自定义事件

# 问题来源

JS经常在一个操作（可能是异步操作，也可能不是）执行完成之后才能继续处理某些代码，或者是在满足一定条件的时候调用很多的组建，这个时候我们通常有两种解决办法。

第一种：采用回调的方式来实现这样的功能，这样的实现有个缺陷就是，当嵌套的时候，回调函数会一层套一层，很难应用到JS面向对象和模块化的编程思想。

第二种：使用promiss对象来实现（await/synch本质上也是promiss）。

这两种处理方式都可以实现以上的功能，我想说的是可以采用另外一种编程方式实现，即自定义事件。

# 自定义事件原理

自定义事件采用观察者设计模式，当某个状态发送变化时会通知所有事件处理者来处理这个变化的事件，自定义事件对象需要向外提供事件注册的接口，并且在其内部满足什么条件的时候应该触发这些事件。

源代码：

function costomEvent(){

this.handlers = {};

},

costomEvent.prototype = {

constructor:costomEvent;

addEvent: function(type, handler){

if(typeof handlers[type] === 'undefined'){

this.handlers[type] = [];

}

this.handlers[type].push(handler);

}

removeEvent: function(type, handler){

if(this.handlers[type] instanceof Array){

$.each(this.handlers[type], function(i){

if(this === handler){

this.handlers[type].splice(i,1);

return false;

}

});

}

}

fire: function(event){

if(!event.target){

event.target = this;

}

if(this.handlers[event.type] instanceof Array){

$.each(this.handlers[event.type],function(){

this(event);

})

}

}

}

# 自定义事件优缺点

自定义事件会使编程变得更加简洁明良，让各模块之间松散耦合，还能传递信息，减少回调函数的嵌套，但是自定义事件不好定位问题，容易产生意想不到的错误。

# 附件：

(function(window,$){

var muphy = Object.create({

isNull: function(data){

if(data === null || data === undefined){

return true;

}

if( typeof data === "String" && data.trim() ==='' ){

return true;

}

if(data instanceof Array && data.length == 0){

return true;

}

return false;

},

nvl: function(data,obj){

if(obj === 0) return 0;

return data || obj || '';

},

each: function(obj,fun){

for (var key in obj) {

if(fun.call(obj[key], key, obj[key]) === false){

break;

}

}

},

addEvent: function(ele,type,handler){

if(ele.addEventListener){

ele.addEventListener(type,handler,false);

} else if(ele.attachEvent){

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

} else {

ele["on" + type] = handler;

}

},

removeEvent: function(ele,type,handler){

if(ele.removeEventListener){

ele.removeEventListener(type,handler,false);

} else if(ele.detachEvent){

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

} else {

ele["on" + type] = null;

}

},

\_dragEvent: function(ele){

var drag = null,

diffx = 0,

diffy = 0,

$e = null;

if($.isNull(ele)) return;

if(!($e = $(ele))) return;

$(document).bind("mousedown",handleEvent);

$(document).bind("mousemove",handleEvent);

//$(document).bind("mouseup",handleEvent);

$.muphy.addEevent(document,"mouseup",handleEvent);

function handleEvent(event){

event = event || window.event;

var target = $e[0];

switch(event.type){

case "mousedown":

if($m.nvl(target.className).indexOf("draggable") > -1){

drag = target;

diffx = event.clientX - target.offsetLeft;

diffy = event.clientY - target.offsetTop;

}

break;

case "mousemove":

if(drag !== null){

drag.style.left = (event.clientX - diffx) + "px";

drag.style.top = (event.clientY - diffy) + "px";

}

break;

case "mouseup":

drag = null;

break;

}

}

},

dragEvent: function(){

var drag = null,

diffx = 0,

diffy = 0;

$(document).unbind("mousedown",handleEvent);

$(document).unbind("mousemove",handleEvent);

$(document).unbind("mouseup",handleEvent);

$(document).bind("mousedown",handleEvent);

$(document).bind("mousemove",handleEvent);

$(document).bind("mouseup",handleEvent);

function handleEvent(event){

event = event || window.event;

var target = event.target || event.srcElement;

switch(event.type){

case "mousedown":

if($m.nvl(target.className).indexOf("draggable") > -1){

drag = target;

diffx = event.clientX - target.offsetLeft;

diffy = event.clientY - target.offsetTop;

}

break;

case "mousemove":

if(drag !== null){

drag.style.left = (event.clientX - diffx) + "px";

drag.style.top = (event.clientY - diffy) + "px";

}

break;

case "mouseup":

drag = null;

break;

}

}

},

\_costomEvent: function(){

var handlers = {};

this.addEvent = function(type, handler){

if(typeof handlers[type] === 'undefined'){

handlers[type] = [];

}

handlers[type].push(handler);

}

this.removeEvent = function(type, handler){

if(handlers[type] instanceof Array){

muphy.each(handlers[type], function(i){

if(this === handler){

handlers[type].splice(i,1);

return false;

}

});

}

}

this.fire = function(event){

if(!event.target){

event.target = this;

}

if(handlers[event.type] instanceof Array){

muphy.each(handlers[event.type],function(){

this(event);

})

}

}

},

costomEvent: function(){}

});

(function(muphy){

muphy.costomEvent1.prototype = {

constructor: muphy.costomEvent1,

handlers:{},

addEvent: function(type, handler){

if(typeof this.handlers[type] === 'undefined'){

this.handlers[type] = [];

}

this.handlers[type].push(handler);

},

removeEvent: function(type, handler){

if(handlers[type] instanceof Array){

muphy.each(this.handlers[type], function(i){

if(this === handlers){

this.handlers[type].splice(i,1);

return false;

}

});

}

},

fire: function(event){

if(!event.target){

event.target = this;

}

if(this.handlers[event.type] instanceof Array){

muphy.each(this.handlers[event.type],function(){

this(event);

})

}

}

}

})(muphy);

window.$m = muphy;

})(window,jQuery);