

事件传送可以分为3个阶段。

（1）.在事件捕捉（Capturing）阶段，事件将沿着DOM树向下转送，目标节点的每一个祖先节点，直至目标节点。例如，若用户单击了一个超 链接，则该单击事件将从document节点转送到html元素，body元素以及包含该链接的p元素。在此过程中，浏览器都会检测针对该事件的捕捉事件 监听器，并且运行这件事件监听器。

（2）在目标（target）阶段，浏览器在查找到已经指定给目标事件的事件监听器之后，就会运行 该事件监听器。目标节点就是触发事件的DOM节点。例如，如果用户单击一个超链接，那么该链接就是目标节点（此时的目标节点实际上是超链接内的文本节点）。

（3）.在冒泡（Bubbling）阶段，事件将沿着DOM树向上转送，再次逐个访问目标元素的祖先节点到document节点。该过程中的每一步。浏览器都将检测那些不是捕捉事件监听器的事件监听器，并执行它们。

利用事件传播（这里是冒泡）这个机制，就可以实现事件委托。

**具体来说，事件委托就是事件目标自身不处理事件，而是把处理任务委托给其父元素或者祖先元素，甚至根元素（document）**

jQuery的事件绑定有多个方法可以调用，以click事件来举例：

click方法

bind方法

delegate方法

on方法

不管你用的是（click / bind / delegate)之中那个方法，最终都是jQuery底层都是调用on方法来完成最终的事件绑定.

**.bind()**

.bind()方法用于直接附加一个事件处理程序到元素上。

处理程序附加到jQuery对象中当前选中的元素，所以，在.bind()绑定事件的时候，这些元素必须已经存在

**.live()**

将委托的事件处理程序附加到一个页面的document元素，从而简化了在页面上动态添加的内容上事件处理的使用。

因为更高版本的jQuery提供了更好的方法，没有.live()方法的缺点，所以.live()方法不再推荐使用

特别是，使用.live()出现的以下问题：

1、在调用 .live() 方法之前，jQuery 会先获取与指定的选择器匹配的元素，这一点对于大型文档来说是很花费时间的。

2、不支持链式写法。例如，$("a").find(".offsite, .external").live( ... ); 这样的写法是*不合法*的，并不能像期待的那样起作用。

3、由于所有的 .live() 事件被添加到 document 元素上，所以在事件被处理之前，可能会通过最长最慢的那条路径之后才能被触发。

4、在移动 iOS (iPhone, iPad 和 iPod Touch) 上，对于大多数元素而言，click 事件不会冒泡到文档 body 上，并且如果不满足如下情况之一，就不能和 .live() 方法一起使用：

* 使用原生的可被点击的元素，例如, a 或 button，因为这两个元素可以冒泡到 document。
* 在 document.body 内的元素使用 .on() 或 .delegate() 进行绑定，因为移动 iOS 只有在 body 内才能进行冒泡。
* 需要 click 冒泡到元素上才能应用的 CSS 样式 cursor:pointer (或者是父元素包含document.documentElement)。但是依需注意的是，这样会禁止元素上的复制/粘贴功能，并且当点击元素时，会导致该元素被高亮显示。

5、在事件处理中调用 [event.stopPropagation()](http://www.css88.com/jqapi-1.9/live/event.stopPropagation.htm) 来阻止事件处理被添加到 document 之后的节点中，是效率很低的。因为事件已经被传播到 document 上。

6、.live() 方法与其它事件方法的相互影响是会令人感到惊讶的。例如，$(document).unbind("click") 会移除所有通过 .live() 添加的 click 事件!

#### .delegate()

为了突破单一.bind()方法的局限性，实现事件委托，jQuery 1.3引入了.live()方法。后来，为解决“事件传播链”过长的问题，jQuery 1.4又支持为.live()方法指定上下文对象。而为了解决无谓生成元素集合的问题，jQuery 1.4.2干脆直接引入了一个新方法.delegate()。

使用.delegate()，前面的例子可以这样写：

$('#element).delegate('a', 'click', function() { alert("!!!") });

JQuery扫描文档查找(‘#element’)，并使用click事件和’a’这一CSS选择器作为参数把alert函数绑定到

(‘#element)上。

任何时候只要有事件冒泡到$(‘#element)上，它就查看该事件是否是click事件，以及该事件的目标元素是否与CCS选择器相匹配。如果两种检查的结果都为真的话，它就执行函数。

可以注意到，这一过程与.live()类似，但是其把处理程序绑定到具体的元素而非document这一根上

那么 http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/0028.png?rev=2.6.1 http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/100/2032.png?rev=2.6.1 http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0061.png?rev=2.6.1 http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/100/2032.png?rev=2.6.1 http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/0029.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/002E.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/006C.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0069.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0076.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0065.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/0028.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/0029.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/003D.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/003D.png?rev=2.6.1

(document).delegate('a') ？

可见，.delegate()方法是一个相对完美的解决方案。但在DOM结构简单的情况下，也可以使用.live()。

#### .on()

其实.bind(), .live(), .delegate()都是通过.on()来实现的，.unbind(), .die(), .undelegate(),也是一样的都是通过.off()来实现的

提供了一种统一绑定事件的方法。

适用性：

适合：click 、mousedown、mouseup、keydown 、keyup、keypress

不适合：mouseover和mouseout虽然也有事件冒泡，但是处理他们要特别注意，因为需要经常计算他们的位置，处理起来不太容易，focus、blur之类的本身就没有冒泡的特性，自然不能用事件委托；

**事件的存储优化：**

jQuery并没有将事件处理函数直接绑定到DOM元素上，而是通过http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/141/002E.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0064.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0061.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0074.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/141/0061.png?rev=2.6.1存储在缓存.cahce上，这里就是之前分析的贯穿整个体系的缓存系统了.

声明绑定的时候：

* 首先为DOM元素分配一个唯一ID，绑定的事件存储在http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/120/002E.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0063.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0061.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0068.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0063.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0065.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/120/005B.png?rev=2.6.1唯一http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0049.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Math/Italic/120/0044.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/120/005D.png?rev=2.6.1http://mathjax.cnblogs.com/2_6_1/fonts/HTML-CSS/TeX/png/Main/Regular/120/005B.png?rev=2.6.1.expand ][ 'events' ]上，而events是个键-值映射对象，键就是事件类型，对应的值就是由事件处理函数组成的数组，最后在DOM元素上绑定 （addEventListener/ attachEvent）一个事件处理函数eventHandle，这个过程由 jQuery.event.add 实现。

执行绑定的时候：

* 当事件触发时eventHandle被执行，eventHandle再去$.cache中寻找曾经绑定的事件处理函数并执行，这个过程由 jQuery.event. trigger 和 jQuery.event.handle实现。
* 事件的销毁则由jQuery.event.remove 实现，remove对缓存$.cahce中存储的事件数组进行销毁，当缓存中的事件全部销毁时，调用removeEventListener/ detachEvent销毁绑定在DOM元素上的事件处理函数eventHandle。