Dom事件

事件流描述的是从页面中接收事件的顺序。事件冒泡：事件开始时由最具体的元素(文档中嵌套层次最深的那个节点) 接收，然后逐级向上传播，直至文档节点。事件捕获，过程相反。

“DOM2”级事件流的三个阶段：事件捕获阶段、处于目标阶段、事件冒泡阶段

Document

Element html

Element body

Element div

1. 事件处理程序

通过节点属性显示声明

<button id=”btn” onclick = “myFunc()”>点击</button>

通过节点属性动态绑定

var btn = document.getElementById(“btn”);

btn.onclick = myFunc; function myFunc(){ }

该方式不同绑定多个click事件，以最后一个处理函数执行。

通过事件监听方式绑定

btn.addEventListenner(‘click’，func1，false)；

btn.addEventListenner(‘click’，func2，false)；

该方式添加事件处理程序，可绑定多个相同事件，执行顺序与绑定顺序有关。

移除事件处理程序

btn.onclick = null;

btn.removeEventListner(‘click’,myFunc，false);

1. Event重要属性和方法

键盘相关状态：altKey(alt键是否按下?)、ctrlKey、shiftKey、metaKey、keyCodewhich(通过键盘码判断具体哪个键被按下)

鼠标相关状态：button(左、中、右)、clientX/clientY/screen/screenY/offsetX/offseY/x/y (获取鼠标具体位置)

其他标准事件：type、target、timeStamp、bubbles、cancelable、currentTarget、preventDefault()、stopPropagation()

【兼容性】

var EventUtil = {

addHandler: function(element,type,handler){

if(element.addEventListener){

element.addEventListener(type,handler,false);

}else if(element.attachEvent){

element.attachEvent("on"+type,handler);

}else{

element["on"+type] = handler;

}

},

removeHandler: function(element,type,handler){

if(element.removeEventListener){

element.removeEventListener(type,handler,false);

}else if(element.detachEvent){

element.detachEvent("on"+type,handler);

}else{

element["on"+type] = null;

}

},

getEvent: function(event){

return event ? event:window.event;

},

getType: function(event){

return event.type;

},

getTarget: function(event){ // 获取事件target对象

return event.target || event.srcElement;

},

preventDefault: function(event){ // 阻止事件默认行为

if(event.preventDefault){

event.preventDefault();

}else{

event.returnValue = false;

}

},

stopPropagation: function(event){ // 阻止事件冒泡行为

if(event.stopPropagation){

event.stopPropagation();

}else{

event.cancalBubble = true;

}

}

}

1. 事件分类

**鼠标事件：** click、dbclick、mousedown、mouseup、mouseover、mousemove、mouseout等；**键盘事件：**keydown、keyup、keypress等；**表单事件：**select、change、submit、focus、blur等；**媒体事件：**play、abort、cancel、playing、waiting等；**html5事件：**online、offline、message、popstate、dragdrop、contextmenu、pageshow和pagehide、hashchange[Ajax中保存变化前后的URL参数]等；**页面事件：** load、error、resize、scroll等； **触摸事件：** touchStart、touchmove、touchend、touchcancl等； **手势事件:** gesturestart、gesturechange、gestureend等；**设备事件：**orientationchange、MozOrientation、deviceorientation、devicemotion等

1. 坐标位置

客户区坐标位置： 事件发生时鼠标指针在视口中的水平坐标和垂直坐标(不包括页面滚动距离) [event.clientX, event.clientY]

页面坐标位置：事件发生时鼠标光标在页面中的位置。页面不滚动时，与客户端坐标相等。[pageX，pageY]

【兼容性】

EventUtil.addHander(div, ‘click’,function(event){

event = EventUtil.getEvent(event);

var pageX = event.pageX,

pageY = event.pageY;

if(pageX === undefined){

pageX = event.clientX + (document.documentElement.scrollLeft || document.body.scrollLeft);

}

if(pageY === undefined){

pageY = event.clientY + (document.documentElement.scrollTop || document.body.scrollTop);

}

});

屏幕坐标位置：事件发生时鼠标相对于整个电脑屏幕的坐标信息[event.screenX, event.screenY]

1. 事件委托

事件委托利用了事件冒泡，只指定一个事件处理程序，就可以管理某一类型的所有事件。

使用事件委托，只需要在DOM树种尽量最高的层次上添加一个事件处理程序。例如：

var list = document.getElementById("mylinks");

EventUtil.addHandler(list,'click',function(event){

event = EventUtil.getEvent(event);

var target = EventUtil.getTarget(event);

switch(target.id) {

case "doSomewhere":

break;

case "goSomewhere":

break;

case "hiSomewhere":

break;

}

});

1. 事件模拟

鼠标事件模拟：createEvent(“MouseEvents”)，初始化initMouseEvent(接收15个参数)

type、bubbles、cancelable、view、detail、screenX、screenY、clientX、clientY、ctrlKey、altKey、shiftKey、metaKey、button、relatedTarget(只存在于mouseover、mouseout)。

键盘事件模拟：createEvent(“KeyboardEvent”)，初始化initKeyEvent(接收8个参数)

type、bubbles、cancelable、view、key、location、modifiers、repeat【Firefox兼容性】

自定义DOM事件：createEvent(“CustomEvent”)，初始化initCustomEvent(接收4个参数) type、bubbles、cancelable、detail(任意值，保存在event对象的detail属性中)

IE事件模拟：

event = document.createEventObject() 添加对象信息event.button = 0(等…) 触发事件btn.fireEvent(“onclick”,event);

注意 添加：调用fireEvent()方法时，自动添加srcElement和type属性。

触发：在目标上调用fireEvent()方法。该方法接收两个参数，事件处理程序名称和event对象