

01 | Chrome架构：仅仅打开了1个页面，为什么有4个进程？

2019-08-05 李兵

《浏览器工作原理与实践》

课程介绍 >



讲述：李兵

时长 18:36 大小 17.04M

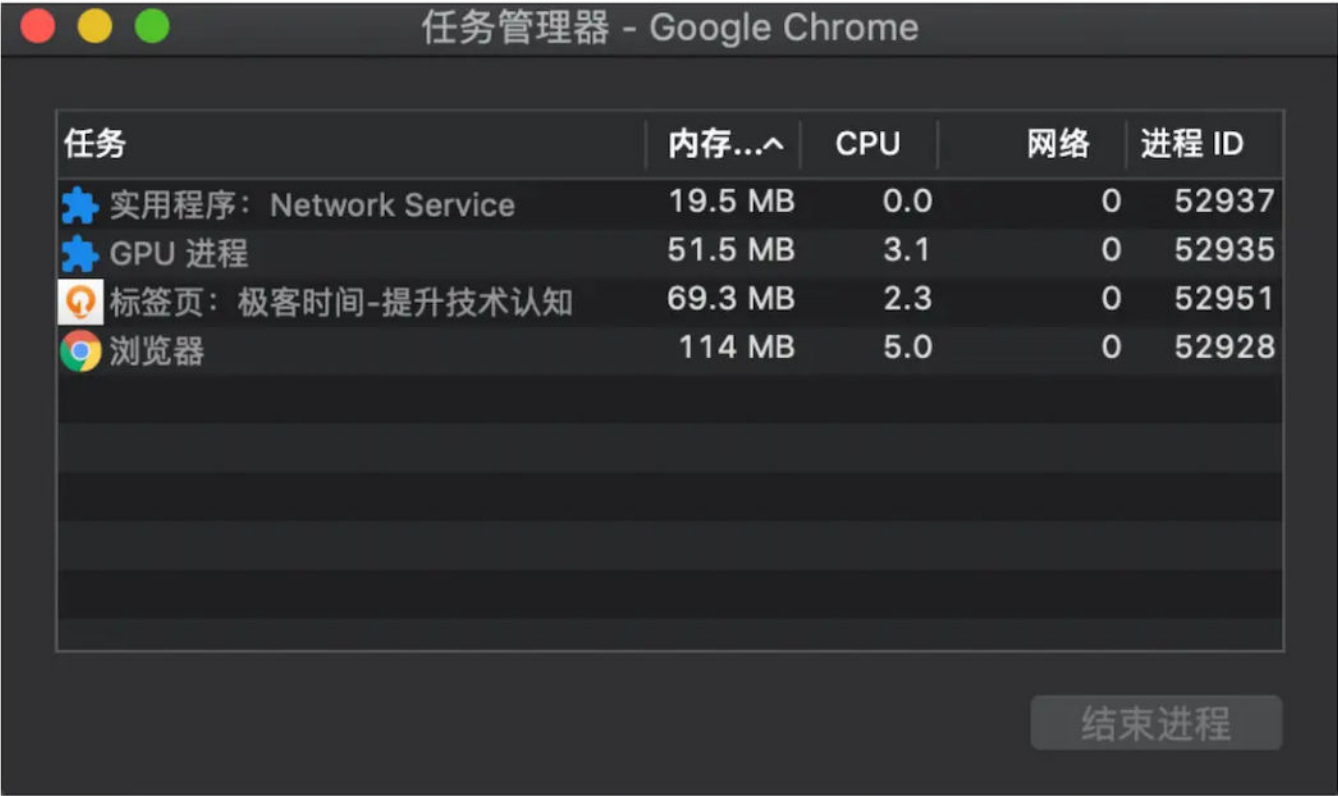


无论你是想要设计高性能 Web 应用，还是要优化现有的 Web 应用，你都需要了解浏览器中的网络流程、页面渲染过程，JavaScript 执行流程，以及 Web 安全理论，而这些功能是分散在浏览器的各个功能组件中的，比较多、比较散，要怎样学习才能掌握呢？通过浏览器的多进程架构的学习，你就可以把这些分散的知识点串起来，组成一张网，从而让自己能站在更高的维度去理解 Web 应用。

因此，学习浏览器的多进程架构是很有必要的。需要说明的是，在本专栏中，我所有的分析都是基于 Chrome 浏览器的。那么多浏览器，为什么偏偏选择 Chrome 浏览器呢？因为 Chrome、微软的 Edge 以及国内的大部分主流浏览器，都是基于 Chromium 二次开发而来；而 Chrome 是 Google 的官方发行版，特性和 Chromium 基本一样，只存在一些产品层面差异；再加上 Chrome 是目前世界上使用率最高的浏览器，所以 **Chrome 最具代表性**。

在开始之前，我们一起看下，Chrome 打开一个页面需要启动多少进程？你可以点击 Chrome 浏览器右上角的“选项”菜单，选择“更多工具”子菜单，点击“任务管理器”，这将打开

Chrome 的任务管理器的窗口，如下图：



Chrome 的任务管理器窗口

和 Windows 任务管理器一样，Chrome 任务管理器也是用来展示运行中 Chrome 使用的进程信息的。从图中可以看到，Chrome 启动了 4 个进程，你也许会好奇，只是打开了 1 个页面，为什么要启动这么多进程呢？

在解答这个问题之前，我们需要了解一下进程的概念，不过由于好多人容易把进程和线程的概念混淆，从而影响后续其他概念的理解，所以这里我就将这两个概念以及它们之间的关系一并为你讲解下。

进程和线程

不过，在介绍进程和线程之前，我需要先讲解下什么是并行处理，因为如果你理解了并行处理的概念，那么再理解进程和线程之间的关系就会变得轻松许多。

什么是并行处理

计算机中的并行处理就是同一时刻处理多个任务，比如我们要计算下面这三个表达式的值，并显示出结果。



```
1 A = 1+2
2 B = 20/5
3 C = 7*8
```

在编写代码的时候，我们可以把这个过程拆分为四个任务：

- **任务 1** 是计算 $A=1+2$ ；
- **任务 2** 是计算 $B=20/5$ ；
- **任务 3** 是计算 $C=7*8$ ；
- **任务 4** 是显示最后计算的结果。

正常情况下程序可以使用**单线程**来处理，也就是分四步按照顺序分别执行这四个任务。

如果采用**多线程**，会怎么样呢？我们只需分“两步走”：第一步，使用三个线程同时执行前三个任务；第二步，再执行第四个显示任务。

通过对比分析，你会发现用单线程执行需要四步，而使用多线程只需要两步。因此，**使用并行处理能大大提升性能**。

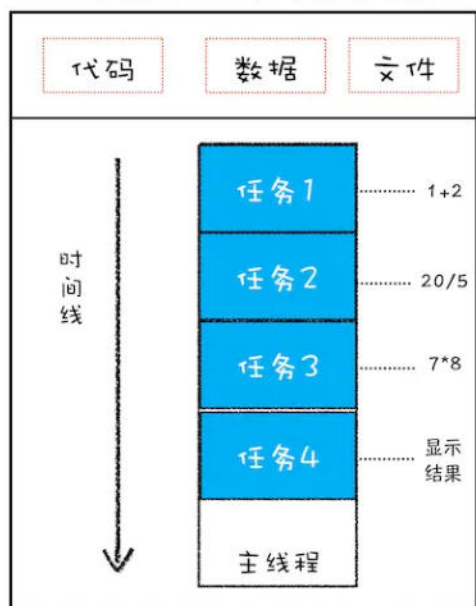
线程 VS 进程

多线程可以并行处理任务，但是**线程是不能单独存在的**，它是由**进程来启动和管理的**。那什么又是进程呢？

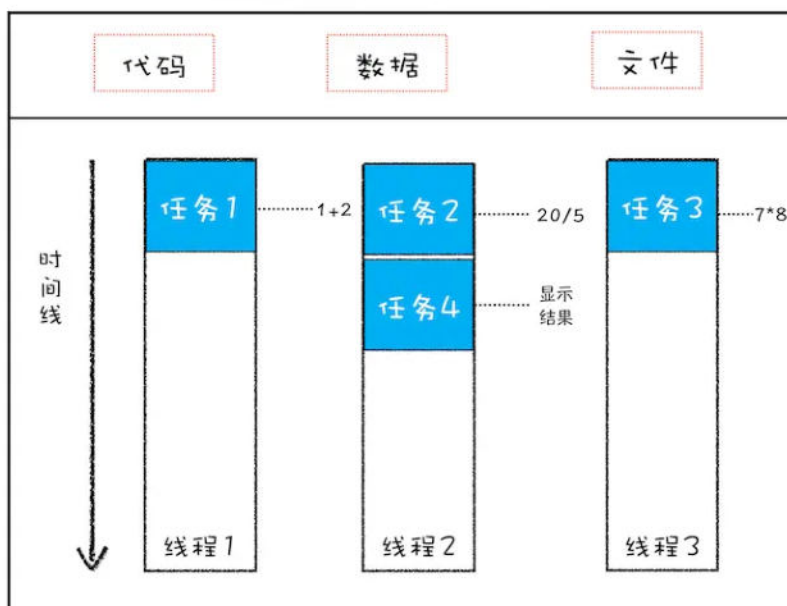
一个进程就是一个程序的运行实例。详细解释就是，启动一个程序的时候，操作系统会为该程序创建一块内存，用来存放代码、运行中的数据和**一个执行任务的主线程**，我们把这样的一个运行环境叫**进程**。

为了让你更好地理解上述计算过程，我画了下面这张对比图：

进程A：单线程处理



进程B：多线程处理



单线程与多线程的进程对比图

从图中可以看到，**线程是依附于进程的**，而进程中使用多线程并行处理能提升运算效率。

总结来说，进程和线程之间的关系有以下 4 个特点。

1. 进程中的任意一线程执行出错，都会导致整个进程的崩溃。

我们可以模拟以下场景：

```
1 A = 1+2
2 B = 20/0
3 C = 7*8
```

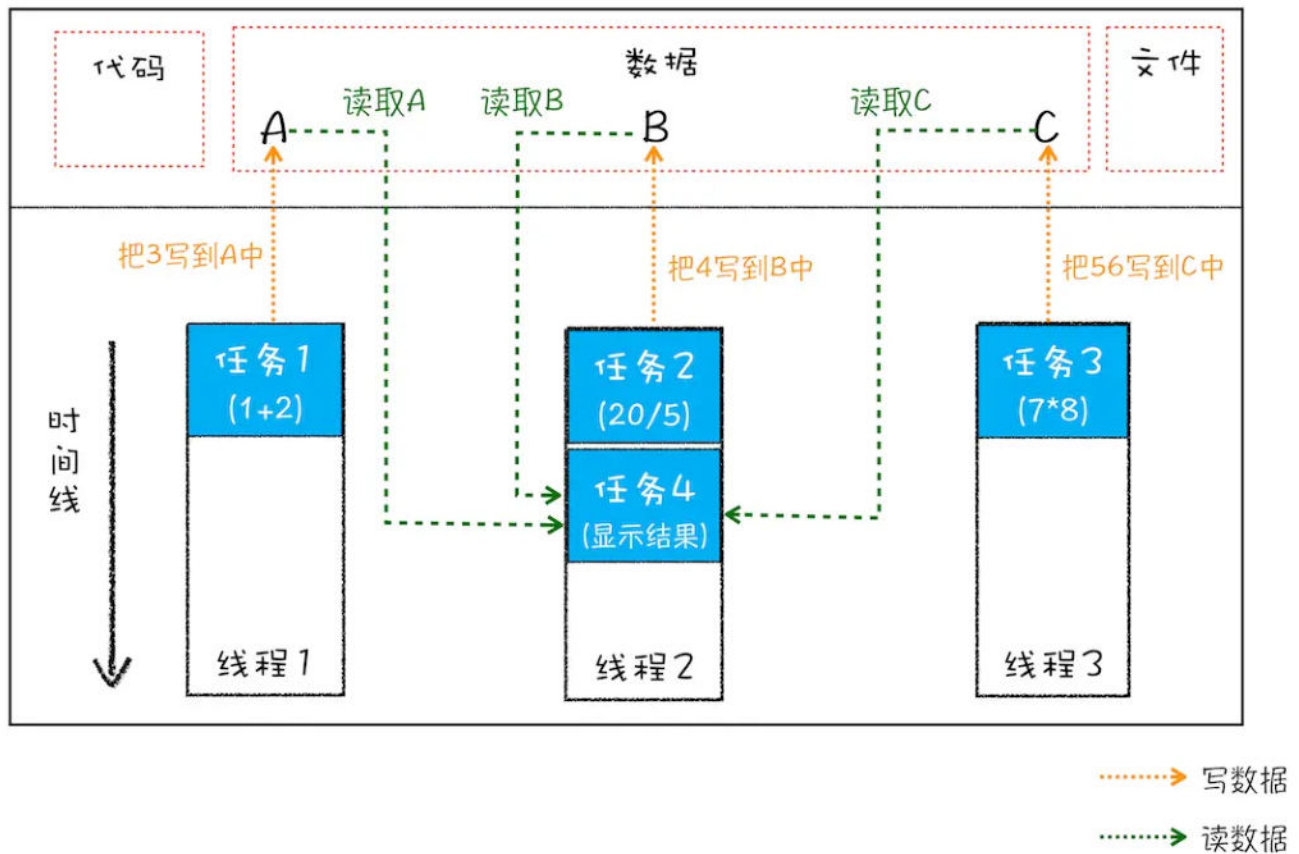
复制代码

我把上述三个表达式稍作修改，在计算 B 的值的时候，我把表达式的分母改成 0，当线程执行到 $B = 20/0$ 时，由于分母为 0，线程会执行出错，这样就会导致整个进程的崩溃，当然另外两个线程执行的结果也没有了。

2. 线程之间共享进程中的数据。

如下图所示，线程之间可以对进程的公共数据进行读写操作。

进程B：多线程处理



线程之间共享进程中的数据示意图

从上图可以看出，线程 1、线程 2、线程 3 分别把执行的结果写入 A、B、C 中，然后线程 2 继续从 A、B、C 中读取数据，用来显示执行结果。

3. 当一个进程关闭之后，操作系统会回收进程所占用的内存。

当一个进程退出时，操作系统会回收该进程所申请的所有资源；即使其中任意线程因为操作不当导致内存泄漏，当进程退出时，这些内存也会被正确回收。

比如之前的 IE 浏览器，支持很多插件，而这些插件很容易导致内存泄漏，这意味着只要浏览器开着，内存占用就有可能会越来越多，但是当关闭浏览器进程时，这些内存就都会被系统回收掉。

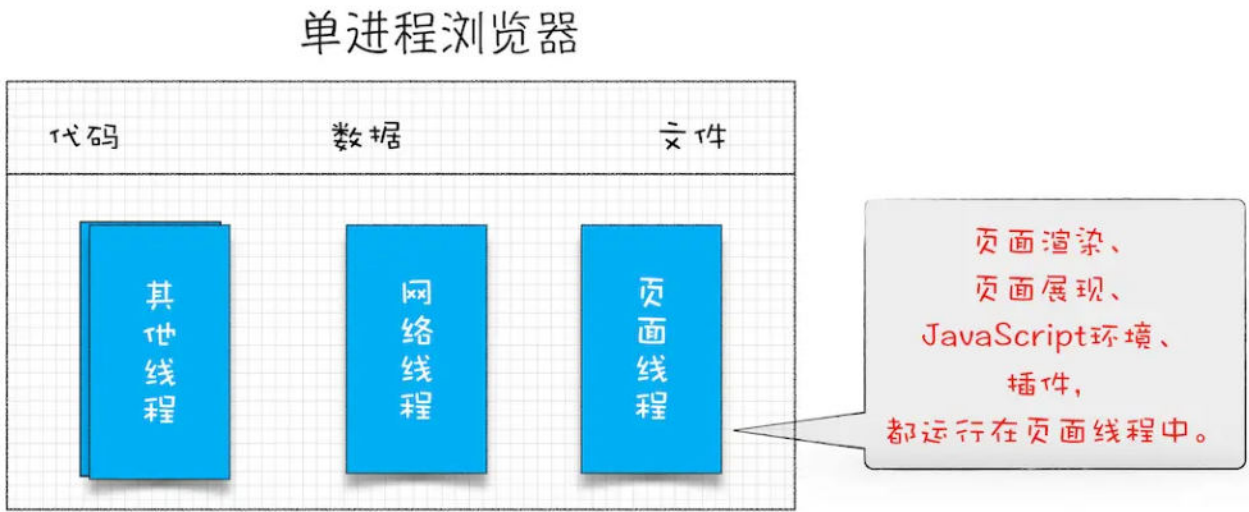
4. 进程之间的内容相互隔离。

进程隔离是为保护操作系统中进程互不干扰的技术，每一个进程只能访问自己占有的数据，也就避免出现进程 A 写入数据到进程 B 的情况。正是因为进程之间的数据是严格隔离的，所以

一个进程如果崩溃了，或者挂起了，是不会影响到其他进程的。如果进程之间需要进行数据的通信，这时候，就需要使用用于进程间通信（IPC）的机制了。

单进程浏览器时代

在了解了进程和线程之后，我们再来一起看下单进程浏览器的架构。顾名思义，**单进程浏览器是指浏览器的所有功能模块都是运行在同一个进程里**，这些模块包含了网络、插件、JavaScript 运行环境、渲染引擎和页面等。其实早在 2007 年之前，市面上浏览器都是单进程的。单进程浏览器的架构如下图所示：



单进程浏览器架构示意图

如此多的功能模块运行在一个进程里，是导致单进程浏览器**不稳定、不流畅和不安全**的一个主要因素。下面我就来一一分析下出现这些问题的原因。

问题 1：不稳定

早期浏览器需要借助于**插件**来实现诸如 Web 视频、Web 游戏等各种强大的功能，但是插件是最容易出问题的模块，并且还运行在浏览器进程之中，所以一个插件的意外崩溃会引起整个浏览器的崩溃。

除了插件之外，**渲染引擎模块**也是不稳定的，通常一些复杂的 JavaScript 代码就有可能引起渲染引擎模块的崩溃。和插件一样，渲染引擎的崩溃也会导致整个浏览器的崩溃。

问题 2：不流畅

从上面的“单进程浏览器架构示意图”可以看出，所有页面的渲染模块、JavaScript 执行环境以及插件都是运行在同一个线程中的，这就意味着同一时刻只能有一个模块可以执行。

比如，下面这个无限循环的脚本：

```
1 function freeze() {  
2   while (1) {  
3     console.log("freeze");  
4   }  
5 }  
6 freeze();
```

 复制代码

如果让这个脚本运行在一个单进程浏览器的页面里，你感觉会发生什么？

因为这个脚本是无限循环的，所以当其执行时，它会独占整个线程，这样导致其他运行在该线程中的模块就没有机会被执行。因为浏览器中所有的页面都运行在该线程中，所以这些页面都没有机会去执行任务，这样就会导致整个浏览器失去响应，变卡顿。这块内容要继续往深的地方讲就到页面的事件循环系统了，具体相关内容我会在后面的模块中为你深入讲解。

除了上述**脚本**或者**插件**会让单进程浏览器变卡顿外，**页面的内存泄漏**也是单进程变慢的一个重要原因。通常浏览器的内核都是非常复杂的，运行一个复杂点的页面再关闭页面，会存在内存不能完全回收的情况，这样导致的问题是使用时间越长，内存占用越高，浏览器会变得越慢。

问题 3：不安全

这里依然可以从插件和页面脚本两个方面来解释该原因。

插件可以使用 C/C++ 等代码编写，通过插件可以获取到操作系统的任意资源，当你在页面运行一个插件时也就意味着这个插件能完全操作你的电脑。如果是个恶意插件，那么它就可以释放病毒、窃取你的账号密码，引发安全性问题。

至于页面脚本，它可以通过浏览器的漏洞来获取系统权限，这些脚本获取系统权限之后也可以对你的电脑做一些恶意的东西，同样也会引发安全问题。



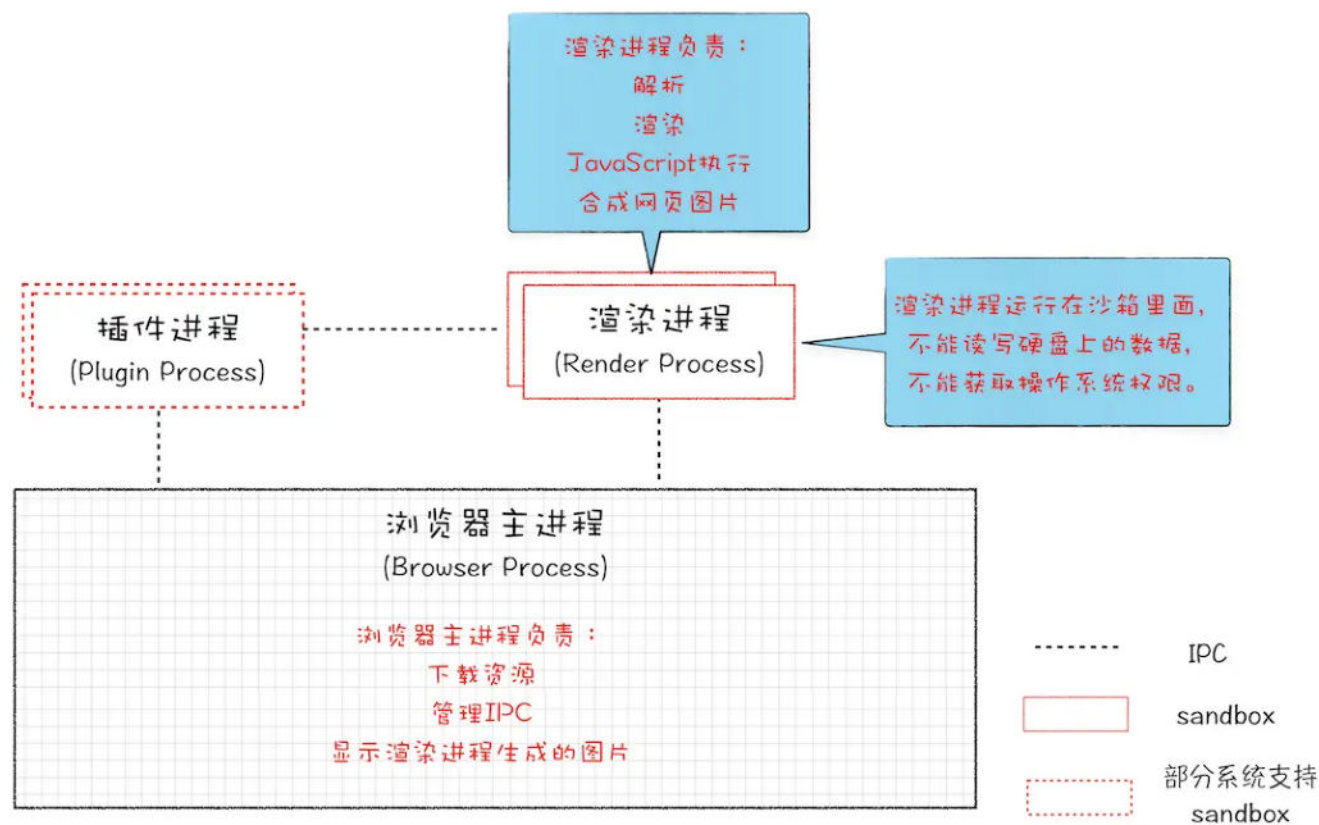
以上这些就是当时浏览器的特点，不稳定，不流畅，而且不安全。这是一段不堪回首的过去，也许你没有经历过，不过你可以想象一下这样的场景：当你正在用浏览器打开多个页面时，突然某个页面崩溃了或者失去响应，随之而来的是整个浏览器的崩溃或者无响应，然后你发现你给老板写的邮件页面也随之消失了，这时你的心情会不会和页面一样崩溃呢？

多进程浏览器时代

好在现代浏览器已经解决了这些问题，是如何解决的呢？这就得聊聊我们这个“多进程浏览器时代”了。

早期多进程架构

你可以先看看下面这张图，这是 2008 年 Chrome 发布时的进程架构。



早期 Chrome 进程架构图

从图中可以看出，Chrome 的页面是运行在单独的渲染进程中的，同时页面里的插件也是运行在单独的插件进程之中，而进程之间是通过 IPC 机制进行通信（如图中虚线部分）。



我们先看看如何解决不稳定的问题。由于进程是相互隔离的，所以当页面或者插件崩溃时，影响到的仅仅是当前的页面进程或者插件进程，并不会影响到浏览器和其他页面，这就完美地解决了页面或者插件的崩溃会导致整个浏览器崩溃，也就是不稳定的问题。

接下来再来看看不流畅的问题是如何解决的。同样，JavaScript 也是运行在渲染进程中的，所以即使 JavaScript 阻塞了渲染进程，影响到的也只是当前的渲染页面，而并不会影响浏览器和其他页面，因为其他页面的脚本是运行在它们自己的渲染进程中的。所以当我们再在 Chrome 中运行上面那个死循环的脚本时，没有响应的仅仅是当前的页面。

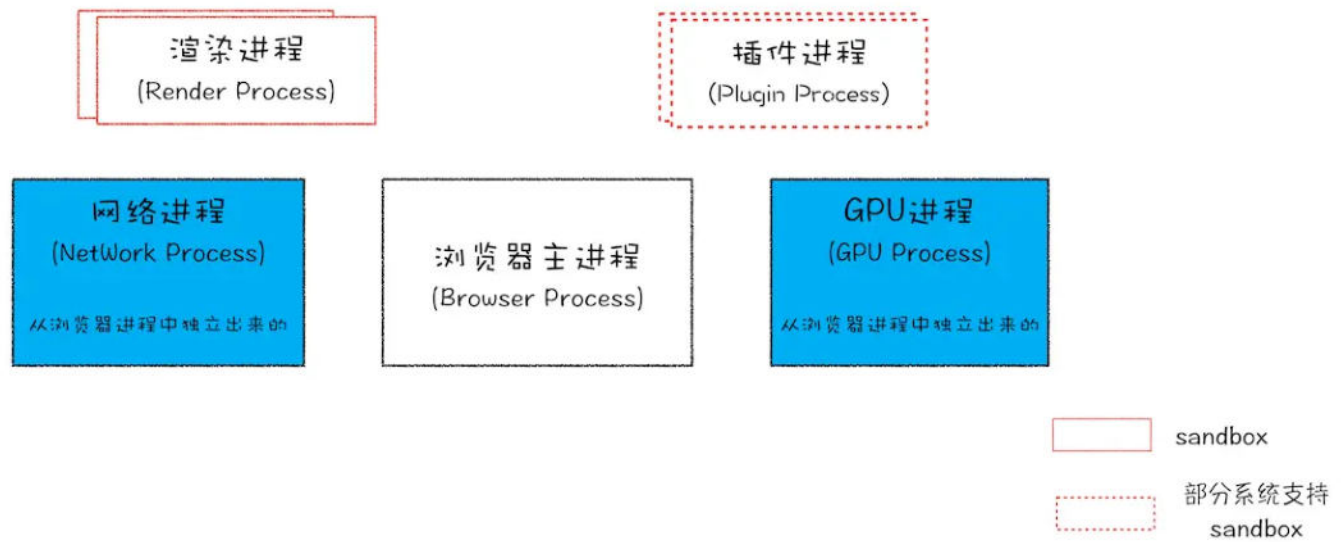
对于内存泄漏的解决方法那就更简单了，因为当关闭一个页面时，整个渲染进程也会被关闭，之后该进程所占用的内存都会被系统回收，这样就轻松解决了浏览器页面的内存泄漏问题。

最后我们再来看看上面的两个安全问题是怎么解决的。采用多进程架构的额外好处是可以使用**安全沙箱**，你可以把沙箱看成是操作系统给进程上了一把锁，沙箱里面的程序可以运行，但是不能在你的硬盘上写入任何数据，也不能在敏感位置读取任何数据，例如你的文档和桌面。Chrome 把插件进程和渲染进程锁在沙箱里面，这样即使在渲染进程或者插件进程里面执行了恶意程序，恶意程序也无法突破沙箱去获取系统权限。

好了，分析完早期的 Chrome 浏览器后，相信你已经了解了浏览器采用多进程架构的必要性。

目前多进程架构

不过 Chrome 的发展是滚滚向前的，相较之前，目前的架构又有了很多新的变化。我们先看看最新的 Chrome 进程架构，你可以参考下图：



最新的 Chrome 进程架构图

从图中可以看出，最新的 Chrome 浏览器包括：1 个浏览器（Browser）主进程、1 个 GPU 进程、1 个网络（NetWork）进程、多个渲染进程和多个插件进程。

下面我们来逐个分析下这几个进程的功能。

- **浏览器进程**。主要负责界面显示、用户交互、子进程管理，同时提供存储等功能。
- **渲染进程**。核心任务是将 HTML、CSS 和 JavaScript 转换为用户可以与之交互的网页，排版引擎 Blink 和 JavaScript 引擎 V8 都是运行在该进程中，默认情况下，Chrome 会为每个 Tab 标签创建一个渲染进程。出于安全考虑，渲染进程都是运行在沙箱模式下。
- **GPU 进程**。其实，Chrome 刚开始发布的时候是没有 GPU 进程的。而 GPU 的使用初衷是为了实现 3D CSS 的效果，只是随后网页、Chrome 的 UI 界面都选择采用 GPU 来绘制，这使得 GPU 成为浏览器普遍的需求。最后，Chrome 在其多进程架构上也引入了 GPU 进程。
- **网络进程**。主要负责页面的网络资源加载，之前是作为一个模块运行在浏览器进程里面的，直至最近才独立出来，成为一个单独的进程。
- **插件进程**。主要是负责插件的运行，因插件易崩溃，所以需要通过插件进程来隔离，以保证插件进程崩溃不会对浏览器和页面造成影响。



讲到这里，现在你应该就可以回答文章开头提到的问题了：仅仅打开了 1 个页面，为什么有 4 个进程？因为打开 1 个页面至少需要 1 个网络进程、1 个浏览器进程、1 个 GPU 进程以及 1

个渲染进程，共 4 个；如果打开的页面有运行插件的话，还需要再加上 1 个插件进程。

不过凡事都有两面性，虽然多进程模型提升了浏览器的稳定性、流畅性和安全性，但同样不可避免地带来了一些问题：

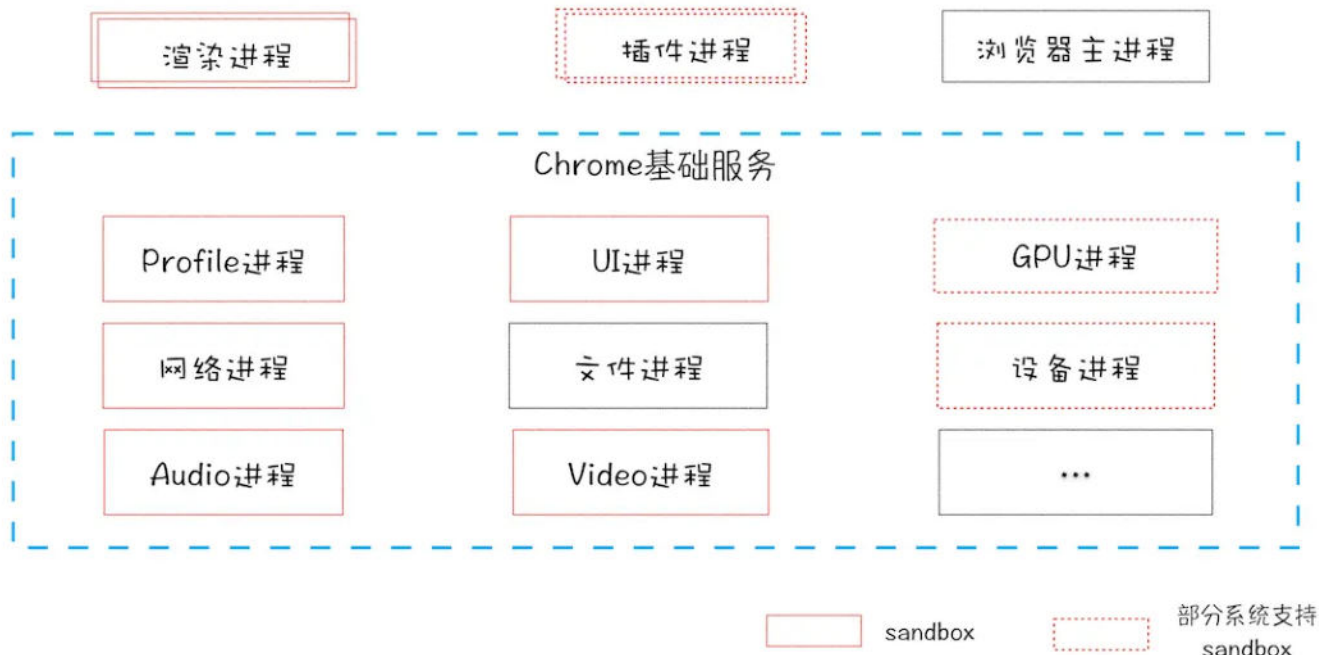
- **更高的资源占用。**因为每个进程都会包含公共基础结构的副本（如 JavaScript 运行环境），这就意味着浏览器会消耗更多的内存资源。
- **更复杂的体系架构。**浏览器各模块之间耦合性高、扩展性差等问题，会导致现在的架构已经很难适应新的需求了。

对于上面这两个问题，Chrome 团队一直在寻求一种弹性方案，既可以解决资源占用高的问题，也可以解决复杂的体系架构的问题。

未来面向服务的架构

为了解决这些问题，在 2016 年，Chrome 官方团队使用“**面向服务的架构**”（Services Oriented Architecture，简称 **SOA**）的思想设计了新的 Chrome 架构。也就是说 Chrome 整体架构会朝向现代操作系统所采用的“面向服务的架构”方向发展，原来的各种模块会被重构成独立的服务（Service），每个服务（Service）都可以在独立的进程中运行，访问服务（Service）必须使用定义好的接口，通过 IPC 来通信，从而**构建一个更内聚、松耦合、易于维护和扩展的系统**，更好实现 Chrome 简单、稳定、高速、安全的目标。如果你对面向服务的架构感兴趣，你可以去网上搜索下资料，这里就不过多介绍了。

Chrome 最终要把 UI、数据库、文件、设备、网络等模块重构为基础服务，类似操作系统底层服务，下面是 Chrome“面向服务的架构”的进程模型图：

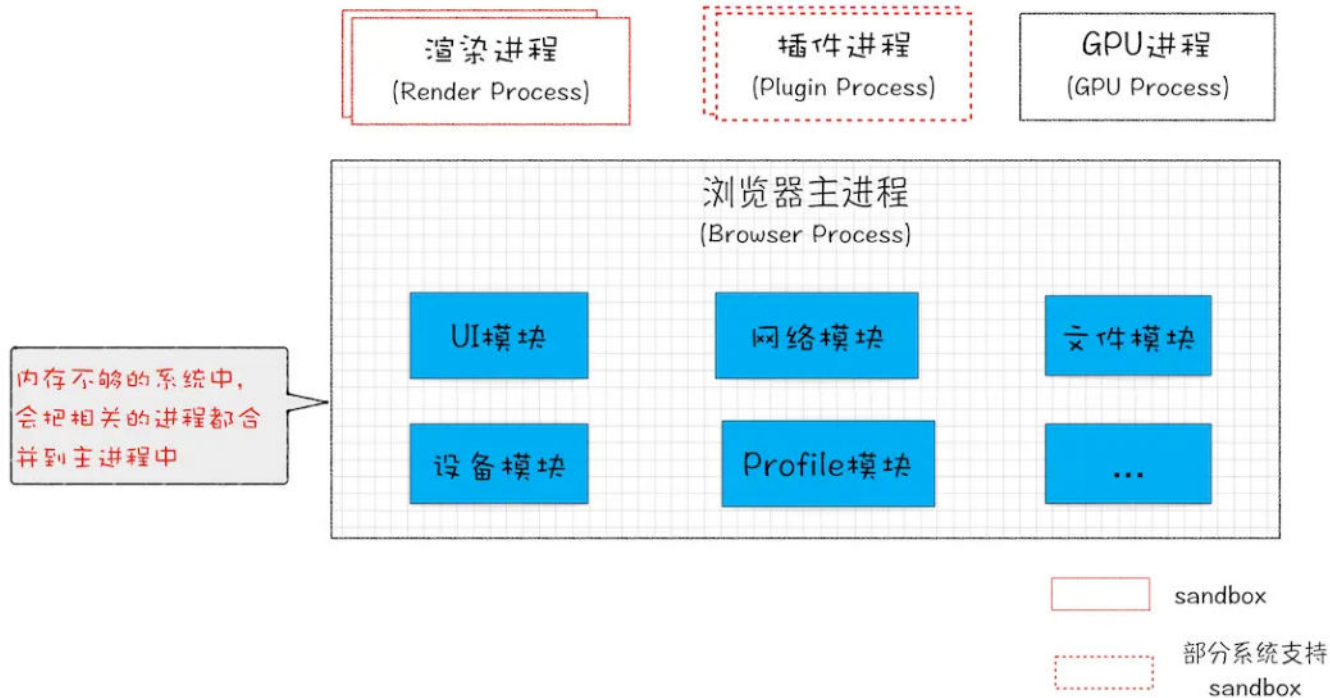


Chrome“面向服务的架构”进程模型图

目前 Chrome 正处在老的架构向服务化架构过渡阶段，这将是一个漫长的迭代过程。

Chrome 正在逐步构建 Chrome 基础服务（Chrome Foundation Service），如果你认为 Chrome 是“便携式操作系统”，那么 Chrome 基础服务便可以被视为该操作系统的“基础”系统服务层。

同时 Chrome 还提供灵活的弹性架构，在强大性能设备上会以多进程的方式运行基础服务，但是在资源受限的设备上（如下图），Chrome 会将很多服务整合到一个进程中，从而节省内存占用。



在资源不足的设备上，将服务合并到浏览器进程中

总结

好了，今天就到这里，下面我来简要梳理并总结今天的内容。

本文我主要是从 Chrome 进程架构的视角，分析了浏览器的进化史。

最初的浏览器都是单进程的，它们不稳定、不流畅且不安全，之后出现了 Chrome，创造性地引入了多进程架构，并解决了这些遗留问题。随后 Chrome 试图应用到更多业务场景，如移动设备、VR、视频等，为了支持这些场景，Chrome 的架构体系变得越来越复杂，这种架构的复杂性倒逼 Chrome 开发团队必须进行架构的重构，最终 Chrome 团队选择了面向服务架构（SOA）形式，这也是 Chrome 团队现阶段的一个主要任务。

鉴于目前架构的复杂性，要完整过渡到面向服务架构，估计还需要好几年时间才能完成。不过 Chrome 开发是一个渐进的过程，新的特性会一点点加入进来，这也意味着我们随时能看到 Chrome 新的变化。

总体说来，Chrome 是以一个非常快速的速度在进化，越来越多的业务和应用都逐渐转至浏览器来开发，身为开发人员，我们不能坐视不管，而应该紧跟其步伐，收获这波技术红利。


思考时间

最后，给你留个思考题：回顾浏览器的进化路线，你认为推动浏览器发展的主要动力是什么？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。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。

分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

 生成海报并分享

 赞 103

 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 开篇词 | 参透了浏览器的工作原理，你就能解决80%的前端难题

下一篇 02 | TCP协议：如何保证页面文件能被完整送达浏览器？

学习推荐

JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费 





Geek_f7f72f

2019-08-06

即使是如今的多进程架构，我偶尔还会碰到一些由于单个页面卡死最终崩溃导致所有页面崩溃的情况，请问这是为什么呢

作者回复: 是这样的，通常情况下是一个页面使用一个进程，但是，有一种情况，叫“同一站点(same-site)”，具体地讲，我们将“同一站点”定义为根域名（例如，geekbang.org）加上协议（例如，https:// 或者http://），还包含了该根域名下的所有子域名和不同的端口，比如下面这三个：

https://time.geekbang.org

https://www.geekbang.org

https://www.geekbang.org:8080

都是属于同一站点，因为它们的协议都是https，而根域名也都是geekbang.org。你也许了解同源策略，但是同一站点和同源策略还是存在一些不同地方，在这里你需要了解它们不是同一件事就行了。

Chrome的默认策略是，每个标签对应一个渲染进程。但是如果从一个页面打开了新页面，而新页面和当前页面属于同一站点时，那么新页面会复用父页面的渲染进程。官方把这个默认策略叫process-per-site-instance。

直白的讲，就是如果几个页面符合同一站点，那么他们将被分配到一个渲染进程里面去。

所以，这种情况下，一个页面崩溃了，会导致同一站点的页面同时崩溃，因为他们使用了同一个渲染进程。

为什么要让他们跑在一个进程里面呢？

因为在一个渲染进程里面，他们就会共享JS的执行环境，也就是说A页面可以直接在B页面中执行脚本。因为是同一家的站点，所以是有这个需求的。

共 19 条评论 >

👍 791



空间

2019-08-06

感觉挺好奇的，单进程浏览器开多个页面，渲染线程也只有一个吗？感觉一个页面开一个线程不是更合理吗？

作者回复: 之前回答的有点笼统，下面是我整理过后的回答：

首先这个问题提的很好，我们从IE6开始讲起，IE6时代，浏览器是单进程的，所有页面也都是运行在



一个主线程中的，当时IE6就是这样设计，而且此时的IE6是单标签，也就是说一个页面一个窗口。

这时候，国内有很多国产浏览器，都是基于IE6来二次开发的，而IE6原生架构就是所有页面跑在单线程里面的，意味着，所有的页面都共享着同一套JavaScript运行环境，同样，对于存储Cookie也都是在一个线程里面操作的。

而且这些国产浏览器由于需要，都采用多标签的形式，所以其中的一个标签页面的卡顿都会影响到整个浏览器。

基于卡顿的原因，国内浏览器就开始尝试支持页面多线程，也就是让部分页面运行在单独的线程之中，运行在单独的线程之中，意味着每个线程拥有单独的JavaScript执行环境，和Cookie环境，这时候问题就来了：

比如A站点页面登陆一个网站，保存了一些Cookie数据到磁盘上，再在当前线程环境中保存部分Session数据，由于Session是不需要保存到硬盘上的，所以Session只会保存在当前的线程环境中。这时候再打开另外一个A站点的页面，假设这个页面在另外一个线程中里面，那么它首先读取硬盘上的Cookie信息，但是，由于Session信息是保存在另外一个线程里面的，无法直接读取，这样就要实现一个Session同步的问题，由于IE并没有源代码，所以实现起来非常难，国内浏览器花了好长一点时间才解决这个问题。

Session问题解决了，但是假死的问题依然有，因为进程内使用了一个窗口，这个窗口是依附到浏览器主窗口之上的，所以他们公用一套消息循环机制，消息循环我们后面会详细地讲，这也就意味这一个窗口如果卡死了。也会导致整个浏览器的卡死。

国产浏览器又出了一招，就是把页面做成一个单独的弹窗，如果这个页面卡死了，就把这个弹窗给隐藏掉。

这里还要提一下为什么Chrome中的一个页面假死不会影响到主窗口呢？

这是因为chrome输出的实际上图片，然后浏览器端把图片贴到自己的窗口上去，在Chrome的渲染进程内，并没有一个渲染窗口，输出的只是图片，如果卡住了，顶多图片不更新了。

国产浏览器这一套技术花了四五年时间，等这套技术差不多成熟时，Chrome发布了 :(

共 17 条评论 >

👍 144



STAN_LIN

2019-08-05

虽然说 Google Chrome/Chromium 的市占率以及代表性都最为突出，但仍然十分希望在讲解 Chrome 各种功能的实现时能够穿插讲解/对比 Mozilla Firefox，尤其是 Firefox Quantum。

毕竟 Google Chrome 也不一定全部都是 best practices 以及此专栏主题为《浏览器工作原理与实践》而非《Chrome 工作原理与实践》不是？



一名 Mozilla Firefox Nightly 用户留。

作者回复: 这个建议很好, 后续课程我考虑按照你这个思路来规划下。

共 3 条评论 >

👍 107



匡晨辉

2019-12-03

老师, 您好。我有个疑惑, 您提到浏览器主进程负责将渲染进程生成的图片显示在ui上面, 就是说渲染进程输出的最终是图片, 浏览器显示的是图片, 那么为什么浏览器中鼠标能选中文字? 如果页面是图片的话文字是选不中的啊, 这里面的机制又是怎样的?

作者回复: 点击鼠标选中文字的时候, 这些消息会传递到渲染进程, 渲染进程再合成选中文字的状态, 然后更新图片!

共 10 条评论 >

👍 97



nissan

2019-08-06

请问老师, 如果打开了 2 个页面, 会有几个进程呢? 是 1 个网络进程、1 个浏览器进程、1 个 GPU 进程以及 2 个渲染进程, 共 5 个吗? 这些进程是可以在浏览器开发者中被实际观察到的吗?

作者回复: 通常情况下会是五个, 但是有很多其他情况:

1:如果页面里有iframe的话, iframe也会运行在单独的进程中!

2:如果页面里有插件, 同样插件也需要开启一个单独的进程!

3:如果你装了扩展的话, 扩展也会占用进程

4:如果2个页面属于同一站点的话, 并且从a页面中打开的b页面, 那么他们会公用一个渲染进程

这些进程都可以通过chrome的任务管理器来查看。

共 7 条评论 >

👍 80



lacaja

2019-08-08

老师好, 我有个疑问, Chrome排版引擎现在是blink, 这一点从哪里可以看到呢? 我在76版本Chrome的navigator属性值里只看到了AppleWebkit, 不理解这是为什么?

作者回复: 你说的是UserAgent, 又称为UA, UA是浏览器的身份证, 通常, 在发送HTTP请求时, UA会附带在HTTP的请求头中user-agent字段中, 这样服务器就会知道浏览器的基础信息, 然后服务器

会根据不同的UA返回不同的页面内容，比如手机上返回手机的样式，PC就返回PC的样式。

你也可以在浏览器的控制台中输入：

```
navigator.userAgent
```

来查看当前浏览器的UA信息。

Firefox中的打印的信息是：

```
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:68.0) Gecko/20100101 Firefox/68.0"
```

Chrome中打印的信息是：

```
"Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Mobile Safari/537.36"
```

安卓系统中的Chrome浏览器：

```
"Mozilla/5.0 (Linux; Android 5.0; SM-G900P Build/LRX21T) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Mobile Safari/537.36"
```

我们知道了服务器会根据不同的UA来针对性的设计不同页面，所以当出了一款新浏览器时，他如果使用自己独一无二的UA，那么之前的很多服务器还需要针对他来做页面适配，这显然是不可能的，比如Chrome发布时他会在他的UA中使用“Mozilla”，“AppleWebKit”，等关键字段，用来表示他同时支持Mozilla和AppleWebKit，然后再在最后加上他自己的标示，如Chrome/xxx。

这就解释了为什么你查看的信息中含有WebKit字样。

共 6 条评论 >

👍 73



许童童

2019-08-06

老师讲得好，我这里放一下自己的总结。

早期浏览器：

不稳定（单独进程）

不流畅（单独进程）

不安全（沙箱）

早期多进程架构：

主进程 渲染进程 插件进程

现代多进程架构：

主进程 渲染进程 插件进程 GPU进程 网络进程



未来：
面向服务架构

作者回复: 总结的很好👍

共 2 条评论 >

👍 45



Artyhacker

2019-08-07

在专栏之外，老师能否提供一些学习浏览器原理的途径？作为前端开发，很早就想好好研究一下浏览器，但除了一些零散的泛泛的博客文章，一直没有找到一个系统的学习方式。

作者回复: 关于参考资料，我先整理下，回头完整发出来

共 11 条评论 >

👍 43



Michael Cheng

2019-08-05

一个角度来说，最大动力就是chrome的出现。曾经的IE像极了诺基亚，chrome就像是横空出世的iPhone，当着IE的面告诉IE，浏览器应该这么玩儿。

另一个角度也是互联网的发展需要，人们所需要的不再是只是简单展示个页面的浏览器，需要有复杂的交互，浏览器应该能做更多的事情，这对浏览器的稳定性、以及性能都有了新的要求。所以出来一个性能符合要求的浏览器也是必须的。

还有就是11年后相对规范的es5的出现，再之后es6.7，web能做的事情越来越多了，web工程化，再后来node的出现，前端体系越来越庞大，

作者回复: 👍

共 2 条评论 >

👍 33



Rocky

2019-08-06

为什么单进程浏览器当时不可以采用安全沙箱？

作者回复: 如果一个进程使用了安全沙箱之后，该进程对于操作系统的权限就会受到限制，比如不能对一些位置的文件进行读写操作，而这些权限浏览器主进程所需要的，所以安全沙箱是不能应用到浏览器主进程之上的。



👍 32



Geek_8476da

老师为什么我这多了一个v8代理解析工具

作者回复:

Chrome支持使用JavaScript来写连接代理服务器脚本, 又称为在线pac代理脚本, pac脚本具体什么样子你可以搜索“PAC代理脚本”, 总之使用pac代理脚本可以实现一些那啥的事。

刚开始的时候啊, Chrome是在浏览器进程里面解析pac代理脚本的, 由于是JavaScript脚本, 所有需要在浏览器进程里面引入V8, 不过把V8运行在浏览器进程似乎不太好, 于是Chrome团队后来就把这个功能独立出来一个进程了, 这个进程就叫着“Utility: V8 Proxy Resolver”。

英文好的话可以看他们提供的官方文档: <https://docs.google.com/document/d/1n5hr4KJhZI2A4MicTfmyiHPdiKp7kmUoWXnRBN8SrZE/edit#>

共 3 条评论 >

👍 27



Brave

2019-08-07

必须订阅了, 首先课程讲的好, 听的明白, 前置技能和概念也都扫盲了, 相信即便0基础也能上手, 订阅了20多个课程, 看了老师耐心的几个长篇回复, 必须点赞, 目前刚好对这方面知识有兴趣, 学习一下做个储备

作者回复: 感谢

共 2 条评论 >

👍 25



Oditto

2019-08-05

对于浏览器中的页面崩溃的原因该怎么定位呢? 很多情况下, 问题并不能稳定复现, 通过浏览器自带的开发者工具感觉很难定位到真正的问题

作者回复: 要定位页面崩溃页面, 我们先要了解下有可能造成页面崩溃的因素, 根据我的实际统计数据看来, 主要有以下三个方便的因素:

首先, 主要因素是一些第三方插件注入浏览器或者页面进程, 拦截了网页的一些正常操作而导致页面或者浏览器崩溃, 如一些杀毒软件, 或者卫士类软件, 或者一些流量劫持软件。

第二个因素是插件, 虽然容易崩溃, 但是通常情况下只会影响到自身的进程, 不过我们以前的统计数据来看, 也会小概率地影响到页面的崩溃, 不过整体数据来看还好了。另外一个方向来看, 插件的使用率已经越来越低了, 所以插件不是个大问题。

第三个因素是浏览器的一些bug, 如渲染引擎, JavaScript引擎等, 不过从统计数据来看, 这类因素导致的崩溃也是越来越低了, 而且随着浏览器的更新升级, 引起问题的因素也是在不断变化的。

所以要直接给出页面崩溃原因是很难的，而且直接从JS的层面来看，也是很难跟踪崩溃原因的。

提一个我之前使用的方法，那就是使用JS来统计页面是否崩溃，这类统计不是100%准备，但是可以通过数据来大致判断页面是否崩溃，然后再找一些典型的用户环境来实地排查。

共 4 条评论 >

👍 21



元斌

2019-08-05

浏览器发展先是标准的发展，然后再是支持标准，可以让需求先统一的提出来，开放的基因，开放的生态，超前的设计理念。还有一个我认为比较重要的一点，计算机性能的发展能比较直接的提升浏览器的体验，从而促进浏览器进一步的发展。

作者回复: 其实，很多标准并不是先制定的，而且先产品化，得到广泛的认可之后，再形成标准，如spdy协议。

世上本无路，走的人多了便有了路。



👍 19



袭

2019-08-05

为什么websql被逐渐抛弃了而使用了Indexdb?

作者回复: 我觉得就是一帮大佬没谈妥

共 3 条评论 >

👍 18



Y s .

2019-08-07

所有的iframe标签都会创建一个新的渲染进程吗?

作者回复: iframe没有单独标签，是潜入在其它页面里面的，比如一个页面嵌入了三个不同域名的iframe，那么这个页面就会拥有四个渲染进程

共 3 条评论 >

👍 15



Will

2019-08-13

请问在chrome task manager里看到的进程数和windows task manager里的为什么不一样呢？我开了一个页面，在chrome task manager 里六个进程，而在windows里有九个chrome相关进程



作者回复: 我在文中提到的四个进程是必须的, 但是通常情况下, 会有更多进程, 如扩展进程, 代理进程, iframe进程, 还会一些如压缩, 视频, 音频等功能进程, 这个根据每个人的电脑环境, 页面环境不同而不同!

另外Chrome有一些辅助进程在任务管理器并没有显示出来的。比如预渲染进程, 会有一个额外的渲染进程, 提前开启, 等有渲染进程需求的时候, 就直接使用改进程, 这样就省去了创建进程的时间了!



👍 14



不矫情不做作那是我

2019-08-05

讲的很清楚, 有点豁然开朗的感觉, 必须点个赞



👍 9



未来系统之父

2019-08-08

老哥快点讲啊, 课程都买了 才发现只有三课, 不讲 关小黑屋哈

作者回复: 吓得我赶紧加快进度

共 2 条评论 >

👍 8



Fortune

2019-08-05

老师后面有讲浏览器是如何运行js的吗? 本人后台, 但是每次大佬用js封装好的方法, 想扩展的好头疼, 看源码

作者回复: 会讲JS的执行流水线, 这点对于理解JS中一些概念很有帮助。



👍 8

