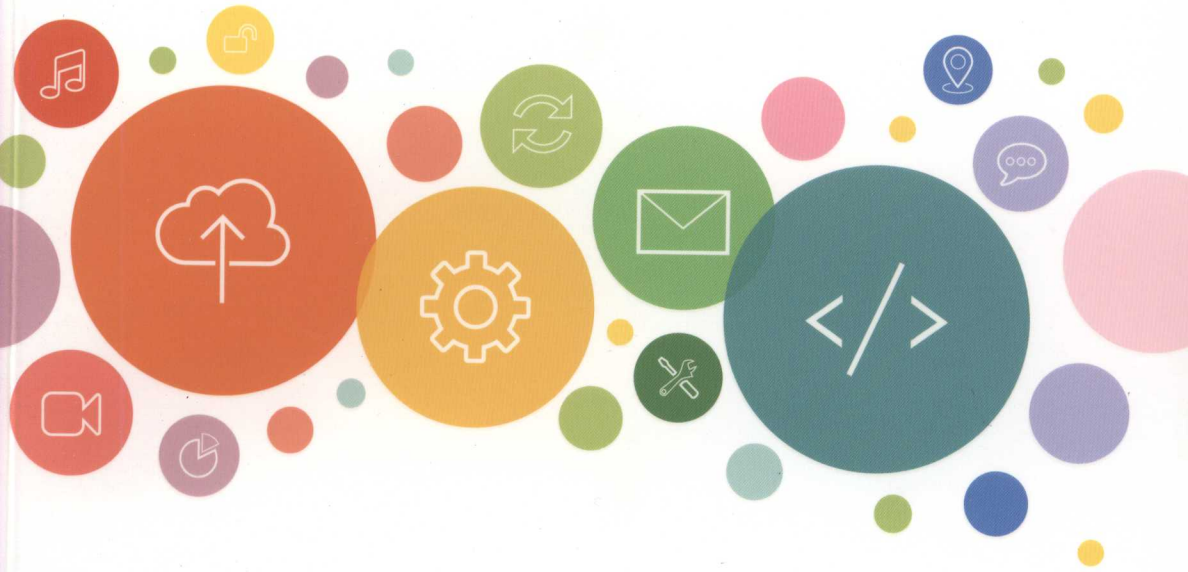


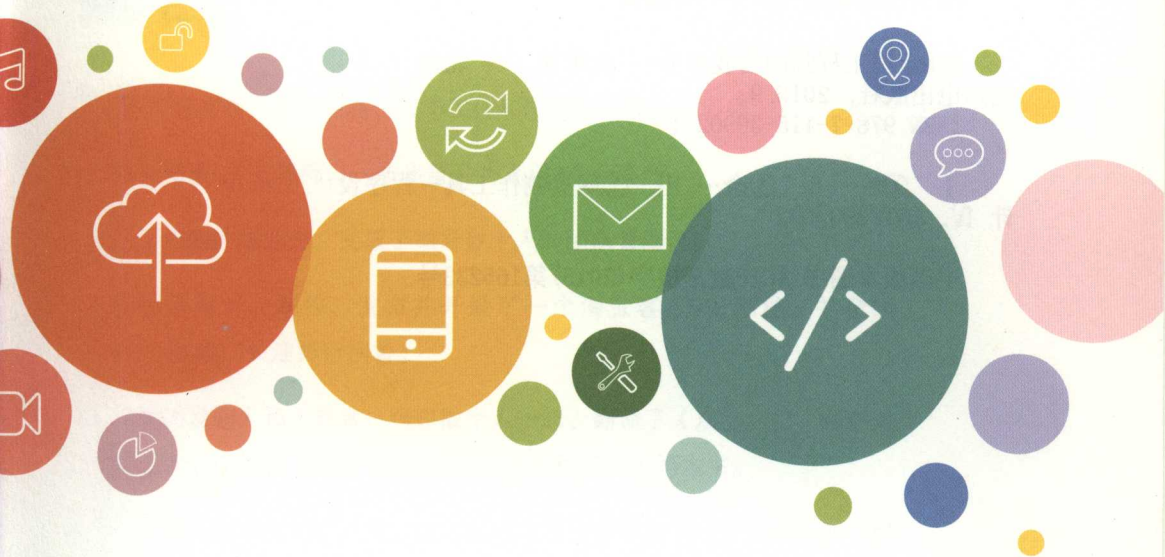
Web全栈工程师的自我修养

余果◎著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

余果

腾讯社交用户体验设计部高级UI工程师，前端开发组负责人，熟悉前端开发、iOS开发、PHP开发和Ruby开发等；曾独立开发iOS APP（撸大师）和CMS（33PU）；翻译有《众妙之门：网站重新设计之道》和《响应式Web设计全流程解析》；平时喜欢编程、写作、演讲、摄影和英语等，希望自己能做一个终生学习者。



Web全栈工程师的自我修养

余果◎著

人民邮电出版社
北京

图书在版编目（CIP）数据

Web全栈工程师的自我修养 / 余果著. -- 北京 : 人民邮电出版社, 2015.9
ISBN 978-7-115-39902-1

Ⅰ. ①W… Ⅱ. ①余… Ⅲ. ①网页制作工具－程序设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2015)第165233号

◆ 著　　余　果
责任编辑　赵　轩
责任印制　张佳莹　焦志炜
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京方嘉彩色印刷有限责任公司印刷
◆ 开本：720×960　1/16
印张：15
字数：340 千字　　2015 年 9 月第 1 版
印数：1－3 000 册　　2015 年 9 月北京第 1 次印刷

定价：49.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前言

您手中的这本书，是我在腾讯五年工作和学习的一些个人心得。

- 我从助理 UI 工程师，一步步晋升为高级 UI 工程师。
- 我从稚嫩的毕业生，变成了领导数十人的团队管理者。
- 我独立设计、制作、发布并开源了一个淘宝客 CMS 系统，并登顶 GitHub 热门关注排行榜首。
- 我作为发起人和主导者，组织数十人一起，翻译了《众妙之门：网站重新设计之道》和《响应式 Web 设计全流程解析》两本书。
- 我从零开始学习 iOS 开发，半年后独立制作并发布了一个 iOS App，赚回了两年的开发者费用。
- 我从一个不敢对陌生人讲话的菜鸟，变成了在几百人面前分享的演讲者。

在这五年中，我最大的收获就是，领悟到做一个“全栈工程师”的快乐。能够做自己喜欢的事情，能够全心投入，能够边学边做，能够不追求完美，能够自我驱动，能够不被自己的头衔局限，能够看到不同技术的联系，能够被老板认可，能够被业界认可，能够相信自己……

由于平时的工作和技术学习都比较跨界，所以在几年前全栈工程师的话题刚刚兴起的时候，我就进行了很多研究和思考。哪些技术对一个组织是真正有用的？全栈工程师的标准能力模型是怎样的？为什么有些人学习和晋升更快？

带着这样的思考，从 2014 年开始，我在豆瓣网发表专栏《谈谈全栈工程师》，发表了 20 篇连载专栏之后，得到了很多读者的欢迎，有五千多人订阅了我的专栏，并且在评论中跟我交流心得、表达感谢。我在开心的同时，也知道自己写得还不够好，文章还有很多语法错误和逻辑不清的地方。于是我打算投入更多心力出一本更好的作品。

经过半年的整理和撰写，这本书终于完成了。我把这本书定义为“轻松的技术杂文集”，希望读者可以以轻松一点的心态来读。书中一小部分内容来

自豆瓣网专栏的扩充，一小部分来自我的博客（http://yuguo.us），一小部分来自这一年多来的梦境和灵感，一大部分想法来自阅读。

本书需要读者有基本的编程基础，能理解基本数据结构，了解一门编程语言的语法。

如果可能，本书尽量不提供某种具体语言的代码实现。此外，读者可能对某一章的内容想作深入的了解，因此我在每一章节的末尾提供了延伸阅读推荐。

关于我

简单说说我自己吧，我从小一直很喜欢读书，高中开始对计算机技术燃起狂热的兴趣。还记得高中时候我每个月必读的两本杂志是《大众软件》和《散文》，即使是最忙碌的高三也没有停止，毕业的时候杂志堆起来一米多高。

《大众软件》话题覆盖面很广，从游戏评测到硬件展览报导，从软件推荐到硬件速递，从手机评测到 CPU 架构介绍……也许从那个时候起，我就养成了对各种新技术来者不拒的习惯吧，这也是我下定决心报考 IT 专业的原因。

小学时候看过很多散文、唐诗宋词，这可能跟我父母都是文科生有关系；可我偏偏热爱并擅长理科，尤其数学和物理，长大后渐渐喜欢看编程类的书。在父母都是文科生的环境下长大，导致我可能有一种感性和理性相结合的特质。

从理性的角度来讲，我做事情非常在意逻辑、证据、数据和对比；从感性的角度讲，我喜欢把我理解的知识用图形化的方式储存在脑海中。还记得高中做数学题的时候，有些关于象限的题，既可以用方程和公式去计算，又可以用图形去推理，我就非常喜欢用图形去推理，看到一个方程式就能“脑补”出解的集合曲线。一个方程组的解就是象限图种几条曲线的交集，因为线是点的集合，所以交点就是既满足方程 A 也满足方程 B 的解，这对我来说是非常容易理解的事情。但副作用是，由于长期不开发自己的记忆能力，所以我很容易忘事，也经常记不住人的名字和脸。理科中，我的生物和化学成绩就不怎么好。

后来我在西安电子科技大学读软件工程专业，西安是一个很美的城市，在沙尘中有一种古老的沧桑感，整个城市也方方正正（处女座最爱），鞋子和衣服在

阳台上放两天就会有一层灰。我很喜欢西安，我在西安度过了美好的四年。

从大学第一天起我就开始写博客，大学生时间比较多，期间折腾了很多域名和很多服务器，以及各种各样的博客程序，也丢过很多内容。在大学毕业那年，我开始启用 `http://yuguo.us/` 这个域名，并抛弃 WordPress，开始用静态站点生成器 Jekyll 生成站点，并使用 GitHub Pages（https://pages.github.com/）提供的免费服务器来托管页面。使用静态页面的最大优点就是访问速度非常快，而且不会出现服务器错误和数据库错误。如果说各种博客程序之间的 PK 就像高手对决，那么 WordPress 这种重型 CMS（Content Management System，内容管理系统）就像降龙十八掌，变化多端，是力量和技巧的极致，有一种无法掩饰的王者霸气。Jekyll 则像是小李飞刀，不会与您正面争锋，但是“小李飞刀，例无虚发”，是速度和精准的极致。

因为非常喜欢折腾网站前端的技术，所以在毕业的时候，我意料之外而又情理之中地选择了前端工程师这一个职业，并且很幸运地在校园招聘中，初次面试腾讯就被录取，并在腾讯工作至今。后来证明，大学的软件工程专业学习很有用。读书时觉得理论知识和后端的知识比较无用，但在工作中却证实，它们非常重要，所以我现在也经常回头复习一些基础知识。

就像乔布斯在斯坦福大学那场著名的演讲里说的，一个人在年少的时候，可能无法看到自己现在做的事情跟自己的未来会有什么关联。您无法预知未来，只能回顾。但是您需要有信心，当您很多年后回头看时，这些点点滴滴会连接在一起，让您朝自己的理想迈进。

我无法预知未来，但回头看过去的五年，我在这期间遇到种种困难，解决各式各样的痛点，帮助项目和团队成长，并成就自己的成长。虽然这些痛点不是每个人都会遇到，世界上也没有完全相同的项目，但是我觉得全栈工程师的理念是通用的，所以我的经验可能对其他人也是有帮助的，这也是我写这本书的初衷。这些思考，我会在本书中一一道来。

最后我想说的是，做您自己感兴趣的事情，学您想学的知识，不要怕走偏了，如果有人说您不务正业，那就让他们说去吧。如果您能远离传统的路子，您将会不同凡响。

目录

什么是全栈工程师

全栈工程师（Full-Stack Engineer），是一个在 IT 行业圈子里越来越热门的话题，无论是像 Facebook 这样的大型公司，还是刚刚起步的初创公司，都开始招募全栈工程师。据说，Facebook 声称：“我们只招全栈工程师！”

Facebook 只招全栈工程师

“全栈”是一个外来词，对于中国读者而言，会觉得它很陌生。当我第一次对某人提到“全栈工程师”时，他一头雾水：“全栈？您是说全端工程师吗？”

其实，“全栈”翻译自英文 full-stack，表示为了完成一个项目，所需要的一系列技术的集合。“栈”是指一系列子模块的集合。这些软件子模块或者组件组合在一起即可实现既定功能，不再需要其他模块。

全栈中的“栈”与计算机数据结构中的“堆栈”不是同一个概念，后者是指先入后出的串行数据结构。顺便说下，“队列”是指先入先出的串行数据结构。

IT 行业之外的人其实很难理解 Web 开发是多么复杂的工程。人们一般认为，在计算机公司或者互联网公司工作的人，就应该能够解决与计算机相关的所有问题：电脑开不了机、应该买什么型号的手机、家里上不了网，等等。在他们眼中，计算机行业的从业者天生就带有“全栈光环”。

但是拿着这本书的您知道，要开发一个 Web 页面，工程师需要掌握的知识至少包括：服务器（比如 Linux）、数据库（比如 MySQL）、服务器端编程语言（比如 PHP）、前端标记语言和脚本语言（HTML、CSS、JavaScript）等。这些技术中的每一个，都需要几年的学习和练习才能达到精通的程度。Web 工程是一个如此大的专业类别，以至于 IT 公司为每一个环节都设置了专门的部门和岗位，来把每一个环节做好。

服务器、数据库、服务器端编程语言、HTML、CSS、JavaScript 等组合在一起就是一个“栈”。这个“栈”是用来制作 Web 站点的，所以又叫 Web 栈（Web-Stack）。[1]

1 最常使用的服务器是基于 Linux 的。Web 发布使用 Apache，数据库使用 MySQL，服务器端编程语言使用 PHP 的组合，所以它们往往一起统称为 LAMP（Linux-Apache-MySQL-PHP）整体解决方案。

如果要开发一个在手机中运行的应用，开发者需要的知识包括：服务器、数据库、服务器端编程语言、iOS 或者 Android 开发技术。这些技术的集合称为 App 栈（App-Stack）。

一个简单的 Web 栈模型：包含前端技术和后端技术。

我们知道，前端工程师就是负责页面浏览器端编程的人，后端工程师就是负责服务器端编程的人，那么什么才是全栈工程师呢？

对于全栈工程师，业界并没有严格的定义，并不是说一定要一种都不能少地具备哪几项知识才能叫做全栈工程师。我倾向于认为，**应该从能力和思维方式两方面，来判定一个人是否是一个合格的全栈工程师。**

国外是怎么定义全栈工程师的呢？在著名的问答网站 Quora 上有人提出了这个问题。一个获得了高票的回答是：

> 全栈工程师是指，一个能处理数据库、服务器、系统工程和客户端的所有工作的工程师。根据项目的不同，客户需要的可能是移动栈、Web 栈，或者原生应用程序栈。

基本上，当客户需要一个全栈工程师的时候，客户需要的是一个**全能的“大神”**。简单来说，全栈工程师就是可以独立完成一个产品的人。当客户让他去做一些舒适区之外的工作时，他敢于迎难而上，并成功完成任务。

我们每一个工程师，进入到公司和企业工作之后，就会有一个职位头衔。我的职位头衔是“UI 工程师”，其他人的头衔可能是“交互设计师”“PHP 开发工程师”，等等。“全栈工程师”不需要头衔。他既有全面的技术能力，也渴望跨界工作的状态。

“全栈”好像是一个遥不可及的梦想，所以对于初次了解“全栈工程师”这个概念的工程师而言，有可能觉得“不可思议”或者抱着“这不可能”的排斥心理。但如果我们回头看看 Web 开发的历史，就知道“全栈”其实没那么难。

Web 开发流程

有人曾开玩笑说，全栈工程师是资本家的阴谋，因为老板想雇一个人来做三个人的工作。

其实在 2000 年第一次互联网泡沫破裂之前，那时候的 Web 工程师也许符合“全栈工程师”的简单定义：**一人包揽整个网站的构建**。

那时的 Web 工程师们所面临的挑战比今天小很多，他们可能只是制作一些静态的页面，不会面对如今富交互的 Web 应用程序。那时网站可能包含数据库和一些 HTML 表单，但仅此而已，甚至只需要将一些静态页发布到服务器上。在网站的前端无需视觉设计和交互设计，因为网站屈指可数，市场竞争很小，工程师仅用一些基本的 HTML 标签和闪亮的 GIF 图片就可以吸引网民的目光。同时，网站访问量都比较小，前端资源的体积也不大，无需关注服务器压力和 CDN，网民对加载速度的容忍度比较高，也不需要过多考虑用户体验。

但随着技术的发展、用户量的增加、客户端种类变多，每一个小小的细节都需要优化和考虑。在海量的访问量面前，也许改变一个按钮的位置和颜色就能影响上千万的订单。如今的互联网产品已不是以一己之力就可以完成的乐高积木了，Web 开发需要以某种可控的方式来管理。

于是，所有认真对待互联网产品的大公司都引入了流水线开发流程，在这条流水线上诞生了多个非常专业的职位。

大中型互联网公司的产品研发流水线。

产品经理：产品经理其实是对一个产品负根本责任的管理者。他通常的工作包括制订产品规划、协调多方资源、把控产品方向和质量细节，等等。有时候，他会从头策划一个新的产品，而更多的时候，他是在优化已有产品的一个部分。总之，在流水线中，产品经理需要从策划跟进到发布，是一个非常重要的角色。

用户研究员：用户研究员的工作是研究用户行为，有时候他会从宏观的角度分析数据，有时候也从微观的角度分解用户场景，有时候会召集一些用户专门来访谈，或者观察用户对产品的使用情况。从输出品的角度来说，用户研究员一般输出用户研究报告来交付给产品经理和交互设计师，作为产品设计的目标参考。

交互设计师：交互设计师常被简称为“交互”。他与视觉设计师最大的区别是，交互设计师更多着眼于如何优化用户界面的信息分布和操作流程。交互设计师的输出品一般是描述用户与网站“交互”过程的流程图，以及描述页面信息结构的线框图。输出的线框图会交付给视觉设计师。

视觉设计师：在细分交互设计师和视觉设计师的大公司，视觉设计师根据交互设计师输出的线框图来做一些润色和设计，输出最终的产品视觉稿之后将视觉稿交付给前端工程师。在一些不细分交互设计师和视觉设计师的小公司，二者被统称为“设计师”，他们的职责就是负责整个用户界面的设计。

前端工程师：产品视觉稿在得到产品经理和交互设计师等多方确认之后，会交给前端工程师，由前端工程师制作页面，实现视觉稿以及交互功能。从头衔上的变化就可以看出，这时候才真正开始编码。前端工程师需要非常熟悉 HTML、CSS 和 JavaScript，以及性能、语义化、多浏览器兼容、SEO、自动

化工具等广泛的知识。[1]

后台工程师：使用服务器编程语言，进行服务器功能的开发。在编程语言的选择上，很多公司都会出于团队已有成员的知识储备、程序员的供给量或者语言性能方面来进行选择。在这一方面，后台语言的选择是相对自由的一件事，不像前端工程师，为了页面兼容性，必须使用 HTML 和 CSS。如果关注各大公司招聘信息的话，您就会了解，不同公司使用不同的后台语言，比如传统的 C# 和 C++、Java、PHP，或者新潮的 RoR 和 Python。小公司的后台工程师除了负责功能开发，可能还会负责服务器的配置和调试、数据库的配置和管理等工作。在大公司，这些工作会分别委派给后台工程师、运维工程师、数据库管理员（DBA）等岗位。

运维工程师：运维工程师是跟服务器打交道的人，他会关注服务器的性能、压力、成本和安全等信息。

测试工程师：顾名思义，测试工程师保证产品的可用性，即使在小公司，这一职位也是不可或缺的。

流水线的优势

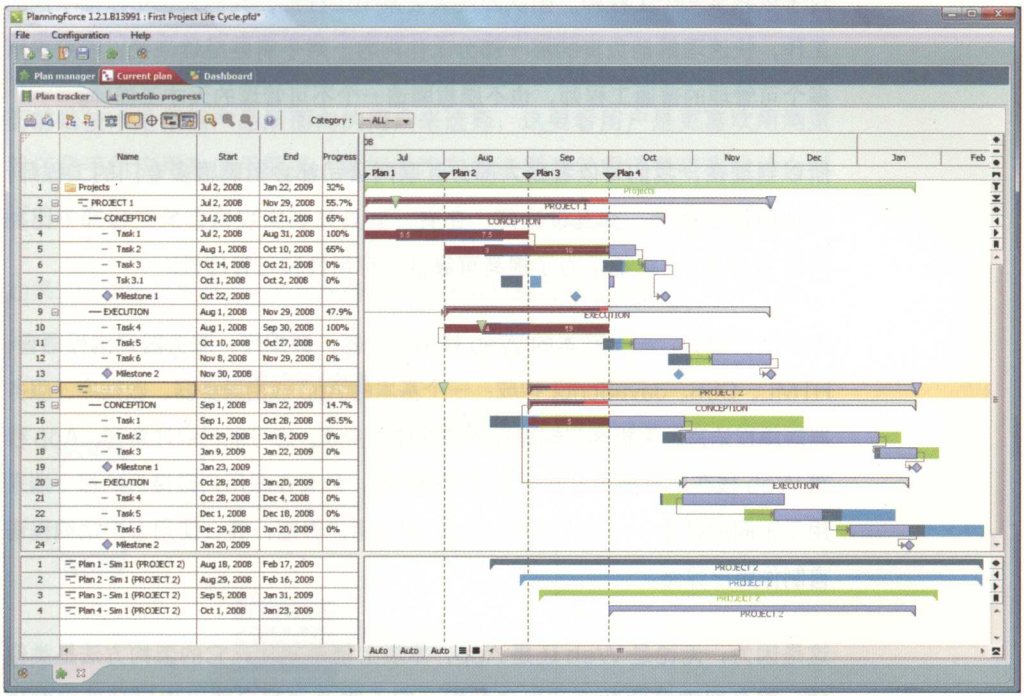
由于有了流水线，其中每个职位的可用工作时间都会作为“资源”来管理，因此需要一位项目经理来把控项目进度，并对人力资源进行调控。比如一个项目立项时，就要预约好这个周期版本需要实现哪些优先级较高的特性，而把优先级不那么高的特性推迟。对于确定在这一周期要实现的特性，就要安排本周进行设计、下周完成开发、下下周进行测试等。

在项目管理中，经常会用到甘特图。甘特图（Gantt Chart）是柱状图的一

1 对于前端工程师这个环节，腾讯公司进行了更进一步的分工，分为“UI 工程师”和“前台工程师”。UI 工程师主要负责 HTML 和 CSS，在制作的过程中要考虑非常精细的设计还原、语义化、页面性能和 SEO 等，而不用考虑 JavaScript 以及页面数据。前台工程师主要负责 JavaScript，他会在静态页面的基础上增加动态数据以及 JavaScript。不是所有的大公司都细分这一职位，在百度和阿里巴巴，前端工程师一个人同时负责页面还原和 JavaScript 开发。

种，显示项目、子项目、进度以及其他与时间相关的系统的进展情况。

流水线在大公司的任何一个严谨的大型项目里都是必不可少的，因为无论是 Web 产品还是 App 产品，它的复杂性都已经超出了单个工程师可以控制的程度。通过把复杂度分解到各个组件，每一个组件就可以进行很好的质量控制。



用于项目管理的甘特图。

Web 页面的生成和传递需要经历复杂的过程，因此容错能力就是首当其冲要考虑的问题。数据从位于深圳某个机房里的服务器传输到用户手机浏览器页面上进行运算和渲染，这个过程中的每个环节都可能出错，所以每一步都要做好容错处理。如果服务器出现错误，是否能在 30 秒内切换到备用机？后台数据异常时返回什么结果给前端，等等。

Web 页面可以在无数种设备上显示。兼容性在此时成为了前端工程师需要考虑的一个重要问题。不同的用户在不同手机上浏览页面，显示的方法会有些许不同，甚至要考虑到如果浏览器不支持 JavaScript，则需要给出特定的提示。

模块化的 Web 开发流程在很大程度上提高了服务的可靠性和可用性，让我们对每一个环节都能单独进行测试。这让大型 Web 开发真正变得可管理、可控制、质量可评估。

流水线带来的另外一个好处是，产品以团队的方式来运作和生产，公司不会过于依赖某一个工程师。团队即使失去某个工程师，其他人也可以接手他的工作，快速理解他负责的那一部分工作内容。对于有些经理来说，宁可雇用多个可管理的普通工程师，也不愿意聘请一个不可管理的天才工程师。

所以到现在，我们可以看到大部分互联网公司都会招聘很多专门的工程师，比如前端工程师、交互设计师，还有一些具体到实现语言的工程师（比如 PHP 程序员），这都是为了提高可靠性、可用性和可管理性。

刚才我们说到，一个基本的 Web 栈由服务器、数据库、服务器端编程语言、HTML、CSS、JavaScript 构成；一个基本的 App 栈由服务器、数据库、服务器端编程语言、手机客户端编程语言等技术构成。您可能已经注意到，App 栈跟 Web 栈在后台技术上几乎是完全相同的，只有在跟用户最接近的那一端采用了不同的技术——要么使用 HTML 制作用户界面，要么使用客户端编程语言制作用户界面。

这是因为，无论是 Web 还是 App，本质上都是软件，它的架构方法是类似的。服务器端接收数据和发送数据，它无需关注客户端采取何种技术制作用户界面。客户端处理用户交互以及显示数据，它不关心服务器使用的是 Java 还是 PHP。如果说开发一款软件就像制造一辆汽车，那么服务器端就像动力系统，客户端就像汽车的车身，不同的动力系统和车身可以自由组合搭配（我不太熟悉汽车的制造过程，这里只是作个比喻）。

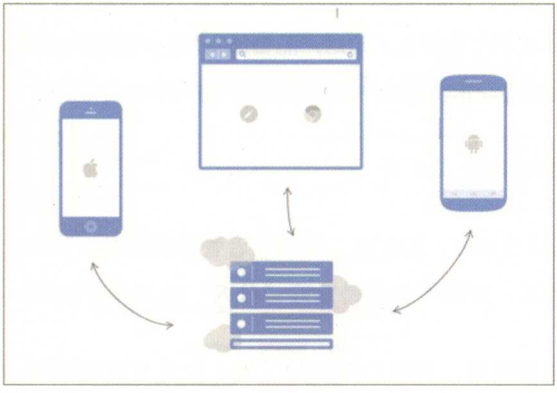
服务器和客户端之间通过 HTTP 协议传递信息。正是因为 HTTP 协议的通用性，使得服务器端和客户端得以实现完全的技术分离。无论是开发 Web 服务还是手机里运行的 App，一套后台开发技术，可以为所有的前端展现方式实现软件的商业逻辑。

HTTP 协议类似于汽车组装过程中的一个通用标准，动力系统和车身都要采用这个统一的标准来实现才可能完美对接。

用户量的大小、服务器承受压力的能力、软件对服务器计算量的要求、对服务器响应速度的要求……诸多因素会影响开发者决定使用哪一种后台技术。汽车的动力性能主要由发动机来决定，汽车厂商也会根据市场需求、消费者定位和制造成本等综合考虑使用哪一种发动机。

而前端技术是根据产品所面向的用户来选择，这要看用户是更喜欢用浏览器还是手机应用来使用服务。就好像汽车的造型要考虑消费者喜欢什么样式的外观。

如果二者功能分离得当，后台服务跟前台服务一般可以自由搭配，互不干涉。



如果服务器逻辑和客户端逻辑分离得当，二者可以自由搭配。

“各司其职”的弊端

虽然流水线式的职业划分和工程管理有很多优点，但是它就像一把双刃剑，在带来高可控性、可用性和可管理性的同时，也给工程师带来了一些困境。

工程师职责不清导致效率低

因为分工太细，所以在不同职业的交接处往往会有一些既不属于上游，也不属于下游的“灰色地带”。

这部分工作没有明确规定由谁去做，所以有时候时间会浪费在沟通上。员工会认为自己的头衔代表了自己的责任边界。比如，一个前端工程师可能会不加思考地实现视觉设计稿，因为他的岗位说明里规定了自己的职责，这其中不包括质疑设计稿，所以他忽视了自己的最终目标：让产品更好。

在一个开放平等的环境中，他实际上可以对影响可用性和性能的设计提出自己的想法。甚至如果他很熟悉这个项目的话，对设计的一致性和一些交互细节都可以说出自己的看法。

- **工程师缺乏主人感导致产品质量差**

流水线工作流程对专精工程师的要求是，能很好地执行动作或者执行任务，而不需要对产品的目标有很好的理解。其实在工程师的初级阶段，执行任务的能力是必需的，因为他还没有能力把握产品的目标，而且也需要更多的练习来提升专业能力。但随着经验的积累，如果工程师还不能对产品整体有自己的理解和贡献，就很容易缺乏主人感，要么他会跳槽，要么产品本身缺乏亮点而导致失败。

- **工程师缺乏全局的视野影响个人成长**

当工程师希望晋升到更高级的职位，如高级工程师或者管理岗位时，公司对他的大局观会有更高的要求，这就不仅仅是做好“分内”的工作就行的。

高级工程师需要有对设计的理解、对后台知识的了解，以及有跨团队推动项目的能力。长期研究专精的专业知识会让一个人视野变窄，变成“学术派”，而不是“实践派”。

- **更多角色导致项目效率低下**

软件工程项目与工业中的标准流程化项目有一个很大的区别：标准流程化项目中每一个流程所接受的输入都是一样的，所需要的输出也都是完全相同的。

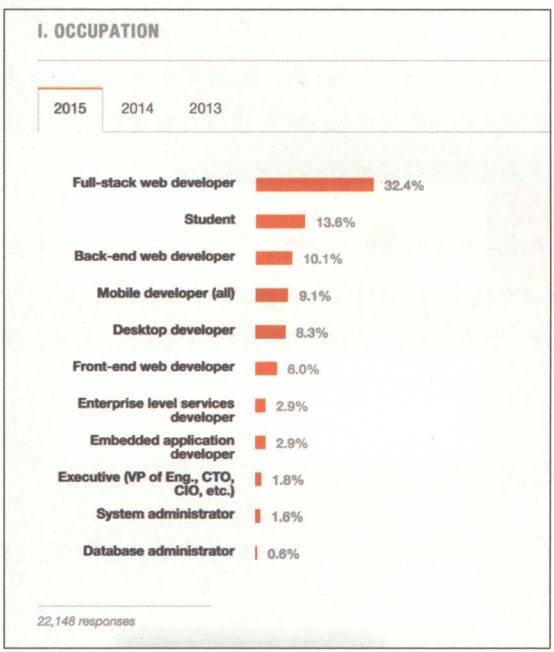
比如，一个汽车生产流水线，将“造汽车”这个任务分解成“造轮胎”“造方向盘”等。流程拆分得越细，每一个工人或者机器人就能做得越快，整

个流水线就会运转得越快。但是在软件工程项目中，我们把任务拆分给多个角色，每一个角色需要同样多的时间去理解需求，在上下游时间的安排中也往往会增加一些缓冲时间，比如周一安排设计，但是为了防止延期风险，会安排周三再制作前端页面。每一个角色的工作时间都会变长，而且交接也增加了缓冲时间，这样整个项目的时间就会被延长。

全栈工程师登上舞台

因为各司其职的工作流程有效率低下、成本高的缺点，所以很多创业公司都不会配备齐全的流水线，而是希望采用更灵活的方式来组建团队，全栈工程师也因此成为了理想的选择。但是全栈工程师的兴起还离不开这两个重要因素：技术的发展，以及提供 PaaS 服务的平台越来越多。

根据 StackOverflow 在 2015 年进行的开发者问卷调查，**有 32.4% 的开发者是全栈工程师，这一比例连续三年来逐年上升。**



您是哪一种类型的开发者？截图来自 2015 年 StackOverflow 开发者调查。

技术的发展

提到全栈技术，不得不提一个代表性的全栈框架——MEAN，它是MongoDB-Express-AngularJs-Node.js 的缩写，是从数据库、服务器到前端页面的一个完整技术栈。

MongoDB 是一个面向文档的、NoSQL 类型的数据库。MongoDB 颠覆了传统的基于表的数据存储方式，而采取了类似 JSON 的文档结构来存储数据，因而它在储存数据时可以更加灵活。

Express 是一个 Node.js 框架，可以创建灵活的 Web 服务，比如单页面应用程序、多页面应用程序和混合型 App。

AngularJS 是一个开源的 JavaScript 框架，由 Google 和开源社区共同维护，它用来创建单页面应用程序。它的目标是使用 model-view-controller 模式来规范 Web 应用程序，让开发和测试富交互的单页面应用程序变得更加轻松。

Node.js 是一个运行在服务器端的 JavaScript 运行环境，它的底层是基于 Chrome 的 JavaScript 运行环境——V8 引擎。Node.js 可以作为服务器端语言，用来创建快速、可扩展的应用程序。Node.js 也可以在本机运行，做一些本地操作，比如加速本地开发流程，或者实现一键发布。

MEAN 可以说是传统的 LAMP 方案的有力竞争者。因为从服务器端到页面端都采用同样的语言（JavaScript）和同样的架构模式（MVC），所以一个擅长 JavaScript 的工程师可以兼顾前后端的开发，并且前端模板代码和后台模板代码是可以复用的。

提供 PaaS 服务的平台越来越多

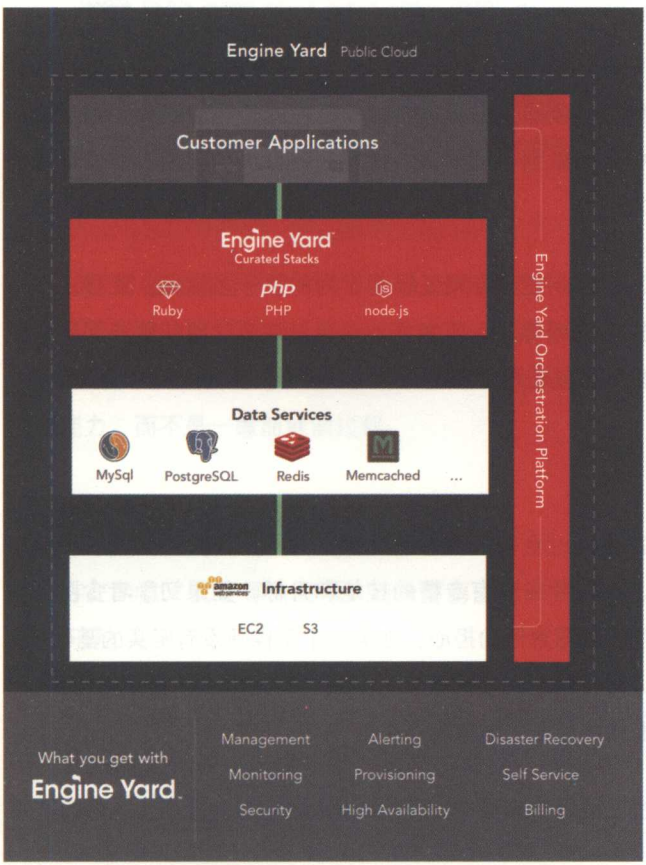
随着 Web 技术的发展和开源社区的积极努力，有很多公司提供便宜又方便的一条龙服务，可以解决独立开发者的大量麻烦。

比如 Amazon 提供的 PaaS（Platform as a Service，平台即服务），就可以

让创业公司的开发者省去架设和维护服务器的麻烦。

而 GitHub 在 2012 年获得了一亿美元融资，也可以看出市场对代码托管市场的信心。可以预期，未来可能会出现越来越多为开发者提供服务的公司。以后，小公司也可以用更低廉的价格获得世界级的 IT 服务支持，毫无疑问，更多的 IT 服务将托管在第三方的服务器上。

VPS（Virtual Private Server，虚拟专用服务器）是把一台物理服务器虚拟成多个虚拟专用服务器的服务。每个 VPS 都可分配独立的公网 IP 地址，运行独立的操作系统，拥有独立的磁盘空间、内存、CPU 资源、进程和系统配置，模拟出“独占”使用计算资源的体验。



EngineYard 为开发者提供一条龙服务（截图来自 www.engineyard.com）。

全栈工程师的发展前景

纪录片“寿司之神”讲述了86岁的顶级寿司制作者小野二郎的故事。小野二郎心怀追求极致的匠人心态，终其一生，他都在握寿司，永远以最高标准要求自己跟学徒，观察客人的用餐状况，微调寿司，确保客人享受到究极美味。他的寿司店只有10个座位，上厕所都需要去店外的公共场所，但是这样一间小店获得了米其林三星的顶级评价，这意味着仅仅为了享受美食就专程来到这个国家都是值得的。我想小野二郎就是专精工程师的代表，他日复一日地磨练和提高自己的技艺，他不会想要上市或者在全国开满连锁店，也不去追逐更大的商业回报，只为了自己内心对完美的追求。

确实，全栈工程师不是唯一成功的方式，也不是所有工程师的最终归宿。无论您是渴求成就感，还是物质回报，都有很多路径可以达到。如果能在任何专精的职业中努力做到名列前茅，就能获得巨大的回报，就像顶级的寿司制作者小野二郎。

而我推崇的全栈工程师则是与专精工程师不同的另一条道路。全栈工程师除了在一个专精知识领域有深入研究之外，还以知识广博和解决问题能力强著称。所以我认为有志成为全栈工程师的学习者，要有这样几个觉悟。

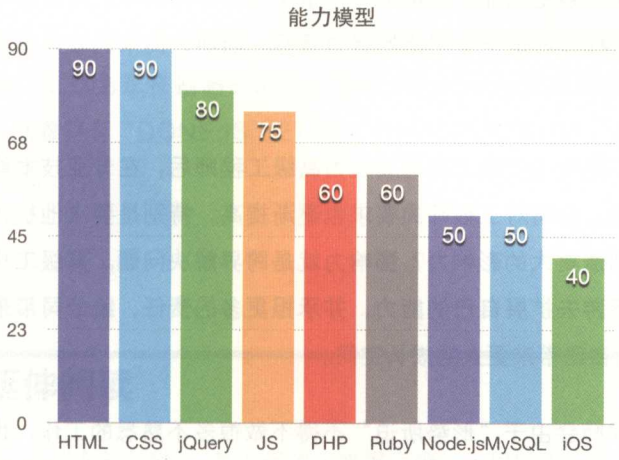
一专多长

我跟一位行业专家讨论过全栈工程师的话题，他不是很赞同全栈工程师这个方向。他认为，工程师应该有专精的技能和目标，如果初学者贪图大而全，反而样样不精。我理解他的担心，如果一个工程师没有坚实的基础（比如专业理论知识，对常用设计模式的理解，或者特定职业的基础知识），那么了解的非本专业技能越多，越容易迷失。

所以我认为，全栈工程师首先要“一专多长”。

一专多长的意思是，工程师首先有一个专精的方向，在这个方向上足够精

通之后（高级工程师级别），以此为突破点去学习更多的知识，增加自己的长处。如果还没有获得某个方向上足够深入的理解，就不要囫囵吞枣地去学习其他领域的知识。



全栈工程师最好“一专多长”。

有些知识需要时间的积累，并不是快速阅读就可以掌握的。“全栈工程师”这个名词可能会引起读者的误解。勿在浮沙筑高台，“全栈”是一个长期积累的过程，是专精型工程师在不断解决问题的过程中积累知识和经验所形成的能力，而不是一蹴而就的过程。

解决问题，而不是醉心技术

公司存在的意义就是解决问题，公司要解决用户的问题，而员工要解决公司的问题。

公司的问题可能是降低成本、扩大用户群、增加成交量、优化性能，等等。不同的问题优先级不一样，投入同样的时间，有的项目能为公司增加上百万的收入，而有的项目却只能增加几万。

互联网领域发展很快，问题的优先级永远都是在动态变化的，所以团队往

往每半年或者三个月就要回顾一下当前形势，并制定新的工作计划。如果新计划不是您擅长的，怎么办？您应该马上开始学习新的技术，这就是我说的关注问题，而不是醉心技术。

无论个人的目标和兴趣是创业，还是单纯希望学习更多的技术，或者学习项目管理，全栈工程师都是一个不错的努力目标。而随之而来的收益也是非常大的。

在大公司，程序员逐渐由初级工程师成长为高级工程师后，在专业技术能力上不断接近极限，公司对工程师的要求也逐渐提高，特别是要求他扩大“影响力”。如何创造更大的影响力？影响力就是跨界解决问题。**高级工程师可以选择往上下游去扩展自己的能力，并承担更多的责任，给公司带来更大的收益，也给自己带来更大的成长空间。**

在创业公司做工程师会由于“形势所迫”不得不做很多不熟悉的工作，也就是“舒适区之外”的工作。虽然说小公司没钱聘请更多员工是最直接的外在因素，但是全栈工程师的成长并不是靠外力，而是自我驱动。**程序员在小公司里主动去承担更多责任，自己跟公司都会获得相应的成长。**假如公司上市扩张，自己能获得巨额的回报，即使公司失败，自己也能获得锻炼。

在自由职业市场，全栈工程师是最闪耀的明星。因为全栈工程师能独立创作产品，所以很容易被市场接纳。比如 WordPress 主题设计、App 开发、网站开发，等等。全栈工程师也能轻松搭建自己的作品网站，而不像后台工程师的作品那样，不太容易展示的后台组件。

全栈工程师还是天生的创业者，因为自己可以独立完成一个产品模型，所以可以用最快的速度去测试自己的想法。从工作中锻炼出来的发现问题、洞察需求、设计解决方案并开发出初始版本产品的能力，是全栈工程师最大的优势。也许这就是为什么有些创业者说“我们就差一个程序员了”的时候，程序员们都会把他当成一个笑话。

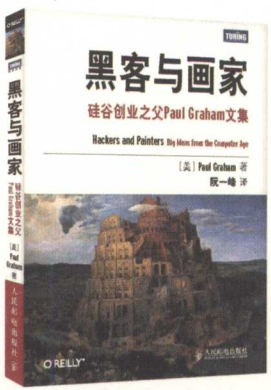
总之，全栈工程师是一个能够在所有场合发光发热、实现个人价值的职业。在未来，中国也会涌现出越来越多优秀的全栈工程师。

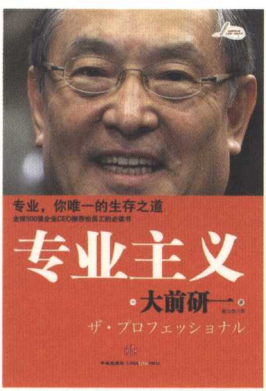
最后，关于上文中提到的“Facebook 只招全栈工程师”是一条谣传。从公司文化的角度来讲，Facebook 鼓励员工以开放的心态解决问题，不因为自己的头衔给自己设限。他们仍然招聘各种职位，您可以在官方网站上找到正在招聘的职位。我在 Facebook 参观的时候，看到一面墙上贴着写有“DONE IS BETTER THAN PERFECT”[1] 的海报，在此与君共勉。

延伸阅读

1. 《黑客与画家》（美）保罗·格雷厄姆，人民邮电出版社

2. 《专业主义》（日）大前研一，中信出版社





1 翻译为：完成比完美更重要。

如何成为全栈工程师

不管您是否承认，除去极少数天赋异禀、骨骼惊奇的天才程序员，我们大部分人都是普通人，都需要遵循“一万小时定律”，才能从平凡变成超凡。

凡人要从一个小菜鸟成长为全栈工程师，只能从少到多、慢慢积累知识和经验。职业生涯的本质，就是在一个专业方向上积累信息。这里我推荐采用“先精后广，一专多长”的流程来学习。采用这种方式来学习，不光可以触类旁通、举一反三，还让我们学习得更快，而且循序渐进更符合一般人的职业生涯发展。

先精后广，一专多长

围绕商业目标

关注用户体验

先精后广，一专多长

“先精后广，一专多长”是指，建议初学者学习全栈技能的时候，**先在一个特定的方向上有比较深入的钻研，然后再将学习目标渐渐推广开来。**比如先从前端方向入手，掌握了基本的 HTML、CSS、JavaScript 之后，不要转头向服务器端语言或者 App 方向发展，而是深入到性能优化、SEO、多种框架、响应式页面等前端细节中去。经过一到两年的深入研究之后，再去学习其他方向。

如果在创业公司做全栈的工作，一般也不会要求一个人处理所有的技术工作，至少会有两三个人组成团队来做项目。大家在分配工作的时候，可以按照每个人的偏好和技术特点，进行前后端的分工，不用完全按照每个人做一个模块的方式来分工。这种分工的界限不一定要很绝对，在不同职位的工作范畴中，可以有一些重合的区域。

如果是毕业生或者初学者，我不建议在刚开始的一到两年接触太多技术，杂而不精，结果可能会对后面的职业道路产生副作用。

为什么我强调在开始的时候有一个专精方向的重要性呢？因为这样您才能在求职的时候有一个“亮点”。

平心而论，程序员在市场上的供求关系比很多其他职业都更有利于求职者，在微博、Twitter、V2EX 上都会有很多引人注目的招聘启示，大家对优秀程序员的需求从来就没有减少过。

虽然优秀的程序员总是能找到工作并且工资不低，但是很多程序员投出的简历都石沉大海，一个主要原因是由于求职者的简历没有亮点，或者说从工作经历中提取不出来一个亮点。

让我们做一个情景假设，作为一个有两年工作经验的全栈工程师，您看到腾讯有一个职位空缺。

腾讯社交用户体验设计部招聘前端开发，要求如下。

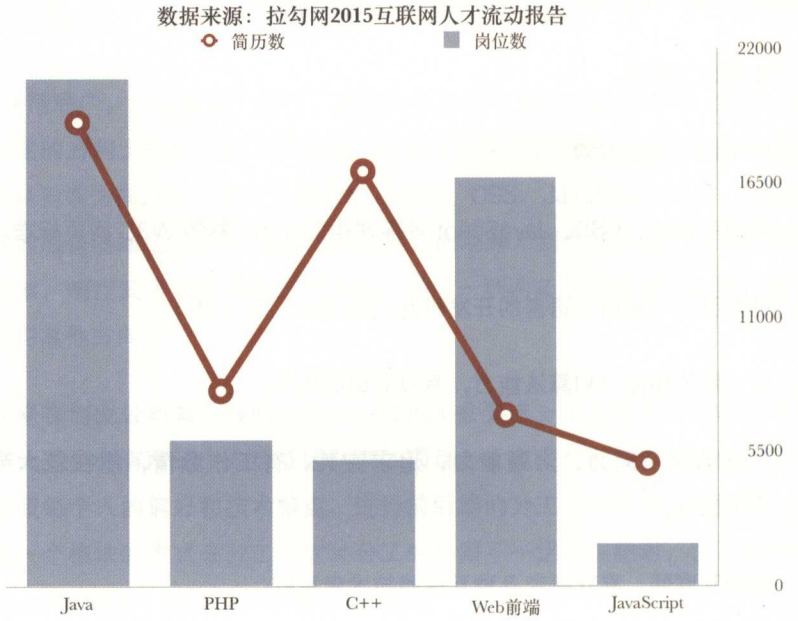
- 本科以上学历。
- 两年以上工作经验。
- 精通 HTML、CSS、JavaScript 等前端相关技术，熟悉 W3C 网页标准。
- 熟悉至少一种后台语言的开发机制（如 Java、C++ 等）。
- 有一定架构能力和算法能力，有良好编码规范。
- 良好的学习能力、沟通能力，追求完美，有工作激情，能在较大强度下工作。
- 热爱互联网，喜欢研究各种互联网技术者更好。

您想，自己完全满足要求啊，于是一封简历就投递到面试官的邮箱，里面用大段文字表达自己全面的能力完全符合这个要求，而且自己还有亢奋的激情和浓厚的兴趣。

但是您从面试官的角度来想想，他收到了多少份简历呢？对于一个大公司的 HR，可能 100 个都算少。

根据中国招聘平台拉勾网“2015 年互联网人才流动报告”，前端相关岗位的简历投递数只有岗位数的一半。与此同时，服务器开发方向（比如 Java、PHP、C++ 等）的简历投递数都大大高于岗位数。从图表可以看出，前端开发仍然处于人才紧缺阶段。

HR 要从 100 个符合要求的人中选择 10 个来面试，您的简历中的哪一点能吸引他呢？有的竞争者有丰富的移动端作品，有的竞争者提到他很擅长页面性能优化、响应式、页面渲染效率，有的写过 JavaScript 框架……而您只是一个普通的满足要求的人。



不同职位的供求关系是不一样的。

您可能会说，我爱好广泛，学习能力强，我会一点 PHP，做过 Wordpress 主题，会一点 Java，毕业设计做过一个小客户端应用，什么都会一点……但最终您仍然会得到一个“无亮点”的评价，被无情地淘汰掉。因为虽然您会的技能很多，但大多只能算是“及格”的东西。

所以，作为一个求职者，无论是毕业生还是社会招聘，仅仅满足招聘要求是不够的。**您需要在招聘要求的方向上以 200% 的能力来得到这个职位。**

一个求职者在整个流程中会受到多方考核：HR 考核您的成本和价值，专业面试官（不是全栈工程师）考核您的专业能力，经理考核您的沟通能力。在所有这些考核中，其实每一环都是漏斗型筛选，会过滤掉一些人。

好消息是，由于程序员的供求关系，只要您的专业能力够强，您就有很大的概率通过整个面试录用流程。我一次又一次提到“供求关系”这个词，是因为在商业社会，所有的商品（包括人才）的价值来自于供求关系，而

不是生产成本。生产成本是准入门槛，但绝不是核心竞争力。

让我再次重复这一点，作为求职者，一定要在某个特定方向上有非常深入的理解。仅仅会做还不够，还要理解背后的原因，还有背后的背后的原因。有些面试官的习惯是，在一个问题上深入地问下去，从经验问到操作过程，再问到技术原理，一直深入到面试官问不下去了，或者求职者答不上来了。所以，理解得越深刻，您就越有优势。

有了一个专长，得到一个能让您成长的工作，进入强大的团队，您就能有自己的阵地，以此为生，然后再逐步学习更加广博的知识，朝自己的个人目标去努力。如果您连阵地都不稳固，就不存在开枝散叶、落地生根的可能性了。

假设您已经在一个中等规模以上的公司找到了工作，那就会有一个专门的岗业。经过几年的工作和练习，您会在专业知识上达到很熟练的程度，日常需求都已经在您的“舒适区”，现在您终于准备好了。既然您的目标是做一个全栈工程师，那么从哪些技术开始入手呢?

围绕商业目标

我的第一条建议是，在考虑做什么项目的时候，围绕商业利益作为目标。归根结底，技术是服务于商业目标的。

在计算机科学诞生的短短几十年中，热门的技术和平台一直在发生巨大的变化。

服务器端的平台和语言从 C 到 C++、Java、Python，再到如今的 Node.js，变化从来没有停止过。

客户端则分浏览器和原生开发两个分支。浏览器方面，Web 标准是一个活的标准，意思是说，有一些新的提案不停地加入到标准之中，这是一个动态滚动的标准，而不是印刷出来的定案。

各种浏览器的市场份额每隔两年就会发生天翻地覆的变化，从 moz 到 Webkit，我们见证了 Webkit 的发展壮大。

移动端设备的市场份额之争更是激烈，曾经的诺基亚和摩托罗拉被新起之秀收购，iOS 和 Android 之争还在继续……

仅仅据我所知，2014 年到 2015 年腾讯就有很多团队进行了从 PC 端到移动端、从 HTML5 到原生 App 开发的各种转型。没有人能说得准下个季度我们团队的目标是什么，每半年就有一次大的调整，而小的调整从来就没有停止过。“变化”是唯一保持不变的东西，每个人都在不停地学习新的技术。

相对来说，商业目标是稳定的。把关注点放在商业目标而不是技术上，就能选择出更适合完成商业目标的技术，这样就能做出更为客观的决定。更重要的是，在这个过程中您学习到的不仅仅是技术，更是一种潜在的思维方式，这种思维方式可以帮助您提升综合竞争力，是一种“硬通货”的能力。

老板雇用一个员工，不是因为他能写程序，而是因为他能帮助自己赚钱。赚钱有两种方法：减少成本，或者增加收入。程序员如果能加快内部系统的运行效率，让产品制作流程更加顺畅，就是减少成本。如果能让用户更容易地购买产品，或者提高服务质量吸引更多用户，就能增加收入。在老板看来，程序员只是一个昂贵的劳动力，他会不会写程序都没那么重要，重要的是能赚钱。

所以如果您想成为一个高级开发者（或者高级设计师），就一定要学会这种思维方式。

所谓“商业目标”要广义地去解读。对于直接制作产品，给用户使用的团队，就需要对外关注如何提高产品质量、降低产品成本；对内应该关注如何优化流程、减少错误率。如果团队输出的成果是公司内其他部门需要的原材料，就要关注下游的需求，研究如何更好地输出成果，如何在流程上使得输出产品的过程更顺畅。

关注商业目标需要持久的练习。等到自己成为全栈工程师，或者成为团队

管理者，更加需要在多个目标任务之中做出选择。全栈工程师需要做和能够做的事情是很多的，他会很多技能，也负责处理很多工作，所以他更需要能力从诸多事情中找到最有商业价值的一个：可能是制作一款工具提升团队效率，也可能是成本上的优化。

全栈工程师可以做得事情越多，就越需要具备判断做什么的能力。如果增加一个用户需要的功能是加分项的话，拒绝一个用户不需要的需求更加值得推崇。

一切都要围绕商业目标来进行，包括您做的项目、您的汇报方式，以及您在学习新技能时进行的取舍。

我在公司的技术通道[1]中会发现有这样一些开发者，他们做项目的驱动力是“技术”本身，而不是“商业”目标。比如说，他们针对微信平台做了一个活动推广页，使用了很多华丽的 3D 旋转和 SVG 动画。好的方面如下。

- 用的技术很新潮，满足了自己的炫技虚荣心。
- 朋友圈（其实都是前端同事）传播很广。
- 在高端机器和大屏幕机器上效果很好。

坏的方面如下。

- 在低端机和慢速网络下效果不好。
- 沉浸在技术的实现中，而忽略用户体验。
- 打开页面就自动播放音乐，让用户感觉很突然。

我老婆是一位财务人员，她每次看到朋友圈这种很炫酷但需要加载的页面就会马上关掉，有时候耐心等待打开之后也是眼花缭乱，不知所以。所以有时候我会思考，一个技术的圈子，在热烈讨论某个推广页又用了某某炫

1 腾讯员工可以根据自己的特长和兴趣，选择走管理的发展通道，也可以选择技术、设计、产品、市场等专业发展通道。在不同的发展等级上，都设计有配套的能力要素。

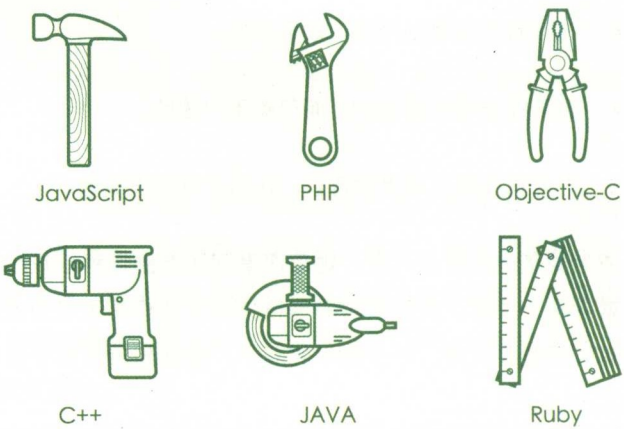
酷新技术的时候，有没有想到普通用户根本不买单呢?

再来说说一个好的案例。

我在面试求职者时遇到一个综合能力不错的候选人，他是一个全栈工程师。我问他，您现在掌握的技术比较多，那您未来的职业规划是怎样的? 他说，他觉得用什么语言并不重要，但是最近一年开始把重心放在 Android 开发上，因为移动端 App 开发是现在的潮流，有很大的需求，在这里可以有所成就。但在未来，不排除改变方向去做别的事情的可能，到时候可能是 iOS 或者其他新的系统。基本上来说，自己掌握的知识体系是可以复用的，但也期待学习新的语言。

我喜欢他这样的态度，对未来有自己的方向，但也知道自己没法看得太清晰。**对商业和市场有想法，而且自己也有足够的技术能力和自信向未来前进。**相比而言，有些候选者的项目经验和学习技能很杂，东一锤子西一榔头，有些时候纯粹是为了折腾而折腾。

记住，当您只有一把锤子，您看什么都是钉子。而如果您痴迷于工具，反而看不到问题所在。因此，要先看看有哪些问题需要解决，然后再补充您的工具箱。永远从商业目标的角度来决定学习哪些东西，而不是纯粹为了锻炼技术能力而去学习。



全栈工程师希望丰富自己的工具箱，而不是用一把锤子处理所有工作。

关注用户体验

我的第二条建议是，从用户体验的角度考虑问题。

用户体验是用户使用产品时的心理、感受、印象、评价。生活中处处涉及用户的体验，闹钟、牙刷、马桶、书包、公交、红绿灯、手机、电脑、键盘、鼠标……**每天，我们都在和产品打交道，每天都在使用和体验产品**。

每一个糟糕的体验背后都蕴含着商机

全栈工程师应该关注用户体验，并且掌握用户体验相关的知识。即使一个技术达人能够以一己之力搭建一个站点，但他如果不关注用户和客户的体验，那么他做的产品可能会很糟糕。这样的产品除了“能用”之外什么优点都没有。

所有优秀的工程师所做的一切都是在优化用户体验：优化性能的开发者是在积极地提升用户体验和交互；设计师有意用颜色、空间、大小和表单的排列方式让用户体验更顺畅好用；而内容运营者认为某些内容重要，某些内容不重要，也是在考虑如何提升用户的体验。

我在 2010 年加入腾讯的时候，公司只有一万多人。那时候，我需要办一些行政手续，需要公司开具薪酬证明，整个操作流程是这样的。

打开公司论坛，搜索“薪酬证明”，搜索到一篇文章，里面讲到找一位人力资源的员工来办理。

我打开 RTX（腾讯内部使用的工作通讯软件，类似 QQ），找到这位人力资源的员工，问他座位在哪里；跑到他座位上，此时已经有几个人在排队了，我排在后面；到我了，我告诉他我要办薪酬证明，并告诉他我的 RTX ID；等待 10 分钟后，他打印出一张薪酬证明，签字盖章后给我。整个过程耗费了我一个小时的时间。

2015 年，我要申请美国旅游签证，需要开具薪酬证明。我从平时的宣传渠道得知，现在人力资源的很多服务都可以在线上办理了，于是我尝试了一下，现在开薪酬证明的流程是这样的。

关注“HR 助手”的微信公共账号，它自动识别出我是腾讯员工，也得到了我的 ID。

选择“我要办证明”→“收入证明”，在证明用途一栏，选择“签证类”→“旅游签证”，并提交一些个人信息。

输入我的办公座位号，提交给系统。

第二天，一个漂亮的红色大信封放在我座位上。打开一看，里面包括中英文两份收入证明，还有我的旅游目的地以及时间，整个收入证明既漂亮又专业，是为签证量身打造。从提交系统到拿到最终的证明，我只花了几分钟，过程顺畅快速，体验非常好。

腾讯“线上申请”业务的体验极佳。

从 2010 年到 2015 年，经过这几年的发展，腾讯的员工规模已经达到了三万多人，翻了三倍。HR 流程如果还以旧的方式运作，可能得加派好几倍的人手，浪费所有员工不知道多少时间。但是现在通过自动化的系统，用户满意度大大提高。一个内部员工使用的系统，尚且有如此的优化空间和投入力度，何况是对外直接出售服务的公司呢？

我这样被公司服务“惯坏”的人，往往对社会上其他服务更加挑剔。此外，在深圳这样一个服务业水平居全国前三的城市居住惯了，去其他城市也经常会有被“怠慢”的感觉。我想这就是所谓“由俭入奢易，由奢入俭难”。

所以，用户现在都被手机中那些提供优质体验的 App“惯坏”了，想让他们再接受陈旧的设计和体验，就更加难上加难了。

用户是谁

“站在用户的角度想问题”这样一句朴实的话，可以指导我们做很多事情，但是很多时候我们忽略了这一步。

就像“体验”泛指所有生活中所有的体验。**这里的“用户”仍然是一个广义的定义：所有您为之服务的人。**

比如做一次演讲或者汇报，第一件要紧的事不应该是做 PPT，而应该是调查听众，站在听众的角度去思考：听众知道什么信息，听众想知道什么。如果给您的老板汇报，您不能期望他了解您所做项目的技术细节，而且他想知道的也不是技术细节，而是项目进度和风险。但是如果在一个技术论坛上分享，您就不能期望听众都知道您的项目背景和目标，需要花一点时间去介绍，听众也不想知道太多细节的东西，只需要介绍一些决策和架构的大方向。

写邮件的时候，收件人（还有可能这封邮件被转发之后的收件人）就是用户，那么写邮件的一些技巧就包括：尽量简短，不要给收件人太大压力；把结论放在邮件的开始，方便对方快速了解情况；如果需要老板拍板，给

出选择题，而不是问答题。总而言之，以对方能理解、会关注的方式来表达自己做了什么。

作为前端工程师，上游的设计师、下游的后台工程师，都可以认为是前端团队的用户。如果细心观察，就可以发现这里面有一些痛点。因为领导没有自己敲代码，所以他可能不会发现这些痛点，也就不会安排您去做优化工作。所以这里需要您自己去观察和优化流程。

很多程序员的第一个想法是做工具，但是想想我刚才说的话，老板雇用您不是因为您能写代码（或者做工具），而是因为您能帮他赚钱。所以您要用一切办法，去优化流程解决痛点，做工具是一个可选的方法，但不应该是您的第一个想法，更不是唯一的办法。假使真的是做了一个工具，最终汇报邮件的时候，不要以“我做了一个工具……”开头，而应该以“我发现了一个问题……”开始。

大巧若拙

老子（两个字都请以三声阅读）说，大巧若拙。

意思是，指**真正聪明的人，不会显露自己，反面从表面看好像还很笨拙**。用户体验不只是界面和交互这样可以直观感受的东西，还包括一些隐藏在用户界面背后的细节和规范。

就像冰山，露出水面的部分只占整个冰山的 1/9，用户看到的只是显露出来的部分。背后的部分一般用户是看不到的：比如用户研究，用研团队会通过调查，输出一些用户画像，影响整个产品的功能方向、设计风格；还有设计规范，设计团队在设计产品的一开始制定了规范之后，新增加的功能和页面都必须遵循已有的设计规范，这样整个产品是统一的，能够给用户专业的感觉。

为什么现在很多商业公司花了大把的钱和精力开发出独立运行的 App，体验却很糟糕，甚至很多用户反馈称 App 还不如微信公共号好用？

用户体验只是冰山上露出的一小部分。

一个很大的原因就是公司不重视用户体验，觉得用户研究和交互这种东西，不用专业人员去做，让设计师搞定就好了；或者老板拍脑袋定方案，做出的东西花里胡哨、炫酷狂拽，但就是让用户摸不着头脑。相反，微信花了很大的精力去做深入的研究，最后设计出了一套看似简单，但是可用性非常好的框架。然后微信开放后台系统给第三方，第三方公共号可以定制的地方有限，只能把功能往里面套，不太容易出错，用户体验自然就上来了。

反观某些银行的 App，几乎每个标签页的设计风格都不一样，而且喜欢自己设计键盘，每次在输入密码的时候都非常不方便，其实这是没有必要的。

做自己会用的产品

创业公司做产品，CEO 一定要是自己的目标用户。因为如果自己都不体验自己的产品，就很难发现用户在使用产品过程中遇到的糟糕体验。我们经常在网上看见网民抱怨办理公共事务时手续很麻烦，很多流程设置得让人抓狂。我想，这里面有一个很大的原因就是，设计公共事务流程的人，自己本身不是目标用户。

网上有个段子，说一般的产品经理没办法把自己代入成“小白”用户，做出的东西只有他自己会用；高级产品经理经过半小时的冥想可以进入小白状态；张小龙和马化腾这样的大师级产品经理需要两分钟；而乔布斯

可以随时切换大师级产品经理和小白的状态。这就是为什么他会说“stay hungry, stay foolish”。

我如果开创一个公司需要招聘“全栈工程师”，我要求的三个能力就是一专多长、关注商业目标、关注用户体验。

有志往全栈工程师方向发展的学生，我推荐从入门简单的前端开发开始学，而且从拉勾网“2015 互联网人才流动报告”来看，职位多、简历少排名第一的职位是前端开发。而且因为前端开发处于互联网开发的中间环节，可以从上下游入手，渐渐地接触 Web 开发的完整流程。第三个原因是，前端开发直接面对最终用户，也可以锻炼自己对用户体验的感觉。

当然，前端并不是唯一的选择，您也可以从其他职位开始，专注地学习这个职位需要的技术，到达一定的深度之后，扩展自己的知识面，往一专多长方向去发展。下一章专门介绍如何从学生转型为全栈工程师。

延伸阅读

1. 《重来：更为简单有效的商业思维》(美)贾森·弗里德 /(丹)戴维·海涅迈尔·汉森，中信出版社

2. 《精益创业》(美)埃里克·莱斯，中信出版社