### 浏览器的回流与重绘 (Reflow & Repaint)

1. 回流重绘的概念

浏览器使用流式布局模型 (Flow Based Layout)。 浏览器会把HTML解析成DOM，把CSS解析成CSSOM，DOM和CSSOM合并就产生了Render Tree。 有了RenderTree，我们就知道了所有节点的样式，然后计算他们在页面上的大小和位置，最后把节点绘制到页面上。 由于浏览器使用流式布局，对Render Tree的计算通常只需要遍历一次就可以完成，但table及其内部元素除外，他们可能需要多次计算，通常要花3倍于同等元素的时间，这也是为什么要避免使用table布局的原因之一。

一句话：回流必将引起重绘，重绘不一定会引起回流。

当Render Tree中部分或全部元素的尺寸、结构、或某些属性发生改变时，浏览器重新渲染部分或全部文档的过程称为回流

1. 回流 的常见场景

页面首次渲染 浏览器窗口大小发生改变 元素尺寸或位置发生改变 元素内容变化（文字数量或图片大小等等） 元素字体大小变化 添加或者删除可见的DOM元素 激活CSS伪类（例如：:hover） 查询某些属性或调用某些方法

一些常用且会导致回流的属性和方法：

clientWidth、clientHeight、clientTop、clientLeft offsetWidth、offsetHeight、offsetTop、offsetLeft scrollWidth、scrollHeight、scrollTop、scrollLeft scrollIntoView()、scrollIntoViewIfNeeded() getComputedStyle() getBoundingClientRect() scrollTo()

1. 重绘 (Repaint) 当页面中元素样式的改变并不影响它在文档流中的位置时（例如：color、background-color、visibility等），浏览器会将新样式赋予给元素并重新绘制它，这个过程称为重绘。
2. 性能影响 回流比重绘的代价要更高。 有时即使仅仅回流一个单一的元素，它的父元素以及任何跟随它的元素也会产生回流。 现代浏览器会对频繁的回流或重绘操作进行优化： 浏览器会维护一个队列，把所有引起回流和重绘的操作放入队列中，如果队列中的任务数量或者时间间隔达到一个阈值的，浏览器就会将队列清空，进行一次批处理，这样可以把多次回流和重绘变成一次。 当你访问以下属性或方法时，浏览器会立刻清空队列：

clientWidth、clientHeight、clientTop、clientLeft offsetWidth、offsetHeight、offsetTop、offsetLeft scrollWidth、scrollHeight、scrollTop、scrollLeft width、height getComputedStyle() getBoundingClientRect()

因为队列中可能会有影响到这些属性或方法返回值的操作，即使你希望获取的信息与队列中操作引发的改变无关，浏览器也会强行清空队列，确保你拿到的值是最精确的。

1. 减少回流重绘的优化方案

CSS

避免使用table布局。 尽可能在DOM树的最末端改变class。 避免设置多层内联样式。 将动画效果应用到position属性为absolute或fixed的元素上。 避免使用CSS表达式（例如：calc()）。

JavaScript

避免频繁操作样式，最好一次性重写style属性，或者将样式列表定义为class并一次性更改class属性。 避免频繁操作DOM，创建一个documentFragment，在它上面应用所有DOM操作，最后再把它添加到文档中。 也可以先为元素设置display: none，操作结束后再把它显示出来。因为在display属性为none的元素上进行的DOM操作不会引发回流和重绘。 避免频繁读取会引发回流/重绘的属性，如果确实需要多次使用，就用一个变量缓存起来。 对具有复杂动画的元素使用绝对定位，使它脱离文档流，否则会引起父元素及后续元素频繁回流。