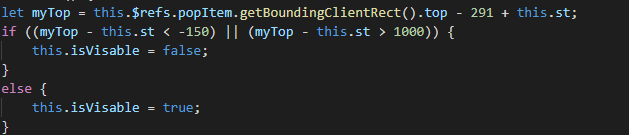
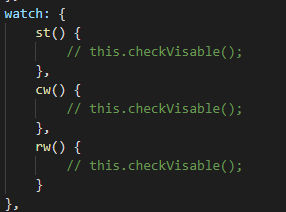
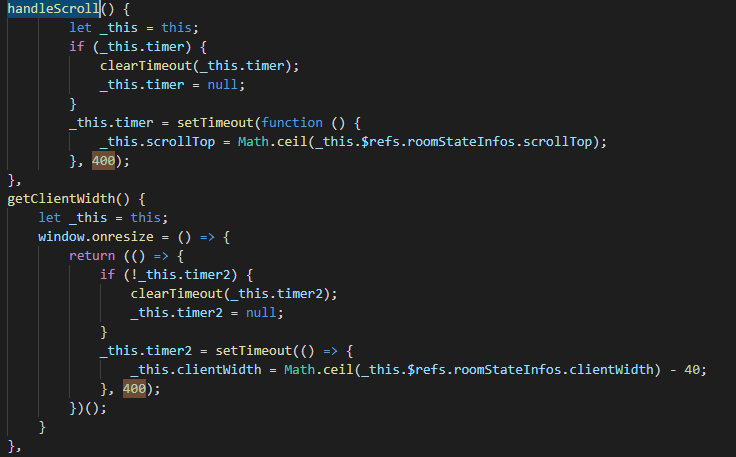
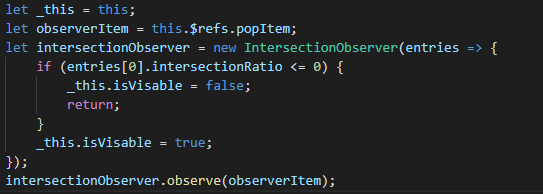


背景： 公寓管理有房态图模块，该页面显示当前公寓的所有房间，公寓类型有3种数据结构（1. 单个公寓无房源，2.单个公寓有房源；3. 多个公寓组成数据）；每个房间元素有多个doom组成；当房间超过上千多间的时候，页面操作的时候比较卡顿，用户体验较差。

思路：

1. 算出整个容器的高度，根据scrollTop来计算出可视区域的元素，把可视区域的元素构建成新的object，只渲染可视区域的元素；
   1. 这样具有几个问题：
      1. 房态图有3种比较复杂的Object，不像一个简单的列表高度固定，容易计算，需要对每种类型都进行特别的运算；而且Object层次比较复杂，只取可视区域的元素构造起来相对比较复杂。结构可能需要完全重新计算。
2. 上述方法不行，想采用骨架占位的方式，可视区域的房间用完整的doom，可视区域外的用一个简单的div占位，这样doom数目相当于原来doom数目的几十分之一。（我们知道doom数量是影响页面性能的最关键因素）。
   1. 最开始我使用的是这种方法：
      1. 监听滚动事件和resize事件，及组件初始化事件；（为了提高性能，对滚动及resize事件进行节流，每0.4s进行一次重新计算）；
      2. 子组件里watch这些属性的变化，每次变化的时候，来计算出自己是否需要显示详情。
         1. 
         2. 
      3. 父组件
         1. 
      4. 这样会有大量的回流和重绘，而且会有0.4s的延迟，效果和性能都不够完美。
3. 使用intersectionObserver
   1. 

其实可以看出来第2种方案，带了大量开销的是多次获取getBoundingClientRect()及scrolltop来获取位置用来判断是否在可视区域。

经过查找我发现IntersectionObserver实例一个异步实例，在浏览器空闲状态的时候触发callback，能解决问题。该方法优势是性能好，提高用户体验。特别在预加载（滚动加载，无限加载），以及懒加载的时候有特别大的优势。