# 加餐四 | 页面性能工具：如何使用Performance?

2019-12-12 李兵

《浏览器工作原理与实践》 课程介绍 >

讲述：李兵

时长 13:50 大小 9.51M

你好，我是李兵。

在分析页面性能时，如果说 Audits 是道开胃菜，那么 Performance 才是正餐，之所这样说，主要是因为 Performance 可以记录站点在运行过程中的性能数据，有了这些性能数据，我们就可以回放整个页面的执行过程，这样就方便我们来定位和诊断每个时间段内页面的运行情况，从而有效帮助我们找出页面的性能瓶颈。

不同于 Audits，Perofrmance 不会给出性能得分，也不会给出优化建议，它只是单纯地采集性能数据，并将采集到的数据按照时间线的方式来展现，我们要做的就是依据原始数据来分析 Web 应用的性能问题。

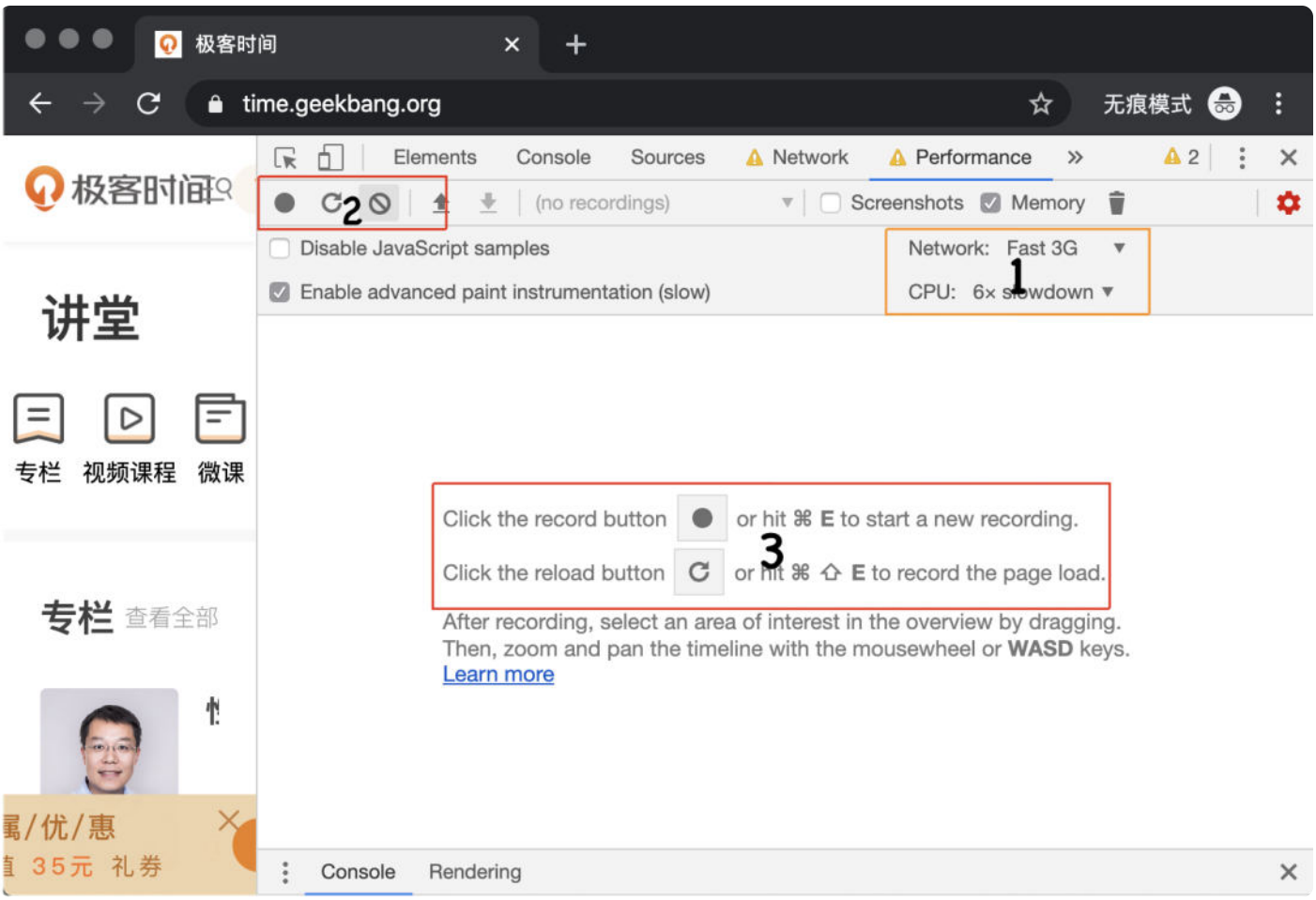
那么本节，我们就继续深入，聊聊如何使用 Performance。通常，使用 Performance 需要分三步走：

1. 第一步是配置 Performance；
2. 第二步是生成报告页；
3. 第三步就是人工分析报告页，并找出页面的性能瓶颈。

接下来，我会根据上面这三个步骤带你熟悉 Performace，并让你了解如何使用 Performance 来分析页面性能数据。

## 配置 Performance

我们在 Chrome 中任意打开一个站点，再打开 Chrome 开发者工具，然后选择 Performance 标签，最终效果如下图所示：




Performance 配置页

上图就是 Performance 的配置页，观察图中区域 1，我们可以设置该区域中的“Network”来限制网络加载速度，设置“CPU”来限制 CPU 的运算速度。通过设置，我们就可以在 Chrome 浏览器上来模拟手机等性能不高的设备了。在这里我将 CPU 的运算能力降低到了 **1/6**，将网络的加载速度设置为“**快的 3G(Fast 3G)**”用来模拟 3G 的网络状态。

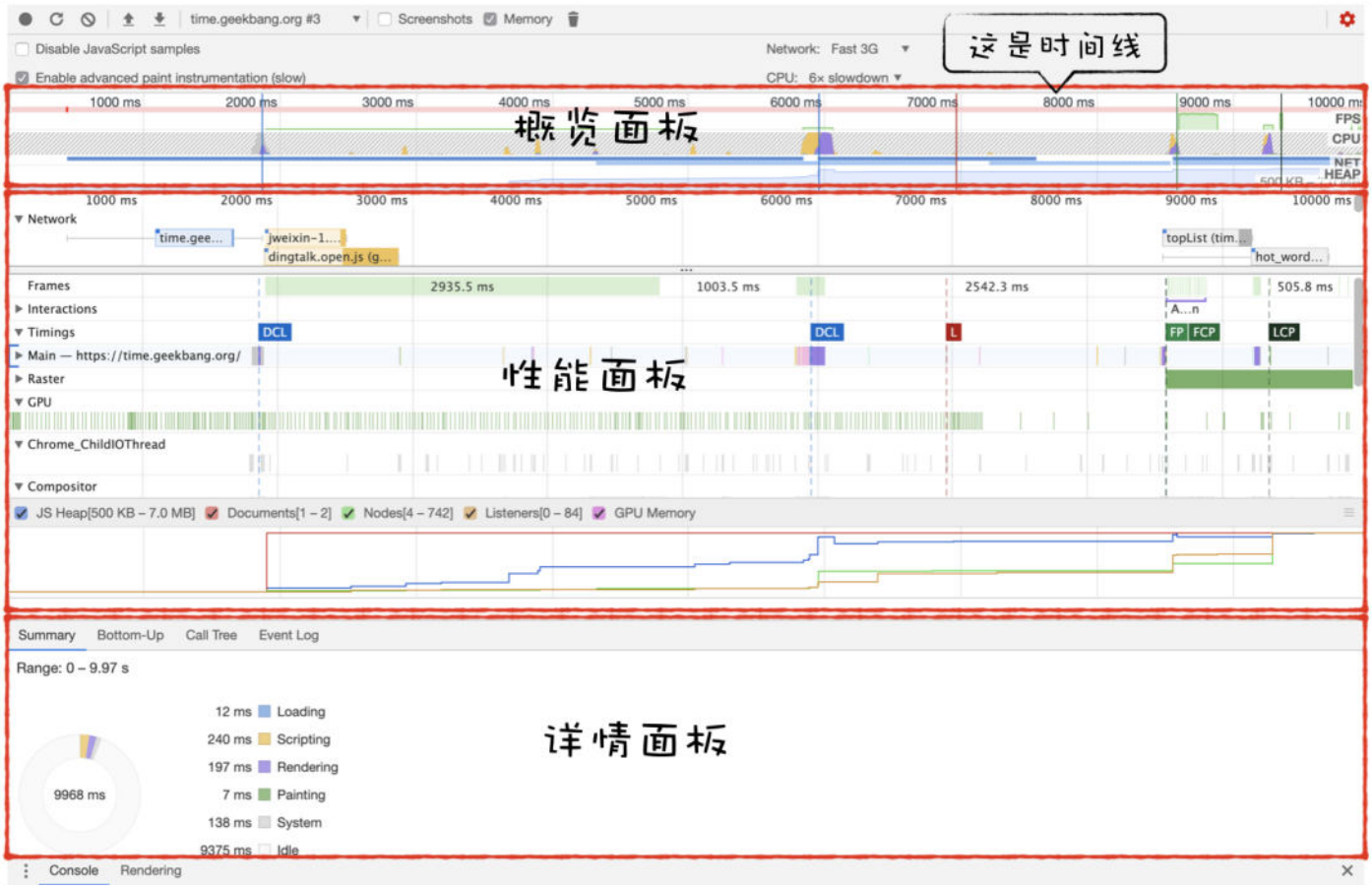
不同于 Audits 只能监控加载阶段的性能数据，Performance 还可以监控交互阶段的性能数据，不过 Performance 是分别录制这两个阶段的，你可以查看上图区域 2 和区域 3，我们可以看到这里有两个按钮，上面那个黑色按钮是用来记录交互阶段性能数据的，下面那个带箭头的圆圈形按钮用来记录加载阶段的性能数据。

另外你还要注意一点，这两种录制方式稍微有点不同：

- 当你**录制加载**阶段的性能数据时，Performance 会重新刷新页面，并等到页面完全渲染出来后，Performance 就会自动停止录制。
- 如果你是**录制交互阶段**的性能时，那么需要手动停止录制过程。

## 认识报告页

无论采用哪种方式录制，最终所生成的报告页都是一样的，如下图所示：




Performance 的报告页

观察上图的报告页，我们可以将它分为三个主要的部分，分别为**概览面板、性能指标面板和详情面板**。

要熟练掌握这三个面板，我们需要先明白时间线的概念，这是因为概览面板和性能指标面板都依赖于时间线。我们知道，Performance 按照时间的顺序来记录每个时间节点的性能数据，然后再按照时间顺序来展示这些性能数据，那么展示的时候就必然要引入时间线了。比如上图中我们录制了 10000 毫秒，那么它的时间线长度也就是 10000 毫秒，体现在上图中就是概览面板最上面那条线。

### 1. 概览面板

好了，引入了时间线，**Performance 就会将几个关键指标，诸如页面帧速 (FPS)、CPU 资源消耗、网络请求流量、V8 内存使用量 (堆内存) 等，按照时间顺序做成图表的形式展现出来，这就是概览面板**，你可以参看上图。

有了概览面板，我们就能一览几个关键的历史数据指标，进而能快速定位到可能存在问题的时间节点。那么如何定位可能存在问题的时间节点呢?

- 如果 FPS 图表上出现了红色块，那么就表示红色块附近渲染出一帧所需时间过久，帧的渲染时间过久，就有可能导致页面卡顿。
- 如果 CPU 图形占用面积太大，表示 CPU 使用率就越高，那么就有可能因为某个 JavaScript 占用太多的主线程时间，从而影响其他任务的执行。
- 如果 V8 的内存使用量一直在增加，就有可能是某种原因导致了内存泄漏。

除了以上指标以外，概览面板还展示加载过程中的几个关键时间节点，如 FP、LCP、DOMContentLoaded、Onload 等事件产生的时间点。这些关键时间点体现在了几条不同颜色的竖线上。

### 2. 性能面板

通常，我们通过概览面板来定位到可能存在问题的时间节点，接下来需要更进一步的数据，来分析导致该问题的原因，那么应该怎么分析呢?

这就需要引入**性能面板**了，在性能面板中，记录了非常多的性能指标项，比如 **Main** 指标记录渲染主线程的任务执行过程，**Compositor 指标**记录了合成线程的任务执行过程，**GPU 指标**记录了 GPU 进程主线程的任务执行过程。有了这些详细的性能数据，就可以帮助我们轻松地定位到页面的性能问题。

简而言之，**我们通过概览面板来定位问题的时间节点，然后再使用性能面板分析该时间节点内的性能数据**。具体地讲，比如概览面板中的 FPS 图表中出现了红色块，那么我们点击该红色块，性能面板就定位到该红色块的时间节点内了，你可以参考下图：


选择时间线上的一段

观察上图，我们发现性能面板的最上方也有一段时间线，比如上面这个时间线所展示的是从 360 毫秒到 480 毫秒，这段时间就是我们所定位到的时间节点，下面所展示的 Network、Main 等都是该时间节点内的详细数据。

如果你想要查看事件范围更广的性能指标，你只需要将鼠标放到时间线上，滚动鼠标滚轮就可以就行缩放了。如果放大之后，要查看的内容如果超出了屏幕，那么你可以点击鼠标左键来拖动时间线，直到找到需要查看的内容，你也可以通过键盘上的“WASD”四个键来进行缩放和位置的移动。

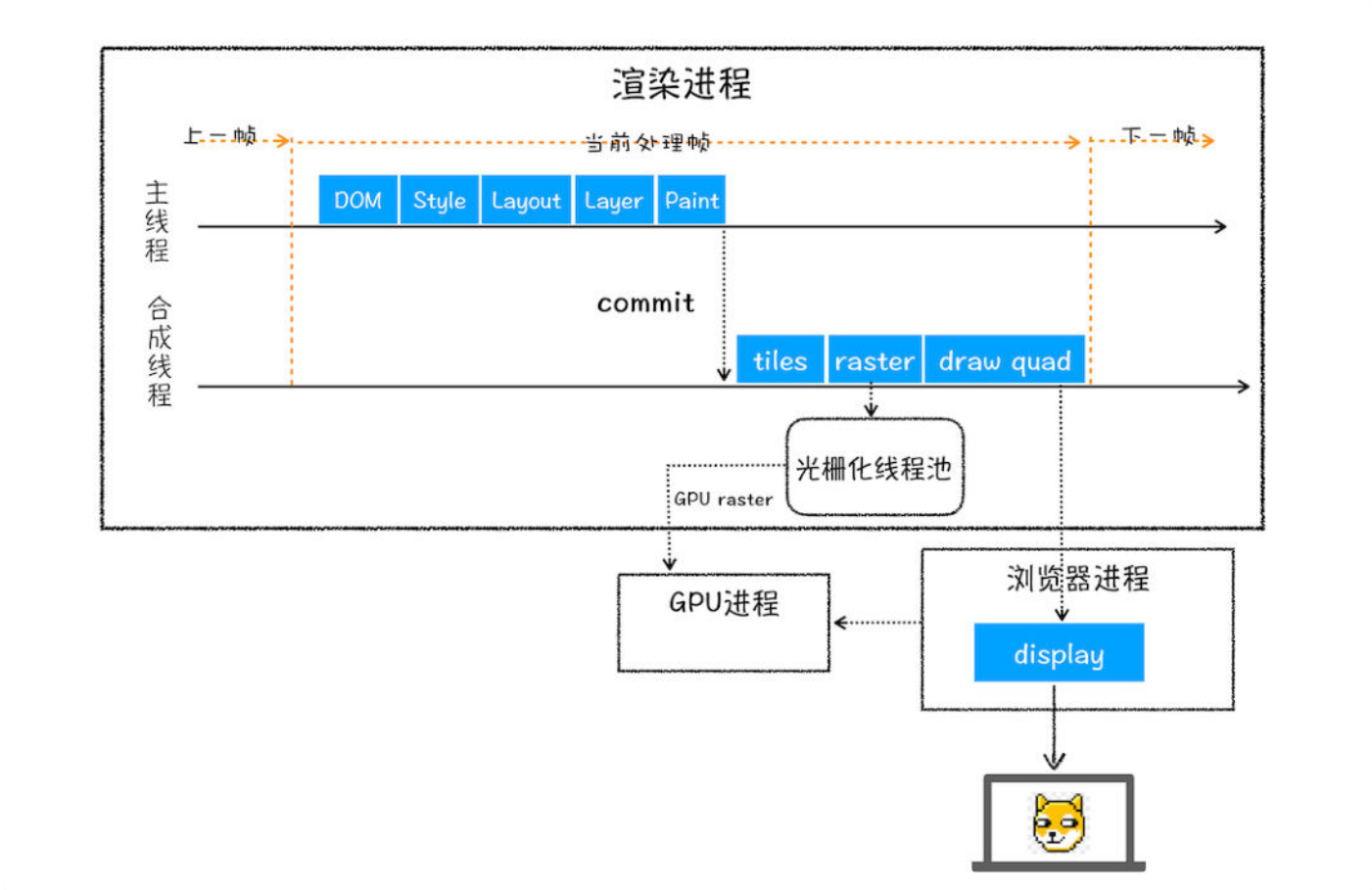
### 3. 解读性能面板的各项指标

好了，现在我们了解性能面板，它主要用来展现**特定时间段内的多种性能指标数据**。那么要分析这些指标数据，我们就要明白这些指标数据的含义，不过要弄明白它们却并非易事，因为要很好地理解它们，**你需要掌握渲染流水线、浏览器进程架构、导航流程等知识点。**

因此在介绍性能指标之前，我们还需要岔开一下，回顾下这些前置的知识点。

因为浏览器的渲染机制过于复杂，所以渲染模块在执行渲染的过程中会被划分为很多子阶段，输入的 HTML 数据经过这些子阶段，最后输出屏幕上的像素，我们把这样的一个处理流程叫做**渲染流水线**。一条完整的渲染流水线包括了解析 HTML 文件生成 DOM、解析 CSS 生成 CSSOM、执行 JavaScript、样式计算、构造布局树、准备绘制列表、光栅化、合成、显示等一系列操作。

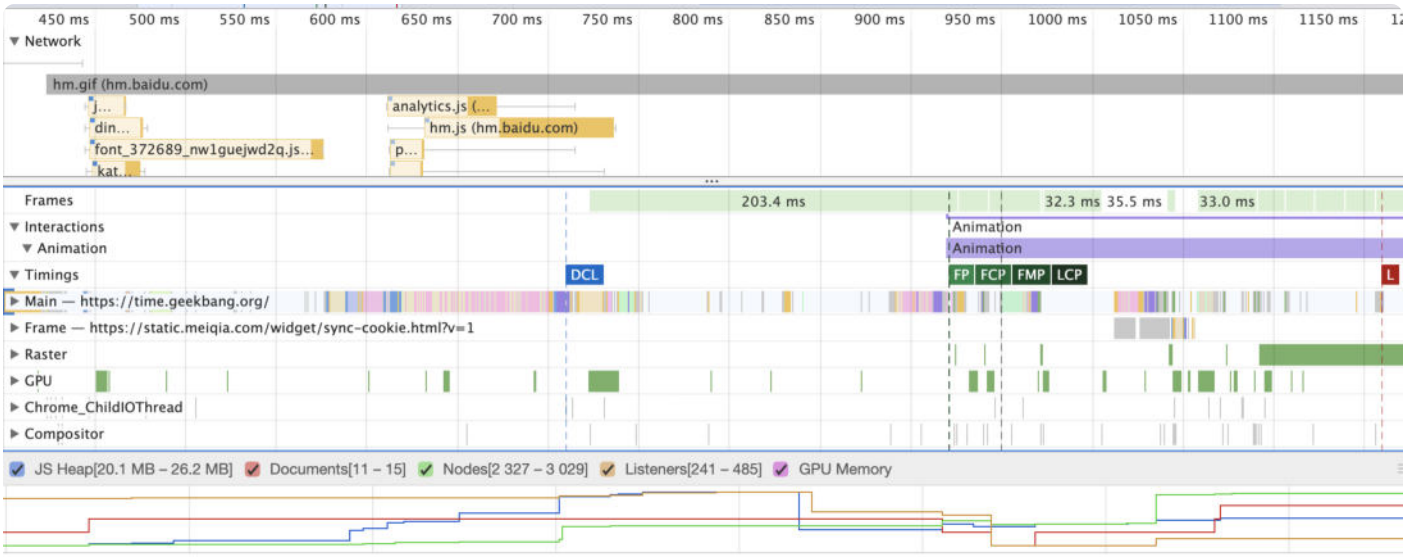
渲染流水线主要是在渲染进程中执行的，在执行渲染流水线的过程中，渲染进程又需要网络进程、浏览器进程、GPU 等进程配合，才能完成如此复杂的任务。另外在渲染进程内部，又有很多线程来相互配合。具体的工作方式你可以参考下图：




渲染流水线

关于渲染流水线和浏览器进程架构的详细内容我在前面的章节中也做了很多介绍，特别是《05 | 渲染流程（上）：HTML、CSS 和 JavaScript，是如何变成页面的？》和《06 | 渲染流程（下）：HTML、CSS 和 JavaScript，是如何变成页面的？》这两节，你可以去回顾下相关章节的课程内容。

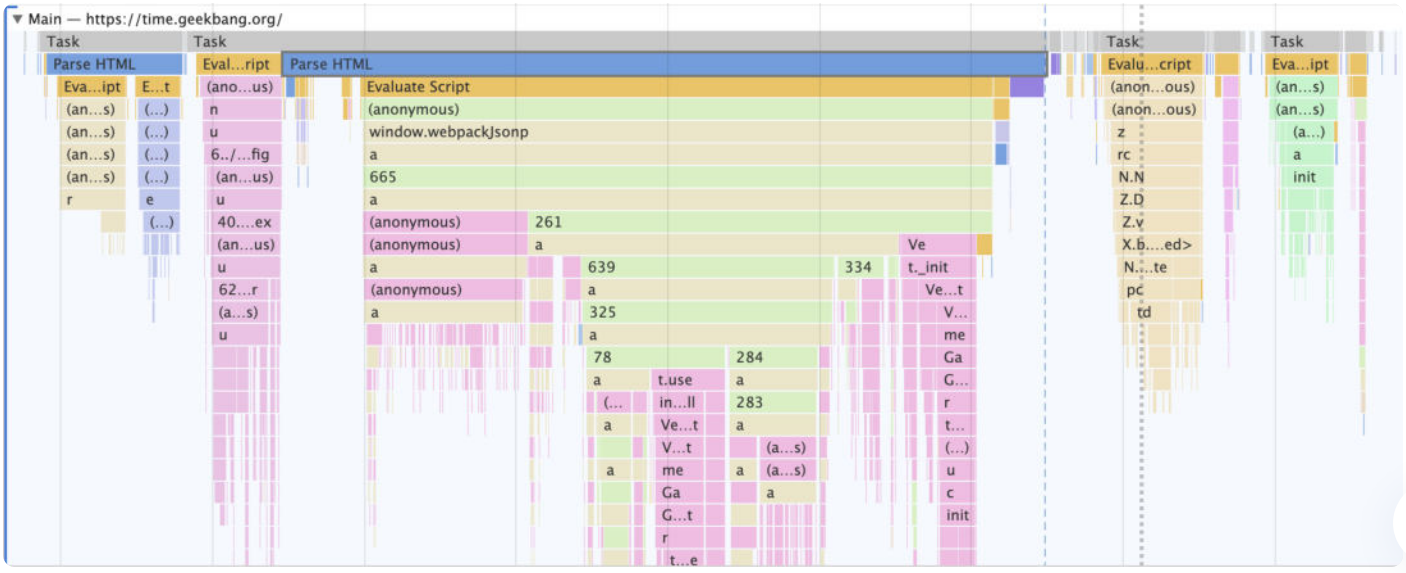
好了，我们简要回顾了渲染流水线和浏览器的进程架构，那么现在回归正题，来分析下性能面板各个指标项的具体含义。你可以参考下图：




性能面板

观看上图的左边，我们可以看到它是由很多性能指标项组成的，比如 Network、Frames、Main 等，下面我们就来一一分析这些性能指标项的含义。

我们先看最为重要的 **Main 指标**，它记录了渲染进程的主线程的任务执行记录，在 Perofrmace 录制期间，在渲染主线程上执行的所有记录都可以通过 Main 指标来查看，你可以通过点击 Main 来展开主进程的任务执行记录，具体你可以观察下图：




Main 指标

观察上图，一段段横条代表执行一个个任务，长度越长，花费的时间越多；竖向代表该任务的执行记录。通过前面章节的学习，我们知道主线程上跑了特别多的任务，诸如渲染流水线的大部分流程，JavaScript 执行、V8 的垃圾回收、定时器设置的回调任务等等，因此 Main 指标的内容非常多，而且非常重要，所以我们在使用 Perofrmance 的时候，大部分时间都是在分析 Main 指标。Main 指标的内容特别多，我会在下一节对它做详细分析。

通过渲染流水线，我们知道了渲染主线程在生成层树 (LayerTree) 之后，然后根据层树生成每一层的绘制列表，我们把这个过程称为**绘制 (Paint)**。在绘制阶段结束之后，渲染主线程会将这些绘列表制**提交 (commit)给合成线程，并由合成线程合成出来漂亮的页面。因此，监控合成线程的任务执行记录也相对比较重要，所以 Chrome 又在性能面板中引入了Compositor 指标**，也就是合成线程的任务执行记录。

在合成线程执行任务的过程中，还需要 GPU 进程的配合来生成位图，我们把这个 GPU 生成位图的过程称为**光栅化**。如果合成线程直接和 GPU 进程进行通信，那么势必会阻塞后面的合成任务，因此合成线程又维护了一个**光栅化线程池 (Raster)**，用来让 GPU 执行光栅化的任务。因为光栅化线程池和 GPU 进程中的任务执行也会影响到页面的性能，所以性能面板也添加了这两个指标，分别是 **Raster 指标**和 **GPU 指标**。因为 Raster 是线程池，所以如果你点开 Raster 项，可以看到它维护了多个线程。

渲染进程中除了有主线程、合成线程、光栅化线程池之外，还维护了一个 IO 线程，具体细节你可以参考《15 | 消息队列和事件循环：页面是怎么“活”起来的？》这篇文章。该 IO 线程主要用来接收用户输入事件、网络事件、设备相关等事件，如果事件需要渲染主线程来处理，那么 IO 线程还会将这些事件转发给渲染主线程。在性能面板上，**Chrome_ChildIOThread 指标**对应的就是 IO 线程的任务记录。

好了，以上介绍的都是渲染进程和 GPU 进程的任务记录，除此之外，性能面板还添加了其他一些比较重要的性能指标。

第一个是 **Network 指标**，网络记录展示了页面中的每个网络请求所消耗的时长，并以瀑布流的形式展现。这块内容和网络面板的瀑布流类似，之所以放在性能面板中是为了方便我们和其他指标对照着分析。

第二个是 **Timings 指标**，用来记录一些关键的时间节点在何时产生的数据信息，关于这些关键时间点的信息我们在上一节也介绍过了，诸如 FP、FCP、LCP 等。

第三个是 **Frames 指标**，也就是浏览器生成每帧的记录，我们知道页面所展现出来的画面都是由渲染进程一帧一帧渲染出来的，帧记录就是用来记录渲染进程生成所有帧信息，包括了渲染出每帧的时长、每帧的图层构造等信息，你可以点击对应的帧，然后在详细信息面板里面查看具体信息。

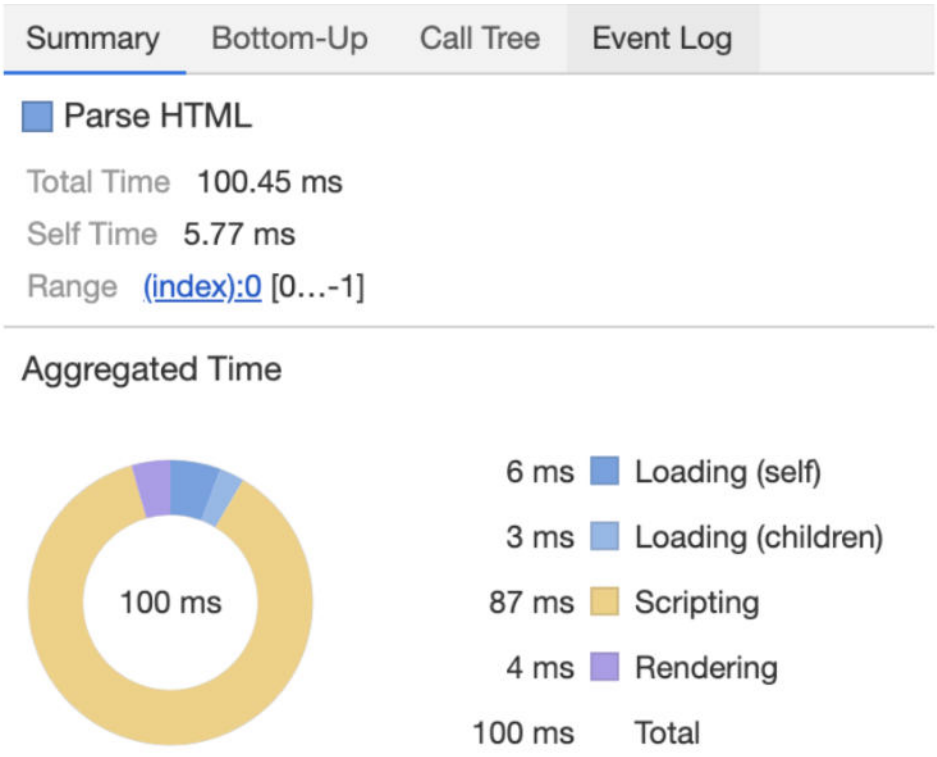
第四个是 **Interactions 指标**，用来记录用户交互操作，比如点击鼠标、输入文字等交互信息。

### 4. 详情面板

通过性能面板的分析，我们知道了性能面板记录了多种指标的数据信息，并且以图形的形式展现在性能面板上。

具体地讲，比如主线程上执行了解析 HTML(ParserHTML) 的任务，对应于性能面板就是一个长条和多个竖条组成图形。通过上面的图形我们只能得到一个大致的信息，如果想要查看这些记录的详细信息，就需要引入**详情面板**了。

你可以通过在性能面板中选中性能指标中的任何历史数据，然后选中记录的细节信息就会展现在详情面板中了。比如我点击了 Main 指标中的 ParserHTML 这个过程，下图就是详情面板展现该过程的详细信息。



由于详情面板所涉及的内容很多，而且每种指标的详细内容都有所不同，所以本节我就不展开来讲了。另外你可以去 Google 的官方网站查看 Performance 的一些基础使用信息。

## 总结

好了，本节内容就介绍到这里，下面我来总结下本文的主要内容：

本节我们首先介绍了如何去配置 Performance 并生成报告页，然后我们将焦点放在了如何解读报告页上。

之后我们介绍了报告页面主要分为三个部分，概览面板、性能面板和详情面板。

我们可以通过概览面板来定位问题的时间节点，然后再使用性能面板分析该时间节点内的性能数据。不过在分析数据时，我们需要弄明白性能面板内各项数据指标的含义，要了解这些，需要了解浏览器渲染流水线、浏览器的进程架构等知识点，因此结合这些知识点，我们接下来分析了性能面板的各项指标的含义。

其中最为重要的是 Main 指标，它记录了渲染主线程上的任务执行情况，不过这块细节内容会非常多，所以我们会在下一节来介绍。

最后我们还介绍了每个指标项的内容都有详细数据，这些详细数据是通过详情面板来展现，你只需要通过性能面板点击相应的数据，就能通过详情面板来查看详细数据了。不过详情面板所涉及的数据也是非常多的，所以本文也就没对详情面板做过深的介绍了。

我把 Performance 比喻成一张网，它能把我们在前面章节中很多知识点都网罗起来，并应用到实践中。

## 思考题

那么今天留给你的任务是，多找几个站点，使用 Performance 来录制加载过程和交互过程，并熟悉报告页面中的各项性能指标，如果有遇到不明白的问题，欢迎在留言区留言与我交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

上一篇　加餐三 | 加载阶段性能：使用Audits来优化Web性能

下一篇　加餐五 | 性能分析工具：如何分析Performance中的Main指标?



## 精选留言 (18)

写留言

wubinsheng
2019-12-18

老师如期而至，感动啊！

作者回复: 😄

11

Mr. Cheng
2019-12-12

每当读到‘会在下一节详细介绍’就莫名的兴奋^_^，感谢老师辛苦的准备加餐课程

9



Geek_f091d5
2020-07-04

不知道老师是不是看过《掌握综合认知能力-面向专业技术的四元教学设计模式》，你的文章比较符合书中的教学设计原则。

4

宗麒麟
2020-05-01

这个performance 面板研究很久了也没搞明白，chrome版本不同总是有些细微的差异，这个面板我总是云里雾里的，朦朦胧胧

4

Tr
2021-09-03

真的写的太好了，好文章 总是让人，反复翻阅

2

zoro
2020-07-28

讲得真好，这篇文章买得真值

2

-_-lll
2019-12-17

V8 的内存使用量在哪memory吗

共 1 条评论>

1

tokey
2019-12-12

老师，跨域的请求，在浏览器 network 选项里观察，有时候可以看到 opinions 请求，有时候直接返回资源，这是为什么啊?

共 6 条评论>

1

Alan He

2022-02-09

比如main指标里，横向记录是每一个任务，单位是时间，纵向是每个任务包含的子任务吗，上下都是包含关系吗?

以上是我的理解。

Geek_panda

2021-08-06

请教下，合成线程、光栅化线程池 (Raster)、gpu进程三者的关系能再详细解释下吗

木偶先生

2021-03-12

考试我想问一下，在chrome中如何查看图片占用的内存呢

皮特尔

2020-10-12

这篇内容很偏实战，应该多用用

sh

2020-05-31

感谢老师

陈十二

2019-12-16

干货满满的一节呀，很期待下一节，老师辛苦了

许童童

2019-12-15

老师真是良心啊

YBB
2019–12–13

辛苦老师了


**許敲敲**
2019–12–12

Chrome真的功能太强大了


**段.**
2019–12–12

深夜更新 给老师点赞

作者回复: 早