李某龙

Keep Learning，Keep Running...

浏览器渲染原理及流程

我们可能都知道浏览器含有一个渲染引擎，用来渲染窗口所展示的内容。默认情况下，渲染引擎可以显示html、xml文档及图片，它也可以借助插件（一种浏览器扩展）显示其他类型数据，例如使用PDF阅读器插件，用于显示PDF格式。但是其具体的渲染原理和流程估计也有很多人都不知道或者不清楚吧。这些天研究了一下浏览器的渲染原理，有了些心得，在这里跟大家分享一下，这里只讨论渲染引擎最主要的用途——显示应用了CSS之后的html及图片。

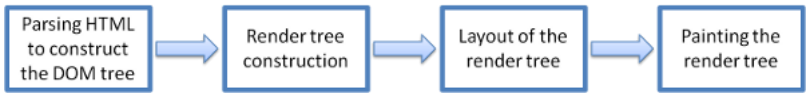
渲染引擎简介

本文所讨论的浏览器——Firefox、Chrome和Safari是基于两种渲染引擎构建的，Firefox使用Geoko——Mozilla自主研发的渲染引擎，Safari和Chrome都使用webkit。

渲染主流程

渲染引擎首先通过网络获得所请求文档的内容，通常以8K分块的方式完成。下面是渲染引擎在取得内容之后的基本流程：

解析html以构建dom树 -> 构建render树 -> 布局render树 -> 绘制render树



这里先解释一下几个概念，方便大家理解：

DOM Tree：浏览器将HTML解析成树形的数据结构。

CSS Rule Tree：浏览器将CSS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。

reflow（回流）：当浏览器发现某个部分发生了点变化影响了布局，需要倒回去重新渲染，内行称这个回退的过程叫 reflow。reflow 会从 <html> 这个 root frame 开始递归往下，依次计算所有的结点几何尺寸和位置。reflow 几乎是无法避免的。现在界面上流行的一些效果，比如树状目录的折叠、展开（实质上是元素的显 示与隐藏）等，都将引起浏览器的 reflow。鼠标滑过、点击……只要这些行为引起了页面上某些元素的占位面积、定位方式、边距等属性的变化，都会引起它内部、周围甚至整个页面的重新渲 染。通常我们都无法预估浏览器到底会 reflow 哪一部分的代码，它们都彼此相互影响着。

repaint（重绘）：改变某个元素的背景色、文字颜色、边框颜色等等不影响它周围或内部布局的属性时，屏幕的一部分要重画，但是元素的几何尺寸没有变。

注意：(1)display:none 的节点不会被加入Render Tree，而visibility: hidden 则会，所以，如果某个节点最开始是不显示的，设为display:none是更优的。

(2)display:none 会触发 reflow，而 visibility:hidden 只会触发 repaint，因为没有发现位置变化。

公告

昵称：李某龙
园龄：2年
粉丝：33
关注：3
+加关注

< 2018年9

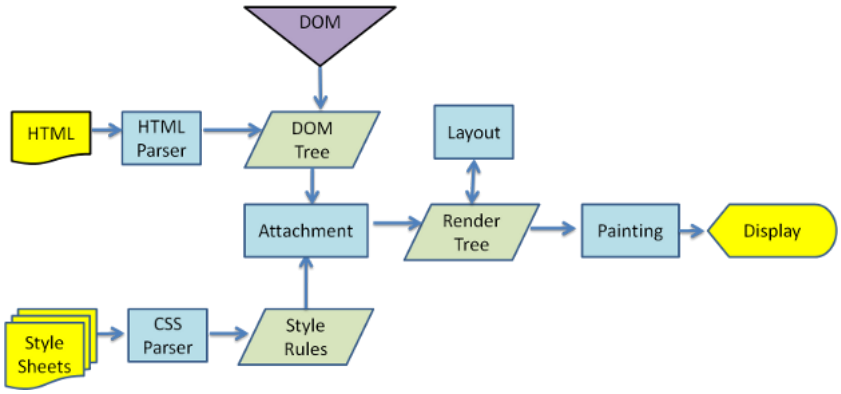
日	一	二	三
26	27	28	29
2	3	4	5
9	10	11	12
16	17	18	19
23	24	25	26
30	1	2	3

我的标签

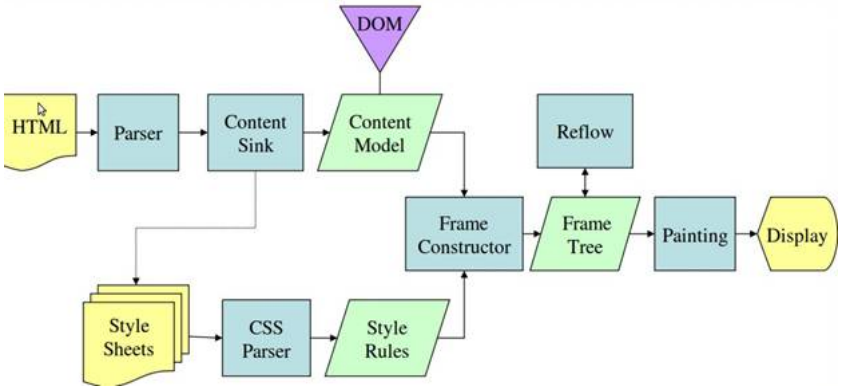
学习笔记(21)
前端(18)
js(17)
node(6)
nodejs(5)
浏览器(4)
Css(2)
闭包(2)
变量提升(1)
调试(1)
更多

(3)有些情况下，比如修改了元素的样式，浏览器并不会立刻reflow 或 repaint 一次，而是会把这样的操作积攒一批，然后做一次 reflow，这又叫异步 reflow 或增量异步 reflow。但是在有些情况下，比如resize 窗口，改变了页面默认的字体等。对于这些操作，浏览器会马上进行 reflow。

来看看webkit的主要流程：



再来看看Geoko的主要流程：



Gecko 里把格式化好的可视元素称做"帧树"（Frame tree）。每个元素就是一个帧（frame）。 webkit 则使用"渲染树"这个术语，渲染树由"渲染对象"组成。webkit 里使用"layout"表示元素的布局，Gecko则称为"reflow"。Webkit使用"Attachment"来连接DOM节点与可视化信息以构建渲染树。一个非语义上的小差别是Gecko在HTML与DOM树之间有一个附加的层，称作"content sink"，是创建DOM对象的工厂。

尽管Webkit与Gecko使用略微不同的术语，这个过程还是基本相同的，如下：

1. 浏览器会将HTML解析成一个DOM树，DOM 树的构建过程是一个深度遍历过程：当前节点的所有子节点都构建好后才会去构建当前节点的下一个兄弟节点。

2. 将CSS解析成 CSS Rule Tree 。

3. 根据DOM树和CSSOM来构造 Rendering Tree。注意：Rendering Tree 渲染树并不等同于 DOM 树，因为一些像Header或display:none的东西就没必要放在渲染树中了。

4. 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。下一步操作称之为layout，顾名思义就是计算出每个节点在屏幕中的位置。

5. 再下一步就是绘制，即遍历render树，并使用UI后端层绘制每个节点。

注意：*上述这个过程是逐步完成的，为了更好的用户体验，渲染引擎将会尽可能早的将内容呈现到屏幕上，并不会等到所有的html都解析完成之后再去构建和布局render树。它是解析完一部分内容就显示一部分内容，同时，可能还在通过网络下载其余内容。*

分类： 浏览器系列

标签： 学习笔记， 浏览器， 渲染， 前端， 浏览器渲染原理及流程

随笔分类
js(7)
lua学习笔记
npm(1)
浏览器系列(5)
你所不知道的JavaScri
随笔档案
2018年8月 (2)
2018年7月 (2)
2018年6月 (3)
2017年12月 (2)
2017年11月 (1)
2017年4月 (3)
2017年3月 (2)
2017年2月 (5)
2016年10月 (4)
2016年9月 (1)
积分与排名
积分 - 27543
排名 - 16871
最新评论
1. Re:js监听浏览器离开
chorme,IE,火狐都是
2. Re:js监听浏览器离开
@王振宇你是使用什么