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

你好，我是李兵，现在是一名创业者，也是一名工程师。

我是2005年开始工作的，基于对新技术的兴趣与敏感性，2008年Chromium项目一开源，我便第一时间下载体验。随后，在创业阶段的我基于Chromium和IE发布了一款双核浏览器：太阳花。

这是国内第一款双核浏览器，使用它，除了能享受到Chrome的快捷之外，还能兼容只支持IE的站点。开发过程中最大的挑战是如何在Chromium中集成IE模块，为此我花了大量时间来研究Chromium的进程架构以及渲染流程，好在功夫不负有心人，最终发布的产品也很对得起我的努力，在没有任何宣传的情况下，日活达到了20多万。

在2011年，我就去了盛大创新院，参与研发WebOS项目。WebOS的愿景是基于WebKit内核打造一个能和安卓并存的操作系统。我在团队中负责HTML5特性的实现，比如实现Web Workers、Application Cache、LocalStorage、IndexedDB、CSS3部分动画效果等。这些工作经历让我对浏览器的整个渲染流程，以及HTML5的发展趋势有了更加深入的认知。不过遗憾的是，这个项目没能最终上线。

再之后，我去了顺网科技。顺网科技是一家网吧服务提供商，在顺网我和团队打造了一款给全国网吧使用的“F1浏览器”，日启量达到2000万。由于网吧的电脑环境异常复杂，页面劫持经常发生，所以对页面安全提出来更高的要求；再加上每天千万级别的启动量，所以页面的加载速度和流畅度也至关重要，它们直接决定了用户的流失程度。这段工作经历，让我对浏览器安全有了全新的理解，同时又对页面性能的优化有了系统性的认知。

算下来，我已经处在这个领域从业十余年，这过程中我踩过不少坑，也积累了不少经验，成长很多。从今天起，我会借此机会将我的工作经验分享给你，希望能对你的工作或生活有所帮助，让你学有所得、学有所用。

对于应用，浏览器的地位一直很重要

1995年，美国网景公司因“网景浏览器”的发布而快速崛起，之后网景试图开发一个依靠浏览器的网络操作系统。这引起了微软的注意和警惕，于是同年微软发布Windows 95，并捆绑了IE，大获成功，到2002年，微软就已经占据了浏览器市场80%的份额。

直到2008年Chrome浏览器横空出世，这种垄断局面才算被打破。Chrome浏览器完全颠覆了之前浏览器的架构设计，在速度和安全性上占据了绝对优势，市场份额占比剧增（据 StatCounter 2019年的统计数据显示，Chrome占据了63%市场份额）。在2010年底，Google还推出了一款网络操作系统——ChromeOS。

可以看到，浏览器自诞生之日起，其地位就一直很重要，而且这种重要性还在不断加强。我从浏览器的发展历程中梳理出了**三个大的进化路线，希望能让你了解目前的Web应用到底能做什么，以及未来能适用于那些新领域**。

第一个是应用程序Web化。随着云计算的普及和HTML5技术的快速发展，越来越多的应用转向了浏览器/服务器（B/S）架构，这种改变让浏览器的重要性与日俱增，视频、音频、游戏几大核心场景也都在往Web的使用场景切换。

第二个是Web应用移动化。对于移动设备应用，Web天生具有开放的基因，虽然在技术层面还有问题尚待解决（比如，渲染流程过于复杂且性能不及原生应用、离线时用户无法使用、无法接收消息推送、移动端没

有一级入口），但Google推出了PWA方案来整合Web和本地程序各自的优势。顺便说一句，PWA也是我个人非常期待的方案。

第三个是Web操作系统化。在我看来，Web操作系统有两层含义：一是利用Web技术构建一个纯粹的操作系统，如ChromeOS；二是浏览器的底层结构往操作系统架构方向发展，在整个架构演化的大背景下会牵涉诸多改变，下面列举一些我认为相对重要的改变。

- Chrome朝着SOA的方向演化，未来很多模块都会以服务的形式提供给上层应用使用；
- 在浏览器中引入多种编程语言的支持，比如新支持的WebAssembly；
- 简化渲染流程，使得渲染过程更加直接高效；
- 加大对系统设备特性的支持；
- 提供对复杂Web项目开发的支持。

也就是说，**浏览器已经逐步演化成了操作系统之上的“操作系统”**。

为什么需要学习浏览器工作原理？

前面我站在大厂的视角，带你回顾了浏览器的发展历程，梳理了浏览器的进化路线，分析了浏览器发展的大趋势。那接下来，我们再一起看看，开发者为什么需要了解浏览器的工作原理。

1. 准确评估Web开发项目的可行性

随着Web特性的极大丰富和浏览器性能的提升，越来越多的项目可以用Web来开发。所以，了解浏览器是如何工作的，能够让你更加准确地决策是否可以采用Web来开发项目。

举个例子，去年我做了一个健身房虚拟教练项目，时间紧，任务重，其中有大量的高速渲染动画和快速交互的场景需求。如果采用传统的C++来开发界面，那基本上不可能按时交付，而且后期的维护也会非常麻烦。于是我决定采用Web方案来开发界面，因为采用Web方案可以降低开发成本，缩短交付周期。最终利用这个方案，我实现了这个项目的提前交付，并且效果也很喜人，大家对实现的效果非常满意。

对于这个例子，我认为我所做的最正确的事就是**选对了方案**，但反过来想，如果我对浏览器和HTML5的内容不了解，那可能我很容易就放弃了这个最优方案。

2. 从更高维度审视页面

作为一名合格的开发者，你还要具备一项重要的技能，那就是：**要能站在用户体验角度来考虑页面性能**。我们看下面几个常见的用户体验指标。

- 当用户请求一个网站时，如果在1秒内看不到关键内容，用户会产生任务被中断的感觉。
- 当用户点击某些按钮时，如果100ms内无法响应，用户会感受到延迟。
- 如果Web中的动画没有达到60fps，用户会感受到动画的卡顿。

这里的页面加载时长、用户交互反馈时长、Web动画中的帧数都决定了用户体验的流畅度，并最终决定了用户体验的效果。在用户体验尤其重要的今天，我们必须能够有效地解决这些体验问题，以免给产品造成不可挽回的伤害。

但通常，这些指标是由一系列的复杂因素导致的。如果你要开发流畅的页面，或者诊断Web页面中的性能问题，那你就需要了解URL是怎么变成页面的，只有弄懂这些之后，你才可以站在全局的角度定位问题或者写出高效的代码。

你当然可以把浏览器看成一个黑盒，左边输入一个URL，经过黑盒处理之后，右边返回你预期的效果。如果你对黑盒一无所知，你倒依然可以写前端代码，也可以使用很多最佳实践的策略来优化代码，这就如同不了解操作系统的工作原理同样可以在操作系统上写应用一样。

但如果你理解了这个黑盒子是如何工作的，那情况就不同了。你可以站在更高的维度审视你的项目，通过全视野快速定位项目中不合理的地方。比如，首屏的显示就涉及了DNS、HTTP、DOM解析、CSS阻塞、JavaScript阻塞等技术因素，其中一项没处理好就可能导致整个页面的延时。

而如果你了解了浏览器的工作原理，更加可以把这些知识点串成线，连成网，最终形成自己的知识体系，练就像专家一样思考问题、解决问题的能力。

3. 在快节奏的技术迭代中把握本质

从2011年到现在，前端技术出现了大爆炸式增长，各种新技术层出不穷。我认为**Node.js是前端发展的一个核心推动力**。Node.js是基于Chrome的JavaScript引擎V8来实现的，它的特点是可以脱离浏览器环境来执行JavaScript，于是大家惊讶地发现，原来也可以使用JavaScript写服务器程序呀！

尽管Node.js的诞生时间不长，但其周边已经形成了一个庞大的生态系统。与此同时，各种新标准、新技术纷至沓来，前端生态空前繁荣。

为什么Node.js能如此快速地发展？根本原因还是浏览器功能以及整个前端的开发环境，不足以支撑日益增长的需求，所以“变化”是这段时期的主旋律。这种变化直接扩大了前端工程师的知识半径，**这也导致很多前端开发工程师变成了爆栈工程师**。

虽然前端技术变化快，不过我觉得这里有更大的机遇，谁能快速抓住变化，谁就能收获这波变化带来的红利。

我相信，随着脚本执行效率的提高、页面渲染性能的提升和开发工具链的完善，接下来的前端会进入一个相对平稳的阶段。通俗地理解就是：**等到核心技术足以支撑核心需求，那么前端生态会进入一个相对稳定的状态**。

如果了解了浏览器的工作机制，那么你可以梳理出来前端技术的发展脉络，更加深刻地理解当前的技术，同时你也会清楚其不足之处，以及演化方向。那么接下来，我们看看前端技术是如何针对这些核心诉求做演进的？

首先是脚本执行速度问题。比如针对JavaScript设计缺陷和执行效率的问题，可以从以下两个途径去解决：

- 不断修订和更新语言本身，这样你就应该知道ES6、ES7、ES8，或者TypeScript出现的必要性。这种修订对目前生态环境的改动是最小的，所以推行起来会比较容易。
- 颠覆性地使用新的语言，这就是WebAssembly出现的原因。WebAssembly需要经过编译器编译，所以体积小、执行速度快，使用它能大幅提升语言的执行效率，但是语言本身的完善，和生态的构建都是需要花很长时间来打造的。

其次是前端模块化开发。比如，随着Web应用在各个领域的深入，Web工程的复杂程度也越来越高，这就产生了模块化开发的需求，于是相应出现了WebComponents标准。我们所熟悉的React和Vue都在渐进地适应WebComponents标准，同时各种前端框架的最佳实践也会反过来影响WebComponents标准的制定。

如果理解了浏览器工作原理，那么你会对WebComponents中涉及的Shawdow DOM、HTML Templates等技术有更深刻的理解。

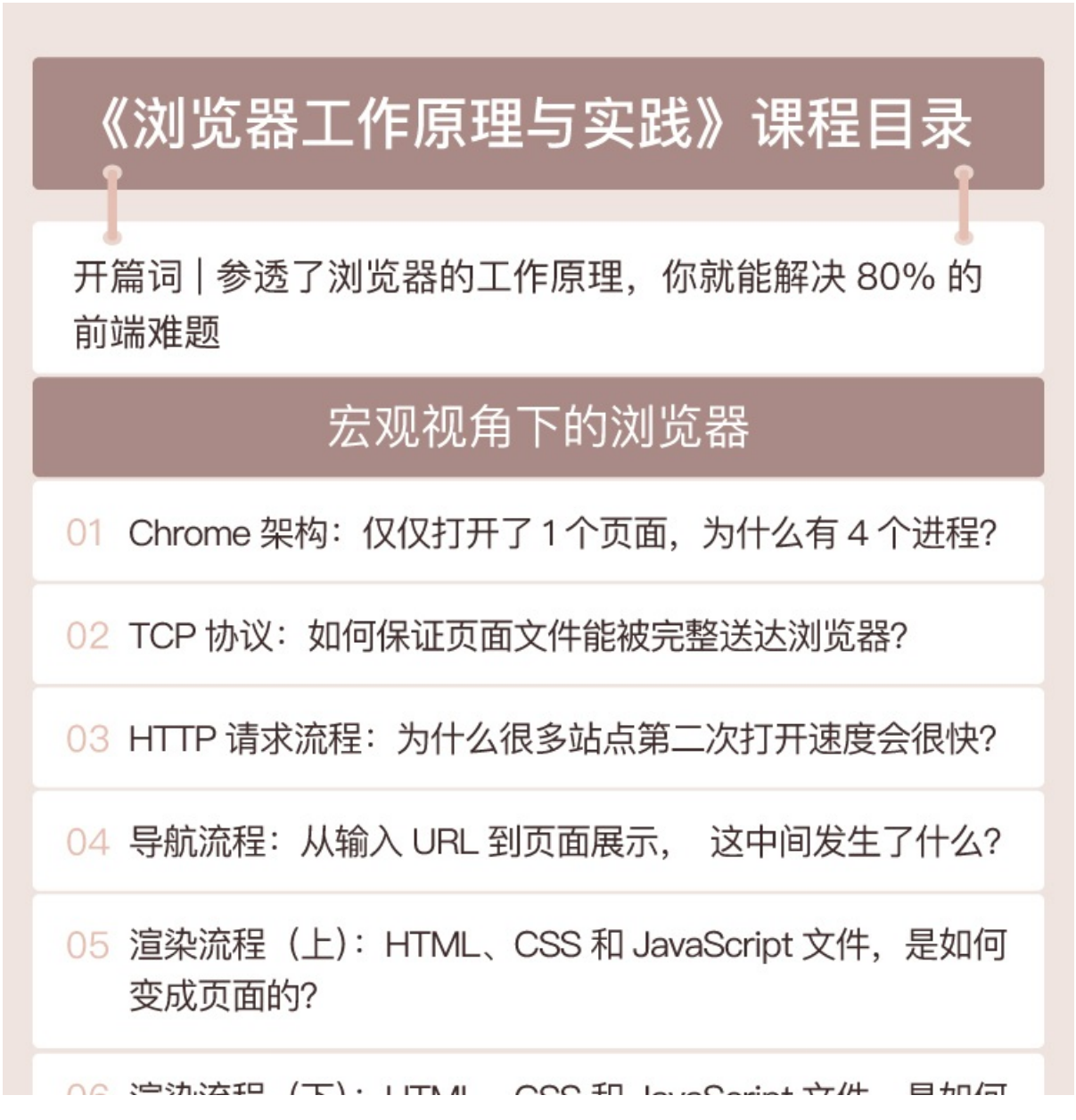
最后是渲染效率问题。同样，如果理解浏览器的渲染流程，那么你应该知道目前页面的渲染依然存在很大缺陷，然后你就清楚如何避开这些问题，从而开发出更加高效的Web应用。与此同时，Chrome团队也在着手改善这些缺陷，比如正在开发的下一代布局方案LayoutNG，还有渲染瘦身方案Slim Paint，其目的都是让渲染变得更加简单和高效。

综上可以看出，触发这些改变的背后因素是当前技术制约了现实的需求，所以**了解浏览器是如何工作的，能让你站在更高维度去理解前端**。

专栏内容

所以，我希望通过这个专栏的学习，能让你系统地掌握浏览器工作原理，并把理论应用到前端实践。

下面就是这个专栏的目录，通过它你可以快速了解下这个专栏的知识体系结构。



36 HTTPS：让数据传输更安全

总结

我希望通过这个专栏的学习，能让你重新认识浏览器，并把网络、页面渲染、JavaScript、浏览器安全等知识串联起来，从而让你对整个前端体系有全新的认识。同时，我会保证用最简单通俗的语言把复杂的问题讲清楚，这也意味着我会在本专栏上花更多时间，所以也希望你能和我一起加油，高质量学完本专栏。

最后我给你留个思考题吧：**你认为现代的前端工程师需要具备哪些核心的基础技能呢？**

欢迎你把你的想法写到留言区，我们一起来交流和探讨，共同进步。

精选留言：

- Geek_0b75a1 2019-08-05 17:13:57
  会有实际操作环节吗？原理再多也没有自己走一遍来的深刻 [4赞]

  作者回复2019-08-05 21:59:14
  会有很多章节来讲实践的

- XIAOXIN 2019-08-05 18:32:19
  请问老师：学习这门课程需要什么基础？是掌握JavaScript的基础知识就行，还是有其它具体的要求呢？[3赞]

  作者回复2019-08-05 22:20:16
  其实有部分JavaScript基础就可以来学了，涉及到的一些主要的概念我都会在文章做相关介绍。
  比如下一篇我会先介绍进程和线程，再介绍浏览器的进程架构。
  再比如后续的讲堆栈，队列相关的类容，我都会先介绍基础的数据结构的用法，再详细介绍相关内容。
  讲网络我会先介绍TCP/IP协议。
  讲V8执行原理时，我也会先科普一些编译原理的基础知识。

以上说的这些基础知识都不难，所以不需要担心学不了，我最期望的大家能通过这门课程的学习，把一个工程师需要的知识体系结构都搭建起来。

- 许童童 2019-08-05 17:39:20
  为老师打Call，前端工程师首先的定位应该是一名工程师，计算机科学基础一定要打牢。 [2赞]

  作者回复2019-08-05 22:00:33
  非常赞同

- 惜心（伟祺） 2019-08-06 08:35:15
  老师会讲blink嘛 或者把blink拨出来 单独做api用 目前碰到这样需求 期待老师讲部分这方面内容

- 业余草 2019-08-06 08:32:38
  谁能快速抓住变化，谁就能收获这波变化带来的红利。作为一个公众号号主，我深有体会！

- 前端路上的小学生 2019-08-06 02:11:19
  一直对浏览器的工作原理很好奇，看到这个必须收藏啊。一定要好好学😀

- Sylvia 2019-08-05 23:52:19
  就在上个月，我也在组内做了浏览器系列课程分享设计，能看到老师更新课程，真是太好了，能够向老师学习，然后更深入去理解和讲解

- Fortune 2019-08-05 23:47:59
  这门课后面会讲一些浏览器怎样加js吗？本人后台菜鸟，每次使用大佬写的js方法，封装好多层，经常出现方法或变量未定义之类，实在痛苦，用工具查看每个页面的加载的js文件，给绕晕哈，见笑哈，但经过自己语法分析，变量提升后才稍微有点思绪，但效率太低哈，感谢老师分享知识哈

- °半月含雪雨 2019-08-05 21:35:50
  我觉得前端工程师主要是利用UI层来展示一些数据和承载交互。但是要是能快速产生利润，前端的项目工程化是一个很重要方面。html，CSS和js做为公认的技能很显然不是一个现在前端工程师的核心技能，而是一个必会技能，核心技能说明了前端的核心竞争力，那么工程化对应的nodejs应该是核心技能。做为主要用到的渲染工具：浏览器也是前端的重头戏。那么提到浏览器就不得说数据结构和算法，无论是http协议这种数据结构还是dom树这种数据结构，都是前端所必须要掌握的。如果说还有技能树需要点亮的话，图形学在前端对应的webgl，canvas，glsl等就又是一片天地了。

- 元斌 2019-08-05 21:27:02
  高性价比的知识

- 浮云 2019-08-05 19:40:24
  看了老师的开篇词，又回来的，看他讲浏览器的轨迹来看web的三大发展历程，太精准了。佩服。个人2年前端开发，一定要把浏览器啃下来。

  作者回复2019-08-05 22:13:26
  加油

- 学习 2019-08-05 19:39:01
  佩服李兵老师，相信学后必有收获。

- Lynko 2019-08-05 19:38:53
  坐等更新

- Geek 2019-08-05 19:15:01
  Firefox的Geoko和Chrome的webkit有没有什么区别啊？

  作者回复2019-08-05 22:50:37
  这里要纠正一下，目前Chrome的排版引擎是Blink，Blink是从Webkit分支独立出来的，起初和Webkit基本是一样的，但是随着Google在Blink上的发力，目前的差异已经非常大了。
  其实站在大的结构层面来看，Blink，WebKit ， Gecko三者的渲染流水线基本是类似的，只是一些术语有些不同。
  但是落实到具体实现层面来看，差别就很大了，如JS引擎的实现，排版过程，绘制过程等都有很大差异。

- 白建国 2019-08-05 17:51:40
  为李老师打Call，浏览器领域的大牛，跟着再学一遍。

  作者回复2019-08-05 22:20:28
  谢谢

- 蒲公英的翅膀 2019-08-05 17:43:33
  如果能有一些实操环节验证这些原理，那就很好了。

  作者回复2019-08-05 22:20:44
  会有的

- 我的偶像是木子 2019-08-05 17:06:11
  前排