**封装库--事件绑定[上]**

学习要点：

1. 问题所在
2. 设置代码

本节课，我们主要探讨一下事件绑定，在此之前我们都使用的是传统的事件绑定。这节点，我们想使用现代绑定对事件进行绑定和删除。

1. **问题所在**

现代绑定中W3C使用的是：

addEventListener和removeEventListener。

IE使用的是attachEvent和detachEvent。

我们知道IE的这两个问题多多，并且伴随内存泄漏。所以，解决这些问题非常有必要。

那么我们希望解决非IE浏览器事件绑定哪些问题呢？

1. 支持同一元素的同一事件句柄可以绑定多个监听函数；
2. 如果在同一元素的同一事件句柄上多次注册同一函数，那么第一次注册后的所有注册都被忽略；
3. 函数体内的this指向的应当是正在处理事件的节点（如当前正在运行事件句柄的节点）；
4. 监听函数的执行顺序应当是按照绑定的顺序执行；
5. 在函数体内不用使用 event = event || window.event; 来标准化Event对象；
6. **设置代码**

//跨浏览器添加事件

function addEvent(obj, type, fn) {

if (typeof addEventListener != 'undefined') {

obj.addEventListener(type, fn, false);

} else if (typeof attachEvent != 'undefined') {

obj.attachEvent('on' + type, fn);

}

}

//跨浏览器删除事件

function removeEvent(obj, type, fn) {

if (typeof removeEventListener != 'undefined') {

obj.removeEventListener(type, fn);

} else if (typeof detachEvent != 'undefined') {

obj.detachEvent('on' + type, fn);

}

}

上面的这两个函数解决了：

1. 同时绑定多个函数；
2. 2.标准event；

上面的这两个函数没有解决的问题：

1. IE多次注册同一函数未被忽略；
2. 2.IE中顺序是倒序；
3. 3.IE中this传递过来的是window

为了解决this传递问题，我们需要使用匿名函数+传递方式参数的方式来解决：

obj.attachEvent('on' + type, function () {

fn(obj);

});

addEvent(oButton, 'click', function (\_this) {

alert(\_this.value);

});

这种方式比较古板，更好一点的方式是使用call来冒充对象。

obj.attachEvent('on' + type, function () {

fn.call(obj);

});

addEvent(oButton, 'click', function () {

alert(this.value);

});

call的用法回忆一下：

fn.call(obj); //this就是obj对象

fn.call(123); //this就是123

fn.call(123,456); //this就是123，第一个参数是456

PS：也就是说，使用了call第一个参数就是this获取，从第2个参数开始，可以通过函数参数获取，以此类推。

使用了call传递this，带来的诸多另外的问题：

1.无法标准化event；

2.无法删除事件。

导致的原因很明确，就是使用了匿名函数。标准化event可以解决，无法删除事件就没有办法了，因为无法确定是哪一个事件。

obj.attachEvent('on' + type, function () {

fn.call(obj, window.event);

});

那么最终有几个问题无法解决：

1. 无法删除事件；
2. 2.无法顺序执行；
3. 3.IE的现代事件绑定存在内存泄漏问题。