**封装库--事件绑定[下]**

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**

上一节课，我们用现代事件绑定封装了事件触发和删除，但还有几个问题没有得到解决：1.无法删除事件；2.无法顺序执行；3.IE的现代事件绑定存在内存泄漏问题。

我们这节课将尝试着通过使用传统事件绑定对IE进行封装。

1. **设置代码**

//跨浏览器添加事件绑定

function addEvent(obj, type, fn) {

if (typeof obj.addEventListener != 'undefined') {

obj.addEventListener(type, fn, false);

} else {

//创建事件类型的散列表(哈希表)

if (!obj.events) obj.events = {};

//创建存放事件处理函数的数组

if (!obj.events[type]) {

obj.events[type] = [];

//存储第一个事件处理函数

if (obj['on' + type]) {

obj.events[type][0] = fn;

}

//执行事件处理

obj['on' + type] = addEvent.exec;

} else {

//同一个注册函数取消计数

if (addEvent.array(fn,obj.events[type])) return false;

}

//从第二个开始，通过计数器存储

obj.events[type][addEvent.ID++] = fn;

}

}

addEvent.array = function (fn, es){

for (var i in es) {

if (es[i] == fn) return true;

}

return false;

}

//每个事件处理函数的ID计数器

addEvent.ID = 1;

//事件处理函数调用

addEvent.exec = function (event) {

var e = event || addEvent.fixEvent(window.event);

var es = this.events[e.type];

for (var i in es) {

es[i].call(this, e);

}

};

//获取IE的event，兼容W3C的调用

addEvent.fixEvent = function (event) {

event.preventDefault = addEvent.fixEvent.preventDefault;

event.stopPropagation = addEvent.fixEvent.stopPropagation;

return event;

};

//兼容IE和W3C阻止默认行为

addEvent.fixEvent.preventDefault = function () {

this.returnValue = false;

};

//兼容IE和W3C取消冒泡

addEvent.fixEvent.stopPropagation = function () {

this.cancelBubble = true;

};

//跨浏览器删除事件

function removeEvent(obj, type, fn) {

if (typeof obj.removeEventListener != 'undefined') {

obj.removeEventListener(type, fn, false);

} else {

var es = obj.events[type];

for (var i in es) {

if (es[i] == fn) {

delete obj.events[type][i];

}

}

}

}