第 **10** 章 高级事件

学习要点：

1

2

3

4

.模拟操作

.命名空间

.事件委托

.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。

blur

change

click

focusin

focusout

keydown

keypress

mousedown

mousenter

resize

scroll

select

submit

mouseleave

mousemove

dblclick

error

keyup

load

mouseout

unload

focus

mouseover

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

//模拟用户执行提交，并阻止的默认行为

('form').triggerHandler('submit');

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：

$

}

2

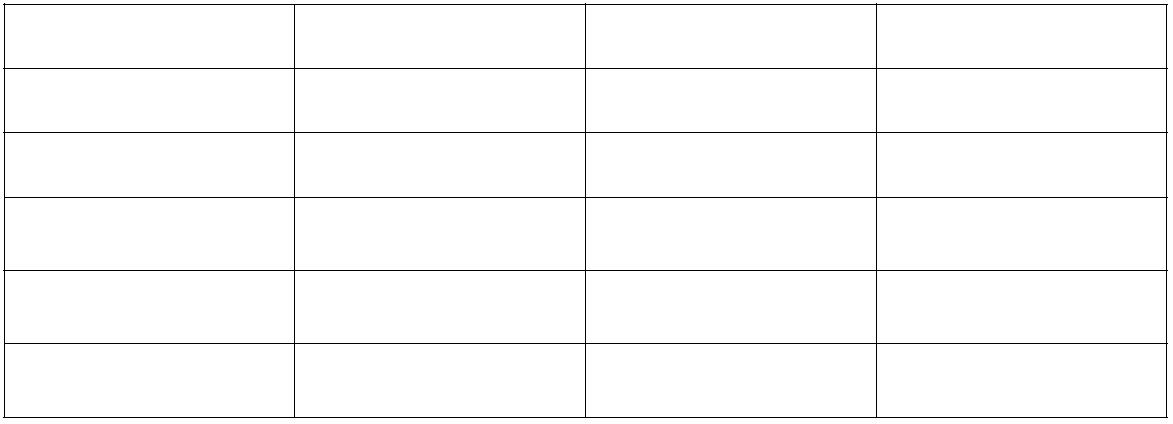
('form').submit(function (e) {

e.preventDefault();

//阻止默认行为

).trigger('submit');

..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

00 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。

1

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。