**页面呈现流程**

  在讨论页面重绘、回流之前。需要对页面的呈现流程有些了解，页面是怎么把html结合css等显示到浏览器上的，下面的流程图显示了浏览器对页面的呈现的处理流程。可能不同的浏览器略微会有些不同。但基本上都是类似的。

1. 浏览器把获取到的html代码解析成1个Dom树，html中的每个tag都是Dom树中的1个节点，根节点就是我们常用的document对象 (<html> tag)。dom树就是我们用firebug或者IE Developer Toolbar等工具看到的html结构，里面包含了所有的html tag，包括display:none隐藏，还有用JS动态添加的元素等。

2. 浏览器把所有样式(主要包括css和浏览器的样式设置)解析成样式结构体，在解析的过程中会去掉浏览器不能识别的样式，比如IE会去掉-moz开头的样式，而firefox会去掉\_开头的样式。

3、dom tree和样式结构体结合后构建呈现树(render tree),render tree有点类似于dom tree，但其实区别有很大，render tree能识别样式，render tree中每个node都有自己的style，而且render tree不包含隐藏的节点(比如display:none的节点，还有head节点)，因为这些节点不会用于呈现，而且不会影响呈现的，所以就不会包含到 render tree中。注意 visibility:hidden隐藏的元素还是会包含到render tree中的，因为visibility:hidden 会影响布局(layout)，会占有空间。根据css2的标准，render tree中的每个节点都称为box([Box dimensions](http://www.w3.org/TR/CSS2/box.html#box-dimensions))，box所有属性：width,height,margin,padding,left,top,border等。

4. 一旦render tree构建完毕后，浏览器就可以根据render tree来绘制页面了。

**回流与重绘**

1. 当render tree中的一部分(或全部)因为元素的规模尺寸，布局，隐藏等改变而需要重新构建。这就称为回流(其实我觉得叫重新布局更简单明了些)。每个页面至少需要一次回流，就是在页面第一次加载的时候。

2. 当render tree中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，比如background-color。则就叫称为重绘。

  注：从上面可以看出，回流必将引起重绘，而重绘不一定会引起回流。

**什么操作会引起重绘、回流**

  其实任何对render tree中元素的操作都会引起回流或者重绘，比如：

1. 添加、删除元素(回流+重绘)

2. 隐藏元素，display:none(回流+重绘)，visibility:hidden(只重绘，不回流)

3. 移动元素，比如改变top,left(jquery的animate方法就是,改变top,left不一定会影响回流)，或者移动元素到另外1个父元素中。(重绘+回流)

4. 对style的操作(对不同的属性操作，影响不一样)

5. 还有一种是用户的操作，比如改变浏览器大小，改变浏览器的字体大小等(回流+重绘)

导致回流的操作：

1. 调整窗口大小（Resizing the window）
2. 改变字体（Changing the font）
3. 增加或者移除样式表（Adding or removing a stylesheet）
4. 内容变化，比如用户在input框中输入文字（Content changes, such as a user typing text in  
   an input box）
5. 激活 CSS 伪类，比如 :hover (IE 中为兄弟结点伪类的激活)（Activation of CSS pseudo classes such as :hover (in IE the activation of the pseudo class of a sibling)）
6. 操作 class 属性（Manipulating the class attribute）
7. 脚本操作 DOM（A script manipulating the DOM）
8. 计算 offsetWidth 和 offsetHeight 属性（Calculating offsetWidth and offsetHeight）
9. 设置 style 属性的值 （Setting a property of the style attribute）

**如何避免回流或将它们对性能的影响降到最低？**

1. 如果想设定元素的样式，通过改变元素的 class 名 (尽可能在 DOM 树的最末端)
2. 避免设置多项内联样式
3. 应用元素的动画，使用 position 属性的 fixed 值或 absolute 值
4. 权衡平滑和速度
5. 避免使用table布局
6. 避免使用CSS的JavaScript表达式 (仅 IE 浏览器)

**尽可能在DOM树的最末端改变class**

回流可以自上而下，或自下而上的回流的信息传递给周围的节点。回流是不可避免的，但可以减少其影响。尽可能在DOM树的里面改变class，可 以限制了回流的范围，使其影响尽可能少的节点。例如，你应该避免通过改变对包装元素类去影响子节点的显示。面向对象的CSS始终尝试获得它们影响的类对象 （DOM节点或节点），但在这种情况下，它已尽可能的减少了回流的影响，增加性能优势。

**避免设置多层内联样式**

我们都知道与DOM交互很慢。我们尝试在一种无形的DOM树片段组进行更改，然后整个改变应用到DOM上时仅导致了一个回流。同样，通过 style属性设置样式导致回流。避免设置多级内联样式，因为每个都会造成回流，样式应该合并在一个外部类，这样当该元素的class属性可被操控时仅会 产生一个reflow。

**动画效果应用到position属性为absolute或fixed的元素上**

*动画效果应用到position属性为absolute或fixed的元素上*，它们不影响其他元素的布局，所它他们只会导致重新绘制，而不是一个完整回流。这样消耗会更低。

**牺牲平滑度换取速度**

Opera还建议我们*牺牲平滑度换取速度*，其意思是指您可能想每次1像素移动一个动画，但是如果此动画及随后的回流 使用了100%的CPU，动画就会看上去是跳动的，因为浏览器正在与更新回流做斗争。动画元素每次移动3像素可能在非常快的机器上看起来平滑度低了，但它 不会导致CPU在较慢的机器和移动设备中抖动。

**避免使用table布局**

*避免使用table布局*。可能您需要其它些避免使用table的理由，在布局完全建立之前，table经常需要多个 关口，因为table是个和罕见的可以影响在它们之前已经进入的DOM元素的显示的元素。想象一下，因为表格最后一个单元格的内容过宽而导致纵列大小完全 改变。这就是为什么所有的浏览器都逐步地不支持table表格的渲染。然而有另外一个原因为什么表格布局时很糟糕的主意，根据[Mozilla](http://www.mozilla.org/newlayout/doc/reflow.html)，即使一些小的变化将导致表格(table)中的所有其他节点回流。

**避免使用CSS的JavaScript表达式**

这项规则较过时，但确实是个好的主意。主要的原因，这些表现是如此昂贵，是因为他们每次重新计算文档，或部分文档、回流。正如我们从所有的很多事情看到的：引发回流，它可以每秒产生成千上万次。当心！