# 交互相关

## 高频事件的节流和消抖

### 消抖

debounce

多次操作，只触发一次

### 节流

限制调用函数触发频率

$(window).on('scroll', \_.throttle(doSomething, 200));

# 渲染相关

1.减少 JS 逐行修改元素样式：

直接改变元素class，或者csstext属性

el.style.cssText = "; left: " + left + "px; top: " + top + "px;";

虽然浏览器会智能的识别短时间内多次reflow，组合为队列，一定时间后flush队列，将多个reflow变为一个reflow。但是最好还是避免逐行写。

2.离线处理 DOM 操作：

①通过 documentFragment 集中处理临时操作；(现代浏览器已优化，不需要使用)

②克隆节点进行操作，然后进行原节点替换；

③使用 display:none; 进行批量操作。

1. 减少某些读操作导致样式的重新计算，即减少 offset\*、scroll\*、client\*、getComputedStyle、currentStyle 的使用，因为每次调用都会刷新操作缓冲区，执行 reflow & repaint。

offsetTop/offsetLeft/offsetWidth/offsetHeight

scrollTop/scrollLeft/scrollWidth/scrollHeight

clientTop/clientLeft/clientWidth/clientHeight

getComputedStyle()

注：

每当试图通过JavaScript获取某个节点的这些信息的时候，为了获取真实信息，会导致浏览器马上flush队列。

4.避免强制性同步布局

在js中读取的布局信息都是上一帧的信息，如果在读操作之前修改页面布局，会导致强制reflow。

因此应当尽量避免读操作与写操作交替出现，读操作应当发在一起。

1. 减少绘制区域

绘制一个大问题是，浏览器会将两个需要绘制的区域联合起来，这可能导致整个屏幕重绘（比如：页面顶部和底部都有绘制元素）

减少绘制区域往往是编排动画，使其不过多重叠（在不同的层中）。

1. 降低绘制的复杂性

一些绘制的开销更大（如涉及模糊的元素）