[**JavaScript中为事件句柄绑定监听函数**](http://www.cnblogs.com/jenry/archive/2010/12/09/1901269.html)

在JavaScript中为Dom元素绑定事件监听函数是一件非常常见的事情，但这里也有许多的Bug。各种浏览器对于事件绑定都提供了很多方法，但可靠的只有3中：

1、传统的绑定方法：

elem.onclick = function( event ){  
 alert(event.type + 'this.innerHTML');  
 };

 　　a、传统的绑定方法，非常简单稳定，函数体内的this指向的也是指向正在处理事件的节点（如当前正在运行事件句柄的节点）。

　　b、一个元素的一个事件句柄只能注册一个函数，如果重复注册，会产生覆盖；而且，传统绑定方法只会在事件冒泡中运行。

2、W3C标准绑定方法：

var elem = document.getElementById('ID');  
 elem.addEventListener('click' ,  
 function( event ){   
 alert(event.type + ' ' + this.innerHTML + 1);   
 } ,   
 false //冒泡阶段执行  
 );  
 elem.addEventListener('click' ,  
 function( event ){   
 alert(event.type + ' ' + this.innerHTML + 2);   
 } ,  
 false  
 );

　　 a、这种绑定方法同时支持时间处理的捕获和冒泡两个阶段；同一元素的同一事件句柄可以注册多个监听函数；而且，监听函数内部this指向当前元素。

　　 b、但是流行的IE浏览器不支持这种注册方法。

3、IE事件句柄注册方法：

var elem = document.getElementById('a');  
 elem.attachEvent('onclick' ,  
 function(){   
 alert(window.event.srcElement.innerHTML + ' ' + this.innerHTML + 1);   
 }   
 );  
 elem.attachEvent('onclick' ,  
 function(){   
 alert(window.event.srcElement.innerHTML + ' ' + this.innerHTML + 2);   
 }  
 );

　 a、这种绑定方法，可以为同一事件句柄注册多次。

　 b、IE的事件模型不支持事件捕获；监听函数体内的this指向的不是当前于元素，而且window.event.srcElement指向的是发生事件的节点，而不是当前节点，并且在IE的事件对象中也没有等价的DOM currentTarget属性。

4、跨浏览器的方法一：

function addEvent(element, type, handler) {  
 if (!handler.$$guid) handler.$$guid = addEvent.guid++;  
 if (!element.events) element.events = {};  
 var handlers = element.events[type];  
 if (!handlers) {  
 handlers = element.events[type] = {};  
 if (element["on" + type]) {  
 handlers[0] = element["on" + type];  
 }  
 }  
 handlers[handler.$$guid] = handler;  
 element["on" + type] = handleEvent;  
 };  
 addEvent.guid = 1;  
   
 function removeEvent(element, type, handler) {  
 if (element.events && element.events[type]) {  
 delete element.events[type][handler.$$guid];  
 }  
 };  
 function handleEvent(event) {  
 var returnValue = true;  
 event = event || fixEvent(window.event);  
 var handlers = this.events[event.type];  
 for (var i in handlers) {  
 this.$$handleEvent = handlers[i];  
 if (this.$$handleEvent(event) === false) {  
 returnValue = false;  
 }  
 }  
 return returnValue;  
 };  
   
 function fixEvent(event) {  
 event.preventDefault = fixEvent.preventDefault;  
 event.stopPropagation = fixEvent.stopPropagation;  
 return event;  
 };  
 fixEvent.preventDefault = function() {  
 this.returnValue = false;  
 };  
 fixEvent.stopPropagation = function() {  
 this.cancelBubble = true;  
 };

 以上代码由是有 [Dean Edwards](http://dean.edwards.name/)[addEvent/removeEven](http://dean.edwards.name/weblog/2005/10/add-event2/)

5、跨浏览器的方法二：

function addEvent( obj, type, fn ) {  
 if ( obj.attachEvent ) {  
 obj['e'+type+fn] = fn;  
 obj[type+fn] = function(){obj['e'+type+fn]( window.event );}  
 obj.attachEvent( 'on'+type, obj[type+fn] );  
 } else  
 obj.addEventListener( type, fn, false );  
 }  
 function removeEvent( obj, type, fn ) {  
 if ( obj.detachEvent ) {  
 obj.detachEvent( 'on'+type, obj[type+fn] );  
 obj[type+fn] = null;  
 } else  
 obj.removeEventListener( type, fn, false );  
 }