，即在持续事务管理过程中的一种决策策略，即根据出现的事件，及时调动相关资源执行相关任务，使不断出现的事件得以及时解决，防止事务堆积。

现在的计算机操作常常是点点鼠标即可执行操作，点鼠标即为事件，操作就是调用事件处理程序。当然事件不仅是鼠标操作，还可以包括很多，如键盘事件、时间事件、以及其他各种软硬件事件等。事件驱动的核心是事件，其基本结构由事件收集器、事件发送器和事件处理器组成。事件收集器专门负责收集所有事件，包括来自用户(如鼠标、键盘事件等)、来自硬件(如时钟事件等)和来自软件(如操作系统、应用程序本身等)。事件发送器负责将收集到的事件分发到目标对象。事件处理器做具体的事件响应工作。在Web程序中，事件收集器主要由浏览器承担，事件分发由浏览器和JavaScript程序承担，事件处理器主要由开发者开发的各种事件处理代码承担。被绑定事件的元素成为事件接收者，其处理函数即为事件处理。

事件驱动模式将流线型程序结构根本性地改变为事件触发模式，这种模式的系统具有很好的灵活性，可以胜任各种离散的、随机的事件。Windows系统是典型“事件驱动”模型，事件收集器由Windows系统完成；事件发送器部分由Windows完成，部分由Windows应用程序承担，事件处理器则由用户编写的应用程序承担。学习Web编程，有助于学习Windows以及目前流行操作系统下的编程，如C#、Java、Objective-C等。

第十节　小结