**jQuery鼠标事件之click与dbclick事件**

用交互操作中，最简单直接的操作就是点击操作。jQuery提供了两个方法一个是click方法用于监听用户单击操作，另一个方法是dbclick方法用于监听用户双击操作。这两个方法的用法是类似的，下面以click()事件为例

使用上非常简单：

**方法一：$ele.click()**

绑定$ele元素，不带任何参数一般是用来指定触发一个事件，用的比较少

<div id="test">点击触发<div>

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

alert('触发指定事件')

})

$("#test").click(function(){

$("ele").click() //手动指定触发事件

});

**方法二：$ele.click( handler(eventObject) )**

绑定$ele元素，每次$ele元素触发点击操作会执行回调 handler函数，这样可以针对事件的反馈做很多操作了，方法中的this是指向了绑定事件的元素

<div id="test">点击触发<div>

$("#test").click(function() {

//this指向 div元素

});

**方法三：$ele.click( [eventData ], handler(eventObject) )**

使用与方法二一致，不过可以接受一个数据参数，这样的处理是为了解决不同作用域下数据传递的问题

<div id="test">点击触发<div>

$("#test").click(11111,function(e) {

//this指向 div元素

//e.data => 11111 传递数据

});

dbclick()的用法和click()的用法是类似的，可以参考以上click()的用法。

**dbclick与click事件不同的是：**

**click事件触发需要以下几点：**

* click事件其实是由mousedown与mouseup 2个动作构成，所以点击的动作只有在松手后才触发

PS:mousedown和mouseup下一节会讲到

**dbclick事件触发需要以下几点：**

dbclick又是由2个click叠加而来的，所以dbclick事件只有在满足以下条件的情况下才能被触发

* 鼠标指针在元素里面时点击。
* 鼠标指针在元素里面时释放。
* 鼠标指针在元素里面时再次点击，点击间隔时间，是系统而定。
* 鼠标指针在元素里面时再次释放。

**注意：**在同一元素上同时绑定 click 和 dbclick 事件是不可取的。各个浏览器事件触发的顺序是不同的，一些浏览器在dbclick之前接受两个 click 事件 ，而一些浏览器只接受一个 click 事件。用户往往可通过不同的操作系统和浏览器配置双击灵敏度

**jQuery鼠标事件之mousedown与mouseup事件**

用户交互操作中，最简单直接的操作就是点击操作，因此jQuery提供了一个mousedown的快捷方法可以监听用户鼠标按下的操作，与其对应的还有一个方法mouseup快捷方法可以监听用户鼠标弹起的操作。**两种方法用法类似**，下面以mousedown()为例

使用上非常简单：

**方法一：$ele.**mousedown**()**

绑定$ele元素，不带任何参数一般是用来指定触发一个事件，可能一般用的比较少

<div id="test">点击触发<div>

$("ele").mousedown(function(){

alert('触发指定事件')

})

$("#test").mousedown(function(){

$("ele").mousedown() //手动指定触发事件

});

**方法二：$ele.mousedown( handler(eventObject) )**

绑定$ele元素，每次$ele元素触发点击操作会执行回调 handler函数

这样可以针对事件的反馈做很多操作了

<div id="test">点击触发<div>

$("#test").mousedown(function() {

//this指向 div元素

});

**方法三：$ele.mousedown( [eventData ], handler(eventObject) )**

使用与方法二一致，不过可以接受一个数据参数，这样的处理是为了解决不同作用域下数据传递的问题

<div id="test">点击触发<div>

$("#test").mousedown(11111,function(e) {

//this指向 div元素

//e.data => 11111 传递数据

});

**mousedown事件触发需要以下几点：**

* mousedown强调是按下触发
* 如果在一个元素按住了鼠标不放，并且拖动鼠标离开这个元素，并释放鼠标键，这仍然是算作mousedown事件
* 任何鼠标按钮被按下时都能触发mousedown事件
* 用event 对象的which区别按键，敲击鼠标左键which的值是1，敲击鼠标中键which的值是2，敲击鼠标右键which的值是3

**mouseup事件触发需要以下几点：**

* mouseup强调是松手触发，与mousedown是相反的
* mouseup与mousedown组合起来就是click事件
* 如果用户在一个元素上按下鼠标按键，并且拖动鼠标离开这个元素，然后释放鼠标键，这仍然是算作mouseup事件
* 任何鼠标按钮松手时都能触发mouseup事件
* 用event 对象的which区别按键，敲击鼠标左键which的值是1，敲击鼠标中键which的值是2，敲击鼠标右键which的值是3

**另外需要注意的是：**

**click与mousedown的区别：**

* click事件其实是由mousedown于mouseup 2个动作构成，所以点击的动作只有在松手后才触发

**jQuery鼠标事件之mousemove事件**

用交互操作中，经常需要知道用户是否有移动的操作。基于移动的机制可以做出拖动、拖拽一系列的效果出来。针对移动事件，jQuery提供了一个mousemove的快捷方法可以监听用户移动的的操作

使用上非常简单：

**方法一：$ele.mousemove()**

绑定$ele元素，不带任何参数一般是用来指定触发一个事件，用的比较少

<div id="test">点击触发<div>

$("ele").mousemove(function(){

alert('触发指定事件')

})

$("#test").click(function(){

$("ele").mousemove() //指定触发事件

});

**方法二：$ele.mousemove( handler(eventObject) )**

绑定$ele元素，每次$ele元素触发点击操作会执行回调 handler函数

这样可以针对事件的反馈做很多操作了

<div id="test">滑动触发<div>

$("#test").mousemove(function() {

//this指向 div元素

});

**方法三：$ele.mousemove( [eventData ], handler(eventObject) )**

使用与方法二一致，不过可以接受一个数据参数，这样的处理是为了解决不同作用域下数据传递的问题

<div id="test">点击触发<div>

$("#test").mousemove(11111,function(e) {

//this指向 div元素

//e.data => 11111 传递数据

});

**mousemove事件触发需要以下几点：**

* mousemove事件是当鼠标指针移动时触发的，即使是一个像素
* 如果处理器做任何重大的处理，或者如果该事件存在多个处理函数，这可能造成浏览器的严重的性能问题

## jQuery鼠标事件之mouseover与mouseout事件

在学JS的时候，大家还记得有两个方法叫移入移出事件吗？onmouseover()与onmouseout()事件~

jQuery当中同样提供了这样的事件来监听用户的移入移出操作，mouseover()与mouseout()事件，两者用法类似，下面一mouseover为例：

**方法一：$ele.mouseover()**

绑定$ele元素，不带任何参数一般是用来指定触发一个事件，用的比较少

<div id="test">点击触发<div>

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

alert('触发指定事件')

})

$("#test").click(function(){

$("ele").mouseover() //指定触发事件

});

**方法二：$ele.mouseover( handler(eventObject) )**

绑定$ele元素，每次$ele元素触发点击操作会执行回调 handler函数

这样可以针对事件的反馈做很多操作了

<div id="test">滑动触发<div>

$("#test").mouseover(function() {

//this指向 div元素

});

**方法三：$ele.mouseover( [eventData ], handler(eventObject) )**

使用与方法二一致，不过可以接受一个数据参数，这样的处理是为了解决不同作用域下数据传递的问题

<div id="test">点击触发<div>

$("#test").mouseover(11111,function(e) {

//this指向 div元素

//e.data => 11111 传递数据

});