```
Animations
Console
                          What's New
                                          Network co
Layer borders
Shows layer border and tiles (cyan)
FPS meter
Plots frames be recond, frame rate distribution, and GPU
Scrolling performance issues
```

Range: 15.60 s - 16.12 s

```
72.5 ms Rendering
              20.6 ms Painting
              66.8 ms Other
528 ms
             367.8 ms Idle
```

前端性能优化

1.html、css、js压缩: 使用YUI Compress进行压缩合并 gzip (所有浏览器都支持)

sdch压缩方式 http://www.bubuko.com/infodetail-595553.html

js会把字节变小。 text(){} 压缩后成 a(){}

2.小 雅虎军规

3.资源(图片、字体、js、css)压缩 基于http 强缓 让后台配合 服务端开启etag (适合不频繁换的)

4.缓存 - > localStorage 激活

如果localStorage超过一半(2.5M),开始卡:

大容量的解决方案(IndexedDB、Web SQL): -> 异步

localStorage扩容

5.cdn js跟本地service不能放在一起,用cdn架设

最多5个

预加载cdn地址,提前解析域名

<link href="//p0.meituan.net" rel="dns-prefetch">

6.v8

7.ajax -> php(削减后台api接口能力)->ip传给后端,后端通过wise,获取用户的网络信息 机型 8.客户端 定位 百度地图 android偷偷启一个本地服务 去请求安卓端127.0.0.1

9.网页端渲染过程

9.1.获取dom元素 分割成多个层

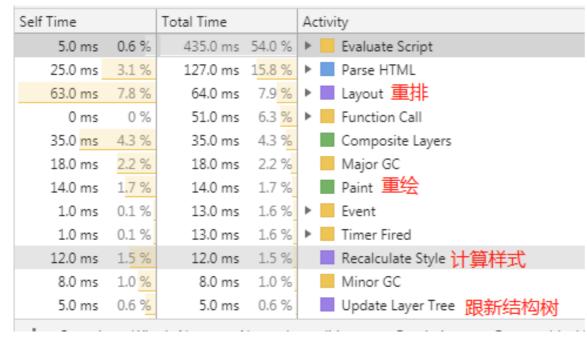
9.2. 对每个图层的阶段计算样式结果 (Recalculate style)

9.3.为每个节点生成图形跟位置 layout

9.4.将每个节点绘制并填充到图形的位置中 paint

9.5.图层作为纹理上传GPU

9.6.符合条件的多个图层生成屏幕的图像 Composite Layers



导致卡:排、绘、生成不了纹理上传不了GPU

哪些元素会生成独立的层

根元素、position\transform\半透明\canvas\video\overflow\滤镜

直接GPU参与,跳过重绘、重排、合成

```
css3d、video、webgl 、trnasform、css滤镜
```

css3d、video、webgl、transform(直接让gpu过来)、css滤镜

gpujs / gpu.js

性能慢丢这个

gpu.js 不在cpu里跑,肯定快

重绘

border-radius box-shadow color\\\

重排

1.改变元素的时候(offsetTop scrollTop width clientTop getComputedStyle),会跳过优化 2.删除 位置 大小 盒子

3.读写分离 var h1 = element.clientHeight; 下一帧去操作 requestAnimationFrame(function(){

xx.style.width = h1*2

});

4.专门处理dom读写分离的 https://github.com/wilsonpage/fastdom

跳过了cpu直接到达gpu

相同之处: 两者都是总线和外界联系, 有自己的缓存体系 以及数组和逻辑运算单元 两者都是为了计算任务而生的

不同之处: CPU主要负责 操作系统和应用程序 GPU主要负责跟显示相关的数据处理

GPU的活cpu都可以干 但是效率低下

10.html 白屏 -> SSR+SPA

多页转单页 -》。 spa pushstate . a/b->c/d ajax

用户访问我的网站,直接吐html swig->最关键的html 。 a 多页 切页就歇菜了 。 SPA单页

span render("ss.html") . 如果是单页,返回执行对象,前端自己拼

H5属性 > navigator.connection

navigator.connection 里面能取到网速

```
NetworkInformation {downlink: 10, effectiveType: "4g", onchange: null, rtt: 50, saveDat
   a: false} [
```

```
downlink: 10
 effectiveType: "4g"
 onchange: null
 rtt: 50
 saveData: false
▶ __proto__: NetworkInformation
```

PerformanceTiming {navigationStart: 1531486062874, unloadEventStart: 153 ▼ 1486063003, unloadEventEnd: 1531486063003, redirectStart: 0, redirectEnd connectEnd: 1531486062915

> performance.timing

performance.timing去监测网络

```
connectStart: 1531486062896
     domComplete: 1531486063563
     domContentLoadedEventEnd: 1531486063262
     domContentLoadedEventStart: 1531486063261
     domInteractive: 1531486063261
     domLoading: 1531486063017
     domainLookupEnd: 1531486062896
     domainLookupStart: 1531486062896
     fetchStart: 1531486062879
     loadEventEnd: 1531486063564
     loadEvertStart: 1531486063563
     navigationStart: 1531486062874
     redirectEnd: 0
     redirectStart: 0
     requestStart: 1531486062925
     responseEnd: 1531486063008
     responseStart: 1531486062998
     secureConnectionStart: 1531486062902
     unloadEventEnd: 1531486063003
     unloadEventStart: 1531486063003
   ▶ __proto__: PerformanceTiming
amp、pwa 谷歌两大神器
```

UA就是 navigator.userAgent

```
> navigator.userAgent
  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML,
   like Gecko) Chrome/67.0.3396.99 Safari/537.36"
```