高级事件

学习要点：

1.模拟操作

2.命名空间

3.事件委托

4.on、off和 one

jQuery不但封装了大量常用的事件处理，还提供了不少高级事件方便开发者使用。比

如模拟用户触发事件、事件委托事件、和统一整合的 on和 off，以及仅执行一次的 one方

法。这些方法大大降低了开发者难度，提升了开发者的开发体验。

一．模拟操作

在事件触发的时候，有时我们需要一些模拟用户行为的操作。例如：当网页加载完毕后

自行点击一个按钮触发一个事件，而不是用户去点击。

//点击按钮事件

$('input').click(function () {

alert('我的第一次点击来自模拟！');

});

//模拟用户点击行为

$('input').trigger('click');

//可以合并两个方法

$('input').click(function () {

alert('我的第一次点击来自模拟！');

}).trigger('click');

有时在模拟用户行为的时候，我们需要给事件执行传递参数，这个参数类似与 event.data

的额外数据，可以可以是数字、字符串、数组、对象。

$('input').click(function (e, data1, data2) {

alert(data1 + ',' + data2);

}).trigger('click', ['abc', '123']);

注意：当传递一个值的时候，直接传递即可。当两个值以上，需要在前后用中括号包含

起来。但不能认为是数组形式，下面给出一个复杂的说明。

$('input').click(function (e, data1, data2) {

alert(data1.a + ',' + data2[1]);

}).trigger('click', [{'a' : '1', 'b' : '2'}, ['123','456']]);

除了通过 JavaScript事件名触发，也可以通过自定义的事件触发，所谓自定义事件其实

就是一个被.bind()绑定的任意函数。

$('input').bind('myEvent', function () {

alert('自定义事件！');

}).trigger('myEvent');

.trigger()方法提供了简写方案，只要想让某个事件执行模拟用户行为，直接再调用一个

空的同名事件即可。

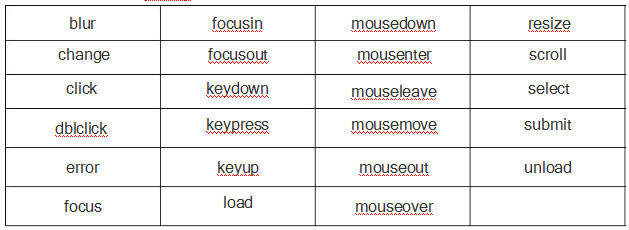
$('input').click(function () {

alert('我的第一次点击来自模拟！');

}).click();

//空的 click()执行的是 trigger()

这种便捷的方法，jQuery几乎个所有常用的事件都提供了。



jQuery还提供了另外一个模拟用户行为的方法：.triggerHandler()；这个方法的使用

和.trigger()方法一样。

$('input').click(function () {

alert('我的第一次点击来自模拟！');

}).triggerHandler('click');

在常规的使用情况下，两者几乎没有区别，都是模拟用户行为，也可以可以传递额外参

数。但在某些特殊情况下，就产生了差异：

1..triggerHandler()方法并不会触发事件的默认行为，而.trigger()会。

$('form').trigger('submit');//模拟用户执行提交，并跳转到执行页面

$('form').triggerHandler('submit');//模拟用户执行提交，并阻止的默认行为

如果我们希望使用.trigger()来模拟用户提交，并且阻止事件的默认行为，则需要这么写：

$('form').submit(function (e) {

e.preventDefault();//阻止默认行为

}).trigger('submit');

1. .triggerHandler()方法只会影响第一个匹配到的元素，而.trigger()会影响所有。

3..triggerHandler()方法会返回当前事件执行的返回值，如果没有返回值，则返回

undefined；而.trigger()则返回当前包含事件触发元素的 jQuery对象(方便链式连缀调用)。

alert($('input').click(function () {

return 123;

}).triggerHandler('click'));//返回 123，没有 return返回

4..trigger()在创建事件的时候，会冒泡。但这种冒泡是自定义事件才能体现出来，是

jQuery扩展于 DOM的机制，并非 DOM特性。而.triggerHandler()不会冒泡。

var index = 1;

$('div').bind('myEvent',function(){

alert('自定义事件' + index);

index++;

});

$('.div3').trigger("myEvent");

二．命名空间

有时，我们想对事件进行移除。但对于同名同元素绑定的事件移除往往比较麻烦，这个

时候，可以使用事件的命名空间解决。

$('input').bind('click.abc', function () {

alert('abc');

});

$('input').bind('click.xyz', function () {

alert('xyz');

});

$('input').unbind('click.abc');//移除 click实践中命名空间为 abc的

注意：也可以直接使用('.abc')，这样的话，可以移除相同命名空间的不同事件。对于模

拟操作.trigger()和.triggerHandler()，用法也是一样的。

$('input').trigger('click.abc');

三．事件委托

什么是事件委托？用现实中的理解就是：有 100个学生同时在某天中午收到快递，但这

100个学生不可能同时站在学校门口等，那么都会委托门卫去收取，然后再逐个交给学生。

而在 jQuery中，我们通过事件冒泡的特性，让子元素绑定的事件冒泡到父元素(或祖先元素)

上，然后再进行相关处理即可。

如果一个企业级应用做报表处理，表格有 2000行，每一行都有一个按钮处理。如果用

之前的.bind()处理，那么就需要绑定 2000个事件，就好比 2000个学生同时站在学校门口等

快递，不断会堵塞路口，还会发生各种意外。这种情况放到页面上也是一样，可能导致页面

极度变慢或直接异常。而且，2000个按钮使用 ajax分页的话，.bind()方法无法动态绑定尚

未存在的元素。就好比，新转学的学生，快递员无法验证他的身份，就可能收不到快递。

//HTML部分

<div style="background:red;width:200px;height:200px;" id="box">

<input type="button" value="按钮" class="button" />

</div>

//使用.bind()不具备动态绑定功能，只有点击原始按钮才能生成

$('.button').bind('click', function () {

$(this).clone().appendTo('#box');

});

//使用.live()具备动态绑定功能，jQuery1.3使用，jQuery1.7之后废弃，jQuery1.9删除

$('.button').live('click', function () {

$(this).clone().appendTo('#box');

});

.live()原理就是把 click事件绑定到祖先元素$(document)上，而只需要给$(document)绑

定一次即可，而非 2000次。然后就可以处理后续动态加载的按钮的单击事件。在接受任何

事件时，$(document)对象都会检查事件类型(event.type)和事件目标(event.target)，如果 click

事件是.button，那么就执行委托给它的处理程序。.live()方法已经被删除，无法使用了。需

要测试使用的话，需要引入向下兼容插件。

//.live()无法使用链接连缀调用，因为参数的特性导致

$('#box').children(0).live('click', function () {

$(this).clone().appendTo('#box');

});

在上面的例子中，我们使用了.clone()克隆。其实如果想把事件行为复制过来，我们只

需要传递 true即可：.clone(true)。这样也能实现类似事件委托的功能，但原理却截然不同。

一个是复制事件行为，一个是事件委托。而在非克隆操作下，此类功能只能使用事件委托。

$('.button').live('click', function () {

$('<input type="button" value="复制的" class="button" />').appendTo('#box');

});

当我们需要停止事件委托的时候，可以使用.die()来取消掉。

$('.button').die('click');

由于.live()和.die()在 jQuery1.4.3版本中废弃了，之后推出语义清晰、减少冒泡传播层次、

又支持链接连缀调用方式的方法：.delegate()和.undelegate()。但这个方法在 jQuery1.7版本中

被.on()方法整合替代了。

$('#box').delegate('.button', 'click', function () {

$(this).clone().appendTo('#box');

});

$('#box').undelegate('.button','click');

//支持连缀调用方式

$('div').first().delegate('.button', 'click', function () {

$(this).clone().appendTo('div:first');

});

注意：.delegate()需要指定父元素，然后第一个参数是当前元素，第二个参数是事件方

式，第三个参数是执行函数。和.bind()方法一样，可以传递额外参数。.undelegate()和.unbind()

方法一样可以直接删除所有事件，比如：.undelegate('click')。也可以删除命名空间的事件，

比如：.undelegate('click.abc')。

注意：.live()和.delegate()和.bind()方法一样都是事件绑定，那么区别也很明显，用途上

遵循两个规则：1.在 DOM中很多元素绑定相同事件时；2.在 DOM中尚不存在即将生成的

元素绑定事件时；我们推荐使用事件委托的绑定方式，否则推荐使用.bind()的普通绑定。

四．on、off和 one

目前绑定事件和解绑的方法有三组共六个。由于这三组的共存可能会造成一定的混乱，

为此 jQuery1.7以后推出了.on()和.off()方法彻底摒弃前面三组。

//替代.bind()方式

$('.button').on('click', function () {

alert('替代.bind()');

});

//替代.bind()方式，并使用额外数据和事件对象

$('.button').on('click', {user : 'Lee'}, function (e) {

alert('替代.bind()' + e.data.user);

});

//替代.bind()方式，并绑定多个事件

$('.button').on('mouseover mouseout', function () {

alert('替代.bind()移入移出！');

});

//替代.bind()方式，以对象模式绑定多个事件

$('.button').on({

mouseover : function () {

alert('替代.bind()移入！');

},

mouseout : function () {

alert('替代.bind()移出！');

}

});

//替代.bind()方式，阻止默认行为并取消冒泡

$('form').on('submit', function () {

return false;

});

或

$('form').on('submit', false);

//替代.bind()方式，阻止默认行为

$('form').on('submit', function (e) {

e.preventDefault();

});

//替代.bind()方式，取消冒泡

$('form').on('submit', function (e) {

e.stopPropagation();

});

//替代.unbind()方式，移除事件

$('.button').off('click');

$('.button').off('click', fn);

$('.button').off('click.abc');

//替代.live()和.delegate()，事件委托

$('#box').on('click', '.button', function () {

$(this).clone().appendTo('#box');

});

//替代.die()和.undelegate()，取消事件委托

$('#box').off('click', '.button');

注意：和之前方式一样，事件委托和取消事件委托也有各种搭配方式，比如额外数据、

命名空间等等，这里不在赘述。

不管是.bind()还是.on()，绑定事件后都不是自动移除事件的，需要通过.unbind()和.off()

来手工移除。jQuery提供了.one()方法，绑定元素执行完毕后自动移除事件，可以方法仅触

发一次的事件。

//类似于.bind()只触发一次

$('.button').one('click', function () {

alert('one仅触发一次！');

});

//类似于.delegate()只触发一次

$('#box).one('click', 'click', function () {

alert('one仅触发一次！');

});