###从输入 URL 到页面加载完成发生了什么事

DNS解析

TCP连接

发送HTTP请求

服务器处理请求并返回HTTP报文

浏览器解析渲染页面

###浏览器应该有的功能

网络;资源管理;网页浏览;多页面管理;插件与管理;账户和同步;安全机制;开发者工具

浏览器的主要功能总结起来就是一句话:

将用户输入的url转变成可视化的图像

###浏览器的内核(渲染引擎)

在浏览器中有一个最重要的模块，它主要的作用是将页面转变为可视化的图像结果。这个模块就是浏览器内核，通常它也被称为渲染引擎。

IE----->Trident

Safari------>WebKit

Chrome;Opera----->Blink

Firefox------>Gecko

###渲染引擎

一个渲染引擎主要包括：HTML解析器，CSS解析器，布局layout模块，javascript引擎，绘图模块

##渲染过程

1.网页URL到构建DOM树的整个过程

1) 当用户输入URL的时候，Webkit调用资源加载器加载URL对应的网页

2) 加载器依赖网络模块建立连接，发送请求并接收答复

3) Webkit接收到各种网页或者资源的数据，其中某些资源可能是同步的或异步获取的

4) 网页被交给HTML解析器转变一系列的词语（Token）

5) 解析器根据词语构建节点（node），形成DOM树

6) 如果节点需要依赖于其他资源，

例如js css 图片 视频等，调用资源加载器来加载他们，但是这些都是异步的，

不会阻碍dom树的继续创建；顺序执行 并发加载

如果资源是css的话

调用CSS解析器解释将CSS解释成内部表示结构(CSSDOM)

如果资源是javascript的话

调用Javascript引擎解释并执行，

7）阻塞

css阻塞

css 在head中通过link的形式引入会阻塞页面的渲染

为什么？

避免闪屏现象

js阻塞

直接引入的js会阻塞页面的渲染

为什么？

Javascript代码可能会修改DOM树的结构

8) 预解析

WebKit 和 Firefox 都进行了这项优化。在执行js脚本时，其他线程会解析文档的其余部分，

找出并加载需要通过网络加载的其他资源。通过这种方式，资源可以在并行连接上加载，

从而提高总体速度。请注意，预解析器不会修改 DOM 树，而是将这项工作交由主解析器处理；

预解析器只会解析外部资源（例如外部脚本、样式表和图片）的引用。

2.从DOM树到可视化图像

1) CSS文件被CSS解析器解释成内部表示结构(CSSDOM)

2) CSS解析器工作完成之后，在DOM树上附加解释后的样式信息，这就是RenderObject树

3) RenderObject在创建的同时，Webkit会根据网页的结构创建RenderLayer，同时构建一个绘图上下文

4) 根据绘图上下文生成最终的图像（这一过程需要依赖图形库）

3.上面介绍的是一个完整的渲染过程，但现代网页很多都是动态的，这意味着在渲染完成之后，由于网页的动画或者用户的交互，

浏览器其实一直在不停地重复执行渲染过程。（重绘重排），以上的数字表示的是基本顺序，这不是严格一致的，

这个过程可能重复也可能交叉。浏览器是一个边解析边渲染的过程

###css图层

浏览器在渲染一个页面时，会将页面分为很多个图层，图层有大有小，每个图层上有一个或多个节点。

在渲染DOM的时候，浏览器所做的工作实际上是：

1. 获取DOM后分割为多个图层

2. 对每个图层的节点计算样式结果 （Recalculate style--样式重计算）

3. 为每个节点生成图形和位置 （Layout--重排,回流）

4. 将每个节点绘制填充到图层位图中 （Paint--重绘）

5. 图层作为纹理上传至GPU

6. 符合多个图层到页面上生成最终屏幕图像 （Composite Layers--图层重组）

###图层创建的条件

Chrome中满足以下任意情况就会创建图层：

1. 拥有具有3D变换的CSS属性

2. 使用加速视频解码的<video>节点

3. <canvas>节点

4. CSS3动画的节点

5. 拥有CSS加速属性的元素(will-change)

6. 元素有一个z-index较低且包含一个复合层的兄弟元素（换句话说就是该元素在复合层上面渲染）

###重绘(Repaint)

重绘是一个元素外观的改变所触发的浏览器行为，例如改变outline、背景色等属性。浏览器会根据元素的新属性重新绘制，

使元素呈现新的外观。重绘不会带来重新布局，所以并不一定伴随重排。

需要注意的是，如果图层中某个元素需要重绘，那么整个图层都需要重绘。

比如一个图层包含很多节点，其中有个gif图，gif图的每一帧，都会重回整个图层的其他节点，然后生成最终的图层位图。

所以这需要通过特殊的方式来强制gif图属于自己一个图层（translateZ(0)或者translate3d(0,0,0)

CSS3的动画也是一样（好在绝大部分情况浏览器自己会为CSS3动画的节点创建图层）

###重排(Reflow 回流)

渲染对象在创建完成并添加到渲染树时，并不包含位置和大小信息。计算这些值的过程称为布局或重排

"重绘"不一定需要"重排"，比如改变某个网页元素的颜色，就只会触发"重绘"，不会触发"重排"，因为布局没有改变。

但是，"重排"必然导致"重绘"，比如改变一个网页元素的位置，就会同时触发"重排"和"重绘"，因为布局改变了。

###触发重绘的属性

\* color \* background \* outline-color

\* border-style \* background-image \* outline

\* border-radius \* background-position \* outline-style

\* visibility \* background-repeat \* outline-width

\* text-decoration \* background-size \* box-shadow

###触发重排(回流)的属性

盒子模型相关属性会触发重布局 定位属性及浮动也会触发重布局： 改变节点内部文字结构也会触发重布局：

\* width \* top \* text-align

\* height \* bottom \* overflow-y

\* padding \* left \* font-weight

\* margin \* right \* overflow

\* display \* position \* font-family

\* border-width \* float \* line-height

\* border \* clear \* vertival-align

\* min-height \* white-space

###常见的触发重排的操作

Reflow 的成本比 Repaint 的成本高得多的多。DOM Tree 里的每个结点都会有 reflow 方法，

一个结点的 reflow 很有可能导致子结点，甚至父点以及同级结点的 reflow。在一些高性能的电脑上也许还没什么，

但是如果 reflow 发生在手机上，那么这个过程是非常痛苦和耗电的。

所以，下面这些动作有很大可能会是成本比较高的。

当你增加、删除、修改 DOM 结点时，会导致 Reflow , Repaint。

当你移动 DOM 的位置

当你修改 CSS 样式的时候。

当你 Resize 窗口的时候（移动端没有这个问题）

当你修改网页的默认字体时。

获取某些属性时(width,height...)

注：display:none 会触发 reflow，而 visibility:hidden 只会触发 repaint，因为没有发生位置变化。

###优化

如果我们需要使得动画或其他节点渲染的性能提高，需要做的就是减少浏览器在运行时所需要做的工作（减少1234中的步骤）

1. 计算需要被加载到节点上的样式结果（Recalculate style--样式重计算）

2. 为每个节点生成图形和位置（Layout--回流和重布局）

3. 将每个节点填充到图层中（Paint Setup和Paint--重绘）

4. 组合图层到页面上（Composite Layers--图层重组）

1.元素位置移动变换时尽量使用CSS3的transform来代替对top left等的操作

变换（transform）和透明度（opacity）的改变仅仅影响图层的组合

2.使用opacity来代替visibility

透明度竟然不会触发重绘？

透明度的改变时，GPU在绘画时只是简单的降低之前已经画好的纹理的alpha值来达到效果，并不需要整体的重绘。

不过这个前提是这个被修改opacity本身必须是一个图层，如果图层下还有其他节点，GPU也会将他们透明化

3.不要使用table布局

4.将多次改变样式属性的操作合并成一次操作

不要一条一条地修改DOM的样式，预先定义好class，然后修改DOM的className

5.将DOM离线后再修改

由于display属性为none的元素不在渲染树中，对隐藏的元素操作不会引发其他元素的重排。

如果要对一个元素进行复杂的操作时，可以先隐藏它，操作完成后再显示。这样只在隐藏和显示时触发2次重排。

6.利用文档碎片

7.不要把某些DOM节点的属性值放在一个循环里当成循环的变量

当你请求向浏览器请求一些 style信息的时候，就会让浏览器flush队列，比如：

1. offsetTop, offsetLeft, offsetWidth, offsetHeight

2. scrollTop/Left/Width/Height

3. clientTop/Left/Width/Height

4. width,height

当你请求上面的一些属性的时候，浏览器为了给你最精确的值，需要flush队列，

因为队列中可能会有影响到这些值的操作。即使你获取元素的布局和样式信息跟最近发生或改变的布局信息无关，

浏览器都会强行刷新渲染队列。

8.动画实现过程中，启用GPU硬件加速

9.为动画元素新建图层,提高动画元素的z-index