# Event对象详解：

event代表事件的状态，例如触发event对象的元素、鼠标的位置及状态、按下的键等等。

event对象只在事件发生的过程中才有效。

event的某些属性只对特定的事件有意义。比如，fromElement 和 toElement 属性只对 onmouseover 和 onmouseout 事件有意义。

例子

下面的例子检查鼠标是否在链接上单击，并且，如果shift键被按下，就取消链接的跳转。

<HTML>  
<HEAD><TITLE>Cancels Links</TITLE>  
<SCRIPT LANGUAGE="JScript">  
function cancelLink() {  
if (window.event.srcElement.tagName == "A" && window.event.shiftKey)   
window.event.returnValue = false;  
}  
</SCRIPT>  
<BODY onclick="cancelLink()">

下面的例子在状态栏上显示鼠标的当前位置。

<BODY onmousemove="window.status = 'X=' + window.event.x + ' Y=' + window.event.y">

属性：

altKey, button, cancelBubble, clientX, clientY, ctrlKey, fromElement, keyCode, offsetX, offsetY, propertyName, returnValue, screenX, screenY, shiftKey, srcElement, srcFilter, toElement, type, x, y

--------------------------------------------------------------------------------

1.altKey  
描述：  
检查alt键的状态。

语法：  
event.altKey

可能的值：  
当alt键按下时，值为 TRUE ，否则为 FALSE 。只读。

2.button   
描述：  
检查按下的鼠标键。

语法：  
event.button

可能的值：   
0 没按键   
1 按左键   
2 按右键   
3 按左右键   
4 按中间键   
5 按左键和中间键   
6 按右键和中间键   
7 按所有的键

这个属性仅用于onmousedown, onmouseup, 和 onmousemove 事件。对其他事件，不管鼠标状态如何，都返回 0（比如onclick）。

3.cancelBubble  
描述：  
检测是否接受上层元素的事件的控制。

语法：  
event.cancelBubble[ = cancelBubble]

可能的值：  
这是一个可读写的布尔值:

TRUE 不被上层原素的事件控制。   
FALSE 允许被上层元素的事件控制。这是默认值。

例子：  
下面的代码片断演示了当在图片上点击（onclick）时，如果同时shift键也被按下，就取消上层元素（body）上的事件onclick所引发的showSrc()函数。

<SCRIPT LANGUAGE="JScript">  
function checkCancel() {  
if (window.event.shiftKey)  
window.event.cancelBubble = true;  
}  
function showSrc() {  
if (window.event.srcElement.tagName == "IMG")  
alert(window.event.srcElement.src);  
}  
</SCRIPT>  
<BODY onclick="showSrc()">  
<IMG onclick="checkCancel()" SRC="sample.gif">

4.clientX  
描述：  
返回鼠标在窗口客户区域中的X坐标。

语法：  
event.clientX

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

5.clientY  
描述：  
返回鼠标在窗口客户区域中的Y坐标。

语法：  
event.clientY

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

6.ctrlKey  
描述：  
检查ctrl键的状态。

语法：  
event.ctrlKey

可能的值：  
当ctrl键按下时，值为 TRUE ，否则为 FALSE 。只读。

7.fromElement  
描述：  
检测 onmouseover 和 onmouseout 事件发生时，鼠标所离开的元素。 参考：18.toElement

语法：  
event.fromElement

注释：  
这是个只读属性。

8.keyCode  
描述：  
（请参考 <http://www.joyist.com/forum/showtopic.asp?TOPIC_ID=31&Forum_ID=2>）   
检测键盘事件相对应的内码。  
这个属性用于 onkeydown, onkeyup, 和 onkeypress 事件。

语法：  
event.keyCode[ = keyCode]

可能的值：  
这是个可读写的值，可以是任何一个Unicode键盘内码。如果没有引发键盘事件，则该值为 0 。

9.offsetX   
描述：  
检查相对于触发事件的对象，鼠标位置的水平坐标

语法：  
event.offsetX

10.offsetY   
描述：  
检查相对于触发事件的对象，鼠标位置的垂直坐标

语法：  
event.offsetY

11.propertyName  
描述：  
设置或返回元素的变化了的属性的名称。

语法：  
event.propertyName [ = sProperty ]

可能的值：  
sProperty 是一个字符串，指定或返回触发事件的元素在事件中变化了的属性的名称。   
这个属性是可读写的。无默认值。

注释：  
你可以通过使用 onpropertychange 事件，得到 propertyName 的值。

例子：  
下面的例子通过使用 onpropertychange 事件，弹出一个对话框，显示 propertyName 的值。

<HEAD>  
<SCRIPT>  
function changeProp()  
{  
btnProp.value = "This is the new VALUE";  
}

function changeCSSProp()  
{  
btnStyleProp.style.backgroundColor = "aqua";  
}  
</SCRIPT>  
</HEAD>  
<BODY>  
<P>The event object property propertyName is   
used here to return which property has been   
altered.</P>

<INPUT TYPE=button ID=btnProp onclick="changeProp()"  
VALUE="Click to change the VALUE property of this button"  
onpropertychange='alert(event.propertyName+" property has changed value")'>  
<INPUT TYPE=button ID=btnStyleProp  
onclick="changeCSSProp()"  
VALUE="Click to change the CSS backgroundColor property of this button"  
onpropertychange='alert(event.propertyName+" property has changed value")'>  
</BODY>

12.returnValue  
描述：  
设置或检查从事件中返回的值

语法：  
event.returnValue[ = Boolean]

可能的值：   
true 事件中的值被返回   
false 源对象上事件的默认操作被取消

例子见本文的开头。

13.screenX   
描述：  
检测鼠标相对于用户屏幕的水平位置

语法：  
event.screenX

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

14.screenY   
描述：  
检测鼠标相对于用户屏幕的垂直位置

语法：  
event.screenY

注释：  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

15.shiftKey  
描述：  
检查shift键的状态。

语法：  
event.shiftKey

可能的值：  
当shift键按下时，值为 TRUE ，否则为 FALSE 。只读。

16.srcElement  
描述：  
返回触发事件的元素。只读。例子见本文开头。

语法：  
event.srcElement

17.srcFilter   
描述：  
返回触发 onfilterchange 事件的滤镜。只读。

语法：  
event.srcFilter

18.toElement  
描述：  
检测 onmouseover 和 onmouseout 事件发生时，鼠标所进入的元素。 参考：7.fromElement

语法：  
event.toElement

注释：  
这是个只读属性。

例子：下面的代码演示了当鼠标移到按钮上时，弹出一个对话框，显示“mouse arrived”

<SCRIPT>  
function testMouse(oObject) {  
if(oObject.contains(event.toElement)) {  
alert("mouse arrived");  
}  
}  
</SCRIPT>  
:  
<BUTTON ID=oButton onmouseover="testMouse(this)">Mouse Over This.</BUTTON>

19.type  
描述：  
返回事件名。

语法：  
event.type

注释：  
返回没有“on”作为前缀的事件名，比如，onclick事件返回的type是click  
只读。

20. x  
描述：  
返回鼠标相对于css属性中有position属性的上级元素的x轴坐标。如果没有css属性中有position属性的上级元素，默认以BODY元素作为参考对象。

语法：  
event.x

注释：  
如果事件触发后，鼠标移出窗口外，则返回的值为 -1  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

21. y  
描述：  
返回鼠标相对于css属性中有position属性的上级元素的y轴坐标。如果没有css属性中有position属性的上级元素，默认以BODY元素作为参考对象。

语法：  
event.y

注释：  
如果事件触发后，鼠标移出窗口外，则返回的值为 -1  
这是个只读属性。这意味着，你只能通过它来得到鼠标的当前位置，却不能用它来更改鼠标的位置。

# 浏览器差异问题：

<script>

function test(event){

var evt = window.event||event;

var src = evt.srcElement||evt.target;

// var code = evt.keyCode||evt.charCode;

alert(src.value);

}

</script>

</head>

<body>

<input type=button value=测试 onclick=test(event) />

</body>

# javascript键盘表：

Button: keyCode: NumLock: CapsLock: ScrollLock:

[BackSpace] 8 - - -

[Tab] 9 - - -

[NumPad 5] 12 no - -

[Return / Enter] 13 - - -

[Shift] 16 - - -

[Ctrl] 17 - - -

[Alt] 18 - - -

[Pause / Break] 19 - - -

[CapsLock] 20 - - -

[Esc] 27 - - -

[Space] 32 - - -

[Page Up] 33 - - -

[Page Down] 34 - - -

[End] 35 - - -

[Home] 36 - - -

[LeftArrow] 37 - - -

[UpArrow] 38 - - -

[RightArrow] 39 - - -

[DownArrow] 40 - - -

[PrintScreen / SysRq] 44 - - -

[Insert] 45 - - -

[Delete] 46 - - -

[0] 48 - - -

[1] 49 - - -

[2] 50 - - -

[3] 51 - - -

[4] 52 - - -

[5] 53 - - -

[6] 54 - - -

[7] 55 - - -

[8] 56 - - -

[9] 57 - - -

[A] 65 - - -

[B] 66 - - -

[C] 67 - - -

[D] 68 - - -

[E] 69 - - -

[F] 70 - - -

[G] 71 - - -

[H] 72 - - -

[I] 73 - - -

[J] 74 - - -

[K] 75 - - -

[L] 76 - - -

[M] 77 - - -

[N] 78 - - -

[O] 79 - - -

[P] 80 - - -

[Q] 81 - - -

[R] 82 - - -

[S] 83 - - -

[T] 84 - - -

[U] 85 - - -

[V] 86 - - -

[W] 87 - - -

[X] 88 - - -

[Y] 89 - - -

[Z] 90 - - -

[Turn Off] (special) 94 - - -

[Sleep] (special) 95 - - -

[NumPad 0] 96 yes - -

[NumPad 1] 97 yes - -

[NumPad 2] 98 yes - -

[NumPad 3] 99 yes - -

[NumPad 4] 100 yes - -

[NumPad 5] 101 yes - -

[NumPad 6] 102 yes - -

[NumPad 7] 103 yes - -

[NumPad 8] 104 yes - -

[NumPad 9] 105 yes - -

[NumPad \*] 106 - - -

[NumPad +] 107 - - -

[NumPad -] 109 - - -

[NumPad ,] 110 - - -

[NumPad /] 111 - - -

[F1] 112 - - -

[F2] 113 - - -

[F3] 114 - - -

[F4] 115 - - -

[F5] 116 - - -

[F6] 117 - - -

[F7] 118 - - -

[F8] 119 - - -

[F9] 120 - - -

[F10] 121 - - -

[F11] 122 - - -

[F12] 123 - - -

[NumLock] 144 - - -

[ScrollLock] 145 - - -

[+] (next to [0]) 187 - - -

[,] 188 - - -

[-] (next to [.]) 189 - - -

[.] 190 - - -

[Æ] 192 - - -

[´] (next to [+]) 219 - - -

[½] (next to [1]) 220 - - -

[Å] 221 - - -

[Ø] 222 - - -

[<] (next to [Z]) 226 - - -

[Wake Up] (special) 255 - - -