函数防抖：事件被触发n秒后再执行，如果在这n秒内又被触发，则重新计时。因此如果一直触发事件就一直清除事件回调，因为在等一个>=n的时间间隔，在这个时间间隔内才能执行到回调函数里。

一直触发事件就一直清除事件回调（因为触发事件的时间间隔没有超过200ms所以被清除掉了），直到有一个时间间隔200ms以后再触发事件，setTimeout中的事件才能执行。

（input输入验证）

*function* debounce(*fn*,*delay*){

*var* timeout;

return *function*(*e*){

*var* context = this,args = arguments;

clearTimeout(timeout);

timeout = setTimeout(*function*(*e*){

fn.apply(context,args);//回调函数里

},delay)

}

}

函数节流：一个单位时间内只能触发一次（对应if），如果在这个单位时间内触发多次则只执行一次，并且触发完了之后还会执行一次（对应else）。（监听页面滚动）

*function* throttle(*fn*,*delay*){

*var* timer,start = new Date();

return *function*(*e*){

*var* args = arguments,context = this;

*var* cur = new Date();

clearTimeout(timer);

if(cur-start > delay){//事件不断触发，没delay时间段触发一次

fn.apply(context,args);

start = cur;

}else{

timer = setTimeout(*function*(){//事件停止触发最后还有再有一次

fn.apply(context,args);

},delay);

}

}

}