浏览器渲染原理

浏览器有一个渲染引擎，用来渲染窗口所展示的内容。可以显示html、xml及图片，也可借助插件显示其他类型的数据，如PDF阅读器插件以及markdown。

渲染主流程

渲染引擎首先通过网络获得请求文档的内容，通常以8k分块的方式完成。取得内容之后渲染基本流程如下：

解析html => 构建dom树=>构建render树=>布局render树=>绘制render树

DOM Tree：浏览器将HTML解析成树形的数据结构。

Render Tree: DOM和CSSOM合并后生成Render Tree。

layout: 有了Render Tree，浏览器已经能知道网页中有哪些节点、各个节点的CSS定义以及他们的从属关系，从而去计算出每个节点在屏幕中的位置。

painting: 按照算出来的规则，通过显卡，把内容画到屏幕上。

回流和重绘

回流（reflow）：浏览器渲染页面，若因用户或者脚本的某些行为改变了布局，需要重新渲染页面，这个过程叫做回流，回流无法避免，只能优化。
重绘（repaint）：元素改变样式时触法发，布局不变，影响比回流小；回流一定会触法重绘，但重绘不一定会触法回流。

触发回流：

1.添加或删除可见的DOM元素
2.元素位置的改变
3.元素尺寸的改变
4.内容的改变--文本或图片大小变化引擎的计算值宽度和高度改变
5.页面渲染初始化
6.浏览器窗口尺寸改变

7.js脚本获取dom的style值（offsetTop、offsetWidth；scrollTop/Left；clientTop/Width，width）；以及请求getComputedStyle()和IE的currentStyle

flush队列

多次回流、重绘时，浏览器并不会每一次触发都立即回流、重绘，而是维护一个队列， 把所有会引起回流、重绘的操作放入这个队列，等队列中的操作到了一定的数量或者到了一定的时间间隔，浏览器就会flush队列，进行一个批处理。这样就会让多次的回流、重绘变成一次回流重绘。

优化：

减少回流、重绘的原理就是减少对render tree的操作，并减少一些对style信息的请求，尽量利用号浏览器的优化策略。

1.直接改变class Name，如果动态改变样式，使用cssText（考虑没有优化的浏览器）
2.合并回流、重绘操作
    a) 使用DocumentFragment进行缓存操作,引发一次回流和重绘；
    b) 使用display:none技术，只引发两次回流和重绘；
    c) 使用cloneNode(true or false) 和 replaceChild 技术，引发一次回流和重绘；
3.减少访问引起浏览器flush队列的属性，访问时尽量利用缓存
4.元素使用绝对定位和固定定位