事件对象

学习要点：

1.事件对象

2.冒泡和默认行为

JavaScript在事件处理函数中默认传递了 event对象，也就是事件对象。但由于浏览器

的兼容性，开发者总是会做兼容方面的处理。jQuery在封装的时候，解决了这些问题，并且

还创建了一些非常好用的属性和方法。

一．事件对象

事件对象就是 event对象，通过处理函数默认传递接受。之前处理函数的 e就是 event

事件对象，event对象有很多可用的属性和方法，我们在 JavaScript课程中已经详细的了解

过这些常用的属性和方法，这里，我们再一次演示一下。

//通过处理函数传递事件对象

$('input').bind('click', function (e) {//接受事件对象参数

alert(e);

});



//通过 event.type属性获取触发事件名

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

alert(e.type);

});

//通过 event.target获取绑定的 DOM元素

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

alert(e.target);

});

//通过 event.data获取额外数据，可以是数字、字符串、数组、对象

$('input').bind('click', 123, function () {//传递 data数据

alert(e.data);//获取数字数据

});

注意：如果字符串就传递：'123'、如果是数组就传递：[123,'abc']，如果是对象就传递：

{user : 'Lee', age : 100}。数组的调用方式是：e.data[1]，对象的调用方式是：e.data.user。

//event.data获取额外数据，对于封装的简写事件也可以使用

$('input').click({user : 'Lee', age : 100},function (e) {

alert(e.data.user);

});

注意：键值对的键可以加上引号，也可以不加；在调用的时候也可以使用数组的方式：

alert(e.data['user']);

//获取移入到 div之前的那个 DOM元素

$('div').mouseover(function (e) {

alert(e.relatedTarget);

});

//获取移出 div之后到达最近的那个 DOM元素

$('div').mouseout(function (e) {

alert(e.relatedTarget);

});

//获取绑定的那个 DOM元素，相当于 this，区别与 event.target

$('div').click(function (e) {

alert(e.currentTarget);

});

注意：event.target得到的是触发元素的 DOM，event.currentTarget得到的是监听元素的

DOM。而 this也是得到监听元素的 DOM。

//获取上一次事件的返回值

$('div').click(function (e) {

return '123';

});

$('div').click(function (e) {

alert(e.result);

});

//获取当前的时间戳

$('div').click(function (e) {

alert(e.timeStamp);

});

//获取鼠标的左中右键

$('div').mousedown(function (e) {

alert(e.which);

});

//获取键盘的按键

$('input').keyup(function (e) {

alert(e.which);

});

//获取是否按下了 ctrl键，meta键不存在，导致无法使用

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

alert(e.ctrlKey);

});

//获取触发元素鼠标当前的位置

$(document).click(function (e) {

alert(e.screenY+ ',' + e.pageY + ',' + e.clientY);

});

二．冒泡和默认行为

如果在页面中重叠了多个元素，并且重叠的这些元素都绑定了同一个事件，那么就会出

现冒泡问题。

//HTML页面

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

<input type="button" value="按钮" />

</div>

//三个不同元素触发事件

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

alert('按钮被触发了！');

});

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

alert('div层被触发了！');

});

$(document).click(function () {

alert('文档页面被触发了！');

});

注意：当我们点击文档的时候，只触发文档事件；当我们点击 div层时，触发了 div和

文档两个；当我们点击按钮时，触发了按钮、div和文档。触发的顺序是从小范围到大范围。

这就是所谓的冒泡现象，一层一层往上。

jQuery提供了一个事件对象的方法：event.stopPropagation()；这个方法设置到需要触发

的事件上时，所有上层的冒泡行为都将被取消。

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

alert('按钮被触发了！');

e.stopPropagation();

});

网页中的元素，在操作的时候会有自己的默认行为。比如：右击文本框输入区域，会弹

出系统菜单、点击超链接会跳转到指定页面、点击提交按钮会提交数据。

$('a').click(function (e) {

e.preventDefault();

});

//禁止提交表单跳转

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

e.preventDefault();

});

注意：如果想让上面的超链接同时阻止默认行为且禁止冒泡行为，可以把两个方法同时

写上：event.stopPropagation()和 event.preventDefault()。这两个方法如果需要同时启用的时候，

还有一种简写方案代替，就是直接 return false。

$('a').click(function (e) {

return false;

});



//判断是否取消了元素的默认行为

$('input').keyup(function (e) {

e.preventDefault();

alert(e.isDefaultPrevented());

});

//取消冒泡并取消后续事件处理函数

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

alert('input');

e.stopImmediatePropagation();

});

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

alert('input2');

});

$(document).click(function () {

alert('document');

});

//判断是否调用了 stopPropagation()方法

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

e.stopPropagation();

alert(e.isPropagationStopped());

});

//判断是否执行了 stopImmediatePropagation()方法

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

e.stopImmediatePropagation();

alert(e.isImmediatePropagationStopped());

});