关于回流(reflows)与重绘(repaints)，我已经在[twitter](http://twitter.com/stubbornella/statuses/1131777322)和[delicious](http://delicious.com/stubbornella/reflow)上发布，但是并没有在演讲中提到或是以文章形式发布。

第一次让我开始思考关于回流(reflows)与重绘(repaints)的问题是在和[ParisWeb](http://www.paris-web.fr/2008/)上的[Mr. Glazman](http://www.glazman.org/weblog/)做一个[firey交换](http://www.flickr.com/photos/kiouv/3035720145/)的时候。我可能有一些顽固，但是我确实听了他的一些理论。:)[Stoyan](http://www.phpied.com/)和我开始讨论如何量化这个问题。

展望性能社区，除了一些典型的黑盒实验外，需要与浏览器厂商有更多的合作。对于性能，浏览器制造者知道哪些是重要的，哪些是不相干的。Opera列出“reflow和repaint是减缓JavaScript的三大主要原因之一”一文，所以其肯定值得一看。// zxx: Firefox浏览器相关内容可以看[这里](http://www.mozilla.org/newlayout/doc/reflow.html)；Safari可以看[这里](http://webkit.org/blog/114/webcore-rendering-i-the-basics/)。

让我们从一些背景资料开始，当一个元素的外观的可见性visibility发生改变的时候，重绘(repaint)也随之发生，但是不影响布局。类似的例子包括：outline, visibility, or background color。根据Opera浏览器，重绘的代价是高昂的，因为浏览器必须验证DOM树上其他节点元素的可见性。而回流更是性能的关键因为其变化涉及到部分页面（或是整个页面）的布局。一个元素的回流导致了其所有子元素以及DOM中紧随其后的祖先元素的随后的回流。

例如：

<body>

<div class="error">

<h4>我的组件</h4>

<p><strong>错误：</strong>错误的描述…</p>

<h5>错误纠正</h5>

<ol>

<li>第一步</li>

<li>第二步</li>

</ol>

</div>

</body>

在上面的HTML片段中，对该段落(<p>标签)回流将会引发强烈的回流，因为它是一个子节点。这也导致了祖先的回流（div.error和body – 视浏览器而定）。此外，<h5>和<ol>也会有简单的回流，因为其在DOM中在回流元素之后。就Opera而言，大部分的回流将导致页面的重新渲染。

Opera原文：Reflows are very expensive in terms of performance, and is one of the main causes of slow DOM scripts, especially on devices with low processing power, such as phones. In many cases, they are equivalent to laying out the entire page again.

**既然它们对性能影响如此可怕，那什么会导致回流呢？**

不幸的是，很多的东西，其中一些还与CSS的书写特别相关。

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）

[Mozilla关于回流的文章](http://www.mozilla.org/newlayout/doc/reflow.html)罗列了导致回流的要点以及何时可以减少他们。

**如何避免回流或将它们对性能的影响降到最低？**

注意：这里我限定了自己只能讨论CSS对回流的影响，如果您是一位JavaScript程序员，我是推荐您读一下我的[reflow链接](http://delicious.com/stubbornella/reflow?tags=on)（zxx: 原作者收藏标记的一些关于reflow的一些文章或页面链接），有一些非常好的东西，没有直接关系到CSS。

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)（Change classes on the element you wish to style (as low in the dom tree as possible)）
2. 避免设置多项内联样式（Avoid setting multiple inline styles）
3. 应用元素的动画，使用 position 属性的 fixed 值或 absolute 值（Apply animations to elements that are position fixed or absolute）
4. 权衡平滑和速度（Trade smoothness for speed）
5. 避免使用table布局（Avoid tables for layout）
6. 避免使用CSS的JavaScript表达式 (仅 IE 浏览器)（Avoid JavaScript expressions in the CSS (IE only)）

**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（感谢Bill Scott提供）。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。

Jenny Donnelly, YUI 数据表格 widget的所有者，建议使用数据表格的固定布局以便更有效的布局算法，任何表格-布局的值除了”auto”将引发一个固定布局，根据CSS2.1规范，这将允许表格一行一行的呈递。[Quirksmode显示](http://www.quirksmode.org/css/tables.html#tablelayout)，大部分的浏览器对表格布局属性支持良好。

In this manner, the user agent can begin to lay out the table once the entire first row has been received. Cells in subsequent rows do not affect column widths. Any cell that has content that overflows uses the ‘overflow’ property to determine whether to clip the overflow content.

[*固定布局, CSS 2.1 规范*](http://www.w3.org/TR/CSS21/tables.html#fixed-table-layout)

This algorithm may be inefficient since it requires the user agent to have access to all the content in the table before determining the final layout and may demand more than one pass.

[*自动布局, CSS 2.1 规范*](http://www.w3.org/TR/CSS21/tables.html#auto-table-layout)

**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！

**进一步的学习**

Yahoo！出色的性能团队做了一个实验，以确定最佳的方法引入外部的样式表文件。我们建议把链接标记放在头部，尽管其比所有其他阻碍进一步渲染的方法慢一秒（6.3至7.3秒）。虽然逐步地渲染不可改变的（用户讨厌盯着一个空白屏幕），但它使我对下载组件和整体响应时间的渲染，重画，回流和CPU的颠簸造成的影响感到好奇。如果我们可以减少加载过程中的回流个数我们可以夺回了失去的时间的十分之一（100毫秒）吗？如果它是多达一半呢？

在SXSW我试图说服史蒂夫的回流是很重要的。我告诉他一个我已经做了很长一段时间的实验，只是没有时间。我希望有人能在我留下来的那个地方继续（提示！提示！）。虽然加载页面我想故意引发不同程度的回流。这或许可以通过切换一个body（实验）标签的class名称来改变body的最后一个child（没有子节点）完成。比较这两者情况，并增加每秒回流的数目，我们可以关联回流反应时间。衡量JS反应对回流的影响将会更难，因为我们所做能够引发回流的事情都可能会影响实验。

最后，量化影响的趣味只有那么一点点，因为浏览器厂商会告诉我们它的问题。或许更有趣的是把注意力集中在回流的原因以及如何避免它们。这将需要更好的工具，所以我需要浏览器厂商和性能社区共同努力，使之成为现实！

**事实说话**

也许你是一个视觉关注者，这些视频通过很酷的方式将回流过程可视化展示出来。//zxx: youtube视频，想看，需翻墙。

1. <http://www.youtube.com/watch?v=nJtBUHyNBxs>
2. <http://www.youtube.com/watch?v=ZTnIxIA5KGw>
3. <http://www.youtube.com/watch?v=dndeRnzkJDU>

**回流消失的疯狂**

为了改进性能，浏览器厂商可能限制回流影响临近的节点或者联合其他几个回流形成一次大的改变，就如[Mozilla的这篇文章](http://www.mozilla.org/newlayout/doc/reflow.html)中展示的。这可以提高性能，但有时也可能导致显示问题。您可以使用我们所了解到的关于回流的知识，在必要纠正相关显示问题的时候触发它们。

例如，我们改变图片优化网站上的选项卡，[http://smush.it](http://smush.it/)，随着选项卡的不同，内容区域的高度是不断变化的，有时候，下面的投影会被留下，这是因为内容区域上的父节点被切换而其容器可能没有回流，下面这张图是模拟的，因为这个bug很难用相机捕捉，任何试图获取操作都会导致回流而纠正显示。如果你自己发现类似的bug，移动背景图片到下面切换的DOM元素上。



另外一个例子是动态添加列表项，当您将列表项从9个增加到10个或是从99个增加到100个，列表的数字将不会正确的排队显示，这在所有浏览器都是。当总数按数量的顺序增加且浏览器不回流兄弟节点时，队列被打破。快速切换整个列表的显示或是增加添加一个class类，即使没有相关联的样式，也会导致回流和正确的队列显示。

**工具**

最近有小部分工具产生了一些波澜，Stoyan Stefanov和我已经一直在寻找好的方法来测量回流与重绘，这里有一些（尽管比较早）。要当心，有的在我正确使用之前把浏览器给搞跛了。大部分情况下，你需要每晚安装下最新的版本。

当Mozilla公开[MozAfterPaint](https://developer.mozilla.org/web-tech/2008/10/13/mozafterpaint/) Firefox API时，互联网界议论纷纷。

* 更新：Google的Lindsey Simon写了个可以在任意浏览器下测试回流时间的[书签工具](http://reflowr.appspot.com/)。非常的赞，注意：所有的震动都是正常的。//zxx: 我测试此链接为500错误
* John Resig写了个[书签工具](http://ejohn.org/blog/browser-paint-events/)来可视化绘制事件。
* Kyle Scholz创建了[一个工具](http://www.kylescholz.com/blog/2008/11/digging_into_paint_events_in_f_1.html)用来在页面加载之前可视化绘制事件
* Yoono的Alex创建了[XUL分析工具](http://blog.yoono.com/blog/2008/12/download-yoono-xulprofiler/)

是否其他人见到过好的关于回流评估的工具？请发送告知我。

还有其他一对不是直接用来处理回流的工具。

* [IE浏览器视觉往返分析仪(VRTA)](http://www.microsoft.com/downloads/details.aspx?FamilyID=119f3477-dced-41e3-a0e7-d8b5cae893a3&displaylang=en)，因为微软起了个这么有趣的名字，所以我们可能还真的会用它。:)
* [专为WPF的性能分析工具](http://msdn.microsoft.com/en-us/library/aa969767.aspx)

最后，我们需要一个跨浏览器的工具来量化并减少回流和重绘。我希望性能社区能够与浏览器厂商合作，使这一工具成为现实。浏览器厂商已经告诉我们有一段时间了，这是我们未来需要看到的，希望在我们手中。

//正文结束

**我补充的点东西**

上面展示了三个YouTube的视频，估计是不翻墙看不了，所以呢，我决定补充一个视频，是IE浏览器开发工程师讲解的关于页面性能的视频，从多角度分析，涉及到CSS，JavaScript，http链接，以及回流重绘等关键知识，相信会有所帮助的。如下：

[IE浏览器开发工程师为您展示页面性能的优化](http://player.youku.com/player.php/sid/XMTQ3NDAyOTUy/v.swf)