**经查阅发现，是先执行的失去焦点事件再执行获取焦点事件**

失去焦点事件中对其中的方法进行了延迟处理，如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | //验证码输入框失去焦点，去除清除按钮;获取焦点时，若有值则显示清除按钮  $('#auth\_code').blur(function() {   setTimeout(function() { //进行延时处理，时间单位为千分之一秒    $('#auth\_code').parent().find('.icon-clear').hide();   }, 100)  }).focus(function(){   if($(this).val() != ''){    $(this).parent().find('.icon-clear').css('display', 'table-cell');   }  }); |

**JavaScript中鼠标事件触发顺序探索**

鼠标的事件有非常多，如PC模式下的click、mousedown、mouseup，手机模式下的touchstart、touchend、touchmove。那么如果将他们全部绑定之后，在PC端和移动端分别做一些鼠标的常用操作，事件触发的情况会是如何呢。

毕竟实践出真知，所以今天我就实验了一下。

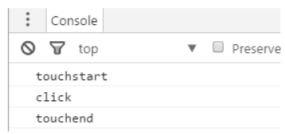
首先打开了百度，然后f12，切换到手机模式，在console控制台中，为body标签绑定上之前提到的所有事件，代码如下：



接下来，我们就可以正式开始实验了：

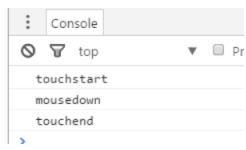
**一、移动端模式下：**

**(1)快速点击一次**



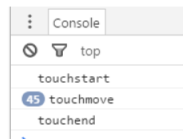
可见移动端单击会先触发touchstart事件，并在touchend前成功触发点击事件。

**(2)长按再松开（不移动）**



可见移动端中，如果长按再松开，将不再触发click事件，而变成mousedown事件。

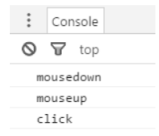
**(3)按住拖动再松开**



不再触发click和mousedown，转而变成touchmove

**二、PC模式下（切换回pc模式实验）**

经过测试，在PC模式下做以上的三种操作，触发效果一致，如下：



总结：经过上边的测试，我们可以发现一些规律。

1.PC模式下，无论如何都不会触发与touch相关的事件。

2.click事件，在pc模式下是在鼠标down再up之后才算触发，而移动模式下则在touch的start之后，end之前触发。

3.移动模式下长按不会触发click事件，如果不移动，则会在touchstart之后不久触发mousedown事件，如果移动则触发mousemove事件。