02

OPEN ORIENTED

凹 凸 实 验 室

throttle 和 debounce

问题由来

对于浏览器来说，高频次地触发某些事件，如果是涉及大量的DOM操作，很容易迅速吃掉系统大量内存，造成卡死现象。

非常普遍的一个场景，当我们需要监听浏览器的滚动 scroll，窗口的调整 resize 等。这个时候就需要考虑如何限制这些高频次的事件触发影响到页面的性能

假如我们需要限制的时间间隔是 200ms

解决方法

解决方法的策略是限制高频次的函数在一定的时间范围内的执行次数。

这里提供两个方法，throttle 和 debounce，那么他们有什么区别呢?

电梯问题

想象每天上班大厦底下的电梯。把电梯完成一次运送，类比为一次函数的执行和响应。假设电梯有两种运行策略 throttle 和 debounce ，超时设定为15秒，不考虑容量限制。

1. throttle 策略的电梯。保证如果电梯第一个人进来后，15秒后准时运送一次，不等待。如果没有人，则待机。

2. debounce 策略的电梯。如果电梯里有人进来，等待15秒。如果又人进来，15秒等待重新计时，直到15秒超时，开始运送。如果没有人，则待机。

解决方法

1. throttle 函数节流

Throttling enforces a maximum number of times a function can be called over time. As in "execute this function at most once every 200 milliseconds."

在200毫秒中，最多执行该函数一次

Throttle应用场景

1. 试图在web页面滚动时进行的DOM操作，都应该做节流限制。

2. 鼠标事件mouseover在某些场景里需要做节流限制

3. window窗口的resize事件

解决方法

2. debounce 防止抖动，防反跳

Debouncing enforces that a function not be called again until a certain amount of time has passed without it being called. As in "execute this function only if 200 milliseconds have passed without it being called.”

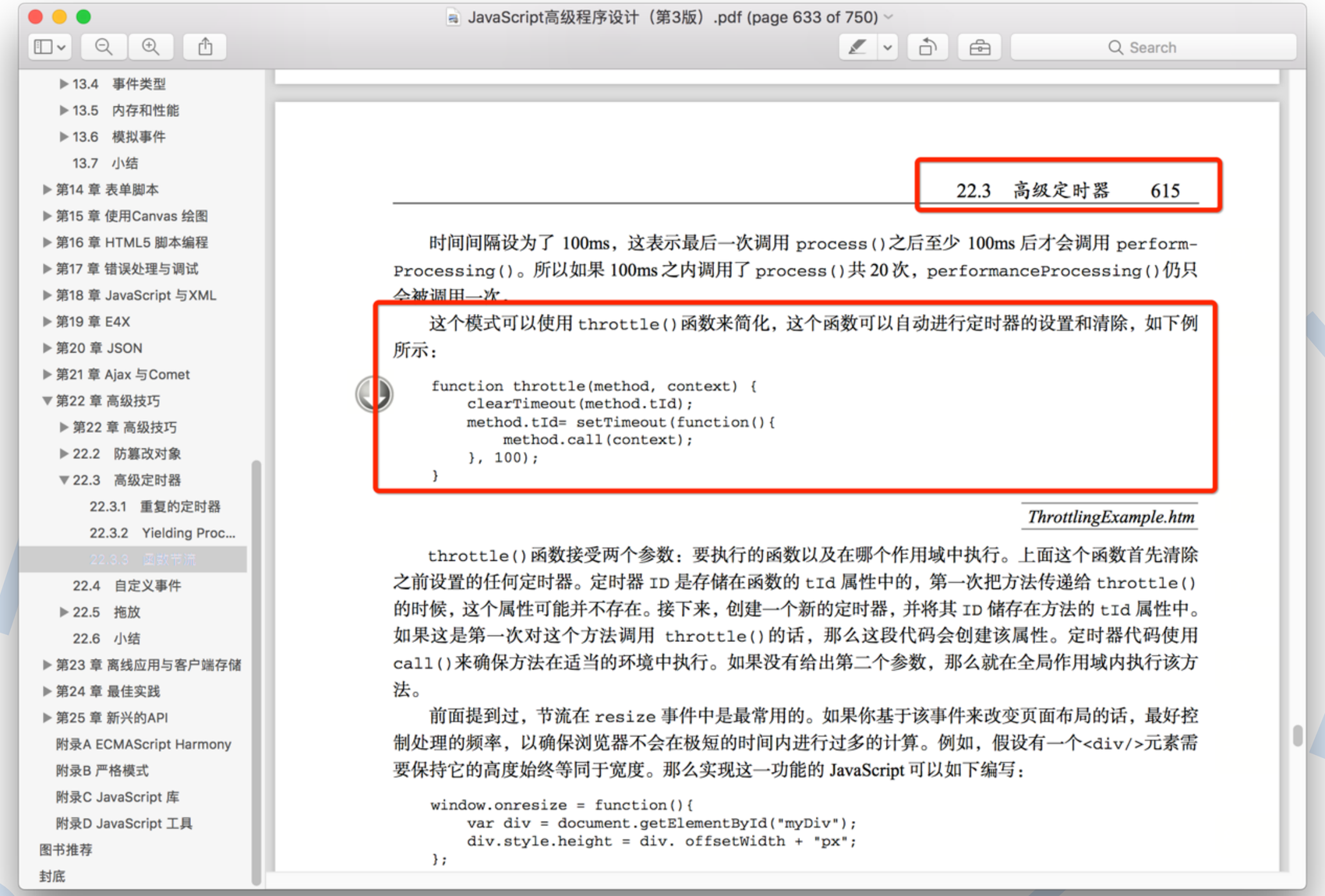
只有在200ms结束前没有再被调用才执行该函数

Debounce 应用场景

1. 在线文档的自动保存。判断用户的键盘事件，当一端事件没有keyup或keydown事件时，进行自动保存

2. 提交数据，防止多次重复请求等

勘误：高级程序设计 P615页里提到的throttle 实际上应该是debounce



THANKS

FOR YOUR WATCHING

02

OPEN ORIENTED

凹 凸 实 验 室