解忧程序员——高薪编程、求职面试与成长转型宝典

安晓辉　著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书是专为程序员而编写的。全书浅显易懂，深入浅出，书中从各个角度，全面地解读了程序员这个特定人群，在日常程序设计工作中遇到的种种问题及解决办法。如果你不知道选择什么技术栈来学习，困惑于怎样在技术上持续精进，想转技术管理却没途径，想有章法地为跳槽加薪做准备，想转型不知道除了技术还能干什么，可以看看本书，它提供的方法和工具可以帮助你找到答案。

适读人群：程序员，准程序员，高校学生等。

图书在版编目（CIP）数据

解忧程序员：高薪编程、求职面试与成长转型宝典 / 安晓辉著. —北京：电子工业出版社，2017.10
ISBN 978-7-121-32610-3

Ⅰ. ①解… Ⅱ. ①安… Ⅲ. ①程序设计－指南 Ⅳ.①TP311.1-62

中国版本图书馆 CIP 数据核字（2017）第 213106 号

责任编辑：高洪霞
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：20.75　字数：397 千字
版　　次：2017 年 10 月第 1 版
印　　次：2017 年 10 月第 1 次印刷
定　　价：59.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

前　言

搞定难题，解决 Bug，项目成功，奖金到手，升职加薪，获得认可，备受尊重……开发者的幸福大同小异，然而开发过程中的迷惘、困惑、痛苦却千差万别，你未曾经历时往往无法想象，你突然面对时常常不知所措。你常常会想，要是有个过来人能和我一起聊聊该多好；你想知道他们是怎么过来的，你还想确认自己并不孤单。这就是这本书存在的意义，它汇总了笔者从软件开发工程师、技术经理、项目经理、项目总监到初创公司技术总监等各种岗位一路走来思考过的各种问题：

- 我适合做开发吗？
- 编程语言怎么选？
- 别人月薪 3 万元，自己只有 3 千元，想拿高薪，怎么做？
- 面对多个 Offer，怎么选择才不后悔？怎么拒绝不喜欢的 Offer？
- 简历投了几十份无人问津，问题出在哪里？怎么优化才能提高通过率？
- 公司都喜欢年轻、敢拼、能加班的程序员，我年龄大了怎么办？
- 程序员一定要转管理岗位吗？怎么转？有什么问题和挑战？怎么应对？
- 想跳槽，该怎么做准备才能找到理想的下家？
- 整天被 Bug 追着跑，怎么破？
- 怎样高效地阅读源码？
- 开发过程中经常要学习新技术，怎么学效率高？
- 感觉在混日子，领导安排任务才会去做，技术水平一般，也没动力学习提升，怎么办？
- 如何避免技术债务？
- 不知道设定什么目标，怎么设定才能让自己积极前进？
- 想参与创业公司，获得预期中的高回报，又怕风险，怎么办？
- 团队不稳，开发人员纷纷离职，我是离开还是留下？
- 想学习提升技术能力，可计划总执行不下去，怎么破？
- 面对多年的老代码，动还是不动？怎么动？

- 怎样激励别人积极工作？
- 想要转型，可除了技术，还能做什么？

这些问题，可能是你正在经历的，也可能是你将要面对的。当你被某些问题困扰、想看看别人怎么面对时，翻开这本书，它会默默地陪伴你，和你一起想办法，让你不再孤单；当你对现状感到迷惑、对开发者的未来感到担忧时，翻开这本书，看看走过山山水水的老炮儿留下的痕迹，它们可以作为镜子，照亮你脚下的路。

无论怎样，这本书，这些问题，这些思考，这些方法，都在这里，等待你需要它们的那一刻。

目　录

自我发现与选择

跳槽与薪水篇

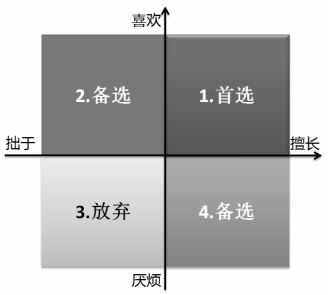
成长之路

管理迷思

自我发现与选择

职业四象限

一个人的职业选择，根据喜欢和擅长两个维度，可以分为四个象限，如下图所示：



该图中，横轴从左到右，表示个人对职业所需技能的擅长程度，纵轴表示个人对职业涉及领域的喜欢程度。

右上角为第一象限，所覆盖的职业自己既喜欢又擅长，是首选。

左上角为第二象限，此象限里的职业自己喜欢但不太擅长，可以作为备选职业。

右下角为第四象限，这个象限里的职业，个人擅长但不太喜欢，也可以作为备选职业。

左下角为第三象限，它涵盖的职业，个人既不喜欢又不擅长，是下下之选，若非不得已，可以直接放弃。

好了，简单介绍完职业四象限后，接下来我们展开一下，谈谈下面三个话题。

- 分析自己的职业象限
- 个人职业在不同象限间的转换
- 职业转换实例

如何定位自己的职业象限

想要判断自己是否喜欢所从事的职业。大体可以从以下四个方面来分析：

- 工作内容和对象
- 工作所用到的知识和技能
- 工作环境
- 工作中的人际关系

通过对上面四个方面的梳理和总结，你就能知道自己是不是喜欢眼下的职业。举个例子，比如杨大胖喜欢健身，李六块喜欢研究与健身相关的各种知识，王爱痴则喜欢到健身房里与教练聊天……这几个人都喜欢健身，但喜欢的东西都不一样。

其实判断自己是否喜欢眼下的职业，往往不用去做特别详尽的分析，只要跟随自己的感受即可。你觉得自己想做，做得开心，那就是喜欢。你觉得懒洋洋的，这也不愿意做，那也不愿意做，面对分配的工作总是想办法推三阻四，那就是不喜欢。

喜欢，但不一定擅长。比如李二狗喜欢养狗，但是他不一定擅长养狗，可能养 10 只死 8 只。比如猿大喜欢编程，但他不一定擅长，有可能写出来的代码脏、乱、差、运行效率低下，结果不符合预期。再比如刘四姐喜欢唱歌，可能不擅长，她唱起来就像驴叫……

擅长不擅长，每个人自己都能感觉出来。假如你遇到问题，能够很快运用知识、技能、资源找到策略，轻而易举地解决问题，那就是擅长；假如你总是感到很吃力，思虑几个月也找不到办法，那就是不擅长。

如果不能通过感觉确定自己是否擅长，那么可以通过与别人对比得出结论。比如同一个问题，别人三分钟搞定，你要三周，那你可能就不那么擅长。

倘若通过对比你也不能确认，还可以请别人帮你评价，让他列出你最擅长做的事情，多找几个人，总能得出结论。

总之，你闭上眼睛想几分钟，就可以判断出自己眼下的职业在哪个象限。

当你明确自己当前从事的职业所处的象限之后，就可以进一步详细地分析喜欢什么、讨厌什么、擅长什么、拙于什么。这个时候可以参考本节一开始提到的四个象限，详细的结果会对你后续的职业转型有很大帮助。

个人职业转型

理想的职业转型，应该是往自己既喜欢又擅长的方向努力。所以，对照职业四象限图，理想的职业转型应该是：**从第二、三、四象限过渡到第一象限**。

另外，从第二象限到第四象限，或从第三象限到第二、四象限，都是可以的。但是从一、二、四象限过渡到第三象限，则是要尽量避免的。

我们来聊聊比较理想的转型。

从喜欢但不擅长到既喜欢又擅长

第二象限到第一象限，从喜欢但不擅长到既喜欢又擅长，这是多么完美的进化啊。

当你处于第二象限时，会因为不太擅长做某些事情而有一定的压力，但因为你喜欢，很乐意去学，从而积极主动地提升自己，压力反倒会成为动力。只要多做事，积极淬炼自己，不断投入时间和精力，持续提升专业能力，就可以得到快速提升。就极有可能晋级到第一象限，遇见完美的职场生活。

从擅长但不喜欢到喜欢又擅长

第三象限的你，有一点点悲催——一直在从事不喜欢的工作。可能是历史原因，比如毕业时就业形势不好，先随便找个机会上班了；比如父母认为公务员好，稳定又体面……

你就在不喜欢中因为长时间的锻炼，对某些不喜欢的事情却很擅长，而这些工作，恰恰又成了你的主要经济来源。

此时你要往第一象限转换，需要放弃你现在擅长的技能，进入到陌生的领域，从零开始学习新技能，开启新的职业篇章。这是需要勇气的，因为很可能经济收入、社会地位、工作压力、人际关系等都会因此发生变化。

你可以从通用和专业两个维度出发，梳理一下自己的知识和技能。

那些可以在不同职业间迁移的技能，属于通用技能，比如英语、写作、开车、演讲、培训、组织、协调、沟通、项目管理、游泳、系统思维、高效学习、时间管理、领导、设计、审美、规划等。

受限于特定工作环境，离开那个环境就废掉的技能，属于专业技能，比如使用 C++编程、焊接、制作牛角梳、维护 IBM Z 主机等就是专业技能。

然后，你可以分析一下自己喜欢的职业，看它需要哪些专业技能、哪些通用技能，专业技能无法匹配的话，就选择那些能与你的通用技能匹配的职业，这样也会比较容易切入。如果一种职业所需的通用技能和专业技能你都没有，那么转换的可能性就会很低。

接下来我们要讲的故事——“饭姐”李雯，就是从第四象限切换到第一象限。她在职业转换时，“设计”“审美”这两项通用技能就发挥了很大作用。

“饭姐”李雯

李雯毕业后在上海一家外企做设计师，头几年她和许多同事一样，每天睡到快迟到才起床，早餐要么在路边随便买个煎饼、饭团，要么就不吃。

在工作了七年后，李雯早上一想到要上班打卡，内心就有点畏惧，甚至都不愿意出门。她担心这样日复一日的煎熬会让自己患上抑郁症，就琢磨找点其他的事情做，让自己的生活过得开心一些。她想起小时候，母亲端上餐桌的早餐总是特别丰盛，包子、煎面包夹鸡蛋、稀饭、豆浆等，热气腾腾香气迷人。于是李雯决定把早餐当作改变的开始，将其当作改变枯燥生活、与幸福邂逅的仪式。

每一天的早餐成了李雯对抗庸俗生活的武器。上班，不再是那么令人焦虑，而幸福感却越来越多。李雯体会到了以前从来没察觉到的幸福，“觉得生命好像被延长了，爱情也更加甜蜜了，感觉很幸福。”

有一天早上醒来，微博上有上千人@她，甚至有人在她发布的微博后面留言，“看到你的早餐，好想娶你回家”。吃货们羡慕而妒忌：“太凶残了，你怎么能把早餐吃得那么浮夸？”

事实是，李雯因为坚持做早餐、分享微博而红了，网友们给她一个“饭姐”的称号。她的每日早餐图也吸引了很多与美食相关的机构前来洽谈合作，于是她发现工作有了更多的选择，爱好也可以和事业结合起来。在 2012 年年底，她索性辞去了原来设计师的工作，做喜欢的事情，专心走进厨房，研发各式各样的菜品。2014 年，她的美食图谱《幸福，从早餐开始》正式出版，走进了大大小小的书店。现在仅当当网上，这本图书就有将近 4000 条评论。

2014 年 4 月，经过三年的积累后，李雯的美食创意工作室终于成立。她全身心地投入工作室的运作，也收获了满满的知足与幸福。

从厌倦的工作到喜欢的事业，这样关键的转变，对李雯来讲，却是从一份用心的早餐开始的。

工作之外，你可以做很多事情，那些你真正愿意做的、能够忘我投入的事情里，蕴含了潜在的机会，它们很可能导引你完成职业转换，让梦想照进现实，让你的生活变得充实又幸福。

李雯的故事，就是这样的，而这也是大多数试图转变却又顾虑重重的上班族可以采取的一个稳妥策略——业余时间丰富知识和技能。

这是非常棒的策略，适合大多数人，但前提是：你足够喜欢你想做的事情，并且能够持续投入时间和精力来丰富相关的知识和技能。

放下你的努力和坚持吧

我们总是非常努力地做一些事情，减肥、读书、跑步、理财、写作……但往往半途而废，功亏一篑。

有没有想过为什么？

这和我们常见的两种目标类型有关。

两种目标

关于目标，心理学家将其分为两种类型。第一种，可称为**接近型目标**，该目标设立的公式表达为“如果我做了 X 事，就能达成 Y 事”；第二种，可称为**回避型目标**，该目标设立的公式表达为“如果我不做 X 事，那么 Y 事就不会发生”。

举个结婚的例子来说明一下接近型目标和回避型目标的区别。

阿妹今年 28 岁，到了结婚的年龄了。如果她自己想找一个男人结婚，享受幸福的家庭生活，那么结婚这件事对阿妹来说，就是她的接近型目标。而阿妹的妈妈比阿妹还急着想让她结婚，妈妈的出发点是害怕女儿大了砸手里，嫁不出去，妈妈的目标，就是回避型的。

再举一个加班的例子来说明一下。

张无忌的目标是成为软件技术专家。大学毕业后他到了一家互联网软件公司做软件开发工程师，他每天晚上加班，不是干工作就是学编程，再不就是看技术书籍。他的目标就是接近型的。而于田英也天天晚上加班，但他是因为公司有加班文化，不加班就不被领导待见，也没有好绩效，甚至还有可能被辞退，所以他为了保住工作，就天天不情不愿地加班。那于田英的目标实际上就是回避型目标。

我们用简单的话概括一下接近型目标和回避型目标的区别：接近型目标源自于内心对个体成长的积极渴望，而回避型目标则往往是出于对无法生存的恐惧。

拥有接近型目标的人，会主动攀援高峰，追求成长和极致体验。

持有回避型目标的人，其所有的努力，都是“不得不”，因为害怕自己不这么做就会掉下悬崖。

你的努力和坚持为了什么

你天天辛苦工作，努力跟上公司“996”的节奏。

你的努力和坚持，到底是为了什么？
你努力和坚持的目标又是什么？
这个目标是接近型的还是回避型的？
请选择下面一种方式来描述你的情况：

- 我想要……所以我努力……
- 我害怕……所以我努力……

比如，我害怕领导批评我不上进，所以我努力和坚持。
比如，我想要成为 Swift 专家，所以我努力学习，哪怕加班加点也毫不在意。
你的选择是什么？

你不想做的，才需要努力和坚持

如果你非常渴望一个东西，你就会主动、自发地去接近它，你就拥有了一个接近型目标，你所做的一切，都是发自内心的渴望，你根本不会觉得做这些事需要坚持和努力——因为坚持和努力是一种自然而然的、自发的行为，你根本不觉得累，也根本不会有坚持努力的感觉。

一句话概括：如果你真的想做，根本没有所谓的努力和坚持。

如果你因为害怕某个东西，或者害怕不做某件事就会到来的某种惩罚而去做，你内心其实并不想做那些事情，而你却因为自己的恐惧而不得不做，那么你的动力必然不足，而没有动力还要完成事情，就需要坚持和努力。

你不想做而又觉得自己不得不做，才需用努力和坚持来克服隐藏在内心深处的阻力。

如果你找到了自己喜爱的事情，那么，你这一辈子都没有一天会觉得自己是在工作。所以，你根本不需要努力和坚持，你只需要做自己喜欢的事。

快快放弃你可怜的努力和坚持吧，现在，去找你喜欢的事，开始做你喜欢的事。

职业连连看模型

最近在分答（在分答搜索“安晓辉”可以找到我）上回答了几个问题：

我现在从事的工作是行政前台，感觉在这里发展空间不大，想换一份工作，但又不知道适合做什么，我该怎么发现自己适合做什么工作呢？

老师，您好，我在一个公司 6 年了，现在是一个主管，感觉没什么上升空间了，工作很轻松，是继续留在这，还是应该走出去闯闯？能给我点意见吗？

对外汉语教师，女，32 岁，本科学历。想转行到互联网，平时喜欢看理财的文章。请老师帮忙规划一下，谢谢！

老师你好！现在我还是一名大二的学生，金融专业，在班级担任学生干部职务，在大学应该怎样培养自己核心竞争力？

这些问题归结起来，其实是下面两个共性的问题：

- 如何找到目标
- 怎样达到目标

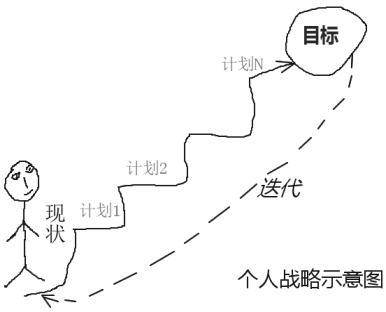
下面就提供一个简单有效的工具——职场连连看，迷惘的读者可以自己操作，迅速找到自己的目标，并规划出一个行动计划。

职业连连看

在开始之前，我们先来看看个人战略是什么。

1．个人战略

个人战略有三个关键词：现状、目标、计划。一句话来讲，所谓的个人战略，其实就是如何从现状出发，抵达你选择的对你具有重要意义的目标。所有的战略都可以简化为下面的个人战略示意图：



个人战略示意图

每个人各种各样的战略（比如你要成为什么样的职业人士，你要过什么样的生活），都可以拆解为上图演示的结构。

现状、目标、计划是上图最重要的三个要素，我们的职场连连看工具，也是基于这三个关键要素而来的。

2．职业连连看工具

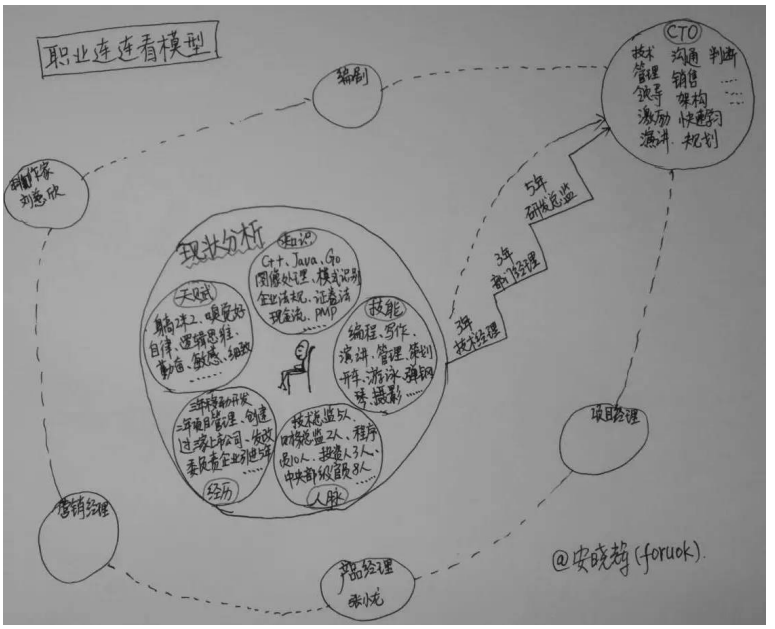
假如你要制定你的个人职业规划（个人职业战略），要完成的事情有以下三大件：

- 找到目标
- 分析现状
- 导出行动计划

目标从哪里来？现状怎么分析？行动计划如何导出？这几个关键问题，都可以在下图连连看模型中找到答案。

在职业连连看模型中，位于内圈的是个人现状，我们可以从知识、技能、经历、人脉、天赋五个方面来分析一个人的现状。

我们以软件开发工程师为例，他可能有比较强的逻辑思维能力、比较细致、自控力强，这些属于天赋，他拥有 C++语言、Java 语言、图像处理等知识，拥有编程、文案写作、项目管理等技能，在人脉方面可能认识很多程序员、项目经理、少数几个技术总监，在经历上，可能做过三年 iOS 开发、两年 10 人以下的团队管理工作。



从事其他职业的朋友们，也可以分析出类似的、具有商业价值的要素。

职业连连看模型的外圈，则代表了你理想的职业目标。比如一个软件开发工程师，他未来的职业目标可能是做一个中型软件公司（200~800 人）的 CTO。那他就可以在内圈和位于外圈的 CTO 那个小圈之间连一条阶梯状的线，这条线上将会填上多个阶段性目标以及对应的策略。

比如软件工程师张三，他的目标是将来做一个科幻小说作家，像刘慈欣那样，写出荣获雨果奖的科幻小说。那么他可以将内圈和位于外圈左上角的科幻作家那个小圈连接起来。

内圈，即现状分析，是有迹可循的，我们每个人在撰写简历时，都会做类似的梳理，现在只需要按照知识、技能、经历、人脉、天赋五个方面来分类即可。订阅号“程序视界”里的文章“程序员这样优化简历，一投制胜”具有很强的参考性。

3. 目标从哪里来

在职业连连看的外圈，放置我们想要的职业目标。那么这些目标可以从哪里来呢？有这么几种可能：

- 现有职业的延伸。比如软件开发工程师可以考虑架构师、技术经理、CTO 等。
- 各个时期的梦想。比如小时候你想长大了当个演说家、演员、歌手、科学家、记者等。
- 对理想工作与生活状态的分析。如果没有明确的职业目标，也可以从想象你最理想的工作、生活状态出发。（比如你希望在干净、明亮的环境里对着很多人分享，那可能适合去做讲师、演说家；比如你希望在一个个城市、一座座山川、一条条河流之间穿梭，拍摄美轮美奂的风景与人物，那摄影师就可能是你的目标；比如你希望在一个温馨的环境里单对单、面对面的与人交流，帮助别人走出心理困境，那心理咨询师可能适合你；比如你就享受领导一个团队完成艰难的目标，就可以考虑项目经理；比如你喜欢讲故事，喜欢看到有人把你故事里的人物呈现出来，编剧也许是个不错的主意……）
- 榜样与偶像。很多人都有偶像，或者有一个内心的榜样，想成为像某个人那样的人。比如你想成为马云那样的人、马东那样的人、姬十三那样的人、罗振宇、黄晓明那样的人……从这些你羡慕的、想成为的人身上，也可以找到你的目标——你可以问问自己最喜欢的榜样具有哪些特质、最羡慕他们拥有的什么东西（物质、知识、技能、职位、身份、社会标签等），列出你想要的那些，可以组合出你的目标。

- 成就事件。大部分人都经历过让自己特别有成就感的事情，可以回想一下，这件事发生的背景是什么？我因为什么而感到有成就感？在这件事中我用到了什么知识、技能？我是怎样做到的？还可以自我推敲一下，类似的事件，在什么样的行业、职业中可能重复发生？没准从这里面能找到非常理想的目标。
- 在没有金钱压力时你想做的事。当你不需要考虑经济压力时还愿意做的事，往往是你内心真正想做的事。那么这些事有没有可能现在就开始做起来？
- 低谷事件。我们都经历过一些让我们挫败、沮丧、情绪低落的事件。分析一下，为什么会这样？我们感受的低谷情绪，是因为这件事挑战了我们哪些方面？当你看到你不想要的东西时，它们的反面，往往会有你想要的。因为想要有时是通过不想要来表达的。
- 未尽的责任。如果你用尽办法，也找不到一个让自己心动的目标，不妨退而求其次，问问自己对父母、妻儿、朋友、社会，还有什么未尽的责任，该如何去尽这些责任。为别人付出，通常会给我们带来自我认同和肯定、自信心和某种程度的存在意义。

4．目标所需技能分析

一旦你找到了几个目标职业，接下来就需要分析：这个职业需要什么样的知识、技能、天赋、人脉、经历。这一点和现状分析是类似的。

有时目标职业与我们不相关或者距离较远，我们缺乏获取相关信息的渠道，无法直接拿到相应信息。下面的途径有助于我们搜集到有用的信息。

- 公司内的岗位职责。很多公司都有岗位描述，每个岗位需要什么知识、技能，有什么职责，都会列出来。如果要评职级，也会根据岗位和职级描述进行相应的考核。职级描述里的信息往往非常明确，可以给我们指引方向。
- 招聘信息。公司在招聘人时提供的招聘信息里，一般都包含所需知识、技能、经历，以及通用能力的要求（如团队协作、沟通、表达、规划、激励等）。
- 各种职位、职业信息百科。如中国教育在线的职业信息大全（http://career.eol.cn/html/sy/zhiye/），CNTV 点亮栏目的职业百科（http://dianliang.cntv.cn/baike/index.shtml），学职平台的职业百科（http://xz.chsi.com.cn/occupation/index.action），类似的还有很多，他们都有行业、职业的说明，相关说明里就有知识、技能等方面的描述。
- 生涯人物访谈。如果你的人脉中就有人从事你想从事的职业，那么你可以直接找到这个人，进行生涯人物访谈，不但能了解到该职业所需知识、技能、通用能力，还

能通过鲜活的样本了解到该职业的工作状态以及对生活的影响。同时，我们还能从这些人那里得到一些过来人的建议，这些建议有时能让我们少走很多弯路。

- 搜索引擎。搜索职业的关键字，往往能看到你要的信息。
- 行业人物访谈。每一个行业都有一些卓越的代表，这些优秀的职场人士，经常会被一些媒体访谈，这也是我们了解信息的有效渠道。
- 名人传记。有时你会对某个名人特别感兴趣，比如李嘉诚、马云、乔布斯、李开复、路遥、贾平凹、马化腾、张朝阳等，特别有名的人往往会有传记。从人物传记中，我们可以看到一个人完整的成长过程，他的性格、特质、习惯、经历，不仅能提供我们所关心的职业的信息，还往往能给我们带来情感和人生方向上的启发。

当你知道了目标职业所需的知识、技能、天赋、人脉、经历时，就能够拿它们和你拥有的知识、技能、天赋、人脉、经历进行对比，找到差距。把这些差距一个一个列出来，就可以形成一条知识、技能演进线路，我们就有了努力的方向。

丰富知识、技能的方法

到这里，你应该已经找到了目标职业，也列出了需要丰富的知识、技能、经历，那么，接下来就是如何提升自己了。

现在互联网如此发达，人与人、人与资源、人与服务的链接如此方便，如果你想要学习某方面的知识，一定会找到很多方法。

像读书、看视频课程、慕课、培训班、主题社群、能得到相关技能的工作、拜师、在尝试中体验……都是非常不错的途径。

职业转换策略

“转行穷三年”告诉我们职业转换的成本很高，也正因为此，很多有换行想法的朋友们举棋不定，“晚上想想千条路，早上起来走老路”，始终不能迈出第一步。

比较常见的职业转换策略有三种，一定有一种适合你：

- 业余时间丰富知识、技能，并利用相关知识、技能获取收入，等到新职业的收入差不多能满足你的生活所需或者可以明确判断出新职业在不远的将来就可以带来较好收入时，果断转换。我在“职业四象限”一节提到的饭姐李雯，用的就是这种策略。

- 先从事目标职业相关的周边工作，适时转换。比如你想做软件开发工程师，但没有任何基础，就可以先到软件公司从事测试工作，然后应用上面的策略，适时寻求内部转岗。再比如你想做投资顾问，可以先从客户经理这个岗位做起。
- 不惧从零开始，直接跳到目标职业。有魄力，我喜欢。但是，可能机会成本比较高。

怎么开始行动

经过前面的分析，我们有了目标、有了计划，可这才是第一步，更重要的是坚定地执行。人生永远没有太晚的开始，但你得有开始才行！

为了能够更好地开始，第一步的行动计划一定要相对简单，可以立即开始，并且能够在较短周期内完成，给你带来成就感，形成初始的正向激励。

比如你想成为心理咨询师，那第一阶段的行动计划可能是这样的：通过国家心理咨询师考试。

而这个短期目标，又可以拆分：

- 了解国家心理咨询师考试的资格要求。
- 报名参加培训。

这个阶段以报名参加培训为完成标记，完成的时间周期，可以设定为两个星期。

一旦你报了名参加培训，接下来就有很多事情可以做：

- 选择课程时间。
- 制定学习计划，比如每周花 10 个小时自学，3 个小时现场培训。
- 选择考试时间。
- 在培训过程中，还可以和授课老师建立联系，了解心理咨询师的工作情况，比如怎么提高实战能力、本地有哪些机构可以提供就业机会、如果想自己执业需要什么条件……

所有这一切，只要你开始了第一步，就会一步一步走下去。所以，最最关键的是，要开始，要尽快开始。

我们迟迟不能做一件事，往往是因为我们没有觉得这件事对我们很重要，我们没有那种不做这件事就没法活的感觉。所以，在确立目标职业时，一定要找到那个你最想要的、错过就后悔终生的，这样就会有很强的内在驱动让你开始行动。

然后呢，梳理现状和目标，找出差距，制定计划，计划要分阶段，距离现在越近的计

划要做得越详细，最后导出立马可以开始的第一步。就像我们刚刚举的心理咨询师的例子，第一步要足够简单，具备可执行性，能够马上去做。

有了这些基础，就可以很快开始了。好的开始是成功的一半！

开始之后，要时刻牢记你的目标和计划，把它们作为头等大事，给予足够的时间和精力，这样，一切都会慢慢好起来，最终就能实现你想要的转换，成为你想成为的那个人。

问答 | 我适合做软件开发吗

最初我在运营微信订阅号“程序视界”时，设置了两个免费的互动栏目：“程序员的职业规划”和“有问有答”。很快就不断有人加我微信，找我聊程序员职场相关的话题。短短两个月内，线下与 5 个人当面讨论过，微信和 QQ 上聊过的人就太多了。

后来我把聊天记录都汇聚在一起，回顾了一下，发现在初入行这个阶段，有一些共性的问题，特此写这篇文章，希望对将要加入计算机软件行业，从事开发工作的朋友和刚刚开始做软件开发的朋友有所帮助。

对于初入行的朋友来讲，有很多的忧虑和担心，我从整理的聊天记录里发现了下面 7 个问题：

- 我是否适合做软件开发？
- 哪个方向好，比较有前途？
- 学哪种语言？
- 怎么学？
- 第一份工作怎么找？

- 怎样才能变优秀？
- 怎么才能开始行动起来？

这次我会着重说第一个问题，捎带着可能会提第 7 个问题，其他的问题我们留到后面去聊。

三位朋友的提问

2016 年 1 月 1 日，“小眼睛”在微信上问：

您好，我是一名 23 岁的建筑系刚毕业学生，对目前的工作确实提不起兴趣，没有成就感，想尝试考研转到程序员行列，不知道自己是否合适，很迷茫，不知道您有什么好的建议吗？

2016 年 1 月 6 日，“想飞的小鸟”在 QQ 上问：

你好，我大学学的专业是电子信息科学与技术，毕业快 5 年，但毕业之后转行从事了建筑建模工作近 5 年，而后跳槽，接着公司破产，现处于待业状态。现在又想转行做回 IT，我马上 29 岁，外面有些 iOS 和安卓开发的培训，我在疑惑这些培训是否靠谱，我的年纪对于重新学习 IT 会不会有点迟……

2016 年 1 月 9 日，“谁夺走了我的爱”在微信上问：

您好！我 29 岁，在北大青鸟培训 Java、C#，初级课程学完，中级开始，但是学得并不是很好。总是处在迷茫中……

前面是几位想要从事软件开发工作的朋友问我的问题，昵称是我随便起的。还有一些朋友问到类似的问题，这些问题归纳起来，其实是一个问题：我是否适合做软件开发？

我是否适合做软件开发

这其实是一个非常难以回答的问题，业界并没有一个标准，说哪种人适合做软件开发，哪种人做软件开发就一定能大有成就。

温伯格在《程序开发心理学》一书的第 8 章，专门讨论了性格因素对程序开发的影响。温伯格认为：

相比较为稳定的智力因素，性格特征对软件开发的影响更大，甚至远远超过人们通常的估计。

他提到了专业程序员需要的一些性格和能力，诸如“在高压力的环境中坚持一个多星期的能力”“适应变化”“爱好整洁”“谦逊”“幽默感”等。然而他后来又说，我们从正在从事软件开发工作的人身上来寻找适合这个职业的性格，可能错过了那些适合这份职业但还未进入这个行业的优秀人员身上表现出来的其他性格。

MBTI 把人分为 16 种性格类型，每一种都有相对适合的工作。其中，INTJ 和 INTP 这两种类型的典型职业里有提到程序员、计算机软件设计/开发人员。

也许我们做个 MBTI 测试就可以明确自己是否适合做软件开发？也不是的，我的 MBTI 类型是 INFP，但我都做了十多年软件开发了……

看起来似乎没有一种明确的方法可以判定一个人是否适合从事软件开发工作……这是一个令人沮丧的结论。

然而我们也不必因此而失落，实际上 MBTI 性格类型、还有霍兰德职业兴趣理论，都提供了一些手段帮助我们界定自己适合的职业范围。温伯格列出的一些性格特点也确实是程序员需要具备的。

所有这些都从群体特征上，帮助我们缩小了选择范围。但对一个个体而言，到底是否适合做软件开发工作，还存在很多的具体因素需要区别对待。

验证适合与否的实操方法

我知道一套具有很强可执行性的实操方法，能让一个个体在比较短的时间内判断出来自己是否适合从事软件开发工作。非常简单，就是：先评估性格和兴趣特征，然后通过实践去尝试。

性格特征评估可以用 MBTI，兴趣特征可以用霍兰德，都有标准的量表，这些理论和量表，得出的结论仅供参考，而且它们通常也不会板上钉钉那般给你一个方向，说“就这个了，去吧，信我者得永生”。

量表测试只提供一个相对适合的参考职业范围，更重要的是我们如何通过实践去验证某个职业真的适合我们。

结合提问的朋友的情况，还有我自己的经历，我觉得要从个人具体情况来谈如何通过实践验证自己是否适合软件开发，大概有三类：

- 即将走向社会的学生。
- 正在从事其他行业的职场人士。

- 暂时无业者。

当下的职场状态，会在很大程度上影响你的选择。上面的划分，正是从个人的职场状态来的。

有的人可能很不满意现下的工作但又忙得像高速旋转的陀螺，根本停不下来，更别说花时间去实践了。而对于学生来说，则会有比较充裕的时间在较轻的压力下来验证（没压力也可能难以有动力^_^）。对于无业者来说，如果没有积蓄，也没有父母或另一半的经济支持的话，生活压力会比较大，会大大地影响选择；而如果还能过得去，就可以相对从容地去验证自己是否适合从事软件开发。

不论哪种情况，都不要随随便便把自己像“行货”那样交出去，政府、家人都不能为你负责，只有你自己才能为你的选择担起责任来。

注：王小波有篇杂文，《“行货感”与文化相对主义》，可以看看。

好啦，现在可以来聊聊实践验证的三个步骤了。

1．职业信息搜集

想必你考虑是否要成为一个程序员时，已经了解了很多信息了。比如计算机软件行业是怎样一个行业、软件是什么、开发人员怎么生产软件、行业就业前景、典型的业内公司的情况等。

那这些信息够不够呢？要回答这个问题，得从职业的定义说起。

通常我们所说的职业指职能，比如医生、教师、程序员，实际上包括两个维度：职能和行业，职业的定义应该是这样的：

$$职业 = 行业 \times 职能$$

行业是指其按生产同类产品或具有相同工艺过程或提供同类劳动服务划分的企业或组织群体的集合，如饮食行业、服装行业、机械行业等。从这个角度看，构成行业的很重要的基础因素就是知识组合，一个行业就有一套知识系统。

职能则着眼于企业内的岗位的技能分工，比如会计、出纳、招聘专员、软件开发工程师，要求的技能都不一样。

基于这样的认识，我们在搜集职业信息时，要了解行业信息、行业内产品信息、知识范畴、行业内的企业构成、行业的前景、企业内的岗位信息、对应岗位的人具体干什么事情、需要什么样的技能、有什么样的社会地位。

所有这些信息都是相对客观的信息，可以从网络、传统媒体、图书等各种渠道获取到。

要大量的，详实的收集相关信息，这将有助于帮助我们判断某个职业是否适合我们。

比如一个人对使用计算机完全不感兴趣，对编程语言完全不感兴趣，那基本上就可以排除做软件开发的可能了。

2．生涯人物访谈

我们搜集了客观的职业信息，还要了解从事某一职业的个体的感受，说白了就是从事某种职业的人的状态如何。

以快递行业为例。当我们想做快递员时，了解快递员的日常工作情况、正在做快递工作的人员的感受，对我们的选择影响非常之大。比如你很难接受在寒风和雾霾中开着电动三轮车走街串巷，那么快递员职业就不适合你。

以软件开发工程师为例。很多人搜集了职业信息之后会对 CBD、高薪、27 寸大显示器、Mac Book、规律、专业、智慧等感兴趣，觉得所有这些点都是自己想要的，就准备报名上培训班了。然而，如果你不实际接触一个从事软件开发工作的活生生的人，就很难了解真实情况：需求经常变更、加班是行业潜规则、技术更新快、多数领导只关心进度不关心人、压力大、Bug 会让你抓掉所有头发、随时可能被炒鱿鱼或者被告知公司关门……

所以，找到从事相关目标职业的人聊一聊非常有必要。我大学毕业时找工作，眼前一片茫然，不知道每个行业如何，也不知道找谁去问，可是我远在农村种地的父亲却深知了解具体工作情况的重要性，居然拐弯抹角托了一长溜人打听了四川成都一家航天研究所的情况。所以，有时我们找不到从事目标职业的人物，只是因为我们没把生涯人物访谈当一回事儿，没有意识到其重要性。

有时我们也会找错人，比如我们想了解程序员的工作状况，如果你找到一个祥林嫂式的人物，他强大的负能量和裹脚布一样长的抱怨，很可能误导你得出错误的结论。所以，我们要找那些看待事物比较客观的人，有时为了避免受单一样本的失真影响，还要多找几个人。

其实每一个人谈论的都是自己。看起来他在说公司、说产品、说团队的氛围、说加班的感觉、说需求的变更……其实都是在说自己的感受，而同一件事对人的影响，在不同的人身上差别很大，有人觉得无伤大雅，有人可能就痛不欲生。所以，我们在做生涯人物访谈时，也要有区别的对待、批判性的思考，不能学过河的小马。

了解、接触软件开发到底是怎么回事儿非常必要，可能你找个行内人聊上一个小时，就能得出自己不适合干软件开发这样的结论。当然也可能你聊着聊着眼睛就会越来越亮，

越聊越发现自己适合干这个，那就可以进入职业情景体验环节了。

3．职业情景体验

现在有很多儿童职业体验馆，可以让孩子们在不同职业体验主题店中扮演各行业成人职业角色，在玩乐中培养职业理想，规划自己的未来。

对想要进入某个行业从事某种职业的成年人来说，职业情景体验非常重要。如果你能在实际工作环境中待上一天，就可能改变你对某种职业的看法。

我个人认为，职业情景体验分两个层次：

- 近距离观察。
- 实作体验。

（1）近距离观察

近距离观察，可以在你无法立刻从事相关工作的情况下采用。比如你已离开学校，进入其他行业，就像前面提出问题的“小眼睛”那种情况，就可以先采取这种方式来观察。具体展开，又有很多不同的方式。

- 到软件园附近，观察过往的人的精神状态。
- 中午混在去吃饭的人群中，留意大家在聊什么，是吐糟需求又改了，还是 Bug 找不着，还是领导又强制加班了，这些信息都能得到，到软件园附近的小饭馆，也可以听到很多有趣的谈话。
- 找一个从事软件开发的朋友，到他的公司待一天，看看大家的工作状况，多观察几个人，看他们的精神状态。
- 到车库、必帮或者 3w 等咖啡馆喝杯咖啡，支楞起耳朵听听周围的人是不是为创业疯魔了

……

只要有心，有很多途径可以让你较为深入地从活生生的人的角度来了解一种职业。

有时你看看从事某种职业的人的表现，就能笃定自己是否适合做这个。同一个行业的人会有很多共同表现，你看到的那些人可能就是你将来的样子，你可以选择是否成为那个样子。To be，or not to be，往往就在一瞬间。

刘邦见到秦始皇出游，说了一句话——“大丈夫当如是也”，后来建立汉朝。

克林顿原本在读音乐系，吹萨克斯管，17 岁的时候遇到肯尼迪总统，后来决定从政当总统，真当上了总统。

这些例子可能离我们较远，但他们确实说明了榜样的力量。你要成为什么样的人，就和什么样的人在一起！所以，你近距离地观察程序员的生活，留意上两天，就可能会做出正确的决定。

（2）实作体验

你观察了某个职业人群，觉得自己就是要成为那样的人，那就可以再往前走一步，看看是否真能做起来。

去做做看，这就是验证是否适合软件开发的终极武器。

正在上班的人可能会说工作挺忙的没时间去做做看，其实时间总是有的，就看你是否真的渴望去做这件事。白天上班，那晚上呢、早上呢、周末呢？如果你真的重视一件事，就可以为它挤出时间来。当一个人不想做一件事时，总会找到理由的。

对于软件开发来讲，试试看的成本很低——因为自己一个人就可以开发点东西出来。

所以，我给那些想转做软件开发的朋友的建议，基本上都是这样的：

先找本教授写的编程语言的书（视频等）来自学，看自己能否学进去，能否自己把环境搞起来做点小实验，坚持一段时间，看看自己是否还有兴趣，如果还有兴趣，那就把它学完。也可以去报一个软件开发培训班，培训可以带你快速熟悉一个技术方向。但不能只依赖培训，更重要的是坚持自我学习和自主练习。

一旦你能通过自我学习掌握一门编程语言，就可以继续往下走，做更深入的体验。比如自己去做一个解决实际问题的小软件（像图书管理系统、记账、博客系统、待办事项等），或者寻找免费兼职的机会，这样你可以锻炼自己掌握的技能，也可以进一步积累经验，判断自己是否适合继续往前走。

如果有机会，能免费进入一个公司去实习或兼职也是非常好的，在一开始，重要的是锻炼的机会。为将来正确的方向，花上一点时间和代价，非常值得。

四句话总结

洋洋洒洒好几千字，其实总结一下，核心意思就四句话：

- 从性格和兴趣角度判断是否适合做软件开发。
- 搜集客观的职业信息，从知识、技能、工作内容上帮助判断。
- 通过生涯人物访谈了解别人对某种职业的感受和看法。
- 通过近距离观察和实作这两种方式的职业情景体验，进一步验证软件开发是否适合自己。

如果我们能按照上面的方法走完整个过程，我们的决定就会靠谱很多。

问答 | 当你选择编程语言时你在选择什么

在“问答 | 我适合做软件开发吗”中提到了“我是否适合做软件开发”和“怎样开始行动”两个问题，这次我们来回答这个问题：我应该学哪种语言？

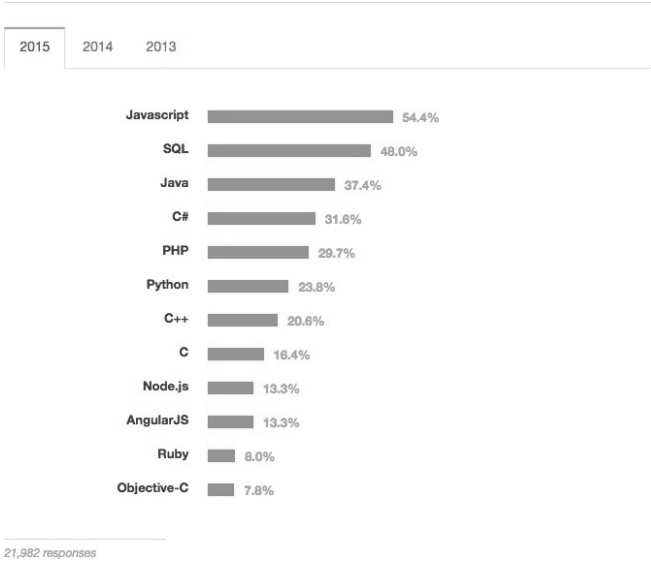
编程语言流行度在说什么

TIOBE 编程语言社区排行榜是编程语言流行趋势的一个指标，每月更新，这份排行榜排名基于互联网上有经验的程序员、课程和第三方厂商的数量。TIOBE 的排名很值得参考，但请注意这个排行榜只是反映某个编程语言的热门程度，并不能说明一门编程语言好不好。下面是 2015 年 TIOBE 的编程语言排行：

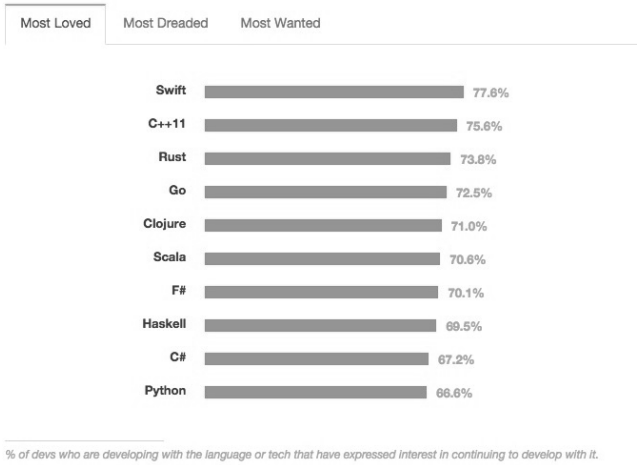
Dec 2015	Dec 2014	Change	Programming Language	Ratings	Change
1	2	^	Java	20.973%	+6.01%
2	1	v	C	16.460%	-1.13%
3	4	^	C++	5.943%	-0.16%
4	8	^^	Python	4.429%	+2.14%
5	5		C#	4.114%	-0.21%
6	6		PHP	2.792%	+0.05%
7	9	^	Visual Basic .NET	2.390%	+0.16%
8	7	v	JavaScript	2.363%	-0.07%
9	10	^	Perl	2.209%	+0.38%
10	18	^^	Ruby	2.061%	+1.08%
11	32	^^	Assembly language	1.926%	+1.40%
12	11	v	Visual Basic	1.654%	-0.15%
13	16	^	Delphi/Object Pascal	1.639%	+0.52%
14	17	^	Swift	1.405%	+0.34%
15	3	vv	Objective-C	1.357%	-7.77%
16	20	^^	MATLAB	1.168%	+0.30%
17	15	v	Pascal	1.147%	-0.03%
18	12	vv	R	1.122%	-0.51%
19	14	vv	PL/SQL	1.103%	-0.23%
20	26	^^	COBOL	0.828%	+0.17%

Stack Overflow 是最火、最专业、最有效的 IT 技术问答网站，很多机构或个人通过它的标签来分析编程语言的流行趋势，也具有很强的可参考性。

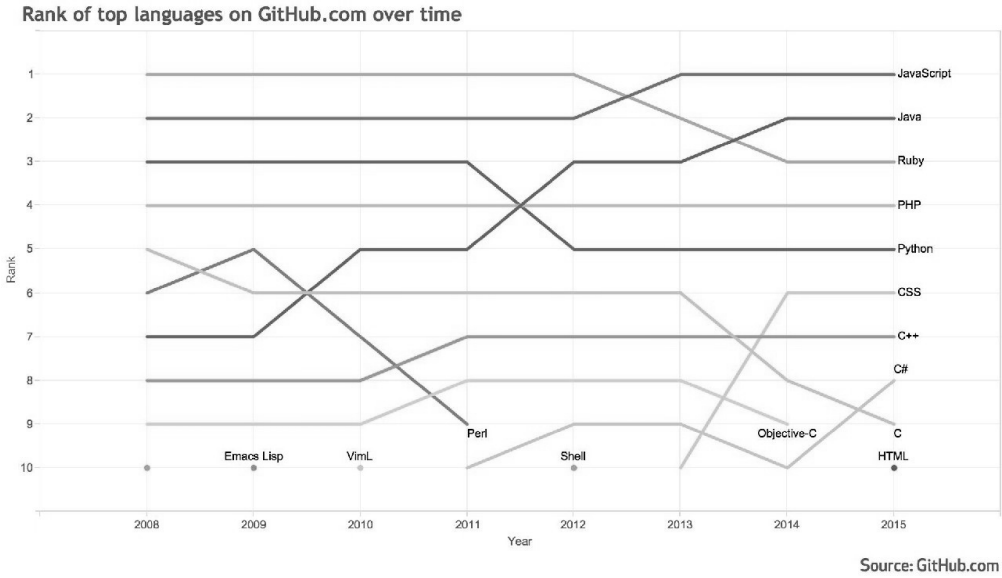
Stack Overflow 2015 年的开发者调查结果中最受欢迎的技术如下：



最被喜爱的语言如下：



Github 是全球最火最流行的开源代码托管站点和社区，下图是其 2015 年 8 月份的数据，包含了最热的 10 大编程语言，以及 2008—2015 年热门语言的变化趋势：



关于编程语言和技术的流行度排名有很多，它们在说什么？我们能否根据它们来选择学习哪门语言？

编程语言的流行度说明了：

- 哪些语言当下一段时间内比较流行。
- 比较近的将来（2~3 年）哪些语言比较有生命力。
- 语言的就业难度。

所以，在选择语言时，编程语言排行榜有一定的参考价值。编程语言的热门程度在很大程度上决定了你能够进入什么样的公司，获得什么样的项目。

选择语言时你在选择什么

对还没入行的人来说，要选择一门编程语言，确实要考虑很多因素。在这里，我把我知道的列出来，供初学者参考。

1．技术图谱

当你选择一门编程语言时，你不仅仅选择了语言本身，还有围绕着这门语言产生的各种各样的应用框架。比如你选择 C++，将来就可能接触到这些框架：

Boost、Qt、Silicon、TuFao、TreeFrog、WTL、STL、libev、Cocos2d-x、Dlib、Ultimate++、Asio、TinyXML、libxml++、CEGUI、FLTK、wxWidgets、Ogre 3D、Cairo、Skia、OpenCV、CxImage、Json++、Json11、CppUnit、OpenSSL、Crypto++、C++React、OpenCL、OpenAL、Vorbis、Memcached、libcurl、SQLite……

除了直接与你所选语言相关的技术框架，还有一些可能经常会与你所用语言搭配的其他语言，比如你选择 C++，那么 Java、C、Python、JavaScript 就是与它搭配使用比较频繁的语言。像 Node.js，就是混合体了；JNI 也在 Java 和 C 之间搭建了桥梁……

你最终会根据你选择的语言构建出自己的技术图谱，你的技术图谱决定了你将来的发展和你在企业眼中的价值。

从这一点讲，应当关注流行度高、应用广泛、有大公司参与的语言。

2．行业

有机构通过分析与 C++语言相关的招聘信息得出的结果，使用 C++语言的前三个行业是：金融、银行和游戏。然后是：Front Office、通信、电子、投行、市场、制造业、零售业。

你选择了某种语言，就可能到该语言应用最多的行业里去。所以，在选择语言时，有必要了解对应语言被应用最多的行业。

也有时是你想做的产品决定了你要用的语言，比如你想么做 iOS 上的游戏，那么选择 Objective-C 就比较靠谱，而如果你想做跨平台的游戏，支持 Android 和 iOS，可能 C++和 Cocos 2d-x 会比较适合。

3．薪水

技术本身无优劣，应当根据特定场景下适合不适合来选择和看待。打个比喻，产品和需求是脚，技术是鞋子。但，不可否认的是，在你择业时，不同的语言，薪水起点是不同的。

比如你选择 Objective-C、JavaScript、Android，起点是不同的，到招聘网站搜索一下对应岗位用人企业给出的薪水范围就可以看出来。

职业规划公司 Gooroo 通过仔细查看了美国、英国和澳大利亚的超过 50 万份的 IT 职位空缺，统计出了需求热度排名前 10 的语言对应的职位的薪水（http://www.sitepoint.com/

best-programming-language-learn-2015-job-demand-salaries/，下面表格中的比率指某种语言在招聘广告中出现的比率，年薪指平均年薪，单位是美元）：

排名	语言	比率	年薪($)
1	Java	18%	100,000
2	JavaScript	17%	90,000
3	C#	16%	85,000
4	C	9%	90,000
5	C++	9%	95,000
6	PHP	7%	75,000
7	JPython	5.5%	100,000
8	R	3%	65,000
9	Scheme	3%	65,000
10	Perl	3%	100,000

语言有地区性差异，上面的数据可能更偏向美国。国内我没找到这样的数据，但语言与薪水有相关性，这是毋庸置疑的。

4．同行多寡

不同语言使用的人数不同，你选择语言时还会选择和哪些人同行、和多少人同行。搞技术无人切磋是寂寞的。所以，使用人数多少也是一个考虑因素。有时一门语言的绝对使用人数并不能影响你，相对的，你身边有多少人使用这门语言可能会对你有很大影响。此时地理因素又很重要，比如 Scala，可能在北京有一些人用，在西安则找不到几个人使用，你要在西安使用 Scala，就很少能找到同伴来沟通。

如果你的身边有很多和你一样学习该语言的人，那么在遇到困难的时候能够及时地寻求帮助。这一点也很重要。

2015 年 4 月 15 日 JetBrains 发布了 CLion，一个跨平台的 C/C++ IDE。还捎带着统计了 C/C++程序员及其他程序员的数量。以下数据是全球范围内的学习相关语言的程序员的数量统计。

- Java 有 900 万人。
- C#大约 780 万人。
- JavaScript 大约 760 万人。
- PHP 将近 600 万人。

- C++大约 440 万人。
- Python 有 400 万人。
- Objective-C 大约 330 万人。
- C 大约 190 万人。
- Ruby 大约 180 万人。

5．气质

语言有性格，会与人的气质相互影响。有人就喜欢 C 不喜欢 Java，有人就喜欢 Java 讨厌 C#，有人就喜欢 C#憎恶 PHP……

为什么有时你对一种技术会有“相看两不厌，唯有敬亭山”的感觉？那是因为，你和这门技术气质相近，和创建者气质相近……

总结

回顾一下，其实选择编程语言时经常考虑下面几个因素：

- 应用范围，即行业和产品所需。
- 薪资多少。
- 同行多寡，和语言有关，有时也和地理属性有关。
- 热门程度。

这些都可以通过互联网信息检索、整理得出，所以，你可以自己决定选择哪一门语言。

领导不在，咱还干不干活

你不是为公司工作，不是为领导工作，也不是为薪水工作。

“世界上大多数人都在为薪水而工作，如果你能不为薪水而工作，你就超越了芸芸众生，也就迈出了成功的第一步。”

这是订阅号“程序视界”每周一书栏目推荐《**致加西亚的信**》时摘录的一句话，值得细细品味。

你是在为自己工作，工作是成就自己的手段，吊儿郎当荒废光阴，是对自己不负责任，所以，出于为自己负责的态度，领导在不在都应该一个样，该干嘛干嘛，一时没工作任务，就自我安排，要么学习提高，要么预测接下来的工作内容、储备相关技能与知识。

请谨记你是为自己而工作，谁都无法阻止你为将来所做的努力，谁都无法剥夺你因此而得到的回报。

为了能最大限度榨干自己的潜力获得更好的成就，我们需要告别没人监督就不干活、没人分配任务就无所事事的状态。

“凡事预则立，不预则废”，古人老早说过了。要告别没鞭子抽着就不干活的状态，就需要对自己有所规划。

规划可以分两方面：

一是工作任务的规划，比如今天做什么，这周做什么，下周做什么，把自己当作领导来俯瞰整个产品或项目，基于已有的任务和整体的目标为自己画一个路线图出来。这样即便没人在旁边监督，也知道自己该干嘛。

二是个人成长的规划，比如我这段时间要学习哪些与工作内容相关的知识和技能，要提升哪方面的软实力（沟通、管理、计划、写作、信息搜集、文案、运营、英语、洞察力、归纳总结、演讲、组织、领导等），有了这种计划，即使领导不在，又没有工作可做时，也不至于混日子。

无论哪种规划和计划，前提是个人得有成长和成就意愿，愿意通过目标、自律、毅力来博一个美好未来，假如某人已决定把自己像行货那样交付给组织和领导，一点也不想对自己负责，总想被动等待发落，我只能说，这种不做选择的选择也是一种选择，而且，后果也还是要落到自己头上，到时再后悔便于事无补了。

假如觉得很难对自己做什么规划，有了计划也很难执行，可以看看程序视界的另一篇文章——**“你的计划为什么执行不下去？怎么破？”**。

请谨记，只有你自己能为自己负责，你的将来，由你现在的选择开始。

什么样的程序员适合去创业公司

2015 年 11 月 25 日早上，我宣布解散团队，结束了我的创业之旅。

吃过散伙饭，喝了二两西凤酒，回到办公室，再一次坐下来，我开始想一个问题：什

么样的程序员适合去创业？

在如今创业公司纷纷倒闭的寒冬里讨论这个问题似乎多少有点不合时宜，然而正因为在这个倒闭潮里无数的程序员需要重新调整心态，再次出发寻找自己的位置，这个问题却恰恰显得重要。回答了它，我们就可以避免在将来的某个时候做出不正确的选择。

从创业公司和程序员两个方面来看，有助于我们理清问题。

创业公司都是什么鬼

我打算从两方面来讲，一是创业公司的分类，二是创业公司的风险。

1．创业公司的分类

如果用万能的二分法，那么这世上的创业公司分为两类：

- 认真打磨产品做事的。
- 讲故事忽悠投资人或用户钱的。

有一些缩写，比如 B2B、B2C、B2B2C、O2O、C2C，讲企业或平台的商业（运营）模式。就是这种说出来别人不太明白、说的人却觉得很牛的、若干年后可能听起来会觉得很傻的缩写词，现在有了新的演绎，叫作 2VC、2pre-A、2 天使。就是用来讽刺那些只想讲故事忽悠钱的创业公司（团队）的。

当然，很少有哪个创业公司会说自己是 2VC 的。就算是真的，打死也不能承认啊。要知道在 2013 年、2014 年，很多人急吼吼地拿着钱要投创业团队，一张嘴就能拿到几百万元的投资，所以产生 2VC、2preA 的团队（公司）也很正常。

不管怎么说，这样的团队不少，花的钱不是自己的，烧起来就不知道心疼，又因为目的不纯正，没有抱持做事业的心态和情怀，所以，对技术、对产品、对研发团队、对程序员，都不怎么用心，也没什么追求，就急着捞一把变现。

这就是第二种。

但第一种是值得敬佩的。我始终认为，产品是王道，能解决用户问题的是王道，向他们致敬。

假如你是程序员，你要选择创业公司，我推荐第一种。不过还是等你看完我这篇文章再来定是否要选择一个创业公司。

另外一种常见的二分法是基于钱来的：

- 不缺钱的创业公司。
- 缺钱的创业公司。

有一些创业公司阵容特别豪华，堪称 MVP 团队，轻易就能拿到投资，比如创始人是来自阿里系、腾讯系、网易系、金山系、小米系、华为系等，那这样的团队多数都很慷慨，所谓和土豪做朋友，总是好的，如果这样的团队找你来谈谈，那么去打打酱油也是可以考虑的。

还有一些团队属于有点想法但没钱的，这时就要另论了，需要再回到前面那个二分法去。假如这样的团队是想做产品的，而且产品也蛮靠谱，那么可以考虑一下。假如他们是盲目创业跟风凑热闹，想 2VC 还没找到门道，那就别浪费你的大好青春了。

2．创业公司带给程序员的风险

有一部电视剧叫作《北京人在纽约》，由郑晓龙、冯小刚执导，姜文和王姬主演，真正的豪华阵容，男的既帅又有内涵，女的既漂亮又有气质，当时火得一塌糊涂。

在这部电视剧里，有一句话当年特别流行：如果你爱他，就送他去纽约，因为那里是天堂；如果你恨他，就送他去纽约，因为那里是地狱。

关于创业公司和程序员，我们可以套用一下：“如果你爱他，就送他去创业公司，因为那里是天堂；如果你恨他，就送他去创业公司，因为那里是地狱。”

是好是坏，因人而异。所以，下面这部分罗列了一些风险与提示，仅供大家参考。

- 创业公司成功概率小，仅 1%或者更低。
- 创业公司变现周期长，比如大家喜闻乐见的股票和期权，这种变现方式，只能等到公司上市或再融资。以上市为例，第一视频 2005 年成立，2006 年上市，那是火箭一般的速度啊；空中网也比较快，2002 年成立，2004 年纳斯达克上市，用了 2 年 2 个月；聚美优品 2010 年成立，2014 年上市，用了 4 年 2 个月……这都是快的，阿里巴巴用了十几年才上市，还有很多公司根本就没希望上市，大多数人到最后都只能感慨着“出师未捷身先死，长使英雄泪满襟”。
- 创业公司也不是人人都有股份和期权……你懂的，即便你选中了 1%的那些公司熬过了变现前的进行曲，也可能到时一切都与你无关……
- 创业公司工作作息不规律，这是非常常见的，比如各种加班，有段子说：结婚的加班到妻离子散，有女朋友的加班到单身，单身的加班到没有朋友……
- 个人定位不清晰，全栈，一个人顶 10 个人用，哪里缺人顶哪里，如果你缺乏适应

性，可能会风中凌乱或精神分裂。

还有一些，不再赘述，如果你身边有创业公司，可以自己观察总结一下。

3．创业公司能带给程序员什么

不能只说不靠谱之处和风险，还得说说创业公司能给程序员带来的好处。其实，就像股市，风险与收益共存。我总结了一下，大概有下列好处：

- 成长很快，这一点不必多说，你要独挡 *N* 面，技术视野会更广阔，你要独自解决问题，技术修为可能会更精深。

公司快速成长带来了员工收入快速上升的机会，处在成长期的行业里，又在成长期的公司里，机会相对较多。

- 参与感、成就感，相对于稳定的大公司，你更容易深度参与产品的设计、开发、运营，你的汗与泪会浸润产品的每一个细节，想象一下用户在使用时发自内心的愉悦和赞叹，就充满了成就感。
- 完整经历产品及公司运作流程的机会。
- 股票、期权变现的可能，这是一夜暴富改变社会财富分配的途径之一，阿里上市诞生了多少千万富翁啊。

……

哪类程序员适合加入创业公司

从创业公司的这面说完了，该从程序员这厢看看了。

1．你想要什么

在决定去创业公司前，最应该想清楚的是自己要什么。通常一个程序员在面对创业公司抛出的橄榄枝时，无非考虑下面几种诉求：

- “钱景”如何。
- 技术成长性。
- 产品调性是否符合个人倾向。
- 能否成为核心人员共享公司未来的成长。

唯有你清楚自己求什么，才能决断一个初创公司是否适合你。这是最根本的，必须慎

重考虑，同时也只能自己决定。我在微信订阅号“程序视界”曾经发布过一篇题为“一招搞定多 Offer 选择问题”的文章，在选择创业公司时有一定的参考价值。

除了这种根本性的因素，还有一些非常现实的因素，从不同侧面与这些根本要素相辉映，会影响到一个程序员的选择。

2．生涯发展理论与创业公司

舒伯于 1953 年提出“生涯”的概念，后来发展出生涯发展理论，将人自我实现的过程分为五个阶段，在每个阶段都有其独特的职责和角色，以及不同的发展任务。详见下表：

阶　段	年　龄	发展任务
成长阶段	出生~14/15 岁	发展自我概念，发展对工作世界的正确态度，并了解工作的意义
探索阶段	15~24 岁	发展相关的技能使职业偏好逐渐具体化、特定化并实现职业偏好
建立阶段	25~44 岁	在适当的职业领域稳定下来，巩固地位，并力求晋升
维持阶段	45~64 岁	维持既有成就与地位，更新知识与技能，创新
退出阶段	65 岁以上	减少在工作上的投入，计划安排退休生活，退休

上表的五个阶段中，每个阶段又可以细分为更小的阶段。对程序员来讲，有两个阶段是要特别注意的。

- 探索阶段中的试行期，22~24 岁，个人初步确定自己的职业并试验其成为长期发展领域的可能性。
- 建立阶段中的安定期，31~44 岁，个体致力于工作上的稳固，大部分人处于最具创意时期，由于资深往往业绩优良。

（1）试行期

我们先说试行期，这往往是大学毕业没多久、寻找适合的职业的时期。这个阶段的主要任务，一个是选择适合自己的工作，一个是快速累积专业技能。

一个程序员在这个阶段，因为年轻，从时间方面看也有一些资本，同时又没有太大的家庭经济压力，快速试错也没什么大不了的，顶多浪费一点时间，所以，可以大胆尝试。在这个阶段，我觉得可以加入创业公司去见识一下。当然，前提还是要结合自己的需求，看公司能提供什么样的平台，你能获得什么样的成长。

（2）安定期

31~44 岁是安定期，这个时期里边又有一段时间是需要特别注意的危机期：稳定工作将近十年（35~40 岁）。在 35~40 岁这个时期，有一部分程序员会发现，向上没有发展空间，

晋升受挫，不可避免地会倦怠、迷惘，于是有想法的人往往静极思动，想要出来看看，这就是所谓的倦怠期。

处于倦怠期的程序员，经济上基本也没有太大压力了。再加上正是年富力强的时候，工作能力和业绩通常也不错，外界会有很多橄榄枝抛过来，自己也春心萌动，“干柴烈火一点就着”。所以，很多人此时也会跃跃欲试，愿意自己创业或加入到创业公司中，谋求将来的规模化收益。所以，我们会看到，很多创业者年龄就落在这个区间，很多创业公司的技术骨干也处在这个年龄区间。

（3）成家立业期

除了前面提到的两个时期，人的一生还有一个危机期，就是成家立业期，年龄段是25~30 岁。

从舒伯的生涯发展阶段理论看，25~30 岁，又是建立阶段中的修正期。这个时期，应该找到一份稳定的职业并逐渐稳定，为后来的安定期做准备。

修正期和成家立业期的重叠，给我们带来很大压力，相信这个年龄段的朋友们都深有体会：如果你还没有另一半、还没有结婚的打算，就会被父母以及七大姑八大姨还有远房小表妹催婚了。

处在成家立业期的程序员，应该需要为成家做准备了（我指一般人，如果你是非主流，请一笑而过），此时应该考虑工作稳定，加入创业公司要小心，越接近 30 岁越要小心！孔子说“三十而立”是非常有道理的。假如 30 岁了还吊儿郎当的，吃了上顿没下顿，那后面就很难“立”起来，也没哪个丈母娘愿意把美丽的姑娘送到你身边。

3．程序员的支持系统

结合我们前面说的创业公司的一些风险，当一个程序员面临创业公司的机会，要不要选择加入，除了上面说的生涯发展阶段可供参考外，个人本身的实际情况也很重要。

首先一点，要有一颗不安分的心，想要追寻梦想。假如你是想有个稳定工作拿个固定薪水过安稳日子的类型，就不用考虑创业公司了。假如你想三五年后实现“让天下没有难做的生意”的目标，同时自己也能身价倍增，收入翻上 *N* 番，那就可以试试了。

其次就是，要有拼劲，到创业公司就是受苦受累搏明天的，你要说每天看个报纸、喝杯茶还能到纳斯达克敲钟，打死我也不信。

再次就是，经济压力的考量。不管哪个年龄阶段，假如你的收入不稳定，你或者你的家庭生活就无法维持下去，那最好还是不要冒险。假如你一人吃饱全家不饿，或者你另一半

有稳定收入能维持生活，又愿意成全你搏个出位，那就算是没有后顾之忧了，放手去干吧！

接下来，就是要有承担风险的预期，你加入的公司有可能一年半载后会关门大吉，工资也拿不到。还有可能会降薪加入一个创业公司，到时公司倒了、期权黄了，你还浮亏多少万元，是否能承受得住？

最后还要有足够的灵活性和适应性。根据前面对创业公司的分析，创业公司往往是瞬息万变、一人多用，你的角色不太可能固定下来，要有到处搬砖的准备，也要有到处搬砖的能力。假如你是那种认为自己是开发工程师，打死也不做运维的活儿的程序员，那还是要慎重考虑一下。

说了不少，不知道对你是否有帮助。最后要提醒的是，创业有风险，创业公司也有风险，选择需谨慎！如果你看到这里，认真地考虑创业机会和创业公司，请参看“程序员参与创业的 *N* 种姿势”。

程序员参与创业的 *N* 种姿势

作为程序员，大多数人在朝九晚五地工作，按部就班地生活。也许今天可以预见到明天干什么，也许很清楚短期内的日程，也许闭上眼睛就能看到下个月自己在忙什么，也许在噼啪复噼啪的击键声中隐约可见未来的那个自己。

然而，世界那么大，我想去看看！

阿里上市了，这世界上一下子多了上万名千万富翁。

暴风影音上市了，连续 30 多个涨停，股价从 7 元涨到 300 多元，不知道多少人因此暴富。

360 私有化要回 A 股……

巨人从纳斯达克退市要回 A 股……

……

在 2014 年 9 月的夏季达沃斯论坛上，李克强总理发出“大众创业、万众创新”的号召，有眼光有魄力、有资源、有积累的草根们也都纷纷参与到创业大军中来了。然后，中关村“车库咖啡”开始驱赶屌丝了……

外面的世界很精彩，荧屏前的自己很无奈。是“上班下班，吃饭睡觉，找女人，谈恋爱，结婚生子，然后死掉”这样平平凡凡过一生，还是“王侯将相，宁有种乎，彼可取而代之”那样拿青春赌一把明天？

一成不变的生活难免令人生出倦怠，摇曳多姿的创业江湖整日播放的传奇之歌蛊惑人心。我心荡漾兮，拔剑四顾兮，欲登太行兮，扶摇（服药）直上兮……

其实，只要你跨出一步，就可能乘风破浪蹈舞沧海。现在，就让我们看看程序员有哪些途径可以参与到火热的创业大潮中，给自己一个从 0 到 1 的突破，让梦想在将来的某一天璀璨绽放。

自己创建公司

这是最直接的。也可能是最难的。还可能是死得最快的，一个鲤鱼打挺，啪！摔砧板上被剁了（参看“创业失败那天我在做什么”）。

不过，作为拥有技术的非正常人类，这也是一条路。

虽然程序员每天和技术打交道，生活相对规律，有一定局限性，视野不够开阔。但是，情到深处人孤独，深夜静思也可能忽然豁然开朗，在某个技术领域辟出一条蹊径。

没错，擅长技术的程序员可以在技术领域创业，比如做一个技术平台，类似环信、融云、云视链、Ping++、Teambition、墨刀等，都有可能。让你的技术在大家都需要的地方生根发芽，别人跑去掘金，我们路边卖水，这也不失为一种极好的策略。

假如你的技术还没那么牛，不足以通过技术创新或卡位来创业，那么还有一种方式：做项目。这其实是很多程序员创业时的常用策略。找几个人，一起接项目，定制开发或者外包，再有点追求的，可能在做项目的同时培育自己的产品。

加入创业公司

近年来，创业公司如雨后春笋般地往外冒，你去参加一个创投会，什么样的公司都能见着。你下班走在路上，从你身边匆匆而过的人可能就是某个创业公司的 CEO、CTO、CIO 及各种 CXO……

创业公司离我们很近，你想加入，只要睁大眼睛去看就能找到。虽然创业九死一生，但认为自己身怀绝技、技高一筹的人还是多如牛毛，所以创业公司此起彼伏，虽然很多创业者已经失败并开始找工作，但还有更多的人正抛弃工作准备开始创业。

所以，你永远不缺乏机会，只要你有一颗不安分的心，怀揣梦想，希望有朝一日忽然绽放。或许多年以后回想，你会记得，人生转折，就在某个下午，那个夕阳西下，心在天

涯的时刻，你忽然遇见了那只能下金蛋的鸡。

加入创业公司又分几种情况：

- 成为技术合伙人，拥有股份。
- 成为核心员工，拥有股份。
- 成为初期员工，分享可能到来的快速成长，将来获取股份。

不管哪种，都需要你有一双慧眼，能够看出创业公司的创始人是否靠谱、他们的产品是否靠谱。这，是一场赌局，你付出的可能是时间和精力，但，赢的可能是将来。

技术投资

现在，还有一种方式来发挥你的技术优势：技术 VC。

所谓技术 VC，是指把技术当作资本，投资给初创公司，获取股份，靠股权变现获得规模收入的一种投资方式。

比如张三丰本人技术很牛，也有一个靠谱的技术团队“武当七侠”，那他完全可以成立一个公司或者工作室，将技术作为资本，和有技术短板或缺少研发团队的初创公司合作，张三丰的技术团队负责实现初期产品，初创公司分享一部分股权给张三丰及他的团队。这样，初创公司既可以快速完成产品开发上市推广，又可以节约成本，而张三丰也可以选择靠谱的产品，承担一定风险，通过股权变现获取规模收入。

股权众筹

2009 年众筹在国外兴起；2011 年众筹开始进入中国；2013 年国内正式诞生第一例股权众筹案例；2014 年国内出现第一个有担保的股权众筹项目；2014 年 5 月，证监会明确了对于众筹的监管，并出台监管意见稿。

目前众筹在国内还处于风口，股权众筹正逐步得到社会的认同。

来看看百度百科对股权众筹的定义：

股权众筹是指，公司出让一定比例的股份，面向普通投资者，投资者通过出资入股公司，获得未来收益。这种基于互联网渠道而进行融资的模式被称作股权众筹。

股权众筹的最核心体现是创造让屌丝赚钱的机会。程序员是屌丝群体的重要且有机的组成部分，自然也可以通过股权众筹，让自己从屌丝变成高富帅，有朝一日走上人生巅峰，迎娶白富美……

当然，故事也可能会是一个令人伤心的结局：获得风投的创业企业失败率可能超过80%，那很可能你没能在股权众筹平台遇见可以最终获得成功的凤毛麟角，结果辛苦钱打了水票，只好拿着一本《程序员修仙秘籍》，在寒冷的冬夜开唱——“谁来买我的火柴，谁把我的希望点燃”，然后，在艳阳高照的早晨，嘴角带着蒙娜丽莎的微笑穿越到没有悲伤的世界……

要相信，美好的事情终将发生在我们身上。现实其实比上面的结果美好很多，最差的结果是股权投资失败，你肯定不会把所有钱都拿出来做这个吧，自然也不会去搞杠杆吧……

还有一个比较好的结局：投资的企业发展顺利，那你就可以坐等企业被收购、下一轮融资或最终上市时兑现收益。当然，也许这需要很长的时间，还要有足够的耐性，并且每天祈祷“God bless me”。

持有创业公司股票

自己创建公司、进入创业公司、股权众筹等方式是有巨大风险的：他们都是找一颗种子种下去，赌秋天能看到花收到果；然而，种子可能根本就发不了芽；发了芽也可能遇上天灾人祸某一天夭折；即便历尽艰辛侥幸成长起来，也可能忽然来一阵子冰雹颗粒无收……

你是得有多坚强的小心脏才能承受这么多的不确定呢……

所以，如果你是怀有梦想的稳健投资人，也可以选择最后这种方式：持有已上市的创业公司的股票。

想跳槽？先看什么样的工作是好工作

过完年了，很多人想换工作，那到底什么样的工作是好工作？

每年春节过后，旧的一年真的结束了，年终奖该拿的拿了、该升职的升了、该调薪的调了，而没有拿、升、调的，或拿了、升了、调了还不满意的，就开始春心萌动准备跳槽了。所以这个时候，到底什么样的工作是好工作这个问题就又冒出来了。

到底什么样的工作是好工作？这是个问题，自从我在微信订阅号“程序视界”上开始发布程序员职业相关的文章后，就一直在和不同的人讨论，什么样的工作更适合自己。然而这个问题的答案从本质上讲，却可以一言以蔽之：

你喜欢的工作就是好工作。

那么问题来了：

到底什么是喜欢？怎样发现你喜欢的工作？如何开始做你喜欢的工作？

下面我将试图快速回答这三个问题。

“喜欢”究竟是什么意思

干工作并不像爱一个人那样简单到可以没有理由，一句“没办法，我就是喜欢她（他）啊”就可以横扫一切。

我们从事什么职业，往往是被很多因素左右、掣肘的，比如钱多、轻松、形象好、地位高、父母喜欢、另一半喜欢、一时找不到更合适的等。当我们在非我的、身外（外在）的因素裹挟下跑得久了，慢慢就习惯了、麻木了，自我发现的能力可能就慢慢丧失了。

所以，很多人对自己的工作，其实是已经无所谓喜欢与否了，你问他（她）“你喜欢自己的工作吗？”，答案往往是“就那么回事吧”“喜不喜欢都得干”“无所谓”“说不上来喜欢还是不喜欢”“不喜欢又能怎样”。

我们多数人都被困在原地，忘了自己。

但说到喜欢，其实还是有讲究的。我粗浅理解，喜欢的工作可分两种：

- 因为喜欢工作带来的附加价值而觉得喜欢某个工作。
- 因为喜欢工作内容、过程及其带来的成就感、自我实现感而觉得自己喜欢某个工作。

第一种，说白了就是喜欢与某种职业相关的附加价值（外部因素），诸如钱多、轻松、福利好、地位高、离家近等。至于其他的，工作内容、工作方式、归属感、价值感等，则不关心或被放置到次要地位。

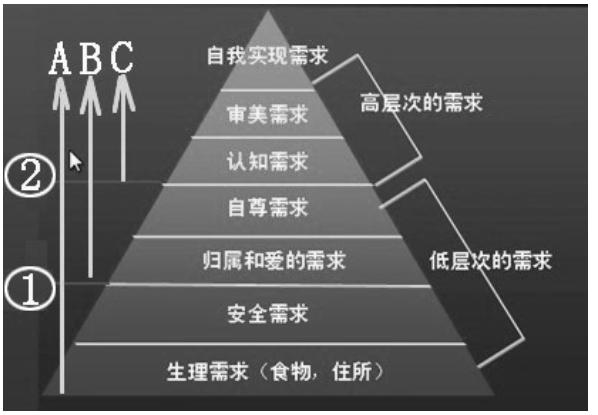
再简单点，一句话：钱多、轻松、社会地位高我就喜欢。

第二种，比第一种上升了一个层次，更关注职业与本身的契合度，是否合乎本心、能否发挥天赋、是否让自己感到有价值成为了主要关注的内容。

其实这两种喜欢，运用马斯洛需求层次理论可以很好地解释。

下图我标注了两条横线，三条竖线。

标号为 1 的是“温饱安全线”。这条线在“归属和爱的需要”之下，“安全需要”之上。



人人都有一条“温饱安全线”。在这条线下，个人急需满足的是生理需求和安全需求。当你的收入没有达到这条安全线时，你的生存就会受到威胁，此时你汲汲渴求的，就是怎样赚更多的钱让生活有所保障，此时谈什么喜欢不喜欢都是奢侈的话题。

此时大多数人，不怎么关心是否喜欢工作本身，关注点在于工作是否能带来财富、改善生活。

过了温饱线，人就慢慢开始关注出于本心的喜欢。然而此时还不强烈，因为还有情感需求、尊重需求未被满足，所以还得忙忙碌碌去追逐。古语说的“仓廪实而知礼节”就是这个意思。

等跨过了小康线（标号为 2 的线），回归自我的倾向就慢慢释放出来了，很多人在这个时候就会感知到内心的召唤，很想做点自己真正喜欢的事了，再考虑工作，工作内容、成就感、是否能发挥自己的天赋和才干，这些内在因素的重要性，开始超越薪酬福利，成为首先考虑的因素。也就是说，到了这个阶段，第二种喜欢就来了。

温饱线开始追求喜欢的工作，对应到标号为 B 的竖线。小康线开始追求喜欢的工作，对应到标号为 C 的竖线。人在而立之年、不惑之年时工作态度往往会发生较大变化，就是

因为循着这样的职业价值观路线发展，低层次需求的满足需要这么长的时间。当然也有的人到老也过不了温饱或小康线，就谈不到追求什么喜欢不喜欢了。

然而，还有一些人，走的是标号为 A 的竖线所示的路线。所谓“不自由毋宁死”“穷且益坚，不坠青云之志”“不为五斗米折腰”“安能摧眉折腰事权贵”，说的都是这种。古往今来很多诗人、作家、艺术家就是这么干的，哪怕穷困潦倒也绝不放弃追求，也没法去刷个盘子挣口饭吃。我对这样的人始终葆有无比的敬意，正是因为他们的存在，这个荒原化的世界才多了些许诗意。

现在人觉醒得早了，很多尚未进入职场或刚入职场的小伙伴就开始考虑做自己真正喜欢的工作了，这完全是有可能的，只是有时需要顶得住压力，耐得住寂寞。

怎样发现适合自己的好工作

其实有了前面的分析，根据自己当下所处的阶段，就能知道适合自己的好工作应该具备哪种“喜欢”的条件。

第一种喜欢其实比较简单，因为较少掺杂内心追求，很容易就可以搞个公式衡量出来（“一招搞定多 Offer 选择问题”这篇文章就讲这个问题）。

第二种喜欢就相对复杂一些，换句话说，就是发现并确认自己喜欢的事。这需要多做一些工作，甚至对相当一部分人来讲，这可能是一个难题。我的微信订阅号“程序视界”里已经有好几篇文章讨论过相关的话题，比如：

- 如何快速定位自己热爱的工作。
- 做自己想做的工作。
- 一招搞定多 Offer 选择问题。

然而，即便讨论了千百遍，确认自己喜欢干什么对很多人来讲还是非常困难的。因为生活中的各种干扰因素，就像程序员收到的来自各个渠道、各种领导的需求一样，优先级很难排，都是最急、最重要的，难以取舍，长时间忙乱无头绪，自己就被淹没了。

其实发现适合自己的好工作的关键是发现自己的天赋。而如何发现你的天赋，“程序视界”里推荐过一本书——《发现你的天赋》专门讲这个，可以看看。我前面提到的文章，“做自己想做的工作”，里面也提供了一些方法可以参考。

如何开始做你喜欢的工作

一旦发现了自己喜欢的事情，就只剩下“取舍”“聚焦”“计划”“执行”这八字箴言了。

关于计划如何制定与执行，最重要的一点，就是找到切实可行的第一个小目标，迈出第一步。

我们不能开始做一件事，往往是因为：

- 把事情想得很困难、很复杂，害怕自己没准备好，常说“等我……就开始……”，结果很多时候成了说说而已，根本无法开始
- 确立了高远的目标，比如五年之内赚到 1000 万元，可还没开始做，就觉得做不到，最后也就成了想想而已

其实，天大的事，都是从小事开始做起的。你需要把期待实现的目标，简化成容易做成的小事，就可以立即开始，这就是目标分解。

比如你想从软件开发转行做心理咨询师，那么这是跨度较大的转变，可以采取从目标倒推的方法来简化出第一个可行的步骤。参考下表：

期待实现的目标	自由执业的心理咨询师，开办自己的心理咨询工作室
阶段目标 1	加入一家心理咨询机构，从助理开始积累经验
阶段目标 2	考取国家二级心理咨询师证书
阶段目标 3	参加心理咨询师培训
阶段目标 4	选择一家提供心理咨询师培训的机构

目标分解至阶段目标 4 的时候，就是很容易执行的事情了。你在网络上搜索一下，就可以找到很多培训机构。

如果你觉得选择培训机构这个任务还是有点大，还可以进一步分解为更小的任务：

期待实现的目标	选择一家培训机构，报名
任务一	确立选择标准（培训费用、师资、离家远近）
任务二	选择三家培训机构
任务三	到培训机构现场咨询具体情况，搜集与你标准相关的信息
任务四	根据选择标准和三家培训机构提供的信息，做出选择

现在是不是觉得容易做了呢？

需要注意的是，你要为你的每一个阶段目标、小任务都标注上时间，比如开始时间、截止时间、需要花费的时间，越详细越好，这样才有利于执行。

女程序员职业发展的特别之处

在“做自己想做的工作”公开课的互动环节，有位女生提了一个问题，大意是“女生是否适合做程序员”，当时我怎么回答的，已经忘得差不多了，大意是性别对是否适合做程序员没有直接影响。课后我又仔细琢磨了这个问题，联想到之前有多位女程序员给我的微信订阅号“程序视界”留言，询问女程序员的职业发展状况，这让我恍然发现，我之前居然一直忽略了女性的具体情况对软件开发的影响。因此，这次，我们来单独聊聊这方面的话题。

下面先谈一谈女程序员有哪些职业发展阶段，然后针对女程序员的特殊性，提供一些职业发展规划的典型策略。

先看看女程序员有何特殊性吧，从以下三方面来谈：

- 性别与性格。
- 女性生理特点。
- 家庭对女性的期望。

性别与性格

据研究，大约 2/3 的女性偏好情感，以人为中心，有同情心，更愿意给别人支持，看重自己和他人的利益，偏好以个人价值来做决定，对客观和逻辑不那么在意。而在男性中，大约 2/3 偏好思维，喜欢有逻辑意义的决定，通常愿意通过客观的分析来决策。

不同性别的个体在做决策的方式上，其偏好会有所差异。通常人们会觉得，男性偏思考，女性偏情感，甚至有人说，千万别和女人讲道理。这可能会导致因为性别而曲解个人的性格偏好。实际上也有很多的女性偏重思维，当我们判断一个个体的性格时，要尽量避免用群体特征来为其贴标签，而忽略个体独特性。

男女之间的普遍的性格差异，对从事软件开发工作有什么影响？

温伯格在《程序开发心理学》一书的第 8 章，讨论了性格因素对程序开发的影响。温伯格认为，相对于较为稳定的智力因素，性格特征对软件开发的影响更大，甚至远远超过人们通常的估计。

那么有哪些性格特征会对程序开发的成功与否产生影响呢？

温伯格的研究很有意思，我觉得他的发现和结论放在现在同样适用。

比如他说，如果一个程序员缺乏在高压力的环境中坚持一个多星期的能力，也许我们就可以肯定地说，他不是一块程序员的材料。

他还提到，由于程序开发工作的多样性，所以如果一个人不能适应快速的变化，就无法胜任程序开发的工作。网络流传的一个段子也可以作为佐证：杀死一个程序员不需要用枪，只要改三次需求就可以了。

我在公司内做主题为“程序员的个人品牌”的内部分享时，也提到了适应并拥抱变化对程序员的重要性。

在 MBTI 性格类型里，第四个维度是关于我们喜欢结构严谨的方式还是自由宽松的方式。这个维度从个人喜好的生活方式来考察性格，区分我们如何适应外部环境，它提出了判断（J）和知觉（P）两种性格偏好。具有知觉偏好的人喜欢多了解世界，灵活、即兴，喜欢更多选择，更容易适应需求、技术、计划等方面的变化；具有判断偏好的人喜欢组织和秩序，喜欢计划、条理，一切事情都早做安排，对临时的、突发的、打破计划安排的事情较为排斥。从现代软件开发的过程来看，适应性好的人更适合做专业程序员。再回到女性和男性的性别角度来看，则没有明确的结论说明男女在判断和知觉两个偏好上存在明显差异。

温伯格还提到了一些典型的性格特点，比如爱好整洁、谦逊还有幽默感。在提到幽默感时，温伯格说，如果某个傻瓜缺乏自嘲的能力，那么在乏味的程序开发过程中他肯定坚持不了多久。互联网上有很多黑程序员的段子和文章，多数出自程序员自己，充分说明了这一点。

我个人认为还有一些性格特征会影响软件开发，比如注重细节、内省、责任。举个简单的例子，有些程序员习惯代码写完、Build 通过就将包扔给测试人员去测，如果他具有内省和责任这两种性格特征，就会意识到他的行为会给测试人员带来多么大的麻烦和困扰，就会进行自我改善。

可能很多人会认为女性更注重细节，男性往往粗枝大叶。女性更爱整洁，男性比较邋遢。这也是性别对性格的影响，但并没有明确的研究和论据来说明这一点。而每个人的性格偏好，又往往是两个方向都有的，只是存在一端强、一端弱，如果环境和工作需要，还可以发展完善另一端的偏好来适应外部环境。

所以，我觉得从性别造成的性格差异这点来看，性别并不能决定一个人是否适合做软件开发工作。有些性格确实会对软件开发造成影响，但如果你真的要做，也可以通过对外

部环境的选择来找到更适合自己性格的行业和公司。比如你灵活性、适应性较差，那软件开发里也有一部分岗位需要更有计划、有组织、有秩序地执行，可以选择具有这种特点的细分岗位。

但是，性别确实也会对整个软件开发过程带来非常重要的影响，尤其是女性，她的生理特点和周期会对其职业生涯带来非常大的影响，下面就来看看。

女性生理特点对软件开发的影响

毫无疑问，女性生理结构和男性有较大差别，比如女人有生理期、要生孩子、有明显的更年期等，这些生理上的差异，对职业生涯会有比较大的影响。

1．女性生理期

我们先谈女性生理期这个贯穿整个职业生涯的生理特点。

一般来说生理期就是指发育成熟的女性每个月都有一次月经，也就是月经期。一般的女孩，生理期会持续 3 到 7 天左右。

在生理期期间，多数女性会有比较典型的身体和心理反应。

身体反应

女性来月经的时候，会消耗很多体能，容易感到劳累，需要注意不要用眼、用脑过度，还要保证充足的睡眠。有些女性会怕冷，有些女性会有头晕、易疲劳、嗜睡等表现，有的女性常有小腹胀痛、腰酸、乳房胀痛、轻微腹泻等现象。凡此种种，不一而足。

假如在这个时候，你负责的产品要上线。上线，意味着需要经常熬夜，尤其是已经发布给用户使用的网站，基本上更新都是在凌晨。此时熬夜会让女性身体不适更为明显，有时甚至难以承受。而身边的男性程序员多数不会注意到这个，男性领导多数也不会注意这个（或者有的注意到了也只能假装看不见），所以，这些都得自己来承受。

心理反应

月经期间，由于体内雌激素与孕激素水平的突然下降，通常会引起人体的一系列不适反应。这点在前面刚说过。如果身体上的不适又遇上开发有状况（加班、调试鬼 Bug、上线等），则会引起心理反应。通常在生理期内，多数女性更容易情绪低落和波动，诸如烦躁、易怒、激动、抑郁、焦虑等反应会比较常见。有些女性甚至在经期前几天就会有明显的情

绪异常，多见于年龄稍长的妇女，如 30 多岁的已育妇女。

当情绪低落、频繁波动时，工作效率就会明显下降。调代码也没心思了，烦烦烦；解 Bug 也捋不出头绪了，烦烦烦；与产品经理或 UI 沟通也言语不到位了，烦烦烦……诸如此类各种状况，在月经前、月经期都会经常遇到。

现代职场男女平等，很少有男性领导和同事会为此而特别照顾女性，而女性此时又需要关注、体谅和照顾，这就形成了反差：“我已经很烦了你还不消停，需求一天改三遍，Bug 一天提几十个，作死啊！”此时女性需要保持稳定的情绪和平和的心态，很难……

女性生理期对软件开发的影响是显而易见的，但又没什么好的解决办法。

2．女程序员职业发展中的几个典型阶段

接下来我们结合舒伯的生涯发展阶段论来看看女性生理特点对其从事软件开发工作的影响。注意，我在谈及这些影响时，是针对女性群体的普遍特征而论的，在谈完群体特征后再来补充个体的差异。

舒伯是世界职业规划发展史上里程碑式的人物。他提出的生涯发展阶段理论，将人的生命发展过程划分为成长、探索、确立、维持、衰退五大阶段，并且给出每个阶段的年龄段、职业发展课题。这个五阶段模型，被广泛应用，长盛不衰，对个人一生的发展与规划，具有极强的指导意义。

先看一张表，这张表是我整理的舒伯生涯发展阶段理论。因为我们讨论的女程序员的职业发展，略去了成长阶段（4~14 岁）。我特意标出了三个阶段，这三个阶段对女程序员的职业发展非常重要，是因为刚好和女性的几个特殊人生阶段重叠。

阶段		舒伯生涯发展阶段理论			
	阶段名称		年龄段（岁）	生涯时期	职业上的发展课题
2	探索阶段	暂定期	15~17	青年中期	·能力与才能的进一步成长 ·学习计划的选择 ·独立性发展 ·适合自己的专业、工作的选择 ·有关专业技能的发展
		转移期	18~21	青年后期	
		试行期	22~24		
3	建立阶段	1.修正期	25~30	成年前期	·逐渐稳定于一项工作 ·确立自己将来的保障 ·发现适当的晋升路线
		2.安定期	31~44	成年中期	
4	维持阶段	3.维持期	45~60(65)	成年后期	·整理成果，维持现有地位 ·为隐退做准备
5	衰退阶段	减速期	65~70	老年期	·逐步退隐的适应 ·闲暇时间的充实与个人兴趣活动技能的学习 ·尽可能维持自立的状态
		隐退期	71~		

我们来分别聊一聊。

修正期与成家立业期

标号为 1 的是舒伯从建立阶段细分出来的修正期，年龄段是 25~30 岁。25~30 岁，又是我们传统意义上的成家立业期。从群体意义上讲，成家立业期对女性的影响比男性大。

在 25~30 岁这个阶段，大部分的女性会谈对象，结婚，还有相当大一部分要生孩子。如果一个女孩到了这个阶段的后期，28~30 岁，还没有男朋友，面临的压力将会非常大。

在这样一种革命形势下，女性更容易不淡定，心理上会产生各种反应，情绪容易波动。

作为一个女程序员，在这样的心理状态下，怎么能一心一意码代码呢？此时此刻，女程序员和男程序员的差别会第一次以比较明显的形态凸显出来。

安定期与育儿期

舒伯给安定期划定的年龄段是 31~44 岁。会遇到另一个非常现实的问题：生儿育女。

女人的黄金生育年龄是 25~30 岁（不小心和修正期又重叠了），再划长一些，有人说黄金生育期是 10 年，那就是 25~34 岁，这和职业上的稳定期又重叠了。女人多磨难，不是白说的。

假定一个女人 28 岁怀孕，怀胎 10 月，29 岁，起码要养三年，等孩子进了幼儿园，精力牵扯和压力才会稍轻，此时 32 岁。就是说，女人因为生孩子，会有 4 年左右的时间，重心不可避免地放在孩子身上。加上备孕的时间，前后得用 5 年左右。

做软件开发，5 年是非常长的一段时间，假如在这样长的时期内，你都不能将精力集中到工作上，那你的成长和发展必然会大受影响。

现实地讲，一个 26 岁的女程序员要跳槽，就会面临下面三个问题：

- 有男朋友吗？
- 结婚了吗？
- 准备什么时候要孩子？

如果你结了婚还没孩子，那么可能很多单位会找理由拒绝你。因为你可能进了这家企业马上就会要孩子，要孩子就有 5 年时间不能全身心地投入快节奏的软件开发工作。

如果你还没男朋友或有男朋友但还没结婚，状况稍微好一些。但考虑较长远的主管还是会预料到不久后即将发生在你身上的事情，这是人生发展的阶段规律……

所以，现实地讲，企业更愿意接纳孩子已经 3 岁以上的女程序员。

这些看起来不公平的潜规则，是由女性的生理特点导致的。如果讲应对策略的话，应该这样：女程序员 25 岁左右就稳定在一家企业，等到结婚生子、孩子稍大后再考虑跳槽。

我有一个女同事，研究生毕业，任何复杂的代码都敢写，执行力和解决问题的能力都很强，但她始终待在毕业时就进入的那家公司。原因非常简单，她生了一对双胞胎，自己和老公两个人照顾两个孩子，工作必须稳定。孩子上幼儿园后，她和公司商量，申请了每天下午四点半下班——要接孩子下学。对她来讲，稳定是第一位的，所以即便从经济角度和未来发展角度来看有更好的机会，她目前也不会考虑。

维持期与更年期

舒伯生涯发展阶段理论图中标号为 3 的细分阶段是维持期，年龄段为 45~60（或 65）岁。维持期与众所周知的、女性特有的一个人生阶段重叠：更年期。

更年期是伴随着女性绝经出现的，一般在 45~55 岁期间。

进入更年期后，女性卵巢功能衰竭，雌性激素明显减少，可能引发体内诸多器官的退化，进而导致各种身体症状的出现。比如潮热、高血压、关节疼痛、骨质疏松、皮肤干燥、毛孔变大、皱纹增多、头痛、眩晕等。这些身体变化，也会带来心理变化，这个时期的女性比较容易烦躁、多疑、焦虑、抑郁，情绪不稳定。

还有一点必须要提，这个阶段的女性，其孩子要么面临中考，要么面临高考，这也会给女性带来比较大的影响——就国内情况来讲，备考不但是孩子的事，还是父母的事，尤其是妈妈的事。除了中高考，孩子们的另一个典型特征是处在青春期，青春期撞上更年期，各种情况都会出现，妈妈的反应首当其冲。

你看，有这么多状况，如果你还在做软件开发工作，工作效率降低是难以避免的，要完成舒伯提出的维持现有地位的职业课题也有一定难度。

更年期对工作和生活的影响已经被广泛关注，不少影视剧也将场景聚焦在这个时期，比如《更年期的幸福生活》、《更年期的战争》等电视剧，里面处于更年期的女性，在生活、工作上，都因更年期而带来各种纷乱、迷惘、难以琢磨的行为。

从常理看，处于更年期的女程序员更需要关心和理解，也更需要心态调试。

家庭对女性的期望

从文化传统来看，我们国家对女性在家庭和社会中的角色有更多的要求。比如一般认为女性应该将更多的时间和精力投入到家庭生活经营中，诸如相夫教子、上得厅堂下得厨房、照顾老人等。虽然现在时代已经进步了，但这种社会文化氛围仍在持续影响着女性的生活和工作。

假如一个女程序员，组建的家庭是双职工类型的。生孩子之前，你搞你的设计，我码我的代码，大家相安无事。可一旦生了孩子，平衡立马被打破。丈夫会更倾向于让妻子来管孩子，比如在孩子小的时候，晚上照顾孩子，给孩子端尿、喂奶，孩子大了接送幼儿园，生病了妈妈陪护，再大了陪孩子做作业、教育孩子……大部分的家庭，都有意无意地把这些事情放在女性身上。

作为女程序员，相对其他职业女性，多数从事的工作节奏快、压力大、技能更新频繁。到了我们前面提到的修正期和安定期，生理特点和家庭对女性的期望，会迫使女程序员不得不将更多的时间、精力放在养育孩子上，这样一来，分派给工作的精力就有限，非常容易有精力不足的表现。其结果往往是两头奔忙，疲惫至极。

女程序员职业发展策略

前面我从性别与性格、生理特点、家庭期望三个方面谈了女程序员可能会面临的一些职业发展方面的困扰，也提出了一些应对策略。最后再来总结一下，看看女程序员在职业发展上应该怎样应对常见的问题。

第一个时期——修正期，25~30 岁

女程序员在这个时期，应该要比男程序员更早稳定下来，让自己稳定在一家企业，为结婚、生子做准备，这样可以避免频繁跳槽面对的不公平待遇。

所以，女程序员要更早注意职业选择问题，选择向好、有发展潜力的行业内有一定规模的、处于成长期或成熟期的公司，这样的工作环境相对稳定，对女性马上要面临的结婚、生子、育儿等课题有非常大的好处。如果你在生养孩子期间工作不稳定，今天公司破产，明天被裁，那就很难受了。

有的女程序员也会在生孩子之前考虑转岗，比如在同一家公司内部，从开发转到需求分析、配置管理、发布管理、质量管理、流程管理、测试等岗位。也有的女程序员从不考虑结婚、生子等问题对自己的影响，她们认为那些事情不会对自己有太大影响。个体可以选择自己的将来，不一定要受普遍规律的约束。

2．第二个时期——安定期，31~44 岁

这个时期的早期，31~35 岁，多数女程序员都有了孩子，但孩子较小，需要妈妈投入

更多时间照顾，比较好的策略是保持工作稳定，把精力向家庭倾斜。

假如你在修正期没能稳定下来，此时就会面临比较大的压力。企业对程序员的要求是不分男女的，你新进入的公司，遇见的领导、同事多半不会因为你要生孩子、有孩子要养就不给你分派任务。而如果你已在当下的公司稳定工作了两年以上，从人情方面讲，领导就很容易考虑到女性的特点，予以适当的关注和照顾，曾经一起战斗过的同事们，也容易理解并照顾你的情况。

当孩子长大之后，比如上了幼儿园，就可以有更多精力投入到开发工作上来，此时便可再度扬帆起航。

注意，生儿育女并不总会影响职业发展。个体之间差异很大，在任何时候，个体都可以选择超越自我、超越一般规律。

这个阶段还应当考虑将来的职业走向。如果已经不再有雄心鹰击长空，那可以着手准备切换到较为轻松的岗位，比如测试、需求分析等。如果还想继续奋斗在一线，那就要经常梳理自己的技能，着意培养顶端优势，为维持期能够维持现有地位做准备。

3．第三个时期——维持期，45 岁以后

我还没遇到过处于更年期的女程序员，从逻辑上讲，此时应该接纳这个特殊阶段出现的身体不适，注意心态调试，避免身体不适带来更多的心理不适。如果身体、心理真的有比较大的反应，影响到工作，也别死拧着不接受，要接纳工作表现可能变差的事实，顺其自然，才能更好地度过。

Offer 那么多，怎样拒绝才好

2014 年 12 月，我离开稳定的工作，以技术合伙人的身份进入朋友创立的公司后，面临的第一个问题就是组建研发团队。在招募队友的过程中，发生了很多有意思的事情，现在回想起来，对那些面试时相谈甚欢而后来莫名被拒绝的经历印象相当深刻。

后来我和一位从事人力资源的朋友聊天，又一次谈到这个话题，从她那里又听到了一些和我经历过的类似的事例。我们都觉得，在拒绝 Offer 这件事上，其实相当大一部分程序员走入了误区。

那么，拒绝一个 Offer 时，什么样的姿势才是正确的？这个问题我们先放下，在本文后面会回答，现在把我和朋友经历的事情先列出来看看。

那些程序员这样拒绝 Offer

有好几个小故事，一个一个来看。

1．玩儿消失

我托朋友帮我介绍一位 Java Web 开发方面的工程师，朋友托他的朋友帮忙，他的朋友刚好有个朋友感兴趣，于是我们就约了时间聊天。

聊天进展很顺利，我和这位工程师有着相似的经历：都是半路转行做软件开发，都是自学编程，都对开发工作很有热情，都喜欢技术，也都是内敛羞涩（用这个词让我有种要被拆穿的恐慌，不过，虽然我有时话痨，可还是内敛羞涩的）的人。所以，我们对聊起的很多问题都有共鸣，包括自学过程中遇到的那些坎儿、对技术的看法、对程序员参与创业公司的认识……

我们当场就谈好了薪水，哥们儿表示没有问题，回去就提离职，一个星期就能加入我的团队。靠谱——当时我就是这么想的，而且根据我在面谈时的观察，这哥们儿虽然内敛羞涩，但应该是一个重视承诺的程序员。

过了两天，我联系他，他说离职申请已提交给项目经理，应该没有问题。我的心放下了，对未来充满了憧憬，但是过了几天，我忽然联系不上这哥们儿了。电话不接、短信不回、微信石沉大海、QQ 总是离线……我知道出了什么问题，可还是抱有一线希望，联系了我的朋友，朋友联系了他的朋友，他的朋友联系了这哥们儿，居然也联系不上了。

2．我离开了这座城市

还有一哥们儿，我们也是直接面谈（我在招人时放弃了笔试），谈得挺好。这哥们儿一看就是做技术的（别人也这么说过我），对技术和未来都还有不小的憧憬，对加入创业公司的风险和收益也有清晰的认识，我也给到了他期望的薪水。然后，这哥们儿说，回去提交辞呈，两周内到岗。

嘿，上一位玩儿消失的程序员带给我的伤痛像梅子黄时雨那样绵延了几个月，现在，

终于雨过天晴了！

然而，过了两天，我给这哥们儿打电话，他说已提离职，领导已批，正在老家休假，休假回来就到岗。

又过了两天，我又给这哥们儿打电话，他说还在休假，家里有些事情，等从老家回来就过来。

我应聘的经历虽然不太多，可也有十回八回。我招过的人虽然不太多，可也有二三十个。我意识到可能又会有意外变故。果不其然，到了约定时间再联系时，这哥们儿说他不打算在西安发展了……

3．薪水的谜

有个哥们儿到朋友所在公司应聘，谈得挺好，朋友给他开出的薪水是他上家公司的两倍还多，这哥们儿看起来挺高兴的，说过几天就报道，朋友给他发了 Offer，约定了报到时间、入职薪水等细节。

过了几天，要报到了，朋友打电话过去，这哥们儿说，“我对公司要做的产品方向不太感兴趣，还要再考虑考虑”。朋友说，那没关系，再考虑几天吧。

又过了几天，朋友打电话过去，这哥们儿说，“公司提供的薪水不太符合我的期望……”。朋友说，“根据你的工作经验和我们笔试、面试的结果，约定的薪水已超过了公司同等岗位同等资历的大部分员工了。”于是这哥们儿说再考虑一下。

后来呢，这哥们儿主动打电话说，“我又深入了解了一下，相信公司所在行业的前景不错，愿意接受 Offer，三天后，下个周一就可以入职。”

朋友延长了 Offer 期限，等着这哥们儿入职。到了那天，这哥们儿真来了，办了入职手续，加入了公司的平台组。然而，这并没有什么用。过了一个星期，这哥们儿找我朋友说，“我觉得不太能融入现在的开发团队，决定离开了……”

拒绝 Offer 的正确姿势

前面我们举了三个我经历过的程序员拒绝 Offer 的例子，我还遇到过很多，不能一一列举，相信你身边也有类似的情景剧不断上演。

经常有人在 CSDN、QQ 或微信上问我类似“怎样不伤情面地回绝一家公司”“怎样和

一家公司沟通来改变已经接受的薪水”等问题，我有时会提供一些思路，有时又不太能理解（我也是矛盾的，我又招人又帮应聘的人出主意，真担心哪天在社交网络上问我的那个程序员就是昨天和我面谈过的那位）……

我自己在应聘时也拒绝过多家公司，我负责招聘时被很多人以各种理由拒绝过，我也和做人力资源的朋友聊过别人怎么拒绝他们，现在，我觉得最好的方法是这样的：面对实际情况，想明白你想要的是什么，实话实说。

比如我们的第一个例子，其实不必玩儿消失，这哥们儿直接告诉我公司为了挽留他开出了更好的薪水即可。再比如第二位，你就告诉我家里觉得创业公司不靠谱就行了。还有第三位，直言我在考虑更高薪水的 Offer 即可。

有时我们担心实话实说会伤害那个青睐你的人，有时我们担心实话实说会失去备胎的爱慕，有时我们担心实话实话别人会看扁自己……

以我的经历来讲，你告诉我你觉得公司给的薪水低比让我抱有期望要好；告诉我公司太小觉得没前途要比骗我说过两周就到岗要好……我不会觉得受到伤害，反倒觉得这样两边都好，都不耽误事儿。

以我朋友的经历来讲，你直说在考虑更高薪水的 Offer 会更好一些，当然那位工程师担心那个更高薪水的 Offer 落实不下来，到时自己又丢了朋友给的那个机会。其实不必，你可以直言，告诉对方我需要半个月或十天时间来考虑、比较几个机会。都是过来人，可以理解。

我们都知道，强扭的瓜不甜，乱点的鸳鸯谱早晚会散，再完美的谎言也有圆不下去的那一天，那么，何不开诚布公地来谈谈呢？

别傻了，人家离职你也离

当你看到身边的同事离职，找到了更好（薪水高或职位高或做的产品好或技术热门或异性同事颜值高……）的工作时，你是否也感到彷徨、迷惘、艳羡、心动进而蠢蠢欲动？

别急，淡定！

适合别人的，不一定适合你。

职业价值观

每个人看重的东西不同，每个人对什么是好的都有自己的一本账，这种倾向体现在职业上，就是职业价值观。从不同的价值观出发，同一种职业在不同人眼里意义就不同了，公司对个人的意义也就不同了。

张三觉得没挑战要走，李四可能觉得驾轻就熟干得舒服。王二觉得钱多该留下，胡大却嫌时间都被公司工作占去了。赵虎觉得单位离家太远，通勤成本高不能承受，周龙觉得离家远才好，回家吃个饭就能睡觉，不用陪着老婆对眼熬。秦一觉得老板不重视自己，黄三却因为老板事事拉自己商量直想躲开……

从不同的职业价值观出发，一份工作就横看成岭侧成峰，燕瘦环肥千秋各异。所以，当我们羡慕别人有更好的机会时，当我们受人影响对当下工作更为不满时，先别盲目行动，问自己几个问题：

- 当时我为什么选择这里，最看重哪两个因素？
- 当初我看重的，现在是否对我不再重要？

如果初衷已不再重要，那现在的工作能否提供我现在看重的东西？能，留；不能，走。

每一份职业，都有你想要的，伴随着也有你不想要的。每一点都匹配个人需求，每一点都让我们满意的工作并不存在。所以，现实中选择职业、适应职业、转换工作时，往往是一种取舍和积极的妥协过程。

找到你当下心中的根，知道自己想要的，就不会随波逐流。下面是 9 种常见的职业价值观：

- 工资高，福利好
- 工作环境（物质方面）舒适
- 人际关系良好
- 工作稳定有保障
- 能提供较好的受教育机会
- 有较高的社会地位
- 工作不太紧张、外部压力少
- 能充分发挥自己的能力特长
- 社会需要与社会贡献大

从上面的 9 种价值观里，分别挑出对你来讲最重要的、次重要的、最不重要的、次不重要的。一旦你挑出来了，心里的秤就形成了，就不会人云亦云随大流了。

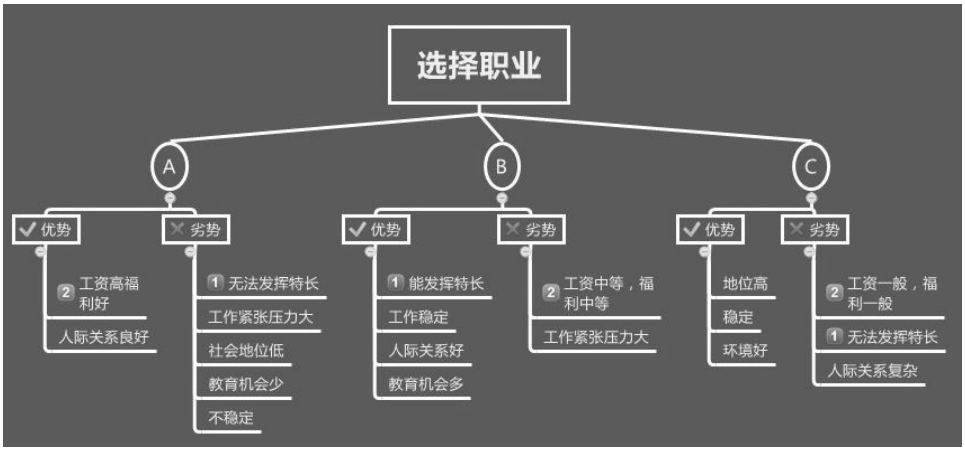
策略方案（取舍之法）

明晰了自己的职业价值观，再掌握一种取舍的方法，在遇到职业选择问题时，两相结合，就能让自己的决定更理性。

我在“一招搞定多 Offer 选择问题”一文中介绍了“生涯平衡单”这个工具，有非常好的参考价值。想了解的请移步过去。

这次我们再介绍另外一个工具：策略方案。

策略方案法的前提是你挑出了你认为最重要的两种价值观，然后分析你面临的几个方案，画出一张策略方案图来。类似下图：



在上图中，每一个方案有优势、劣势两个列表，优势在左，劣势在右。优势和劣势，从上到下的排列顺序，代表了某种价值观在该方案中的强度由强到弱。

一旦你画出这张图，选择就相对简单了。假如你认为最重要的和次重要的都在劣势列表中，那这个方案直接可以剪掉了。假如你认为最重要的和次重要的都在优势列表中，那对应的方案就是首选，就是最理想的。如果像上图那样，最重要的和次重要的刚好分布在优势和劣势列表中，那就优先琢磨分布在劣势中的那个因素失去之后你是否能忍受，如果不能，那就毙掉这个方案。

我们简单地说了职业好坏与人的匹配性，并引入了职业价值观来帮助我们找到定盘星，还介绍了生涯平衡单和策略方案两个工具帮助我们进行选择，希望能对面临职业选择的朋友有所帮助。

你的计划为什么执行不下去？怎么破

2015 年 12 月 17 日，我做了一次题为“做自己想做的工作”的公开课，有朋友提了个问题，大意是说他想学编程，可是每次有时间时，他宁愿背会儿单词，也懒得开始学编程。

有个朋友决心要减肥，每天吃很少，晚上甚至断食，坚持了两个星期就见效果了，瘦了将近 10 斤，我夸他有决心、有执行力，可又过了一周，再见到他，又反弹回来了。

我们经常会希望自己有所改变，能往更好的方向发展，达到自己希冀的某个目标，于是就会自己制定一些计划，踌躇满志地准备执行。可往往第二天早上起来就放弃了，或者执行十天、半个月就坚持不下去了，继而不了了之。

有没有想过，为什么你的计划执行不下去？怎样才能让一个计划顺利完成？

我也曾经定过很多计划，在 2014 年之前，大部分计划都中途夭折了，在 2014 年之后，一些计划执行下来了。

2015 年 12 月 21 日，我在公司内部做了一次分享，提到我的微信订阅号“程序视界”每周发布三篇文章，大部分都是我的原创。有同事当时问我是怎么做到的，分享结束之后我又思考了这个问题。

目标是不是你真正想要的

计划一定指向一个目标，或者先有目标，才有计划。那确认这个目标是不是你真正想要的，就是至关重要的。假如你对这个目标的渴望达到了类似“不自由毋宁死”的热切程度，那么你的计划十有八九能如愿执行完毕。

若某一个目标是你发自内心想要的，那就容易将对应的计划落实到行动上。如果这个目标对你来讲并不是不可或缺的，就很容易放弃。

我在做《你好哇，程序员》这本书时，萌生了学漫画的想法，觉得要是我既能写又能画，就可以把程序员的生活更形象地表达出来，更有吸引力，一时间心潮澎湃，不能自已，马上买了一套林晃的《漫画技法终极向导》、一套 wacom 的手绘板，还买了描图纸和专用的铅笔，准备大干一场，每天学习 20 分钟。结果只是在我女儿的带动下（她喜欢我买的书和描图纸）画了几天，这几个月根本没再动过……根本的原因就是，能画漫画并不是我真正想要的。

目标是否适合你

有时我们做一件事，仅仅是因为想要。比如有的人想要创业，就去创业了，最终却失败了。

想要某个东西，想实现某个目标，是非常好的事情。但还要看这个目标是否适合你，唯有你的目标适合你的性格、天赋、兴趣，你才会充满热情、自发投入、不计较投入与回报，才会更容易成功。

假如一个看见书就头疼的人，给自己定下一年看 50 本书的目标，那很难实现，这根本不适合他。

目标的有效性

在“设定目标的 SMART 原则”一节中，我会介绍什么是有效目标及怎样设定有效目标。

有效的目标要具体明确，要可以衡量，要有实现可能性，要有时间期限，还要和其他的目标有一定的相关性。

比如一个软件开发工程师设立的目标——要在 5 年内成为 XXX 公司 XXX 研发部门的经理，就是有效的目标，既有时间期限、又是其职业通道的一个自然发展方向、也可以度量。

而不能执行下去的计划往往是因为目标无效、不切实际。比如我 2015 年 5 月 9 号开通了微信订阅号“程序视界”，曾经发布了一篇 QQ 说说，定下了一个月内发展 10000 关注量的目标。那时我对怎么运营订阅号根本一无所知，只是觉得我写的文章很多人爱看，博客也有很多人关注，应该很快能有不少朋友跟过来关注。

结果呢，自然是被打脸了……

关键的第一步要具有可执行性

临渊羡鱼不如退而结网。当你的目标既是你想要的、又适合你、又有效时，就应该导出行动计划了。

目标可以高远，但计划一定要接地气，尤其是第一步怎么做至关重要。很多人空有美

好的愿景，最终无法实现，因为左思右想、举棋不定、迈不出第一步。第一步迈不出去，计划有 90%的概率会半途而废。

将大目标拆成小目标

我们想要达到的目标一般都相对较大，需要一段较长的时间并需要花费较多的精力才能达成。那些 10 分钟就能完事的，通常不需要做计划。

高远的目标一定要拆小，可以使用 XMind 之类的工具，运用自顶向下的方法来分解，分解的结果就是一连串方便执行的小目标。我个人的经验，一般能做到一周一个小目标就行了。也有的牛人每天都有目标，苟日新，日日新，又日新，隔上三天你就不认识他了。但是我想大部分人是做不到的，能做到每周完成一个小目标就不错了。

比如你定下每年要读 50 本书的目标，每周读一本书就可以了。每周读一本书是相对容易的，以我为例，早上六点起来，到七点半，这一个多小时可以用于读书；中午吃完饭，到下午上班前，有半个小时左右可以读书；晚上去掉陪孩子陪家人的时间，有一个小时左右的时间；早上开车送孩子上学，会在路边等十几分钟，一般也会看书；这样算下来，一天可以有两三个小时可以支配，工作日内就有 10~15 个小时，再加上周末，如果抽半天看书，一周可用的看书时间就有 20 个小时左右，看一本书不成问题。

每周能看一本书，那么一年 50 本自然就不是问题了。

我在“想跳槽？先看什么是好工作”一节中“如何开始做你喜欢的工作”中提供了一个转行做心理咨询师的例子来演示目标分解，也可以参考。

保持不断的正向激励

小孩子要多鼓励，比如他把糖果皮扔进垃圾桶了，你就夸他不乱扔垃圾；他拿拖把拖地，你就夸他通过努力把地拖得很干净；你和他约定每天看 20 分钟电视，如果他在时间到时自己关电视，你就夸他遵守约定。通过鼓励正向行为，可以让一个孩子感到自己的行为被认可，他就会继续强化这种行为，从而形成良好的习惯。

其实不止是小孩，成年人也是如此。我们在执行计划时，一方面要遵循上面讲的把大目标拆成小目标，另一方面，每一个小目标都要有可以度量的结果。就像我们做软件，当你把整个软件拆成一个一个小的、两三天就可以完成的子模块时，如果每两天的结果都能

看到、能集成、能测试，就会大大提高成功的概率；而如果你要干 30 天，你拆出来的结果是前 28 天都显示不出来有形的、可测的东西，那么一定越做越没谱。

每一个小目标都可以测量，当结果达到时，就会对你形成正向反馈，就会让你获得成就感，并促使你继续前行。这样不断的正向激励，会引导你走向最终目标。

2013 年年底我恢复博客更新，访问量不断上升，排名每天都会变化，有时一天上升 200 多名，有时几十名，这种直接的刺激也是我能坚持下来的原因之一。

目标需要不断调整，让它对你始终能产生吸引力，带来前进的动力。不然一旦某个目标实现之后，可能就会陷入无所事事的空虚之中。

现在我的博客排名进入 200 名以内，再想每天或隔几天就有排名上的提升已比较困难，我的目标也做了调整：坚持更新，期待关注访问量能突破 200 万人次。如果突破，我会再次调整目标。

杜绝自我怀疑及缺乏自信现象

我有一个朋友，对现有工作非常不满意，一心想换个环境去做自己喜欢的工作。为此他一度满怀热情地去发掘自己的兴趣，去了解职业规划，在相当长的一段时间内觉得自己很快就会有所突破。可是前几天再次见到他，他忽然告诉我，他对自己很失望，觉得理想的工作虽然存在，但不可能降临到他身上。

我们要相信，每个人从出生时就具有某种独特的天赋，这种天赋会指引你完成计划，实现目标。不同的是，有些人能很快发现自己的天赋和才干，能够充满自信地去执行计划，走向目标。而有些人则可能需要不断地寻找和实践才能发现自己的特点，找到适合自己做的事。对这些人来讲很常见的一种情况就是：因为自我怀疑或缺乏自信而放弃自己想要的目标，在计划开始执行之前就打退堂鼓。这些人特别需要经历完成某个计划而带来的成就体验，特别需要通过经验性成功事实带来的鼓励和肯定来增强自信。前面我们提到的目标拆分、迈出第一步、保持不断的正向激励，都有助于我们看到成功的希望，帮助我们建立自信。

运用可视化技术

许多成功的人在睡觉前花几分钟展望他们正在工作的项目，想象一个积极的成果展。

奥普拉·温弗瑞是世界顶级的高效人士，他使用可视化技术来勾画明天的成功，明确即将面对的挑战和障碍。所以，每天晚上花几分钟时间想象第二天的成功，想象目标实现时的那个不一样的自我形象。这会有助于激励你让它成为现实，因为你已经在你的脑海里看到它了。

如果你担心纷繁的工作会淹没你个人的计划和未竟的目标，可以将目标、行动计划、每一个阶段性小目标都写在便签纸上，贴在所有你能看到的地方，比如冰箱、床头、办公桌的隔档上。随时随地能看到，就能不断提醒、激励自己。

保持节奏

我们天生就处在一个充满节奏的世界里。一年有四季交替，一周有七天的轮转，太阳每天升起又落下，月亮每月都会从缺到盈再从盈到缺，这些自然和宇宙的节奏支持着我们的生活。我们春耕秋收；我们周一到周五工作，周六周日休息；我们在太阳升起时起床，在太阳落下后睡觉。我们的生活因为有了节奏而得以稳步展开，我们所做的事情都要符合一定的节奏才可能成功，我们的计划也是一样的。当你每周都能完成一个小目标，建立起这种节奏之后，连续几周，就会形成习惯，习惯会与你融为一体，带给你新的力量，让节奏更平稳地运行，让计划更有保障。

保持节奏还有一个特别的好处，它使得你专注于当下在做的事情，不会因为目标过于高远而焦虑不安。你跑马拉松，一上来就百米冲刺，总想着何时结束，总想着还有很远很久，反倒可能慢慢丧失信心，最终很可能到不了终点。你不紧不慢，调整呼吸和步伐，保持节奏，反倒容易坚持下来。节奏，让你知道并且习惯，凡事都有其该有的过程，去享受这个节奏，去体会这个过程，结果自然会到来。

抵制诱惑

人天生有放松和享乐的欲望，放纵多数时候都比“端着”更容易。

比如你计划每周读一本书，可每次回到家里，老婆在看电视，孩子在打游戏，这些就会松弛你的神经，增大你放弃计划的可能；比如你想减肥，可朋友拉着你去哈根达斯吃冰淇淋，你去还是不去？比如你想在五年内攒够 100 万元付房子的首付款，可每次经过商场看到漂亮衣服都会想买一件回家，根本控制不住自己。

当我们执行自己的计划时，总会面临各种各样的诱惑，“就这一次，下不为例”“别人都那样，我何必如此辛苦”“就算我这么坚持也不一定有结果”等想法很可能在一秒之内俘虏我们，我们一定要想办法抵制这种诱惑。

当我们面对各种各样的诱惑时，要牢记自己的目标，收心敛性，将力量导向目标，落到计划执行上。

有一些简单的办法可以帮助我们。一种是远离诱惑，比如你老想看电视，那就把广电的服务退了，把电视封起来；比如有朋友老拉你买冰淇淋，就别陪她去冷饮店或者干脆离她远一点。还有一种是想象放弃计划的后果，当你想象目标达成的画面和形象时，你就成了未来的自己，你觉得自己得到了某个东西，而一旦你想到放弃计划就会失去那个已经得到的，就会本能地想保住已有的成果。这是人怕损失的本性在起作用。以我为例，我现在买牛仔裤，32 码的，腰围刚好，长短刚好，从店里拿了就能穿，这让我感觉超好，一旦我要多吃，就想到买裤子时要拿 33、34 的甚至 35 的，就会失去那种刚刚好的完美感觉，就更加能坚持适当饮食。

摆脱别人的期望

通常我们并不是独自一个人生活，有家庭，有工作，有家人，有同事，还有朋友，我们周围的每个人都对我们抱有期望，他们会觉得你是这样或那样一个人，应该做这样的事，应该那样做事。一旦你的计划和目标与他们的期望不符，他们就会有形或无形地给你施加压力，他们的期望就会成为你的阻力。

比如你喜欢上了刻章，买了雕刻刀和一堆石头，准备在 3 个月内掌握阳刻、阴刻，能熟练使用篆文刻印，可你的家人居然不支持你，认为你买那些寿山石完全是花冤枉钱，还说学了刻章也不能当饭吃。你怎么办？

美籍华人伍绮诗有一部很火的小说，名字叫《无声告白》。它的封面上就印了这么一句话：我们终此一生，就是要摆脱他人的期待，找到真正的自己。

每个人都是独一无二的，每个人都应该成为他自己。如果你的计划和别人的期望不符，那就让别人失望去吧，没什么关系。如果他真的关心你爱你，就应该会很高兴看到你成为你自己。

找到你的社群

世界上没有人会跟你一模一样，也没有人会和你过一样的生活。但是，很多人会因为共同的兴趣、爱好、目标与你发生联系。

当你和与你有着相似的目标的一群人在一起时，就能在彼此的肯定中一起成长，群体中的他人也会指引你，让你看清自己的方向，你还能不断从别人身上看到希望、受到鼓舞，另外很重要的一点是你们还可能一起合作相互支持，让各自的创造力和激情迸发。

你热爱 iOS 开发，那就找 iOS 的小伙伴，比如 QQ 群、微信群、社区、网站、线下组织，尤其要加入有高手潜伏的社群，这样才能更好地成长。

前阵子我脱产参加了一个职业规划师的培训，持续 9 天，来自全国各地的老师聚在一起，共同学习、研究、实践，现场那种气氛给了我很大的触动。那 9 天是我参加工作十几年来第一次感到和一群人一起学习是一件美好的事，有很多的感触，也有很多的期待，甚至不希望结束的时刻到来。培训结束后，我意识到了同好社群的重要性。如果我想坚持做职业规划这件事，就要和同行的老师保持密切的联系，相互支持，共同前进。唯其如此，才能在一个正在培育的市场中经受住时间的考验，才能最终有所成就。

所以，如果你有一个目标，你制定了一个计划，如果能找到志同道合的社群，把你的计划发布到社群内，发布给在做类似事情的伙伴，彼此的肯定、指引、支持、鼓舞、合作，会让你的计划更容易成功。

如何快速定位自己热爱的工作

我前几天发了篇文章——“月薪 3 万的程序员都避开了哪些坑”，有很多朋友看了，反馈很多，我汇总了一下，有两方面的意见比较突出：

- 文中说的都是大道理，妇幼皆知，知易行难
- 不喜欢开发，怎么去找自己热爱的事业

感谢所有反馈的朋友，让我加深了对这些问题的认识，我决定再写几篇文章来谈相关的问题。这篇文章要谈的，是怎样判断你是否喜欢软件开发以及怎么去找自己热爱的事业。

庄子与惠子游于濠梁之上。

庄子曰：“儵鱼出游从容，是鱼之乐也。”

惠子曰：“子非鱼，安知鱼之乐？”

庄子曰：“子非我，安知我不知鱼之乐？”

惠子曰：“我非子，固不知之矣；子固非鱼也，子之不知鱼之乐，全矣。”

庄子曰：“请循其本。子曰‘汝安知鱼之乐’云者，既已知吾知之而问我，我知之濠上也。”

庄子和惠子的这段话很多人都有印象，其中最广为人知的，可能是“子非鱼，安知鱼之乐”，大概意思是不要总是以自己的眼光看待他人。

其实我的理解和庄子类似，就软件开发而言，我认为可以从精神状态对应的外在表现看出一个人是否喜欢软件开发。但从另一方面来讲，我又支持惠子，我觉得我很难去揣测别人怎么判断他是否喜欢自己的工作。因为每个人的经历、学识、家庭背景等都是不同的，这决定了每个人看待问题的角度不同，做事的方法也千差万别。我可以观察到一个人是否喜欢软件开发工作，但却不能臆测背后的原因，我能做的，只能是从我自己的经验出发来总结一些我认为可行的判断方法，抛砖引玉。

关注自己的感受

喜不喜欢是主观的，是感情、情绪方面的，是一种感受，要了解自己对一件事情的感受，可以通过各种问题问自己做这件事时的感觉，然后分析一下就能知道是否喜欢。下面是我总结的几个问题，通过问自己这些问题，可以判断自己是否真的喜欢软件开发工作。

- 写代码让你觉得时光飞逝如箭还是一秒犹如一万年？
- 看到代码是否有“似曾相识燕归来”的温暖？
- 隔一段时间不写代码，是否会充满怀念，有想打开 IDE 写点什么的冲动？
- 是否经常有这样的时刻：看着自己的代码，有种“相看两不厌，唯有敬亭山”的喜悦？
- 有没有一些时候，你看着自己的代码，会不自觉地想：这里或那里改改是不是更好一些？
- 当你看到令人眼前一亮的 APP 或网站或其他软件时，会不会想象“要是我来做该怎么做”？
- 你有没有想让别人阅读你代码的冲动？
- 你有没有读别人代码的冲动（想看到更好的代码）？
- 别人指出 Bug、错误或设计瑕疵，你会生气、拒绝还是接纳感激？

- 修复一个 Bug，你是为这个 Bug 被解决掉高兴多一些还是为你的代码（软件）更完美而高兴多一些？
- 听到新语言、新框架、新系统、开发者大会等相关的消息，你是很想了解还是懒得搭理？
- 有技术大咖在你身边出现时，想去结交还是懒得理他？
- 看见别人的烂代码，你是吐糟然后绕过还是想怎么改好？
- 看见别人的优秀代码，会不会羡慕，会不会想要是自己也能写出这么漂亮的代码就好了？
- 当你完成一个模块、功能、系统，解决一个问题时，是有成就感还是有“终于交差了”的感觉？
- 想到你开发的软件可以帮助别人解决问题带来好处你是否感到期待、兴奋？
- 你是否想建立属于自己的软件资源（比如工具、类库）？
- 你是不是像蜜蜂一样总是把看到的与软件相关的好东西收藏起来？

可能还有很多问题都能帮助我们发现自己是否喜欢编码，是否喜欢软件开发，不一一列出，欢迎感兴趣的朋友补充。

也许有人觉得“我的每一行代码都是诗，哪怕它们不能 Run”，自己特别喜欢，真是好！好喜欢啊！

我们的问题，是针对已经变身为程序员的兄弟姐妹们的。对于准备从事软件开发工作还没进场的朋友们并不适用。有些事必须得亲身体验了才能知道是否喜欢，理性的分析是没有用的。

怎么快速找到自己热爱的事业

前面我提供了一些问题，用来测验是否喜欢软件开发工作。假如你不喜欢，那么接着往下看，必有所获。

我在每周一书栏目里推荐过一本书，书名是《A Life At Work（这辈子，我最想做的事）》（订阅号内回复 10007 可查看）。这本书非常棒，从精神层面出发，系统地介绍了找到终身事业的理论和方法，推荐一看。

除了书，我这里提供一个非常简单快捷又相当有效的方法，也可以让你思索怎么找到自己热爱的事业。

1．从“不想要什么”开始

很多人会说，“我不知道自己想要什么”，这很正常，我自己也是不断实践、不断思考才慢慢知道自己想要什么、想干什么。

不知道自己想要什么，这没什么关系，这是很正常的事情。但假如你想知道自己想要什么，那么可以从一个问题开始：我最不能忍受的是什么？

当你越清楚自己“不想要什么”的时候，就越接近“想要什么”。

现在，找两张白纸，一支笔，两张纸顶端对齐放在书桌上，给左边的纸写上标题“不想要清单”，给右边的纸上写上标题“想要清单”。然后，我们先来完成“不想要清单”。

比如你是一个程序员，你的“不想要清单”可能是这样的：

- 我不想天天困在电脑前。
- 我不想老是加夜班上线新系统。
- 我不想周末和节假日接工作电话。
- 我不想老重复做一个东西。
- 我不想一成不变地老看着这么几个队友。
- 我不想月月拿死工资。
- 我不想“月光”。
- 我不想被女朋友说我“一辈子就这样了”。
- 我不想被人指使。
- 我不想有 Bug 时被批评。
- 我不想产品经理老改需求。
- 我不想老是延期交付，老是被老板说无能。
- 我不想和一帮没水平的家伙共事。
- 我不想在 Level 太 Low 的领导下干活。

……

也许还有其他的，嗯，必然还有其他的。自行脑补吧，每个人都不一样。你可以不看我上面列的单子，列你自己的。

2．如何得出想要什么

现在我们来完成“想要清单”，方法也很简单，对着“不想要清单”，一条一条来找下面的内容：

- 对立面是什么。
- 你之所以不想要的背后原因。
- 这一条对应的情感诉求是什么。

对应这些问题，一一找找看，然后大多数人就能完成“想要清单”。比如我们上面的不想要清单对应的想要清单可能是这样的：

- 我想要接触更多的人。
- 我想要作息规律的工作。
- 我想要不断挑战新东西，想要成就感。
- 我想要未来越来越有希望，比如职位不断晋升、收入规模上升之类。
- 我想要更高的收入。
- 我想要更被尊重。
- 我想要和牛人一起工作。
- 我想要和高水平的领导共事。

……

“想要清单”可能很长，一个不想要可能衍生出多个想要；也可能很短，因为多个“不想要的东西”可能会指向同一个“想要的东西”。没关系，只管做下去

一旦你分析出了“想要清单”，就可以进行下一步了。

3．成就事件

再找一张白纸来，写上“成就事件”作为标题，然后把纸放到“想要清单”右边。现在来回忆你的工作生涯中那些让你充满热情、真诚投入、感到愉悦的时刻吧。

每个人都会有一些因为有成就感而开心的时刻，比如你帮助客户解决问题后很开心，比如你给同事讲解了一门技术课程反馈不错你很开心，比如你卖出去了一部手机很开心，比如你写了篇文章很开心，比如你做了一道菜很高兴，比如你画了一幅画很自豪，比如你自己完成了一个 APP 很高兴……很多，用心回忆，一定可以找出来。

在寻找成就事件时，重点关注那些“即便没有物质回报也愿意投入去做的事”。这些事就是你感兴趣的，能给你带来成就感的，能让你有归属感的，很可能它们当中就有某件事会成为你终身热爱的事业。

一旦你列出了那些成就事件，就停下来，把即便没有物质回报也愿意投入去做的事特别标注出来。现在可以休息一下了，后面还有更耗神的事做。

4．寻找你想做的事

好，现在把你的成就事件清单和“想要清单”来比对，看哪个成就事情与你的想要清单中的事项能联系起来。

如果没有能联系起来的，好奇怪啊——那些曾让你感到有成就感的事、开心的事，居然和你“想要的什么”都没什么关系。如果是这种结果，那可能你分析得还不够，回头再分析一下不想要清单、想要清单、成就事件清单。

反复来，直到能建立联系。

如果你真的没办法建立联系，那么改天再来试吧。

你到了这里，说明已经找到了你想要的什么与让你开心的事之间的联系了。太好了，把你找到的那些事件、那些时刻单独挑出来，仔细分析，看看哪些事可以作为你的事业来做。假如你找到的不是一个可以当作事业（职业、工作）来做的事情，那就进一步分析这个时刻：

- 为什么那时我会喜悦？
- 这样的时刻是偶然还是必然的？
- 这样的时刻可能出现在哪些工作、职业、事业中？
- 这样的时刻怎样才能重复？

通过不断的发问、分析，你可以找到一个热爱的、乐意做的事情、工作。假如找不到，你的工作也不会白费——找不到可能是由于你没有使用正确的方法、不了解其他职业或自我意识能力稍差，而不是没有适合你的事业。这个时候，你有两种选择：

- 改天再重复这个过程，死磕。
- 找职业规划师聊聊。

我的建议是等一段时间再重复，如果重复几次都不能找到自己想做什么，那就找职业规划师聊聊。职业规划师有很多方法论和分析工具，可以帮助你梳理自己，找到你想做的事。

5．去做你想做的事

然而，无论你自己找到方向，还是职业规划师帮你找到方向，这都仅仅是第一步：定位。接下来的事情更为重要：假如你的方向与你现在的工作不符，你能不能真的放弃现在的工作，去做你想做的事。因为离开现在的工作，你很可能就会失去收入来源，生活质量会下降；并且你找到的事，真的去做了，可能也会遇到各种各样的不美好，最终发现不是你想要的。然而，只有亲身去体验了，你才能验证你的分析，才能真的找到你想要的。

然而，离开现有工作做你想做的事，其实是相当难的，需要勇气，也需要家人和朋友的支持，祝你早日迈出这一步。

一招搞定多 Offer 选择问题

12 月 9 号在论坛看到一位毕业生问了一个问题：

“现在两个 Offer 一个在南京，搞.NET，电气行业，税前收入一年加起来差不多 7.2 万元。一个是在上海，搞 Java，公式属于金融行业，薪资一年税前 9 万多元吧。大家可以积极给点意见吗？”

刚好我最近也经历了 Offer 选择的问题，谈谈吧。

从前面那位同学的提问中可以看到下列信息：

地域、薪资、行业、公司、职能、知识技能、年龄。

这里面还有至关重要的两点是有待补充的：个人倾向和家庭愿望。这个从帖子里看不出来。下面我们就从理论和方法上来讲讲如何进行 Offer 选择。

大学毕业生的特点

我们先看看大学毕业生的一些共同点：

- 第一次求职。
- 年龄在 21~25 岁。
- 对职业（行业*职能）不甚了解。
- 目标可能不清晰。

专业技能欠缺。

我毕业时就是这个状态，找工作很盲目，后来稀里糊涂就进行了电信行业。现在想，当时主要是考虑了行业因素。

大学生的这些特点会影响到求职，很可能会拿到不同行业的公司的 Offer，不同的 Offer 给出的岗位职能也很可能大相径庭。此时就会面临如何选择的问题。

我们以前面提问的那个同学为例来说明一下。

选择 Offer 要考虑哪些因素

我们从职业价值观和具体影响因素两方面来看。

1．职业价值观

从多个 Offer 中选择一个，首先要弄明白自己的职业价值观。所谓职业价值观，就是价值观在职业方面的体现。所谓价值观，简单点说，就是“你看重什么”“你觉得什么是好的”。

下面是 9 种常见的职业价值观：

- 工资高，福利好。
- 工作环境（物质方面）舒适。
- 人际关系良好。
- 工作稳定有保障。
- 能提供较好的受教育机会。
- 有较高的社会地位。
- 工作不太紧张、外部压力小。
- 能充分发挥自己的特长。
- 社会需要与社会贡献大。

从上面 9 种价值观里，分别挑出对你来讲最重要的、次重要的、最不重要的、次不重要的。一定要挑出来，挑不出来你就死磕，死磕还挑不出来就算了。

很多时候搞明白这个，答案就出来了。但也有一些人是比较纠结的，因为这个想要那个也想要。所以，还要再细化一下，看看更具体的考量因素。

2．影响 Offer 选择和将来发展的若干因素

根据前面那位同学的帖子及我们的分析，选择 Offer 有下列因素需要考虑：

- 地域
- 薪资
- 行业
- 公司
- 职能
- 知识技能

- 年龄阶段
- 个人倾向
- 家庭愿望

我们先看个人倾向和家庭愿望。

个人的性格、兴趣爱好、价值观、使命、需要等会影响个人求职时的倾向。外向的人和内向的人在选择职业时的倾向就不一样，事业心强的人和随遇而安的人也不一样。在这块有很多经典的理论和工具能帮助我们理清自己的倾向，比如性格类型（MBTI）和霍兰德职业兴趣，感兴趣的可以了解一下，也可以联系我畅聊职业规划（关注我的微信订阅号“程序视界”，回复 foruok 可加好友，或者到在行 APP 里搜索我的名字）。

父母会将自己的价值观加到孩子身上，这是不可避免的。大学生找工作尤其容易受父母的影响，一方面是因为我们前面提到的那些特点导致我们缺乏足够的信息支撑我们进行选择；另一方面是父母总觉得要为你负责，不能让你走错路，不能让你受苦，不能让你将来不幸福，他们有强烈的为你负责的意愿，想要帮你设定将来。

所以如果父母在南京，不让你去上海，那就会影响到你的 Offer 选择。父母是电力行业的，不愿你去互联网金融这种比较动荡的行业，也可能影响你的选择。家庭整体收入较低，就会提升薪资因素在职业选择中的占比。女朋友去了北京，估计你上海、南京都得放下，转而找北京的单位。

接下来我们看年龄阶段会对选择 Offer 有什么影响。按照舒伯的生涯发展阶段，大学学习（18~22 岁）、第一份职业（22~24）这两个时期，都处在探索阶段。探索阶段的职业发展课题有：

- 能力与才能的进一步成长。
- 学习计划的选择。
- 独立性发展。
- 适合自己的专业、工作的选择。
- 有关专业技能的发展。

所以我们看，当你面临多份 Offer 时，应该更多地考虑知识、专业技能等内在因素的发展，哪一份 Offer 与你的专业更贴近，哪一份 Offer 对积累知识、技能更有帮助，哪一份 Offer 对个人能力的锻炼与成长更有利，就选择哪一份。因为像薪水、职务这些外在的目标，是由内在目标决定的，即内决定外。所以，哪怕你一开始薪水福利较差、职务等级较低也没关系，适应、探索、成长三两年，知识、技能、能力等内在的、最具价值的方面积累够了，你可以很快在薪水和职务上得到提升，到时在企业内部和外部都会有很多机会。机会只青睐有准备的人，你武功已成，还怕什么。

当你考虑知识、专业技能时，自然和行业和公司联系起来了。你选择某个行业，会决定你能学到、用到什么样的知识，发挥发展什么样的技能。

以发帖那位同学为例，电力电气行业属于传统行业，进入之后，即便是做软件开发，也一定是会更多地积累电力电气方面的行业知识。要知道，软件开发只是一项基础技能，一项基础服务，要想解决现实问题，一定要和行业结合起来，甚至有些行业里的软件，80%是业务，20%才是软件开发技能。所以，如果我们从事应用软件开发工作，一定要明白这一点，软件是基础技能，业务和产品积累非常重要。

再说金融行业，进入后会积累金融方面的知识。金融是高大上行业，是财富行业，是现在和将来都会不断发展向上的行业。互联网+金融是很好的方向。金融行业会更累，但会更有钱。

最后我们说地域因素。地域会影响你积累知识、技能，进而影响将来的发展。比如你选择小城市，那么你如果想做金融行业，肯定空间很小；想做软件，机会也不多。很多产业会集中在某个地方，你要在某个产业中发展，最好到该产业最集中、最发达的地方。

生涯平衡单

前面说了那么多，最后介绍一个非常棒的工具来帮助选择 Offer：生涯平衡单。

先看一个示例表单，然后我们再解释如何使用。

姓名：任盈盈			方案 A			方案 B		
日期：2015/12/9								
序号	项目内容	权重(1-5)	得	失	加权和	得	失	加权和
1	薪水	5	5	0	25	4	-1	15
2	兴趣满足	5	3	-2	5	4	0	20
3	上下班距离	3	0	-5	-15	4	0	12
4	家人相处	3	0	-2	-6	3	-1	6
5	成就感	4	3	0	12	4	0	16
6	福利	3	4	0	12	2	0	6
7								
8								
9								
10								
合计			33					75

先看生涯平衡单的结果怎么解读：方案 A 得分 33，方案 B 得分 75，任盈盈应该选择方案 B。

结果是怎么来的呢？细细看。

项目内容

在项目内容这部分，列出你自己认为的任何重要的内容。比如任盈盈看重薪水、兴趣、成就感等。换个人可能看重的东西不一样，这需要结合我们前面讲的东西仔细捋捋才能列出来。

权重

不同的人对他看重的每一项东西，赋予的权重都不一样（取值范围 1~5），这也是我们前面让选择职业价值观的原因。以任盈盈为例，最看重薪水和兴趣满足，权重都是 5。这个权重，到时是用来和某个方案的得失之和相乘的。

方案

这里的方案，一般指某个职业，或者某个公司的某个岗位。

选择每个方案，都会有利弊得失。为了量化，设定得分范围是-5~5。

加权和是将得失分数加在一起乘以权重。比如任盈盈设定薪水的权重为 5，方案 A 的利为 5，弊为 0，那加权和就是 (5+0)×5 = 25。

合计

生涯平衡单会对每个方案计算总分，合计的分值是方案的加权和的总值。

这就是生涯平衡单的用法，希望它能帮助你选择合适的 Offer。

大龄程序员的未来在何方

大家都对大龄技术人员的未来非常关心，有的迷惘，有的坚定，不一而足。因此，专门来谈谈这个问题。

大龄程序员的界定

老早网上有人说，软件开发干不过 30 岁，后来又有人说干不过 35 岁，后来又有人说

干不过 40 岁，后来又有人说干不过 45 岁……各种说法很多，我的订阅号“程序视界”里有一篇题为“程序员的年龄怎么着了”的文章，感兴趣的可以看看。

这里我采取通俗的说法，认为过了 30 岁的程序员算是大龄程序员，只是为了讨论方便，不同意的请保留自己的意见。

人生的阶段发展理论

《论语 • 为政》篇：

子曰：“吾十有五而志于学，三十而立，四十而不惑，五十而知天命，六十而耳顺，七十而从心所欲，不逾矩。”

这是至圣先师孔老夫子对人一生的发展阶段的精辟概括，里面谈到三十而立。所谓三十而立，是指人在三十岁之后，就应该在稳定在某一个职业上，有所建树。

舒伯对人的生命发展过程，提出了以成长、探索、确立、维持、衰退为中心的 5 个阶段模型。每个阶段都有不同的职业课题需要完成，当前阶段的职业课题没有完成的话，就会影响后续的职业发展和生活。

在舒伯的生涯阶段里有个确立阶段，25 岁~44 岁。在这个阶段，职业上的发展课题有这些：

- 逐渐稳定于一项工作。
- 确立自己将来的保障。
- 发现适当的晋升路线。

确立阶段又可以细分为两个小阶段：

- 25~30 岁，修正期。
- 31~44 岁，安定期。

我们所说的大龄程序员，就是 30 岁往后，即过了孔老夫子所说的三十而立阶段的程序员，大龄程序员基本也处在舒伯老前辈提出的安定期内。你要说你超过 44 岁了，按孔老夫子所说，应该已经不惑了。

31~44 岁这个安定期，其实是人生最富有创造力的阶段，如果能稳定到一项工作上，一定可以有所成就，为将来打下保障。

30 岁这个年龄，是每个人都会遇到的，你不是一个人在困惑，程序员，世界上所有的人都会经历这个阶段。而且，已经有人经历过并提出了一些有效的应对措施。

还有一点很重要，25~32 岁是人的婚育高峰，所以，此时人会面临工作、生活的双重压力，事业家庭两头忙，身心俱疲。假如两头都不定，那就基本没法愉快地生活了。

技术人生的三个方面

本节会谈到对程序员来讲至关重要的三点：

- 知识、技术
- 技术能力和阅历
- 业务积累

前面咱说过，30 岁是所有人都会面临的人生转折节点，那么对于程序员来讲，这样的节点有什么特殊的表现和含义呢？

软件现在已经成为各行各业的基础服务，具有非常特别的属性：软件本身形成了一种产业，又和其他产业结合形成了交叉领域。

说白点，纯粹的代码没有意义，解决现实问题是软件存在的最大意义。而现实问题来自各行各业，所以，大部分的软件产品，是软件技术和行业需求的有机结合。

所以，多数程序员的工作方式是这样的：使用某种开发工具、通过编程语言来实现一个解决特定问题和需求的软件。

1．知识、技术

程序员需要掌握编程语言、应用框架、开发工具等这些具体的知识和技术，这是最基本的。

不同的语言和技术，都有特定的应用场景，这种语言解决对应问题效率高，那种语言解决其他问题效率高。

随着人类社会的不断发展，随着信息化和互联网化的不断深入深化，现实问题越来越多，越来越复杂，老的知识、技术在面对新的问题时可能力不从心或效率低下，所以不断有新的语言和技术问世，比如 Go、Scala、Swift、ROR，都才出现没多久，都是因为特定领域问题而出现的。没有最好的语言，只有最合适的语言。

面对这样的现状，程序员就需要保持学习，为了更好地解决问题，可能需要掌握多种语言和技术，而且会随着社会的发展和技术的发展不断地调整自己的知识、技术图谱。

所以，早在几千年前，我们的儒家经典《大学》就预料到了将来会有程序员这种人群，

对他们的生活做出了概括性的预言：“苟日新，日日新，又日新”。

这就是程序员面临的知识、技术现状，所以有人觉得过了 30 岁奔四张去的时候，家庭事业两头忙，身心疲惫精力不足没时间充电没时间学习不再适合做程序员了。

这是一种现状，当然它对某些人如此，对另外一些人则不然。其实，31~44 岁，正是人年富力强精力旺盛创造力爆棚学习能力焕发第二春的黄金阶段。不信你去看看舒伯的生涯彩虹图，一看便知。

举个例子，王江民就是这个时候（38 岁）转做软件开发，后来（45 岁）一骑独行白衣飘飘杀进中关村创造了江民杀毒的传奇历史，塑造了一代软件神话。

在 30 岁到 44 岁这个阶段，如果你还在做程序员，对自己的学习模式一定有所了解了，加上之前的积累，学起新东西来很快，应该会不断收获举一反三、触类旁通的愉快体验。

2．技术能力与阅历

对程序员来讲，知识、技术是一方面，是容易习得的，是较浅的层面。较深的层面，就是技术能力与阅历。

技术能力是指对具体的知识和技术的运用水平。它在很大程度上决定了一个程序员身上的技术价值。

技术能力是在不断地运用知识、技术解决现实问题的过程中培养出来的。在这个过程中，有的人爱琢磨、好总结、能升华、技术能力提升很快，一年可能收获一般人两三年的技术经验；有的人可能会停留在写段代码、写完了事这种层面，可能干三年不如人家干一年。

《天龙八部》中的神仙姐姐王语嫣，很多少年看了都很喜欢。从 IT 的角度看，她实际上就是一个掌握了很多知识、技术的字典型程序员，但不会实际开发。当然，她也可以是很妙的程序员鼓励师，既能极大激励程序员的干劲，又能在必要时提供字典式的帮助与指导。

与王语嫣对应的另一个人物是扫地僧，超越了知识、技术，阅历极深，不拘泥于招式，一抬手一投足就拍“死”了慕容博和萧远山这种超强高手。他是我们专业技术者的梦中梦。

扯了这么多，我想说的是，技术能力和阅历，随着程序员年龄和工作经验的增加，其重要性和价值将超越知识、技术本身。这是我们必须意识到的，也是老江湖的价值所在。

你可以在三两个月学会服务端开发，但如何应对大用户量、大业务量、大数据、大并发带来的挑战，绝对不是一个小白三两个月能搞定的，不积累 5~8 年，不随着企业的产品、

服务的发展而经历技术架构的变迁，你很难有能力去解决这些问题。

3．业务积累

大部分软件是技术和业务的结合，甚至有的行业软件，开发技术只占 20%，80%的都是业务层面的知识、流程。所以，对于一个程序员来讲，熟悉业务也是非常重要的。做电商网站和测绘软件绝对是天差地别。

业务和行业紧密相关，你选择一个行业，选择一家企业，就会决定你能积累的产业、业务知识、经验。而这部分业务积累，是程序员的重要价值所在，它和技术阅历一样是经得起时间考验的。

有句老话，早已告诉了我们业务积累的重要性：隔行如隔山。还有一句老话也说明了同样的道理：男怕入错行。

学一门编程语言容易，深刻理解业务却没那么简单。而你不理解业务，空有编程语言和应用技术框架，实际上没什么用。这就是我们强调业务积累的原因。

企业的分类

软件企业一般分类如下。

- 外包型
- 项目型
- 产品型

这里结合前面提到的“技术人生的三个方面”再啰嗦一下。

外包型公司，通过承接别的企业的部分或全部软件业务来发展。这导致了业务不稳定，编程语言、技术框架等技术方面也不稳定。

外包型公司做软件的心态，是“干活、交活、拿钱、完事”。在这样一种心态支配下，程序员较难有机会仔细打磨产品，对技术能力的积累会有一些影响。另一方面，因为业务随时会变，对行业知识积累也会有较大影响。

还有，从企业角度讲，如果考虑成本的话，更愿意用成本低的年轻人，所以，大龄程序员的将来，在这种公司不够乐观。一个 35 岁的程序员和 22 岁的程序员，做同样的 APP，

老板不会觉得你 35 岁就比 22 岁做出来的东西好多少，他会认为年轻人多加个班多改改就差不多了，这时候会产生劣币驱逐良币的效应。

项目型公司通过承接电信、银行、电力、政府或其他单位的软件项目为生，比外包型稍强一些，技术上可以自己选择，也能够在某个行业长期积累经验。如果这类公司的项目规模小而多、行业不集中，那就与外包型公司类似。另外，做项目的心态和外包有些类似，对成本的考量较多，对程序员的持续技术发展不利，比如有新技术出现，公司不一定会用，可能为了快和规避风险而选择较老较成熟的技术来完成项目，这一方面会减缓开发人员的技术更新周期，另一方面也减少了锻炼机会，降低了技术成长速度。

产品型公司是最好的，一个产品，要产生竞争力，要么靠对业务的深刻理解，要么靠技术上的领先优势。在这类公司工作对程序员的技术阅历、业务知识的提升都很有好处，我认为是程序员的较好选择。

大龄程序员的将来

通过前面的分析，我们知道程序员这种技术性职业，价值体现在三点：

- 语言、技术
- 技术能力与阅历
- 业务积累

用一句话来概括，程序员最大的价值就是运用技术解决问题的能力。而这种能力的构成里，技术能力与阅历、业务这两方面是具有经久价值的，是相比语言、技术更耐得住时间考验的。

程序员的将来，与价值维系有关，可以参看发布在我的微信订阅号“程序视界”的文章“程序员保值的四个秘密”。在我们进一步展开阐述之前，需要先看看四大职能取向。

1．常见的四种职能取向

一般来讲，有常见的四种职能取向：

- 管理者
- 专业技术者
- 自由职业者
- 创业者

管理者又分为职能管理者和全面管理者。CEO、总经理之类的角色属于全面管理者，开发经理、研发部门经理、项目经理、项目总监、HR 经理等属于职能管理者。

专业技术者指靠技术吃饭的人群，比如程序员，测试工程师，UI 设计师，会计，律师，编辑，摄影师……这些人的一个共同点就是拥有独特手艺和技术，能用自身所掌握的技术为别人解决问题创造价值，他们通常会依附一个组织来工作。

自由职业者首先是一个专业技术者，然后他脱离了特定组织，自己跑单帮了，自己安排自己，今天想干就今天干，今天不爽就明天再干。

创业者是特殊的一类职能取向，他们特别想拥有自己的产品和服务，于是就拉一帮人成立一个组织，协调各种资源来实现自己的梦想。

程序员的职能取向，多数会落在专业技术者上。我个人做了 6、7 年管理，现在还是回到了专业技术上来，因为做具体技术工作会让我更自然、更自在也更快乐。每一个程序员都应该理清自己的职能取向，只有职能取向清晰了，谈将来才比较靠谱。

2．所谓“成功”

有人说，三十出头的程序员，如果混得不够成功，面临的压力就会很大。话听起来没错，不过也不尽然，这里面有个关键的问题需要澄清，那就是“成功”的定义。

世俗的成功，古时候是“十年寒窗无人问，一朝成名天下知”，是“春风得意马蹄疾，一日看遍长安花”。现在也差不多，财富更多、社会地位更高……你看流布甚广的“当上 CEO，迎娶白富美，走上人生巅峰”之类的说法就是这种观念的反映。

但我这里说的成功，是指找到适合自己的职业，完成自我实现。简单点说，就是你在干你想干的事，既有成就感又快乐，就这么简单。

我们不必活在社会统一的价值取向里，也不必活在别人的期望里，你的人生是你自己的，哪怕父母非要你成为一个什么样的人，也不必去听。

有了这样的基本认识，就可以继续了。

3．程序员在企业中的发展

程序员的直接发展通路如下：初级工程师→中级→高级→架构师（专家）→技术总监→CTO。

技术总监和 CTO 带一些管理职能，也可以从另外一条路上来。这条路就是程序员的管理之路：工程师→项目组长→项目经理→项目总监→技术总监→CTO。

一条技术通道，一条管理通道。这是程序员常见的职业晋升通道。

除此之外，还有一些交叉发展的机会。比如开发转测试，开发转产品经理，开发转售前，开发转售后，开发转销售……

最后，还有一个选择，就是离开软件行业，到别的行业里自由自在地飞翔。

走管理路线的人十不其一，暂且不谈；那么下面就说说技术这条路的将来吧。

4．走技术路线的程序员怎么办

其实这原本不是一个问题。你说，哪行哪业没有老将？

而这之所以又成为一个问题，是因为程序员是“有知识、有技术、有理想、有焦虑”的四有新人，觉得自己已然学了那么多，付出了那么多，将来总不能停滞不前，走下坡路吧，总得蒸蒸日上吧，所以就焦虑这件事，担心、困惑，觉得这行当是青春饭，吃不了几年。

其实根据我前面的分析，结论已经呼之欲出了：保持学习能力，丰富技术能力与阅历，积累行业知识与经验，就可以持续走下去。即使有衰退阶段，那也是正常的，人生就如花一样，有含苞待放、有盛放、也有枯萎的过程，我们需要自然接纳。

5．程序员的二八定律

有结论只是第一步，我们还要讲清楚一个事实：程序员群体中的二八定律。

先说技术路线和管理路线的选择，基本上也是二八开（或者一九）。

小部分人走了管理路线，剩下的大部分程序员，走技术通道。那么有多少人能成为高级软件工程师、架构师、技术专家、CTO 呢？

20%。

那剩下的 80%，在技术路上不能走得更远，但还得维持自己的竞争力，保持价值不衰退，这样才可能继续做下去。怎么维持竞争力，前面已经说过了。

大量的大龄程序员将面临无法晋升和难以维持竞争力的问题，这是每个人都必须看到的事实。

从晋升角度看，任何一个行业的从业人员的分布，都是金字塔形的。大锅饭不存在，共产主义按需分配也还没到来，我们必须接受、接纳这个事实，然后才能在此之上谋求发展。

6．怎样面对将来

前面已经确认，技术路线可以走下去。现在我们来看，如果一个程序员的四大职能取

向定位到专业技术者，到 30 多岁时，该如何走，如何适应？

选择企业

就做技术来讲，如果想维持竞争力，在选择企业时就应当作一些考量，选择将来向好的行业，选择产品型、重视技术的公司，这种选择非常重要。它会严重影响你将来能在技术路线上走多远。

我前阵子找工作，就根据行业、企业、产品，只选了三家来面试。我也 35 岁了，需要稳定下来。

有人说可否选择创业公司，我的观点是，如果你能承受一定的风险（金钱和时间成本），创业公司的创始团队和产品又都比较靠谱，可以尝试。小米当初创立时，从摩托罗拉等公司找了好大一批开发工程师。

稳定于一项工作

根据舒伯的职业生涯阶段理论，31~44 岁是安定期，程序员应该稳定于一项工作，发现适当的晋升通道，确立自己将来的保障。

在这个时期，频繁地跳槽对将来的发展不利，这是毋庸置疑的，所以选择要慎重。尤其你到了 40 岁还没在哪个行业长时间待过，技术岗位也换来换去，再出去找工作，就会遇到比较大的困扰。

到招聘网站溜一圈，看看招聘信息中对年龄的要求，你就知道我所言非虚，除了技术总监、架构师、技术专家、高级软件开发工程师等岗位的年龄区间落在 30~45 岁，其他岗位，绝大部分都要求你 30 多以下。所以，如果你在不同行业、不同公司晃来晃去，到了三十四五岁还没稳定，也没能达到高级软件开发工程师应有的水平，再找工作肯定会遭遇比较尴尬的状况。

不能晋升怎么办

技术路线，可以初级、中级、高级、架构师（技术专家）……这么走下去，假如一个程序员发现自己到一定程度无法再走下去，就会产生挫折感或倦怠感。通常这种情况会在你稳定一项工作 7 年左右出现，所谓“七年之痒”。对本科毕业来讲，这个年龄一般在 33~35 岁。

此时怎么办？这是很现实的问题。

要具体问题具体分析：是你很牛，限于企业环境无法晋升？还是你自身能力到了天花板无法晋升？

不同的原因对应的行动是不一样的，若是前者可能应该考虑换一家公司。后者的话，

要调试心态，建立第二生活中心，在职业之外发展其他的兴趣爱好来平衡。

在这个年龄阶段，有一大批程序员会因为晋升受挫或倦怠而成为创业者或参与到创业中来。这是需要留意的现象。

竞争力有区域性和相对性

程序员的价值和竞争力，其实是相对的。比如你所在的公司，牛人太多，无法晋升了，其实别处可能有鸡头的位置等着，比如其他行业内的小公司的技术总监、部门经理，如果你看重职位和头衔，就可以去尝试。这也是一种典型的路线，有部分程序员在大公司镀镀金，回头到其他行业的相对规模较小的公司去做管理或技术专家。

如果你很在意职位等级，很在意比别人更受尊重，那就找一个能凸显你价值的环境。比如有一些国外的业余足球运动员或退役的足球选手到中国来，迅速成为热门选手受到重视，这都是一个道理。

所以，对于大龄程序员来讲，为了发挥价值和竞争力，还有下面的路径可选择：

- 到其他公司，不换行业，利用自己的技术优势，谋求管理职能，走管理通道。
- 到能凸显自己技术价值的公司，继续做开发。
- 到平均技术水平较差的地域，继续做开发，彰显自己的优势。

学习其他技能

当然，如果你有时间，也可以学习其他可以赚钱的技能，因为，虽然我们说大龄程序员可以一直做技术，但能一直做到退休的，估计只有 20%。大多数人都是那 80%，当你不得不离开心爱的开发岗位时，就必须做好准备而不至于忽然失重。

职业之外的兴趣

这一点是我们必须要谈的，程序员的人生里不只有软件和工作，生活是多元中心的，要有一些其他的兴趣爱好，能够滋养心灵，修复纯技术工作给心灵带来的磨损。

提笼架鸟、雕刻、书法、围棋、游戏、写作、读书……很多人（包括我）缺乏信仰，不能连兴趣爱好都缺失，这样生活才不至于枯燥乏味。

你值得不迷惘的职场

有几个朋友通过公众号“程序视界”的后台和 QQ 问了我一些问题，列出几个比较典

型的来看看。

我是工作一年多的 Java 程序猿，最近有些疑惑和烦恼，技术方向也不太明确，想和你聊聊。

我去年 7 月毕业，跳过一家公司了，现在在一家中小型公司做 4G 通信模块的维护，负责内置协议栈（TCP/IP），感觉很难学到东西，每天遇到的技术点都不同，很难深入学一项技术！是不是要考虑离职？

我今年刚刚毕业，在广州一家私企，已经做了一年了，老板对我还不错，不过现在感觉公司没有什么发展前景，未来也不知道怎么去发展。我主要学 Java。现在学了一些新的东西，Docker，ElasticSearch 这些。目前的感觉就是任何东西都是自己去摸索，学不了一套完整的体系。也不知道过完年后怎么办。

我学的电气工程专业，毕业一年了，我现在下决心转行做前端开发。一个人来北京学习前端。但是招聘都需要一年以上工作经验，而且一看转行都不看好。怎么破？

……

我选择的这些问题有几个共同之处：

- 提出问题的朋友都是刚入职场没多久的。
- 对将来的方向感到迷惘，不知道该怎么办。

迷惘是非常正常的，大多数人都在迷惘中忙着追求。

然而我们最好不要从青春到年老、从职场新人到退休老炮儿一直处于迷惘，所以，接下来我准备聊聊初入职场那些事，看看怎样在迷惘中寻找方向。

- 对于刚刚踏入职场的新人，什么是最重要的？
- 如何寻找方向？
- 怎样快速提升技术能力？

让我们从一个真实的故事开始吧。

2014 年 3 月份我们公司来了个聪明、勤勉、自律又敦厚的实习生 XJ，他之前的考研经历，让我和小伙伴们都衷心赞叹。

2012 年 XJ 本科毕业后进入东软，在 IA 事业部研发中心负责算法模块，每天朝九晚六从不加班。

在 XJ 负责机器学习智能算法部分时，每天被各种数学公式虐，深觉数学有用，可惜本科专业课没好好学，有些遗憾。另外大连距离甘肃太远，他想离家近点，有意离开大连。

看到一起住的两个小伙伴离职后，XJ 决定考研，找厉害的导师学数学。考虑到读书和工作地点与家乡的距离问题，他选择了西安电子科技大学又年轻又厉害的博导陈为胜老师。

XJ 决心考研时，2014 年 9 月份已经过去好多天了，距离考试只有不到三个月时间。而他还要靠现有的工作维持生活，必须继续上班，他能用的时间只有晚上八点以后和来回公交车上的 50 分钟。于是他制定了下面的学习计划：

1). 晚上八点到十点学习数学分析，十点到十二点学习高等代数。前一个半月仔细看书，后半个月巩固，最后一个月做题。

2). 来回上班路上用手机背英语单词，最后半个月背作文模板。

3). 早上到公司先看半个小时左右时政新闻熟悉政治，最后一个月背题。

对 XJ 来讲，有很多实际的困难：

首先，没有队友一起奋斗，需要自己坚持，非常考验毅力。

其次，习惯了一帮人每天一块吃饭、玩耍、聊天、打游戏。

再次，每天白天正常上班，累是一方面，晚上回去由动转静的过渡很煎熬，半天静不下心来，他的做法是每天吃完饭看个游戏视频然后看书。

还有，要照顾宿舍另外两个一起上班的小伙伴的感受，他们累了一天回来看电影聊天他得忍住不去聊，而且不能苛责和约束他们。

最终 XJ 完成了自己制定的学习计划，顺利通过了西安电子科技大学的分数线（305），成功进入复试并通过了。

XJ 也是职场新人，也遇到了问题，然而他通过寻找目标、制定计划、坚决执行，成功改变了现状。他的经历，能够回答我们前面那几个问题。

职场新人，什么最重要

从校园走入职场的新人，年龄在 22~24 岁。这个年龄段，按照舒伯的生涯发展阶段理论，属于探索阶段。探索阶段的职业发展课题有：

- 能力与才能的进一步成长。
- 学习计划的选择。
- 独立性发展。
- 适合自己的专业、工作的选择。
- 有关专业技能的发展。

这是前辈总结的，具有极强的参考意义。想想我们大多数人毕业找工作时都以薪水多少为导向，真是误入歧途了。谨记，对职场新人来讲，应该用最快的速度让自己更值钱，

而不是斤斤计较自己现在赚多少钱。

至于怎样才能让自己更值钱，参考前面的职业发展课题，可以用一句话总结：找到适合自己的职业，打磨专业技能，不断提升能力，尽快独挡一面。

XJ 在工作中发现算法对数学要求很高，他决定以后从事算法相关的工作，为此进一步选择读研来提升数学水平，这符合“让自己更值钱”的原则。

如何寻找方向

在“想跳槽？先看什么是好工作”一节中，我们讨论了什么是好工作。其实对大多数人讲，契合自己天赋和才干的职业，可能是最好的职业。所以，要找方向，就得从这一点入手。

荣格这句话——“小的时候，做什么事能让时间过得飞快并让你快乐，这个答案就是你在尘世的追求。”——很可能一语惊醒梦中人。

“如何快速定位自己热爱的工作”一节中也提供了一些切实可行的方法来帮助我们找到自己喜欢做的事情。

然而现实情况是，对有些人来讲，发现自己的天赋，找到自己喜欢的工作并不那么容易，很可能要反复分析、探寻、尝试才能拨开迷雾，整个过程会非常艰辛，艰辛到随时都想放弃。我只想说，别放弃希望，在路上也是一种修行，你曲折的航道会让将来豁然开朗的那一刻更加珍贵。

这里有三个问题：

- 你能干什么？
- 你想干什么？
- 你适合干什么？

假如你对自己的目标不清楚，就不断地问自己上面这三个问题。每周分析自己现在做的事情，问自己一次这三个问题，把答案记下来。持续去做，直到有所发现。

假如一直找不到自己想做的职业，那就：爱你所干的，围绕它发展自己的技能，让自己成为这方面的高手；假如有方向但出于种种原因不敢毅然决然地去追求，那就在现有工作之余先为你日后的方向“忙活”起来，储备相关知识和技能。具体可以：阅读相关书籍、看视频资料、参加培训、参与相关活动、参加相关社群。

对 XJ 来讲，他的方向，来自于对现有工作内容、感受的分析与总结。这是我们寻找方

向时最常用的一种方法。你有不满、有痛苦、有各种“不要”，反向也会推出你想要的。

其实不管你找了多少方向，最重要的是：去做。要相信你可以做到并开始行动，并且要坚持。仅流于想想、说说，是不可能有真正改变的，即便机会来了也不可能落到你头上。

最后，推荐几本和这个话题相关的书吧：

- 《发现你的天赋》。
- 《持续的幸福》。
- 《现在，发现你的优势》。
- 《现在，发挥你的优势》。

怎样快速提升技术能力

塞利格曼在《持续的幸福》一书中总结了成就公式：

成就=技能 × 努力

这里的努力指“花在目标任务上的时间”，而不是“早上八点到单位晚上十点回家中间都在打游戏”这种看起来很努力的努力。XJ 的努力正是这样：把时间尽可能多地花在你认为最重要的、与你的目标正相关的事情上。

这里的技能，则取决于你的职业方向，你做什么，就会有相关的技能。你做 Android 开发，Java 和 Android Framework 可能就是你的技能。你做移动端游戏开发，Unity 3D 或 Cocos 2d 可能就是你的技能。你做电子商务，J2EE、PHP、MySQL、Apache 等可能就是你的技能。

我们假设你已经选定了技术方向，然后来看怎么提升技术能力。据我个人的经验，大体有这么几点需要注意：

- 在实践中运用你的技能（没机会要发现或创造机会）。
- 做笔记，常回顾，常总结，发现不足，有针对性提高与完善。
- 制定进修路线，以 C++为例，语言本身基本学习路线可能是“C++基础语言、数据结构、常见算法→设计模式→STL→C++11→函数式编程与泛型算法”，结合工作需要，还有很多基于 C++的框架需要学习，比如 Qt、ACE、Boost、FFMPEG、OpenH264、WebRTC、CEF 等。
- 与优秀的人一起工作，远离混日子的那些家伙。
- 别人可以领你进门，但修行一定靠自己。

- 每一个技术方向都可能带来成就，但行动可能是艰苦、复杂的，也可能是长期的。要耐得住寂寞，沉下心去努力，今天看张三做前端拿钱多就转去前端，明天看李四做 iOS 开发赚钱多就转 iOS，往往几年下来反倒不如坚持做精一件事能让自己更值钱。
- 学习、学习、再学习，上班闲暇学，下班闲暇学，人家打游戏、看电影、压马路时你还学，努力 10000 小时之后你就有飞跃了，然而之后还要持续学习，终身学习。

确定性这剂“毒药”，你喝过没

假如你现在想换工作，你会怎么选择呢？薪水够多、工作够轻松、不加班、目标职位需要的技能你刚好具备、离家近、奖金多？满足这样要求的单位就可以去了，去了就会如你所愿一切都很美好？

假如你是个程序员，不想做软件开发了，想转行，眼下有以下几个选择：IT 培训讲师、NLP 教练、开咖啡馆、淘宝开店卖家具，你会选择哪个？

我去参加职业规划师的培训，同期的学员中有很多都是在单位里从事某种工作，HR、财会、猎头、开发、引导师，然而会有多少人立马出来专门从事职业规划师这个职业？我和几位老师聊过，因为所处地域的市场还在培育阶段及当下工作能够满足他们较为丰裕的生活等原因，他们暂时都还没有专职去做职业规划师的打算。

你想做某件事，却没有去做，最大的原因就是：未来并不确定。这是我个人的亲身经历，也是我从身边人身上观察到的。

你觉得现状不好，想要改变，最终却没有改变，很大的原因是：究竟能不能改变成功，短期内能不能看不到预期的结果，是个未知数。

所有这些，都是因为：我们中了一种叫作确定性的毒。

这种毒怎么解？听我慢慢道来。

确定性中毒的征兆

我的微信订阅号“程序视界”开了“有问有答”栏目后，通过订阅号、微信、邮件、

QQ，收到了不少朋友的留言，其中有一类就关乎选择问题。

有人问我在西安学 J2EE 好还是 Android 好；有人问我学 Android 好还是 iOS 好；还有人说他在大学里学习的是嵌入式，可是现在 Android 和 iOS 开发很火而嵌入式前景不知道怎么样，要不要坚持……

大多数时候我们喜欢“确定性”的事物，这样会让人感到有安全感，或者感觉踏实、舒服。对于不确定性，我们会经过一番利益得失的衡量之后选择回避或放弃。

我们已习惯在确认某种改变肯定会带来预期结果时才会开始行动，因此我们会找权威、找专家去咨询：这样做好吗？没问题吧？一定能成功吧？当别人点头或者告诉我们能成时，我们就欣欣然回去了，准备开始行动。然而有时从路上到家那段路程，我们甚至又会心生疑惑：这样做真的可以吗？真的没问题吗？

这就是确定性中毒的表现：常常因为不能确认改变是否能够百分之百成功而无法做出改变。我们只想要一个万无一失的完美结果。

真正的转变从不确定中来

让我们先来看看什么是真正的转变。

我推荐过克里希那穆提的《生命之书》在这本书里，十月二十八日，“真正的转变”这天，提到了两种改变：

- 从已知进入已知
- 从已知进入未知

来看看原文：

只有从已知进入未知才能带来真正的转变。请和我一起来探索这件事。从已知进入已知的改变一定会产生权威之见——你知道而我不知道，因此我崇拜你。我跟随你是因为你回答了我想知道的事，你交给我一种行为上的准则，以便制造出某种成果及成就。成就便是一种已知，我很清楚成就是什么，而这正是我想要的东西。因此我们总是从已知进入已知，其中必定有权威之见——制裁、意见领袖、上师、阶级次第、某人知道而他人不知道——这个已经知道的人能保证我修行有成，满足我的需求，让我得到快乐。这不就是我们想改变的动机吗？……

突变或转变是从已知进入未知，其中是没有权威的，而且不保证一定成功。……

虽然是断章取义，但确实有很多真正的转变是从踏入未知的那一刻开始的。

1985 年，杭州要修一条通往安徽阜阳的高速公路，参与此事的美国一家投资公司迟迟没能按合同付款。杭州政府聘请英语老师马云到美国同该公司接触。到了美国之后，马云发现那家公司是家骗子公司。

马云在买机票回国时决定到一家互联网公司瞧瞧，这一瞧，世界从此改变了：马云认为互联网这个行业将来肯定有戏，回国后就邀请了 24 位交情很深的朋友来聊互联网，在只有 1 人认为可以试试的情况下决定创办互联网公司。

1995 年 5 月 9 日，中国第一家商业网站“中国网页”诞生。

马云从此走上了互联网创业的道路，几经波折，不改初心。现在庞大的阿里巴巴商业帝国已经改变了几亿人的生活。

马云说起当初孤注一掷做中国黄页：其实最大的原因并不是我对互联网有很大的信心，而是我觉得做一件事，经历就是一种成功，你去闯一闯，不行你还可以掉头。但是如果你不做，总是走老路子，就永远不可能有新的发展。

什么情况下更容易做出改变

许多企业使用 360 度考核法（我曾经工作过的一家单位也用过几年），罗列 *N* 项指标，让同事们给你打分，然后你就会看到自己在哪些指标上分数很低。

企业的管理者认为，当你知道自己哪方面做得很差劲并且影响到你的绩效时，你就会改变自己。这也是多数人的看法：当遭遇失败、事情一团糟时，我们就会改变。

No，其实并非如此，脸上挨一巴掌也就疼几分钟或几十分钟，一般超不过三天。三天之后，好了疮疤忘了疼，该怎么着还是怎么着，还是会回到老路上，该迷茫的迷茫。真正因为 360 度考评而改变的员工少之又少。

你愿意改变并且能改变的真相是这样的：

当发现自己最好的部分时，当有具体的方法可以更多地发挥优势时，我们更倾向于、更乐于去改变。

你知道吗，那些成功改变并做出应有成绩的人，都是发现了自己的天赋。所以，要给自己机会，要拥抱不确定性，像马云说的那样，有一线希望、有一丝兴趣都可以去试试，如果你总走老路子，就永远也不会知道自己的天赋是什么，也永远不可能有新的发展。

尝试不同领域的可能性是发现天赋的必经之路，而“不确定性”则是这条路上的常客，假如你因为中了确定性的毒而听任机会从你身边飘然离去，那很可能你永远也不知道自己

的长处是什么，永远也体会不到做自己喜欢做的事情时投入到无暇思考的状态，那你就不会因为发现了自己某方面的美好而积极做出改变让自己变得更美好。

英国有一位叫理查德·弗思的画家，就是因为一次偶然的机会迷上了绘画，结果从一名屠夫一步步变身为国际顶尖画家。

理查德·弗思 13 岁时成为一名肉铺学徒，16 岁时辍学成为一名全职屠夫。25 岁时，在给一位海洋画家送货时，被对方画作深深吸引，于是开始自学绘画。到 40 岁时，他每幅画作 4 万英镑（约合 40 万人民币）。如今，理查德·弗思被誉为全球排名前 3 的顶尖海洋画家。

我们再举一个程序员的例子吧：

山姆·里奇（Sam Ritchie）曾是美国短程皮划艇代表队中的队员之一，并在 2009 年的世界锦标赛中折桂而归。

山姆·里奇的叔叔是丹尼斯·里奇（Dennis Ritchie），C 语言的缔造者，Unix 系统的开发者，是计算历史上最重要的软件开发者之一。Dennis Retchie 于 2011 年 10 月 12 日去世。

在丹尼斯·里奇去世以前不久，山姆·里奇加入了 Twitter，并在这家公司中与一位名叫奥斯卡·博伊金（Oscar Boykin）的前量子物理学教授联手，共同开发出了新时代开发工具 Summingbird。Summingbird 是一个大规模数据处理系统，支持开发者以批处理模式（基于 Hadoop/MapReduce）或流处理模式（基于 Storm）或混合模式（即组合前两种模式）以统一的方式执行代码。

山姆·里奇说，“我接纳了丹尼斯·里奇的鬼魂。”

山姆·里奇并不是接纳了 Dennis Ritchie 的鬼魂，而是发现了自己的天赋。当你运用自己的天赋时，就会自发地积极投入，就会愿意沿着这个方向去发展、去改变。

解掉确定性的毒，拥抱变化

前面说了那么多，其实就几点：

- 真正的转变是从已知到未知的改变。
- 人更愿意顺着自己的好处（长处、天赋）去改变。
- 接纳不确定性，尝试不同领域的可能性会帮助我们发现天赋。
- 发现天赋，你就能积极投入，解掉确定性的毒，更容易不断改变。

最后，还要推荐两本书，也是我订阅号“程序视界”里推荐过的：

- 《发现你的天赋》。
- 《生命之书》。

你永远都有更好的选择

每个人心里都有一团火——成为更好的自己。即便是强盗，也不愿意别人叫他强盗。可是很多人的行为却往往与“成为更好的自己”背道而驰，有时他做出这种行为是不自觉的或自觉没有更好的选择，但其实是他的生活态度和生活模式决定了他遇到事情就会做出违背内心的选择。当他意识到自己的选择具有不良后果时，往往又会归因于环境或他人，用自己“不得不如此”来开脱。

我非常喜欢弗兰克尔的《活出生命的意义》，他在描述自己的集中营经历时，写道：

在集中营生活的经验表明，人还是有可能选择自己的行为的。有足够的例证（常常是英雄性质的）说明，人可以克服冷漠，克制暴躁。即使是在可怕的心理和生理条件下，人也能够保持一定的精神自由和意识独立。

我们这些在集中营生活过的人，都记得那些走过一个个屋子安慰别人、把自己最后一块面包给了别人的人。这样的人在数量上可能不多，但足以说明一点：有一样东西你是不能从人的手中夺去的，那就是最宝贵的自由，人们一直拥有在任何环境中选择自己的态度和行为方式的自由。

正如弗兰克尔所说，在任何环境中，你都拥有选择自己的态度和行为方式的自由。对更好选择的探寻和坚持，终将让你成为更好的自己。弗兰克尔自己就是绝好的例证。

在当下，我们大多数人一辈子都不会经历特别严峻的环境，多数时候都在工作和生活中随意地做出一些小的选择，然而这些看似小的、不值得煞有介事地讨论的选择，却往往具有深刻的意义和决定性。这就是萨克雷所说的：“人生一世，总有些片段当时看着无关紧要，而事实上却牵动了大局”。

我在订阅号“程序视界”里发布过很多文章，比如：

- 混日子不是你的错，根源在这里。
- 领导不在，咱还干活不啦？
- 十年的老代码，你敢动？
- 入职薪水对你的影响有多大。
- 这 10 个问题去哪啦？

其实都在讲“选择”，本意也是想说：“你永远都有更好的选择，并且，只有你能为你的选择负责”。这次我还是想讨论工作中遇到的选择，看看我们有哪些常见的选择误区，有

没有更好的选择，怎样做出更好的选择。

工作中的选择时刻

《你好哇，程序员》一书中有一书讲了一个开发人员因为联调而与另一个开发人员在办公室打架的故事。其中一个人就是我。而我本该有更好的选择，我可以用更好的方式来了解他手头正在做的事情，区分几件事的优先级，或者让领导出面协调……

打架的例子比较极端，而下面这些问题比较常见，我们经常会遇到并且会不自觉地做出糟糕的选择，我们需要仔细思考这些问题，有意识地做出合理的、对自己的目标有价值的选择（其中部分问题只有程序员才会遇到，其他职业可以类比）。

- 领导不在，工作还需要认认真真做吗？
- 面对难以卒读被标记为禁忌的老代码，动还不是动？
- 工作过程中有松有紧，闲暇时刻，是上网、聊天、游戏、睡觉，还是做点儿其他事情？
- 下班后，是追剧、打游戏还是针对自己技能的缺口有意地学习和练习，或者把发展自己的兴趣、做一些自己想做又非工作需要的事情？
- 你的意见未被采纳，是埋怨决策者、阳奉阴违还是保留自己的意见，全力以赴完成通过的方案？
- 你被邀请参加“无休止、无意义”的会议，拒绝还是接受？你会主动设定会议主题、会议持续时间还是听任开会变成无目的、无结果、无行动的瞎聊？
- 上线的软件服务系统出现 Bug，是先批评追责还是解决问题？
- 公司不加薪，是消极怠工还是继续努力让自己更值钱？
- 得知自己的入职薪水较低，是埋怨、怠工还是继续专注做事体现自己的价值？
- 同等资历和能力的同事升职而你没有，是埋怨领导不公平、怒视同事还是罗列同事优点与贡献，对照提升自己？
- 与同事讨论行动方案，为尊严各持己见争论不休还是谨记目标，抛却对“你的方案”“我的方案”的执着转而寻求更好的方案？
- 用人单位因你没学历或学历低而拒绝你，是埋怨单位还是正视现实，以能力的提升洗白学历低的实际情况？
- 看见老代码或别人代码中的烂窗户，是小心绕行、视而不见还是遇佛杀佛、遇鬼杀

鬼见一个改一个？

- 一开始你就觉得领导拟定的交付日期不可能实现，是默然接受、用无谓的努力与加班体现出“我已经尽力了”进而让自己有所交代，还是积极寻找合理的方式讨论确认一个合理的结果？
- 你觉得公司决定开发的新产品从一开始就注定要失败，是做不讨人喜欢的直言者还是跟着众人在错路上走到尽头，让必然的死亡结果呈现给管理者？
- 你想做 iOS 开发，领导却安排你做了 Java Web 开发，不满地接受、断然地拒绝还是接纳现实但正当表达自己的意见以便在有机会时变换工作内容？
- 别人批评你的报告不严谨，是愤起反击还是闻过则喜？
- 领导对张三谈笑风生，对你公事公办、不假辞色，是讨厌张三、讨厌领导还是坦然接纳、分析原因继续努力沟通？
- 觉得领导管理水平低，是认为自己屈居人下而满腹怨言、得过且过，还是磨炼自己，让自己更完善？
- 公司产品不好用，问题多多，睁一只眼闭一只眼还是努力向上反馈？
- 公司制定了不合理的制度，是当沉默的大多数等着别人站出来，还是自己不卑不亢地提出意见？

……

工作中的很多问题都有更好的选择和解决方案，只是我们当时来不及想或者放任了自己的情绪拒绝去想。比如一个软件周五上线，周四在做最后的测试，测试人员提出了一个关键性的 Bug，可能导致周五无法上线，此时项目经理焦大可能想都不想就会对程序员王二说：“为什么老是到上线前出问题？到底怎么回事？”

焦大这样做有用吗？显然没有。但他会不断地这样做，只要发布版本前出现 Bug 他都可能这样做，他和开发人员之间已经形成了这样的对话模式。而实际上他有更好的选择——牢记解决问题是最重要的，以解决问题为出发点来指导自己的行为。比如他可以问王二：“咱们该怎么查这个 Bug？”接着给出建议：“你看咱们从 APP 本地的 Activity 调用和服务器的轮播海报配置两方面来查，怎么样？”

像前面这些问题，如果我们在做出选择、采取行动前问问自己“这样做对我预期的结果有帮助吗”“我这样做对我的将来有何影响”“有没有更好的选择”，结果可能就会发生变化。然而有时我们是“想也不想”就下意识地做出了选择，怎么办？

怎样做出更好的选择

工作中有很多时刻需要我们做出选择，往往决定的当时我们没时间考虑那么多，事后才会追悔当时为什么那么做。这是可以避免的，只要我们使用一些方法，就可以规避很多糟糕的选择。

最常见的方法，就是知道自己当下想要什么，知道自己的目标，做选择前先问自己一句：**“我这样做对我的目标究竟有什么帮助？”**

有人会说，事情发生的那一刻，哪有时间去梳理自己最想要的东西啊？都是一闪念的事。没错，当时来不及琢磨那么多。你最想要的东西，你的目标，这些都是在平静时梳理出来、固化到内心深处的，在选择时只需调取即可。

按照下面这三步来做，就可以有意识地做出有价值的选择：

- 梳理自己最珍视的人、事、物，列出自己最想要的财富清单（参考《当时忍住就好了》）。
- 我做了某种选择会有什么后果。
- 这个后果是不是会与我最想要的财富清单有冲突。

如果你觉得这种方式还是麻烦，那么还有更简单的：

- 列出你最看重的三种东西，划掉一个，保留两个（参见“一招搞定多 Offer 选择问题”中提到的职业价值观）。
- 想象你的选择是否与自己最看重的东西冲突。

在应用上面的方法时，很多时候我们还是会说：“我真的好纠结，我真的做不到啊，我真的确定不了到底哪个才是我想要的”。那就先把每一个想要的都放大，去想“没有它会怎样？我能不能接受？”不能接受，觉得自己会死掉，觉得选择了它就能够“虽九死而犹未悔”，那就是你想要的。要是这样想来想去还做不出结果，那也正常，没什么好怕的，你不是一个人在纠结，没必要因为一时半会确认不了自己最想要的而烦恼。人生永远没有太晚的开始，可以慢慢来。

一旦我们按照上面的方法梳理出了最重要的财富清单，接下来只要掌握“暂停”的方法，就可以在多数时候有意识地做出有价值的选择。

按下暂停键

有时我们明明确认过自己最看重的要素，可工作中真的遇到问题，还是想也不想就做

出了坏选择，怎么办？

武侠情景喜剧《武林外传》里的郭芙蓉擅长排山倒海掌法，与人话不投机就要给人两掌。后来她为了控制自己这种惯性行为，就在觉察到自己愤怒时暂停三秒，对自己说：“世界如此美妙，我却如此暴躁，这样不好不好”。

这就是一种暂停方式。

暂停的关键，就是从你最珍视的、最想要的、最不想失去的人、事、物中提取简单、快捷、有效、能瞬间调取的触发器（说明结果的短句或画面，能瞬间唤起你的情绪让你为它而行动），能够在闪念之间出现在你脑海里，截断你的惯性反应链，阻止你做出不良选择。

张三目前的工作薪水丰厚又不太忙碌，朝九晚五比较规律，但是他觉得经理总是对他挑三拣四，每次都能从他相比其他同事已经好很多的工作报告里挑出几个无可厚非的小毛病来批评他，这让他无法忍受，一到周会这样可能当众被领导批评的时刻，他就有种“老子不干了”的冲动，而且这种冲动越来越强烈，眼看就要控制不住了。

张三可以怎么做呢？什么是更好的选择呢？

张三当初选择这家公司，就是因为氛围宽松薪水中等，他可以照顾怀孕的老婆，而现在老婆生了孩子，他目前最重要的是有能力养活自己、老婆、孩子，还要有时间陪伴他们。如果这是他最重要的愿望，那他可以用“决不能因为买不起奶粉与老婆吵架”作为触发器或者设想一个“孩子哇哇哭，老婆拿着空奶粉罐冲他摇晃”的画面，这样的触发器对他就是管用的，只要他脑海里闪现这个画面，就可以抑制住自己辞职的冲动。一旦他没了辞职的念头，就能更好地审视“领导对他挑三拣四”这件事，就可能反观自身与内心，通过把工作做得更好或提升沟通与表达能力来消除他与领导之间的沟通问题。

再举个例子。我以前对交通规则不太在意，看见路口没车就想冲过马路。现在我非常遵守交通规则，哪怕夜半三更路上没一辆车，我也要等绿灯才过马路。因为我觉得生命和健康最珍贵，万一冲来一辆车把我撞了，自己就有可能死去或残疾，家庭就会因此万劫不复。所以当我有要闯红灯的念头时，脑海里迅速就会闪现一张我躺在血泊中的画面，我就能停下脚步。

没有选择的选择

稻盛和夫大学毕业后找不到工作，后来托关系进了一家濒临倒闭的陶瓷厂。这家陶瓷厂经营惨淡，员工工资都发不下来，旁边小卖部的老板都说在这里上班老婆都找不到。稻盛和夫对瓷器不感兴趣，又拿不到工资，吃穿用度都成问题非常苦闷，一度徘徊在黑社会

堂口前，想要加入他们。

过了没几个月，与稻盛和夫同时加入这家陶瓷厂的几位同事先后离职，很快只剩他和另外一位同事。两人看看在陶瓷厂待下去不是个事，就去参加自卫队干部候补生学校的考试，两人都通过了，只要从家里拿到户口簿就可以报道了。那位同事的家人很快寄来了户口簿，稻盛和夫写给家里的信却石沉大海——他哥哥非常恼火，认为“家里节衣缩食把你送进大学，多亏老师介绍才进了京都的公司，结果你不到半年就忍不住要辞职？真是一个忘恩负义的家伙。”，气愤之余拒不寄送户口簿复印件。

后来只剩稻盛和夫一个人继续留在破败的陶瓷厂。此时他已经无法选择，只好从自己身上开刀了。他不再抱怨工作和没有工资，决心要先埋头工作，于是铺盖卷一卷，锅碗瓢盆都搬进实验室，睡在那里，昼夜不分，极度认真地投入工作。

后来稻盛和夫接受了一项新任务，研究开发一种叫作“镁橄榄石”的新材料，这是一种新型陶瓷，绝缘性超好，特别适合于高频电流。但这种材料极难合成成型，因为没有合适的粘性材料。当时全世界只有通用电气（GE）一家成功合成“镁橄榄石”。

稻盛和夫在像样的实验设备都没有的境况下夜以继日地工作，反复试验，结果却总是不理想。为了解决“粘性”的问题，稻盛和夫每天思考、试验，绞尽脑汁，百思不得其解，到了几乎痴狂的地步。有一天他在走进实验室时被某个容器绊了一下，差点跌倒，下意识一看脚下，鞋上沾了试验用的松香树脂。正当他要喊“谁把松香搁在这个地方！”时福至心灵，立即架起一个简单的锅，将陶瓷原料和松香放入锅中，一边加热一边混合……

成功了！令人头痛的难题就这样解决了，稻盛和夫称那一瞬间为“神的启示”。

稻盛和夫后来创立“京瓷”，上市，成为世界五百强企业。再后来又创立 KDDI，再度成为世界 500 强。稻盛和夫也成了唯一一位创立两家世界五百强公司的人。再后来 78 岁时出任日本航空公司董事长兼首席执行官，把日本航空公司重新带回世界 500 强。

当你没有任何选择时，还可以选择以什么样的心态面对眼前的现实。这是稻盛和夫在松风工业公司的经历给我们的启示，你自己的心态转变了，人生的转机就来了。

而阿德勒在《自卑与超越》中的话则从另一个方面对此做出了解释：

一个精神正常的人，当他在一个方向努力受阻时，他总能找到新的方向。只有神经症患者才会只认定一个目标，他们会说：“我只能这样，此外别无他法。”

永远都有更好的选择

无论何时，无论你处在什么境况之下，都拥有选择的自由。最重要的是，不要自我设

限，破除进退维谷、非此即彼的模式，因为，更好的选择永远都在。只要你能梳理出自己最珍视的人、事、物，抽象出对你有效的触发器，就能使自己在面临工作中的选择时暂停，思考“还有没有更好的选择”，有意识地做出对自己更有价值的选择。

当诱人的工作机会来临

有个朋友，工作了 10 年左右，春节后换工作，拿了三个 Offer（西安）：

- 通信行业的一家研究所，软件开发工程师，月薪 7000 元，承诺有月奖金、年终奖金。
- 一家做大数据的公司，软件开发工程师，月薪 15000 元，13 薪。
- 一家做外包的企业，项目经理，月薪 20000 元，13 薪。

他问我应该选择哪家……他正准备结婚，需要买房买车，还想早点生个小程序员……各位说他应该选择哪家呢？

恐怕每个人都有自己的答案，也都有自己的选择理由。

大仲马说：“一个人一生中会有这样的时刻，这一刻将决定他整个的未来。然而不论这时刻多么重要，人们却很少有思想准备并且按自己的意志去行动。”

我们每个人都会面临这样的时刻，在类似的时刻来临时，各种诱惑因素会同时出现，交织着影响我们，使我们难以做出长远来看有价值的人生选择。

“一招搞定多 Offer 选择问题”一节介绍了职业价值观，还提供了“生涯平衡单”这个工具来帮我们做选择。然而有一点我们当时没说，现在需要单独列出来，因为它比“生涯平衡单”“决策树”SWOT 分析等都更重要、更有效，那就是：提前预测你的职业选择。

提前预测职业选择的要点在于：

- 你知道在未来的某个关键时刻，必须要做出职业选择。
- 你会预测你在关键时刻时你将做出的职业选择。
- 你会把这些预想的选择储存在脑海中，这样当选择时刻来临时你就能即时将这些预设的选择提取出来加以利用。

一旦你做好了提前预测，选择就非常轻松了，不会因为重要时刻出现的一些情感因素妨碍你做出理智的判断，大大降低失败选择概率及失败选择对你产生的负面影响，使整个的职业生涯朝着更符合你长远规划的方向发展。

提前预测让我果断放弃管理职位

2015 年 11 月底，我和朋友解散了创业团队。之后我根据多年的工作经验和历次职业转换的经历，经过慎重思考和规划，圈定自己将在“专业技术”方向发展，做一个快乐的匠人。后来我重回软件开发岗位，并且告诉自己，三年之内不论谁再拉我去做管理工作，我都不会答应。

2016 年 3 月初，一个朋友打电话说深圳有家软件公司准备在西安开设分公司，他推荐我做分公司的经理，薪水丰厚且执掌一方，我果断拒绝了。因为我在几个月之前已做出了决策：我的职能取向是专业技术人员，我希望通过在专业技术上的努力和探求来体现自己的价值。所以我可以轻松做出选择。

像类似的情况，我回归开发岗位后还发生过两次，我都毫不犹豫地拒绝了——因为我已提前做好了选择并刻印在脑海里，相关事件发生时只需要提取选择分支，走个 if-else 就好了。

那么，怎样提前选择呢？这里又可以引申出来两点：

- 提前选择的基点。
- 预测清单。

当你弄明白自己职业选择的依据，提前列出了预测清单后，提前选择就能发挥作用。

提前选择的基点

为什么我们能够提前做出职业选择？

其实奥妙就是“一招搞定多 Offer 选择问题”一节中介绍的职业价值观。简单点说：

只要你知道自己真正想要得到什么、真正想成为什么样的人，你就可以在心中设想你真正做出有价值的、诚实的职业选择的情形。

为了提前做出选择，我们可以问自己几个问题：

- 最想要什么。
- 最不想要什么。
- 最珍视的人、事、物。
- 最想成为什么样的人。

在“如何快速定位自己热爱的工作”一节中，我们介绍过“想要清单”和“不想要清

单”，并且提供了从“不想要清单”推演“想要清单”的方法，感兴趣的朋友可以看看。

除了“想要”和“不想要”这两个清单，我们还要弄明白自己最珍视的人、事、物，这一点其实和想要清单有所重叠。比如你最重视女朋友，工作单位与女友的距离无论如何不能超过两公里；比如你最重视自己才六个月的孩子，在孩子上小学前，周末、晚上都不能加班，因为陪伴孩子是最重要的；比如你最重视健康，一定要杜绝吸烟喝酒……就是类似的东西，这些你最珍视的人、事、物，将会导引你做出有利于人生的正确选择。

无论我们怎样分析自己“想要什么”“不想要什么”“最珍视的人、事、物”，目的其实都是想找出在职业上“我最想成为什么样的人”：一辈子机械地拷贝粘贴的码农、在设计和实现上都精益求精的匠人，还是走上管理岗位成为 CTO？

你在职业上最想成为什么样的人，你对职业形象的设想，是影响你选择的最重要的因素。假如你想成为一个匠人，想法非常坚定，那么你在面临管理职位附带的权力、地位、高薪时就不会动摇。就像我现在的想法一样。

一旦我们根据上面四点完成了对自己的梳理（这种梳理是阶段性的，也是动态变化的，需要不断迭代），就能形成我们的预测清单，当重要的时刻来临，要做出选择的时候，就可以按下暂停键，拿出我们的预测清单，回避潜在的各种诱惑，确保做出的选择符合我们之前慎重思考、分析得出的结论。

请谨记，我们现在每一次努力的梳理，都是为了未来我们最珍视的财富和梦想出现在眼前时，我们能拿出最佳状态做出正确的选择。机遇只青睐有准备的人，这句话应该像弹幕一样时刻飘在我们眼前。

预测清单

预测清单，说白了就是你根据自己“最珍视的人、事、物”“最想要的”“最不想要的”和“最想成为的人”列出来的一个能帮助你提前选择的单子。比如类似下面这样：

- 绝不在受到薪水诱惑时做出职业选择。
- 工作单位和家的距离绝不能超过 5 公里。
- 周六绝不能加班，这是陪伴孩子（女友、父母）的时间。
- 如果要开车，绝不沾酒。
- 晚上绝不能吃过 9 成饱，也不能吃油腻、过咸、过甜的食物，因为它们会导致发胖、血脂高、血压高。

- 不管如何疲惫，每周至少要有四天去公园走两圈。
- 决不从事管理工作。
- 决不招募颜值高的女秘书。

……

每个人都有自己的清单，每个人的清单都是不同的，也都是经过不断的努力梳理整理出来的。对有些人来讲，可能很容易就列出一份清单；对另一些人来讲，可能需要不断的、往复的分析，甚至需要有专业人士的帮助才能列出自己的预测清单。然而不管付出多少精力，都是值得的。

罗曼·罗兰说："人们通常觉得准备的阶段是在浪费时间，只有当真正机会来临，而自己没有能力把握的时候，才能觉悟自己平时没有准备才是浪费了时间。"

所以，做再多的准备都不为过，何况一个预测清单？

我们的预测清单，可以帮助我们在面临岔道时做出有长远价值的选择，引导我们走上正确的道路。

所谓选择，就是"因为更想要某种东西而放弃你原本想要的另一种东西"。而在某些时刻，有些因素和情绪会被即时放大，瞬间击溃我们原来计划好的、看重的条款。比如你想保持体型，却和朋友一起走进了哈根达斯，被冰淇淋诱惑；比如你原本坚信金钱是要脚踏实靠自己的实力赚的，可你的朋友忽然向你推荐了几只轮番涨停的股票，你看人家不费吹灰之力一天赚好几千元，脑子一热就杀进去了，结果过了两天就开始连番跌停……

所有这些情形，都是因为在特定的时刻，某种人、事、物、场景会触发并放大你的某种情绪，当那种情绪不受控制地在瞬间无限放大时，你的脑子就会短路，就会抛开一切、不顾后果只求速爽。而如果你已经将预测清单深刻于脑海，就可以在最后时刻通过**快速的后果预测**让灵台保持清明。比如你希望通过技术安身立命，打算花 10 年之功来成就自己，那么这一点可以缩短为"绝不能身无长技被人耻笑"，这样当你在疲惫时看到别人通过人际关系技巧扶摇直上九万里时，就不会因为心理失衡而转变方向去走不适合自己的管理路线。

我们在每一次做选择时，都应当考虑"这样选择到底是为了什么""这样选择到底会让我成为什么样的人"，还要问自己"这样的选择是否偏离了我的预测清单""这样的选择是否有助于我成为我想要成为的那个人"。希望通过这样的方式，能够做出有价值且诚实的选择，帮助我们实现自己想要的人生。

我为什么放弃管理重回软件开发岗位

2015 年 12 月，我从创业失败中走出来，重回软件开发岗位。

我做过 7 年左右的技术管理工作，当过项目经理、部门经理、项目总监、技术总监，在技术管理这条路上摸爬滚打了这么多年，按常理来讲，接下来还应该走管理路线。

在我找工作时，朋友们都说，你可以轻松找到一个经理职位。有一些猎头也联系我，让我考虑某些公司的经理或总监职位。然而我最终没有那么做，我选择了回头去做一个软件开发工程师。

这种选择，从社会普遍的价值观来看，是一种倒退——在人们的意识里，管理者的收入和社会地位都比技术人员要高。

然而我的考虑并非如此。我运用所学的职业规划技术对自己进行了梳理，经过深思熟虑得出了对自己的定位。在思考的过程中，我曾经在微信订阅号“程序视界”发布了一篇题为“大龄程序员的未来在何方”的文章，感兴趣的可以读读。

那篇文章并没有提我为什么放弃管理重回软件开发，现在，在这里，我把自己的定位过程晒出来，希望对同样迷惘的人有所帮助。

两个关键问题

职场人士要想找到自己的职业定位，必须弄明白两个问题：

- 我想要做什么？
- 我能做什么？

很多人不知道自己想做什么，他要么随大流，去做别人所做的事；要么被别人支配，去做别人希望他做的事。

我人生的前 33 年也是如此。

维克多•弗兰克尔在《活出生命的意义》一书中介绍“意义疗法”时提到，人实际需要的是为追求某个自由选择的、有价值的目标而付出的努力和奋斗。

我非常认同这一点，你需要什么、你选择什么、你为你的选择而努力和奋斗，非常关键。

从 2013 年下半年开始，“我想要做什么”这个问题在 11 年之后再度出现在我面前（大学时曾考虑过），经过又一次的探索，终于有了答案。

现实多数时候就像一个玩笑，有时你知道自己想要做什么，可却不一定适合。所以，还是要从现实情况出发，弄明白“我能做什么”这个问题。

“我能做什么”这个问题，更多地关注你自己的特质（潜质）、你拥有（擅长）的技能。以我为例，2005 年我离开技术支持岗位，寻找软件开发工作，我的老领导接收了我。我从未写过什么程序，连 C 语言都是找工作前花了一星期学的。他为什么愿意让我到他的单位去做软件开发，他为什么愿意冒这个大多数人避之唯恐不及的风险？

我没有问过他，但我想他从他的角度分析，我身上一定有某些潜质很适合做软件开发。结果也证实了这一点，我很快就入行了，并且做得相当不错，还获得过公司的 S（Super）员工荣誉。

你想做点什么，也明确了，但，你能干好吗？在你没干过之前，你自己也不确定。所以，很多时候，我们又会从一个人所擅长的技能来考虑他能做什么，有时这是一种出于现实考虑的妥协——因为他能做的并不一定是他想做的。

有个哥们儿上大学时父母代为选择了英语专业，为了不让父母失望，他苦学英语，过了专八，发音纯正，口语流利，用英文与人沟通就像母语般顺畅。然而现在他根本不愿提起英语，一想到就恶心，他现在的工作也和英语没什么关系，甚至根本用不上。当你知道他英语极好，劝他找一份能发挥他英语特长的职业时，他甚至都不愿意谈。

很多人都是这样，因为家庭、社会的原因在不喜欢的岗位上工作多年，练就了他不喜欢却擅长的技能。所以通过分析技能树，判定一个人适合做什么，存在一定的风险，有可能你告诉他他能做这个，他马上就反驳说他不喜欢。但我们又必须分析一个人现在擅长做什么，只有弄明白他现在所处的位置，才能更顺畅地走向将来的目标。这是积极的妥协。即便一个人厌恶他拥有的这些技能，但如果再次利用这些技能是为了更顺畅地切换到其想做的职业时，他通常也可以接受。

挖掘自己想做什么、能做什么

我在微信订阅号“程序视界”发布过一篇文章“如何快速定位自己热爱的工作”，在那篇文章里提供了一些方法，帮助我们发现自己想做的工作。里面提到了成就感事件，这是一种简单有效的方法。我对自己的定位，也从这里开始。

那些让你特别有成就感的事件中隐藏着你的职业兴趣、喜欢的技能、擅长的技能，可以挖掘出你今后乐意从事的职业方向。

我们可以遵循 STAR 原则来回顾成就事件：

- S（Situation），背景情况，包括面临的障碍、限制或困难。
- T（Task），任务，目标，想完成的事情。
- A（Action），行动计划与步骤，如何克服障碍、达成目标。
- R（Result），对结果的描述，重点关注你取得了什么成就。

一旦有成就事件，就可以继续挖掘。可以按照下面两点来分析：

哪一个具体的点让你特别有成就感，比如“达成结果受到领导表彰”“独立克服某个技术难题”、“用自己的 XXX 帮到了某某某”“成功组织大家齐心协力达到目标”……

- 用到了什么知识、技能。

通过这样的分析，你就可能找到自己的最佳技能和工作中最在意的点在哪里，而有了这些，就可以根据他们来确立新的职业目标。

我在重新定位自己时，分析了与工作相关的三个成就感事件：

- 转行做软件开发。
- 第一代互联网机顶盒优化。
- 写作《Qt on Android 核心编程》和《Qt Quick 核心编程》。

1．转行软件开发

成功切入软件行业，这是我大学毕业后的第一个让我特别有成就感的事件。

- S：2005 年，厌倦了做技术支持，准备转做软件开发，没有任何编程经验。
- T：找到一个软件公司，进入软件行业，从事软件开发。
- A：尝试在西安大唐电信内部转研发，也托一个朋友做了推荐，被拒绝；辞职；找到正在读软件学院研究生的同学，从他那里拿到一本《Java 2 核心技术卷 I》，自学了一阵子，放弃；转向《C 程序设计语言》（the C Programm Language），花了一星期，掌握了基本语法；遍地撒网投简历，持续将近 2 个月，不断参加笔试，不断复习 C 语言。
- R：不断的笔试，让我更熟悉 C 语言，在 1 个月后，基本上都能通过笔试关；后来遇见我转行之路上的“贵人”谢总，他让我完成一个马踏棋盘的 C 程序后找他，我完成了，进入西安信利软件科技有限公司，开始软件开发之路。

在这个事件里，最重要的发现有三点：

- 有自主学习的能力。曾经我以为自己丧失了自主学习的能力——大学四年玩了两年

游戏泡了两年 BBS，考试基本都是 60 分。

- 对软件开发感兴趣，这是我能学习下去的关键因素。（但当时之所以想转到开发工作上来，其实并没有深入分析，只是扫了一遍所在公司的岗位，觉得除了开发，其他自己都干不了）。
- 能承受转换成本和风险，这是我的一大优点！当时转做开发，很可能失败，到一个多月时其实已经忐忑了，不过还好没放弃。还有，做了开发后，每个月到手的工资，只有做技术支持时的三分之一，不过都挺过来了。

2．第一代互联网机顶盒优化

我在信利软件负责开发的第一代互联网机顶盒，投放市场后，视频播放功能存在严重影响用户体验的 Bug，带队加班加点三个多月，解决了相关问题，用户体验获得很大提升。这是另一个成就事件。

- S：第一代互联网顶盒的播放功能频繁出现 Bug；芯片方开发技术支持不到位；我是研发部门经理；当时孩子只有 2 岁。
- T：解决关键 Bug，提升机顶盒使用体验。
- A：重构播放器框架，对芯片方的 SDK 里相关部分做逆向工程，参考 ffmpeg 设计插件式的播放器框架；设计实现 httpserver，代理转发互联网视频；持续三个月加班，一周五个工作日四个晚上到 10 点或更晚，周六全天、周日半天也要献给工作。
- R：重构成功，机顶盒可用度大大提升，终于可以说它是一个“产品”了。

在这个事件里，我的发现：

- 我发现自己更喜欢做具体的技术工作：主导播放框架的设计，实现核心模块的开发；完成流媒体代理 httpserver 的设计和实现。这些工作给了我很大成就感。当我和团队一起干这些事情时更开心，反过来，当我做管理，要去组织、领导、激励别人时，远没这么开心。
- 我喜欢做产品，能够克服现实障碍，投入地做自己喜欢的事（感谢我媳妇对我的支持）。
- 我关注并享受实现的过程，乐而不觉其苦。

3．独立写作技术图书

写作《Qt on Android 核心编程》和《Qt Quick 核心编程》带给我很大影响，到现在还

在影响着我。

- S：公司产品不温不火，作为研发部门经理，时间相对充裕但时感无聊；希望保持技术竞争力。
- T：保持技术敏感度和竞争力。
- A：研究产品开发中用到的 Qt 框架，跟踪其在移动端的最新进展，学习、记录、更新博客；制定写作计划，规划写作时间（晚上到 12 点半或 1 点，早上五点多，周末两天），严格执行；之前没写过书，全凭自己摸索，没有反馈。
- R：出版《Qt on Android 核心编程》和《Qt Quick 核心编程》。

我发现：

- 我喜欢技术，也喜欢写作
- 我享受写作的过程，对结果不甚在意。当我完成第一本书《Qt on Android 核心编程》时，觉得很空虚，虽然过程很累，可我居然还愿意重新体会那样投入的过程，所以我又给自己定了写作《Qt Quick 核心编程》的任务并很快开始执行计划。
- 我希望自己能通过写作分享自己的东西，影响别人。

成就感事件关注的是个人的高峰体验，与之对应的，还有低谷体验。低谷体验也可以帮助我们分析自己适合做什么——分析出不适合的，就接近了适合。

我曾经到一家公司做过项目总监，管理三个团队，40 来个人，工作性质属于纯管理。我只做了一个月，就离职了。我当时做得最多的是找人聊天，推动团队内部及团队之间的协作，拟定制度，想办法改进开发过程……我心里长满了野草，有种即将荒芜的感觉，看不到自己的价值所在，不能从这样的工作内容中体会到充实、愉悦。这样的经历进一步确认了我做技术的倾向。

通过对成就事件的分析，我明白了自己的职能取向是专业技术者，我还发现了即使没有回报也愿意热情投入的事情——写作，所以，我的将来，就是一边做软件开发，一边写作。至于软件开发的的前景，我发布在订阅号“程序视界”里的文章——大龄程序员的未来在何方——已经做了展望，我一点也不悲观。

希望成就事件分析法能够帮你找到自己想干、能干的事情。

跳槽与薪水篇

月薪 3 万元的程序员都避开了哪些坑

程序员薪水有高有低，有的人一个月可能拿 3 万元、5 万元，有的人可能只有 2 千元、3 千元。同样是有 5 年工作经验的程序员，可能有的人每月拿两万元，有的人拿 5 千元。是什么因素导致了这种差异？我特意总结了容易导致薪水低的九大行为表现，避开这些大坑，你就离高薪不远了。

习惯即刻回报

他不懂得只有春天播种，秋天才会有收获。刚刚付出一点点，甚至还没有付出，就想要得到回报。技术刚刚掌握，能一边百度一边干活了就计算该拿到多少报酬。找工作先想着多薪资，入职没干几个月就想着要加薪，一看没涨就放弃，准备通过跳槽加薪，即使不跳槽，往往也会因为没加薪而牢骚满腹，工作敷衍了事。

一个程序员的价值，是通过他带给公司的价值体现的。先要给公司带来价值，然后才会反过来在薪水上体现出自己的价值。公司都很现实，很少会为你的潜力买单、在你还没有体现出价值时就给你很高的薪水。

在生活和工作中，一定要懂得付出，不要那么急功近利，马上就想得到回报。天下没有白吃的午餐，你想获得什么，就得先付出什么。唯有种下种子，然后浇水、施肥、除草、杀虫，然后才能等来收获。

缺乏学习热情

很少有哪个岗位的人像程序员这样需要持续不断的学习，软件开发的技术日新月异，而每一项技术又往往博大精深，不持续、深入钻研是很难掌握的，更别谈精通了。如果你

对一项技术不能深刻理解、熟练应用，表现出来的水准仅仅是能干活、还行，那很难说会有公司愿意为“还行”付出大的代价，只有脱颖而出，才可能备受重视。

假如你对学习、掌握、精通技术没有兴趣，面对不断涌现的新语言、新技术、新框架没有学习欲望，那单就软件开发这个工作而言，你不但眼下不太可能拿到高薪，将来也不会。在这样一个快速变化的时代，只有不断地学习才不会被抛弃。

不够努力

虽然我们都知道努力学习可以改变我们的技能水平，持续努力、不懈坚持可以让自己有所建树，可还是有很多人浅尝辄止，三天打鱼两天晒网，搞两下能“Run”就放下了。

人和人在聪明才智上的差距并没有想象中大，甚至很多时候，从大多数人的努力程度来看，根本还轮不到拼天赋。如果两个人的实力半斤八两的话，热情工作、努力坚持的人，一定比较容易成功。

畏难

做事拈轻怕重，不愿挑战。殊不知能力就是在不断挑战、不断突破自己的过程中历练出来的。在一个公司里面，经常承担高难度任务的程序员，一定是成长比较快的，薪水增长也一定比较快。越是困难的事情，越能体现出个人价值，也越能带给个人成长。

万事起头难，不要害怕困难。事情做不好往往不是因为没有能力，而是由于缺乏恒心。只要不怕困难，坚持前行，一定会有不一样的收获。

缺乏责任心

工作上不管什么事，认为反正不是自己的事，缺乏责任心，干好干不好都无所谓，对交付承诺、产品质量都不在意，没什么事能让他上心。

一个人的责任心如何，决定着他在工作中的态度，决定着其事业的好坏和成败。如果一个人没有责任心，即使他有再大的能耐，也做不出好的成绩。

消极，抱怨

工作稍有不顺，就怨气沸腾，这个怎么样，那个怎么样，而我怎么就这样，任务不公平，资源不公平，谁谁不支持，谁谁不配合……

抱怨不能使事情变好，反之，它会让负面情绪蔓延，蚕食你的精力和时间，让你产出更低。成功者永不抱怨，抱怨者永不成功。立刻停止抱怨，早一分钟停止，你就离目标近一分钟。

没有时间管理观念

每个人的一天都只有 24 小时，人和人的差别就在于如何利用时间上。

有的人每周都有目标，每天都有计划，早上起来会想当天要做的几件重要的事，晚上会回顾当天完成的事，总结干成了什么干坏了什么，还会有计划地学习新知识、新技能，这样日积月累、不断坚持，每一天都是高效的，每一天都朝着更丰富、更完美的自己前进。

而有的人则漫无目的，走到哪算哪，到了公司，上午基本做不成事，到下午了还不知道要做什么，晚上也发愁如何消磨时间……

为薪水工作

虽然工作的一大目的是获得薪水，养活自己并供给家庭所需；但这只是工作最直接的报偿，同时也是最低级的目标。

如果我们为薪水而工作，将注定是短视的，也将注定受到最深的伤害。假如你看不到工资以外的东西，斤斤计较于薪水、福利、职位等，那么外界的风吹草动就可能让你像浮萍一样飘来荡去，你很快就会失去平衡，失去信心，失去热情，失去平和，进而在工作时总是采取一种应付了事的态度，能少做就少做，能躲避就躲避，觉得只要对得起自己的那份薪水就成了。长此以往，你追求的高薪水反倒得不到。

我们进入一个公司工作，是为了自己，不是薪水也不是别人，比薪水更重要的，是成长和成就自己的机会。我们一定要明白，公司、企业、组织，都是锻炼自己、修炼自我的平台，我们不是为薪水、为老板、为家人而工作，而是为实现自我而工作，是为更完美的

自己而工作。

唯有志存高远，方能风行天下。

其实不喜欢软件开发

有一部分人从事软件开发工作，并不是因为喜欢，也并没有干着干着从不喜欢变成喜欢。他们可能是喜欢软件开发附带的高薪水——平均薪水比其他行业高。人做一件自己不喜欢的事情时，心理上没有亲近感，不会想着怎样把事情做得更好，往往是差不多就成了，不太可能有精益求精、积极向上的追求。因为在做不喜欢的事情时，情感上是拒绝的，情绪上是想逃离的，总想着早做完拉倒，每一天去单位时不是充满期待，而是充满各种担忧、烦躁、畏惧，到了单位，稍有困难或不顺心，就会消极、抱怨、抵触、拒绝……

做喜欢的事，能最大限度地发挥一个人的潜能和热情，会最快速地通向成功，成就自己。而做不喜欢的事，一开始就注定了事倍功半，最后也往往会痛苦不堪或半途而废。

程序员如何谋划出月薪 3 万元

我发表过一篇文章“月薪 3 万元的程序员都避开了哪些坑”，在我的微信订阅号“程序视界”、CSDN 博客、简书等平台都获得了非常广泛的关注，点击量和评论多得出奇。有一些朋友觉得很难做到。其实，月薪 3 万元是可以一步一步谋划出来的。个中关键，从大的方面来说，有两点：

- 个人商业价值的挖掘与修炼。
- 职业机会的探索与把握。

在展开之前，我们需要先澄清三个问题：

- 商业价值包含哪些因素。
- 职业机会包哪些要素。
- 内生涯与外生涯是什么。

本节从整体上分为两部分，第一部分是和大家一起弄明白上面三个概念；第二部分，

我们会一起研究如何通过商业价值的挖掘、内生涯的修炼及职业机会的寻找来实现月薪 3 万元的目标。

关键的四个基本概念

小米在发展过程中，挖过两个比较著名的人物，一个是谷歌 Android 副总裁胡戈·巴拉，一个是新浪总编陈彤。我们来看两个问题：

- 小米看上了这两个人的什么？
- 这两个人看上了小米的什么？

1．商业价值要素

胡戈·巴拉身上有几个很有价值的点，这是他能加盟小米的关键：

- 在谷歌工作 5 年。
- Android 产品管理副总裁。
- Android 操作系统团队最具代表性的公众人物之一，他曾经频繁出席谷歌的新闻发布会及谷歌的 I/O 开发者大会。比如 Nexus 7 平板电脑就是由他登台演示的。
- 熟悉海外市场。

相信这也是小米看中胡戈·巴拉的重要原因。而这些原因，其实就是胡戈·巴拉身上体现出来的商业价值。具体来看，谷歌的工作是其个人经历，为其积累了广泛而有价值的人脉，同时他也在 Android 产品管理方面有独到的知识和技能。

个人商业价值中非常重要的一个要素，那就是天赋。天赋指人与生俱来的某些特质。在胡戈·巴拉身上，人际交往能力和个人形象气质就属于天赋。在姚明身上，他的身高算是天赋之一，特别有助于他的篮球事业。在罗纳尔多身上，他的爆发力就是天赋，所以你经常看到他突然加速把对方球员甩在身后然后狂奔几十米形成单刀射门的优势。如果一个人能发挥其天赋，将其与职业关联起来，必能实现自我，有所成就。

总结一下，商业价值包含 5 个要素：

知识、技能、天赋、经历、人脉

我们每个人身上都有这些东西，如何挖掘出来，如何着重培养某方面的价值，就是我们获取高薪的关键。

2．职业机会

现在我们看看胡戈·巴拉为什么会选择小米。我猜测有这么几点：

- 小米处在中国这个经济快速增长的发展中国家。
- 智能手机的发展在中国处于上升期。
- 小米公司上升势头强劲。
- 负责小米国际业务拓展。
- 薪水可观。

第一点是宏观环境的东西，只有宏观环境足够好时，机会才会多。

第二点是产业环境，在一个好的宏观环境里，一个好的、处于快速发展期的产业是最有吸引力的，到这样的产业中去，个人必将随着产业的发展而有所成就。所谓“站在风口上猪都能飞起来”，就是这个道理。

第三点可以说是组织环境，一家处在好的宏观环境下、好的产业环境里的公司，又正处于上升期，对个人来讲，是天赐良机啊。到这样的公司里，你就是躺着不动都可能比一般人达到的高度高。

其他都是小米提供给胡戈·巴拉的职位相关的东西，属于职业资源。另外还有一点是家庭环境，一个人的家庭，可能成为他选择新职业的助力，也可能成为阻力。比如有很多学生毕业后就被父母通过人脉安排到银行、电力等国有企业工作，如果他们要跳槽，父母一般都会成为阻力。

好了，现在职业机会的 5 个要素都出来了：

宏观环境、产业环境、组织环境、职业资源、家庭环境

3．内生涯与外生涯

内生涯与外生涯是职业规划中非常重要的一组概念，理清了它们，我们就知道在哪个方向上努力可以提升自己，进而提升薪资待遇。

内生涯与外生涯，最直接的就是用你的身体区分，我们常说的“身外之物”，其实就是外生涯。我们常说的“内在”，就是内生涯。比如你现在是华为的软件工程师，负责相机模块开发，月薪 2 万元，这属于外生涯；而你熟悉 C++、Android 的 Camera 框架、图像处理算法，这些就属于内生涯。

属于外生涯的那些身外之物，是别人、别的组织给予你的，很容易因为外界环境的变

化而被剥夺。比如你曾经是诺基亚北京研发中心的开发人员，那么 2014 年 8 月份诺基亚北京研发中心大裁员，你瞬间就失去这个身份。

而我们归属于内生涯的那些内在之物，一旦拥有，就是你的，别人很难夺走。比如你掌握 C++这门语言，熟悉 Android 应用开发框架，精通数据库调优，你工作负责，为人诚信，勇于担当，这些知识或技能或心态，别人不可能从你这里拿走。

现在对内生涯和外生涯我们应该比较清楚了，下面总结一下它们都包含哪些内容。

外生涯包括（但不限于）职务目标、经济收入、工作内容、工作环境、工作时间、工作地点、企业文化、薪酬福利、通勤状况等。

内生涯包括（但不限于）知识、技能、工作经验、心理素质、内心情感、行为习惯、视野、观念、职业心态（爱与感恩、责任、忠诚、诚信、勇气、担当）、职业成熟度、心灵成长等。

如果仔细品味内生涯包括的东西就会发现，它们和个人商业价值的某些要素是重叠的，尤其是知识、技能、天赋这些东西。而外生涯包含的一些东西，其实也和个人商业价值相关，比如你的职务目标、工作内容、工作过的企业等，都是你的经历。你在选择一份新的职业时，不论内外，只要能体现商业价值的内容，都会被重度参考。

4．内生涯与外生涯的关系

我们知道了内生涯和外生涯都包括哪些内容，接下来就是它们之间的关系。弄明白它们之间的关系，就可以导出如何规划自己的职业发展与方向了。

内生涯和外生涯包含的一些东西，既可能是企业选择你的依据，也可能是你选择新职业时的目标，影响你的职业选择。比如你在 22~24 岁时可能更看重知识、技能、工作经验的积累，积累到一定程度，你的职务目标、薪酬就会自然上升，工作内容也可能发生变化。而你的职务、工作内容，也会影响到你积累什么样的知识和技能。

举个我自己的例子。我 2005 年开始做软件开发，做了两三年，积累了 C++、Windows 开发、网络编程、MFC、WTL、Windows CE、多媒体等方面的知识和技能，2008 年时开始带团队，工作内容分成开发和管理两部分，慢慢积累了项目管理和团队管理方面的知识和技能。在这个变化过程中，经济收入也发生了变化。当我再次选择职业时，我的知识、技能、曾经的工作经验、薪酬福利等，共同决定了我能找到什么样的工作：别的企业会看我的知识、技能、经历，我也会本着个人商业价值可持续发展的角度去选择职业机会。

OK，现在可以来说明内生涯和外生涯的关系了：

- 内生涯决定外生涯。
- 外生涯拉动内生涯。

古代的“学而优则仕”，是内生涯决定外生涯的一种典型情况。其实类似的情况我们身边也有很多，普通的开发工程师知识、技能、经验修炼到一定程度，就可以做架构师、技术专家，这也是典型的内生涯决定外生涯。

当外生涯高于内生涯时，虽然有压力，但也会促使你提升自己的知识、技能等，使内生涯与外生涯匹配，最终内生涯超越外生涯，可以进一步在组织内或组织外发展，获得更高的外生涯。这就是外生涯拉动内生涯的一种表现。

说到这里我们再展开一下。当内生涯略高于外生涯时，工作会驾轻就熟，容易出成绩，感到轻松舒适。当内生涯高过外生涯一大截时，个人就会觉得怀才不遇，想谋求更好的发展，如果长时间处于失配状态，跳槽指数就会增高。当内生涯低于外生涯时，工作会感到吃力，人容易焦虑，需要不断提升自己。如果不能有效提升，就可能会被剥夺外生涯。

高薪的谋划之道

因为内生涯决定外生涯，所以，程序员要想获取高薪，最根本的策略是修炼内功，发挥自己的性格优势，挖掘自己的职业兴趣，找到适合自己的职业，发挥天赋，不断提升知识、技能，让自己的商业价值不断爬升。

而一个人的商业价值能否提现出来，和所处平台又有非常大的关系。当你在一个好的宏观环境里，在一个前景光明的产业里，在一个处于上升期的企业里时，你的价值很容易就能体现出来，你能获取到的回报（外生涯）也会超越大多数人。所以，除了修炼内功，还要懂得如何寻找职业机会，让自己有用武之地。这就是程序员谋取高薪的指导性原则。

下面我们展开来讲如何修炼内功及如何寻找职业机会。

1．修炼内功

内功的修炼，其实又分为三部分：

- 发挥性格优势。
- 挖掘职业兴趣。
- 积累知识和技能。

假如你现在已经是软件开发工程师，那么就可以略过前两步了。

假如你虽然是软件开发工程师，但不确定是否适合做下去，那么可以看看“如何快速定位自己热爱的工作”一节，肯定会有帮助。

假如你是还没入职场的小鲜肉，可以来找我聊聊，我们可以一起来看看你是否适合做程序员。

说了这么多“假如”，现在要关注的就只有第三点——积累知识和技能了。

积累知识和技能有两个原则：

- 职业目标相关性。
- 持续性。

职业目标相关性

这一点很容易理解，假如你就想做 iOS 应用开发，那么你学 C#估计就没什么用，学 MFC 就更没用了。一样知识，一种技能，只有它和你的目标相关时才是有用的。没用的知识对你来讲，再多都是枉然，假如一种知识不能落到应用上，那么它就不是知识。

技能其实又分为两类：

- 专业技能。
- 通用技能。

专业技能是对某种专业知识的应用能力，与特定职业相关。比如你能用 MFC 在 Windows 下开发客户端应用，现在你要去做 Android 开发，那么用处也不大。

通用技能是可迁移的，也就是你会做的事。比如你很会写 PPT，这种技能到哪里都用得上；比如你很善于沟通，总能与别人达成一致；比如你善于当众演讲；比如你的社交能力强；比如你很会指导别人；比如游泳……这些都是通用的技能，可以在不同的工作中广泛应用。

通用技能对于你能否找到理想工作至关重要。

我们从程序员的世界来看，那些很牛的人，比如马克·扎克伯格，一开始也是程序员，后来创立 Facebook，他的创新、谈判、指导、说服等通用能力一定很强。另外他还会中文（语言技能也是通用技能，和驾驶一样）。还有雷军，写了 10 年程序，现在在做什么呢？如果他只会使用 C 语言开发，能做到这样吗？

所以，我们在积累技能时，既要立足于现在的职业，强化职业相关的专业技能，比如你是做 Web 前端的，HTMP、CSS、JavaScript、各种 JS 框架（比如 JQuery、AngularJS 等）、前端框架如 Bootstrap，都可以玩得很熟，这是专业技能，必需的，只要是做前端就会用得上；又要留意专业技能之外的通用技能，通用技能是一个程序员的软实力，比如发现自己

的学习模式，培养自我学习的能力，与人沟通的能力，口头表达能力，写作能力，信息检索能力……

那么，问题来了，怎样才能知道一个职业都需要什么样的知识、技能呢？有三种途径：

- 企业内的岗位描述。
- 业内前辈访谈。
- 招聘网站的招聘信息。

一般的企业都会有岗位（职位）描述，说明这个职位的职责，需要的知识、技能。有的企业还有一条晋升通道，比如软件开发工程师会有初级、中级、高级、资深、专家等级别，每个级别的任职资格说明里通常都会有对技能水平的说明。这是我们可以接触到的第一手资料。比如我曾经任职过的公司就有这种说明，高级开发工程师会要求你有 C++、概要设计、文档、数据库调优、授课、指导、管理等知识和能力。

第二种途径是找一个同岗位的前辈聊一聊，他很可能会从和第一种途径不一样的视角，根据他的经验告诉你什么重要、什么不重要，该培养什么、不该培养什么。

第三种是非常有效的途径，不但可以弥补第一种途径的不足（有的公司会没有，有的公司会很简单而流于形式），还可以从中梳理出某个技术栈的发展态势。像智联招聘、拉钩、猎聘、大街网、51job 等网站都会有大量软件开发工程师的招聘信息，可以结合我们自己的技术方向，拟定关键字进行搜索，然后看看别的企业对某个岗位都有什么要求，通过不断分析，就可以列出一张知识、技能清单，拿着这个清单，就可以去有针对性地发展自己的技能，该自学的自学，该培训的培训，该参加开源项目的参加开源项目……

持续性

植物的顶芽优先生长而侧芽受抑制的现象，在植物学上称为顶端优势。为了维持顶端优势，可以人为干预植物的生长，比如一颗泡桐树，要想让它长得又高又直又粗，就需要不断砍掉树干上的侧枝。

在企业管理领域存在顶端优势现象，处在优势的实权部门会抑制其他相关职能部门的发展，处于权力顶端的人往往抑制着处于下端的职权，处于优势的核心产品也会抑制其他产品的发展壮大……

对于程序员个人的知识和技能积累来讲，通常也需要维持顶端优势。

一个软件开发工程师，在自己的知识图谱与技能树中，如果存在顶端优势现象，那么当别人问你擅长什么时，你就可以信心满满地回答出来。而如果你的知识和技能还能在企业内超越其他程序员，形成群体内的比较优势或者顶端优势，那么你的光芒一定照耀四方。

我承认，能做到第二步这种程度的人相对较少，所以，我们只讨论第一步：在自己的知识图谱与技能树中打造顶端优势。

毋庸讳言，软件开发工程师跳槽频率比大部分职业的从业者高一些，在不同的企业、不同的行业、为不同的用户开发不同的产品时，用到的知识和技能通常是不同的。这就会导致一种情形：什么都懂一点，什么都不精通，什么都能干一点，什么都干不专业。而知识越精深越有价值，技能越熟练产出率越高，现在及将来是专业主义时代（参见大前研一的《专业主义》），如果我们能沿着一个方向积累知识锻炼技能，就可以形成竞争优势，随着不断用心打磨，就会产生顶端优势，就越来越能解决问题，不可替代性就会越来越强，商业价值就越来越高，薪酬福利自然越来越好。

所以，工作一段时间之后，就要思考自己的职业目标，梳理自己的知识和技能，选择几样，着重培养，持续精进，形成优势。

2．寻找职业机会

修炼完内功，我们该来寻找外部机会了。

在向外看寻找机会时，有三个递进的层次需要注意：

- 行（产）业选择
- 企业选择
- 职位

我们一个一个来讲。

行业选择

现在软件已经成为支撑各行各业发展的服务，几乎每个行业都会用到软件。行业里的企业在使用软件服务时，要么买现成的，要么自己开发。现成的软件，比如 CRM、ERP、OA 等，很多行业里的公司都用，也一般都是采购来的。除采购软件外，还有一些行业的公司选择自己开发，这个时候就需要软件开发工程师，也就是程序员了。

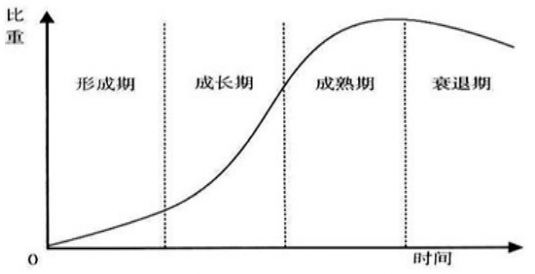
必须注意，不同行业的程序员的平均薪酬待遇差距很大。行业越有前景，个人的发展越好回报越好。越赚钱的行业，处在其中的程序员薪资水平越高。这是一般性规律，我们程序员在择业时也需要考虑。

以房地产行业为例，2003—2013 是中国房地产黄金十年，处在这个行业里的从业人员，有很多都赚得盆满钵满。以商品房销售为例，售楼小姐曾经是收入非常高的职业，而 2013 年后，房地产销售每况愈下，现在几乎是举步维艰。

再说说我曾经待过的电信行业，1995—2012 年是黄金阶段，其中 1997—2005 年，是固定电话业务和宽带业务大发展的时期，国内产生了华为、中兴、西安大唐电信、普天、UT 斯达康、烽火等知名企业。我的前辈们说，1999、2000 年左右，奖金比工资多，出差都是飞机。而我 2002 年加入某电信公司后不久，固话和程控交换业务开始走下坡路（2002 年之后移动通信大发展）。所以，你进入一个产业的时机非常重要。

那么，我们该怎样选择行业？

先看一张图：



从产业曲线图中可以看出，一般产业都有形成、成长、成熟、衰退四个时期。我们在选择一个产业时，成长（发展）期进入是最好的，成熟期也可以，衰退期就要慎重考虑了，除非你已无太多要求，只想随便干两年退休。

所以，作为程序员，不能只盯着技术，还要看行业大势。“女怕嫁错郎，男怕入错行”这种老话是很有道理的。

2015 年，最有发展前景的行业是互联网、金融、医疗、教育、新能源、智慧产业、高端制造等。而一些传统行业，比如煤炭、电力、房地产、石油，都在走下坡路。再说下房地产，别看那些楼盘死撑着不降价，其实一个月也不见得卖出一套房去，很多房地产厂商想跑路都跑不掉，工地停工，薪资拖欠，各种新闻不断。

企业选择

与行业类似，企业也有投入、成长、成熟、衰退这样的发展周期。我们选择企业时也需要考虑目标企业的当前状况，是在快速成长还是正在衰退。对不太有冒险精神的程序员来讲，最好的进入时机是快速成长期，此时企业飞速扩张，各种机会很多，产品要不断迭代形成技术优势，对技术人员需求很大，职位上的晋升，技术上的积淀，都会有很多。对于想拿青春赌明天的程序员来讲，也可以在一个公司的投入期进入，如果这家公司能突围，

那作为初期的核心人员，回报是难以想象的。你只要看看阿里巴巴的“十八罗汉”就知道了。

选择朝阳行业，选择非衰退期的企业，这是寻找职业机会时必须要首先考虑的，只要你选对了行业、进对了企业，个人的成长和回报是早晚的事。

我们还要展开来说一下，行业是由若干从事同一类或相近性质产品生产的企业组成的，在这些企业当中，一定有龙头老大，一定有前三甲，一定有前五、前十，选择排名靠前的企业，一般来讲会更好一些。因为实力越强的企业，占有的资源越多，市场覆盖越广，盈利能力越强，现金流越好。企业盈利，企业现金充裕，员工收益自然也大。

可我为什么说“一般”呢？因为一个行业还有细分，还存在一些垄断细分领域的企业，虽然在整个行业中综合实力排名不靠前，但因为卡位好，你想干那个领域的事就绕不开它，所以它也能活得很滋润。选择这样的企业，也相当不错。

职位

选对了行业、企业，接下来就是选择企业内的职能了。

前面我们说互联网行业很好，金融行业很好，互联网金融也很好，你进了这些行业中的某家企业，也不一定有什么大发展。因为这里面还有职能之分。比如你在 Camera 360 做前台，在网易做行政专员，在腾讯做保洁，是不是回报没那么高呢？

在一个企业里，一定是创造价值最多、距离核心价值链最近的职位的员工拿到的薪水最多。

有人说对于程序员来讲，根本没有选择，因为到哪个企业里都是做开发啊。

其实不然，还是有得选的。

比如你到一个房地产公司去做网站前端开发，你觉得怎么样？房地产公司最重要的部门是什么？肯定不是维护网站的技术部啊，你作为一个软件开发工程师，很可能受到的尊重、拿到的薪水远不如销售人员。

对互联网公司来讲，最重要的是产品，那就对应有两个职位——产品经理和软件开发——很受重视。所以你到这样的公司里去做软件开发，就和到房地产公司感觉不一样。

再说说我们前面提到的华为，华为内部有核心网、终端、大数据等不同的部门，你觉得哪块的软件开发待遇更好？我猜是大数据和终端。

现在再来想，是不是有得选呢？

补充一点，其实我们在选择行业和企业时，还要考虑宏观环境里的地域因素。比如你在西安，互联网氛围就不太好；比如你在郴州，整个软件行业就很差……所以有时为了更

好的发展，有些程序员会选择到机会更多的城市，比如北京、上海、深圳等。而如果你不想换城市（像笔者一样），有时就得妥协喽。

没有一滴水分的总结

总结归纳要点如下：

- 个人的商业价值体现在知识、技能、经历、天赋、人脉等方面。
- 宏观环境、产业、组织、职业、家庭等要素的综合会影响职业选择。
- 知识、技能、天赋、经验等属于内生涯，是你的内在质量，没人可以剥夺。
- 职务、薪酬福利、工作环境、工作内容等属于外生涯。
- 内生涯决定外生涯，外生涯可以拉动内生涯。
- 通过对知识、专业技能、通用技能等的不断积累，形成顶端优势，可以获取更好的外生涯目标。
- 累积知识和技能时，注意职业目标相关性和持续性。
- 要想获得高回报，选择前景好的行业中处于投入期、成长期的企业，在企业内选择靠近核心价值链的职位。

当我们谈论跳槽时在谈论什么

2016 年 3 月 9 日，我的微信上突然收到一条消息：

“公司给涨了几千元工资，且答应让我自己完成公司的 DLP 的网络驱动……终于能锻

炼了。”

我一看，是原来在微信上向我咨询过职业选择问题的一个朋友。

当时他在微信里说，他喜欢做底层开发，可所在公司安排他做应用开发，他觉得没意思，学不到什么东西，担心将来没竞争力，想要跳槽去做自己喜欢的事情。

当时我考虑到他学历不是太高，劝他先工作几年积累经验，用经验和能力洗掉学历对自己的影响之后再跳槽。我的建议出于现实情况，是比较保守的。他犹豫着，觉得非离职不可。

后来他提到可以跟公司谈一下，因为公司也有一些 Windows 驱动开发工作。我建议他试试。

再后来，一晃两个月过去了，春节过完了，在跳槽季我收到了他的信息。由此我知道，他成功地在组织内实现了转型。这是最好的结果：既做了自己喜欢的事，又没付出什么成本，还涨薪了。

这位朋友的经历，让我想起这次要聊的话题：跳槽、换工作与职业转型。本节大概会说三个方面内容：

- 职业、转型、跳槽、行业等概念。
- 职业转换的分类。
- 成本。

职业、跳槽与转型的概念

先理清两个概念：职业；转型。

关于职业，有一个最简单的定义：

职业=行业*职能

行业是由若干个经济活动性质相同或相近的组织所构成的，比如软件行业，就是由做各种软件的公司构成的。

职能，通常牵涉到对业务活动的细分与归类，以职位（岗位）的形式体现，指一个职位所承担的任务、作用和功能。以软件企业为例，Windows 驱动开发工程师，就是一个明确的职位，Android 应用开发工程师，也是一个明确的职位。

我们通常说的工作就有职业的含义，只不过它是一种模糊的说法，多数时候指代某个公司某个岗位。

明白了行业和职能的含义，就可以来讨论转型了。当我们换工作时，就牵涉到“转型”这个概念。不是所有的换工作都是转型，只有更换了行业或职能才能叫作转型，两个都没换而只换了公司的，叫跳槽。比如你从软件行业转换到了快消品行业、金融行业，就算是转型了；你从 Android 应用开发工程师转换为产品经理或售前技术支持，也算是转型；而你从通信软件公司 A 的信令开发工程师换到通信软件公司 B 的信令开发工程师，就是跳槽，不是转型。

现在行业、职能、职业、转型、跳槽这几个概念都清楚了。

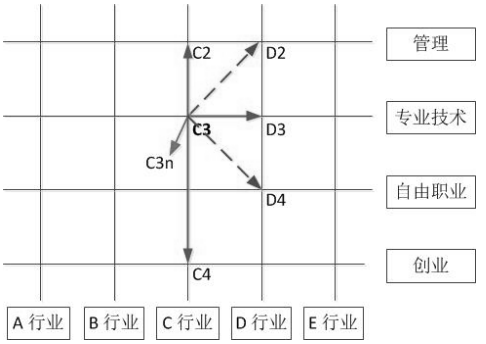
转型的分类

我在“大龄程序员的未来在何方”一节中提过四大职能取向：

- 管理者
- 专业技术者
- 自由职业者
- 创业者

如果你更换工作时，产生了上述职能取向上的变化（比如从软件开发工程师这种专业技术者转到创业者），那就明确无疑是转型了，而且是大的转型。

我画了一幅图，以职能取向为纵轴，以行业为横轴，可以清楚地说明转型的概念：



如该图所示，当前职业为 C3，在 C 行业内从事专业技术工作。C3n 与 C3 行业、职能都一样，公司不同；C2、C4 与 C3 职能不同行业相同；D2、D4 与 C3 的行业不同职能也不同；D3 与 C3 行业不同职能相同。

那么图中标注的职业转换就有四种模式：

C3→C3*n*，同行业内换一家公司原职能领域发展，比如你从酷派的 Android Framework 工程师跳到华为去做 Android Framework 工程师，就属于这种。

C3→C2 或 C3→C4，行业不换，职能转变。比如一位女士原来是在医疗行业的一家医院做护士，过了一阵做护士长了，就属于这种；一个 iOS 应用开发工程师跳到另一家公司做项目经理，也属于这种。

C3→D3，职能不变，行业转换。比如一位程序员原来在通信行业做网上营业厅的开发，现在转到教育行业做直播教学类的软件，就属于这种。

C3→D2 或 C3→D4，行业、职能都转换。比如有位朋友原来在运输行业开长途汽车，后来通过自学进入一家油服公司做测井软件开发，就属于这种；再比如有朋友在彩票行业做 APP 开发，搞累了回家包了 100 亩地种果树，也属于这种。

现在我们把换工作这件事说清楚了。接下来说更为重要的问题：成本。

成本，成本，成本

“成本”一词我们用滥了，百度百科是这么解释的：

成本是商品经济的价值范畴，是商品价值的组成部分。人们要进行生产经营活动或达到一定的目的，就必须耗费一定的资源，其所费资源的货币表现及其对象化称之为成本。并且随着商品经济的不断发展，成本概念的内涵和外延都处于不断的变化发展之中。

现在我们说的成本，换工作的成本，用的是“成本”一词的外延涵义。具体讲，有如下两种：

为达到一定目的而付出或应付出资源的价值，可用货币单位加以计量。我们叫它可见成本。

为达到一种目的而放弃另一种目的所牺牲的经济价值，通常叫作机会成本。

回到职业转换上来，我们还要考虑几种隐形的成本：在新的工作环境中与他人和组织建立信任获取晋升的成本；适应新工作环境的经济、生理、心理成本。

一个人可以自由地在行业和职能间转换，想干嘛干嘛，但跨行业时行业间不同的产业属性会直接影响转行的难度，跨职能时个人的性格、价值观、需要、兴趣、知识、技能、经历等是否与目标职能匹配也会在很大程度上影响发展。

现在来讨论职业转换可能发生的成本吧。可见成本大概有下面几种：

- 时间成本，比如你计划离职后三个月找到工作，你月薪 2 万元，那么成本是 6 万元。

- 为目标职业准备知识、技能的成本，比如你从 Web 前端转做 Android 驱动开发，可能要参加培训，培训费用 1.6 万元；比如你想从移动开发工程师跳到另一家公司做项目经理，可能需要考个 PMP 证书，考试费用 3300 元，培训费大概 4000 元。
- 试用期产生的各种成本，比如有公司试用期不交五险一金，三个月可能让你有 7000、8000 元的损失，另外试用期还可能存在试用期工资，3 万元月薪，3 个月试用期，发 80%工资，就是 1.8 万元。

比如你在华为，在拿到公司年终奖前走人了，可能损失 15 万元。

比如你在阿里，有价值 50 万元的股票，还没过锁定期你就离职了，可能将损失 50 万元。

机会成本就比较难计算了，大抵有下面几类：

- 现在公司准备给你加薪 40%并且你的职级将从 P3 升到 P4，而你跳槽到了另外一家公司，那家公司给你加薪 20%，那机会成本就是每月 20%薪资。
- 两个 Offer，一个薪水高加班多，一个薪水低工作轻松，你选了薪水高的，但周期性的生病花大笔医疗费，计算下来反倒是薪水低的那份工作收益更高。
- 两个 Offer，一个是继续做软件开发工程师，另一个是做分公司经理，但薪水比软件开发工程师低 5000。你选了分公司经理，机会成本就是每月 5000 元。如果你很适合做管理，可能干上两年回报就超过另一家的软件开发岗位了。但如果你后来发现自己对事务性的管理工作很厌烦，干不下去了再转回来，成本就更高了。

隐形成本因人而异，就更难计算了。

按一般的经验来计算职业转换成本，由低到高应该是下面的顺序：

- C3→C3*n*，行业不变，职能不变。
- C3→D3，行业变，职能不变。
- C3→C2 或 C3→C4，行业不变，职能变。
- C3→D2 或 C3→D4，行业、职能都变。

关于 2、3 的排序，我是从“一个人做与自己性格、兴趣、价值观等符合的工作时更自然更容易投入更可能快速发展”这一点来考量的。行业转变职能不变（C3→D3），成本在于学习新行业的知识、技能等东西，这是相对容易的；而职能转变，则牵涉到与个人性格、兴趣等是否合拍，不合拍的话转换就常常会有很大的失败概率，比如一个研究型的算法工程师去做部门经理，有 90%失败的可能性。

根据上面的顺序，“行业不变，职能不变”成本是最低的，假如你对自己大的职能取向没什么意见，那么这就是普通的跳槽行为，没什么难度也没什么好考虑的。付出的成本基

本上就是试用期工资、找工作周期折算出的工资、新公司建立信任付出的隐形成本。

如果你对行业不满，对职能也不满，那么一切从零开始吧，跨过“转行穷三年”的坎，重生就在下一个街角等着你。

排除两种极端的，剩下的就是大多数人的选择了：换行业或换职能。这种难度居中，成本也不太大，因为你总有一方面的积累作为你的价值陪着你。至于成本，就根据前面说的那几种，计算一下就知道了。要特别说明的是，这里面其实有一条近乎零成本的转型路径：组织内转型。这也就是本文一开始提到的那种类型了。简书签约作者“彭小六”，原来是一个程序员，后来在同一家公司内成功转型为产品经理，就属于这种。

还有一点值得注意，现在很多公司其实是跨行业的，比如阿里，就有电商、音乐、电影、云计算等，在组织内实现转行也是很有可能的。如果有组织内转型的机会，优先考虑，积极去试，因为可见成本和隐形成本都最低。

不是结束的结束

本节想通过介绍职业、转型等概念，引入职业转型的分类，供大家在换工作时参考去计算成本，以便做出慎重选择。

打听别人工资的 7 个话题，让你薪水更高

你打听过同事的工资吗？你的同事问过你的工资吗？你怎么看待这件事呢？这就是下面要聊的话题，包括但不限于：

- 同工不同酬现象。
- 打探工资的常见途径。
- 你为什么会打听别人的工资。
- 职业的本质。
- 商业价值与工资的本质。
- 如何面对同事工资比自己高。
- 怎样凸显自己的商业价值。

同工不同酬

五号是公司发工资的日子，阿巧心情超好，就等待着下午工资到账那个短信。可是等来短信后没多久，她无意中看到了同事阿洁的工资短信，一下子整个人都不好了！同等学历，在同一个部门同一个小组，做同样的工作，前后脚进公司，阿洁甚至做得不如她好，凭什么工资比自己高那么多？

其实，同工不同酬到处可见，并非阿巧一人所遇。当别人遇见时我们都能表示理解，但发生在自己头上时，情况立马变了……

虽然禁止相互打听工资是潜规则，但还是有很多途径可以打听别人的工资，也有很多人热衷于干这件事。我顶讨厌打听别人工资，也顶讨厌别人打听我的工资，当然也讨厌别人把我作为消息来源到我这里打听其他人的工资。然而思考一下下面的问题。

你为什么会打听别人的工资

一般人打听别人工资，表面看无非是这么几种原因：

- 担心自己被不正常对待。
- 想确认自己的优越感。
- 建立信息壁垒，换取其他资源或充当八卦谈资。

抛开后面那种，前面两种其实更深层次的原因是：对自己的价值没有清醒的认识，倾向于在比价中来定位。

说白了，就是习惯在比较中生活。从小我们就被告知邻居家的孩子怎么样，我们已经有一种习惯，总想着打败身边的人，最不济也不要被身边人打败。

当我们陷入“比较”的怪圈中后，就很少去考虑职业的本质到底是什么了。

职业的本质

职场人士本质上讲和商品差不多，你有价值人家才给你合适的价格。与商品不同的是，职场人士的价值是在通过劳动制造其他商品的过程中体现的，老板们“买你”的时候，其实并不确定你值不值那个钱——他只能根据你过去体现出来的价值来出价。而这种做法其实是不那么科学和合理的……

简单说，作为职场人士，你首先就要给别人创造价值，你的价值，是通过为别人创造

价值体现的。一言以蔽之，职业的本质是价值交换。再说得现实点，作为企业主，给你的钱一定会少于你能创造的价值。

所以，理解并接纳这一点，就能正确地对待“自己到底值多少钱”这件事。

然而，你有价值，却并不一定能拿到高工资。

商业价值与工资的本质

个人的（商业）价值，体现在其知识、技能、天赋、经历、人脉 5 个方面，这是个人内在的价值，在某一时期内是相对稳定的。

企业给员工定工资时，基于前面提到的价值交换原理。简单说，就是你对我有价值，我对你有价值，咱俩具有匹配性，根据匹配性定工资。所以，从这一点讲，一个人能拿多少工资，很多时候不是由其泛泛的价值决定的，而是由企业需要的那个价值点决定的。

你上晓天文下知地理，古通三皇五帝今明互联网思维，可我现在就是要找个搬砖的，所以，我只能给你个搬砖的钱，对不对？

到这里我们就明白了工资的本质：工资是企业为自己需要的那个价值点而支付员工的劳动报酬。

认清这一点，作为员工，就能理解：薪酬不代表个人的整体价值，要想通过薪酬最大化体现自己的价值，就要寻找与自己商业价值最匹配的环境。这样在通过职业实现价值的道路上，就会少走很多弯路。

如何看待“同事的工资比自己高”

别人的工资比你低你肯定能接受。单说阿巧的情况——同事的工资比她高。

理解了职业的本质、工资的本质，这个问题其实已经不是问题了。

阿巧认为阿洁不如她，但在企业里，判断一个员工相对于企业的价值，是由这个员工的经理、公司的管理部门等决定的。他们的标准，往往和阿巧看阿洁的标准不一样。这就是薪资评定的相对论。

当阿巧发现阿洁工资比自己高 500 元时，气愤、不平，是可以理解的。但情绪过后，还需要想几个问题：

- 别人的薪水，和自己有什么关系。

- 自己相比别人，有哪些不同的价值。
- 公司更需要哪种价值。

如果没想明白就义愤填膺以请辞来要挟涨工资，很可能领导高高兴兴给你俩字：同意。

如果你想明白了——自己的商业价值与公司的需求匹配度低，那就很好办了，找一个能更好地发挥你的价值的地方，薪水必然高啊。

假如你认为目前你的商业价值和公司已经比较匹配，可薪水还是偏低，那我还要说另外一个商品属性：商品价格会回归价值本身。

具体是什么意思呢？正常情况下，商品的价格体现其价值。在较长的时间周期内观察，商品价格总是围绕其价值上下波动。

对于职场人士来讲，薪资也最终会与其商业价值匹配。

所以，你无须不平衡，一时的差错（比如入职薪水没谈好、进入企业那年整体经济状况低迷等）导致的薪酬与个人价值失配并不可怕，可怕的是失衡的心态及由此产生的错误行为。只要你正确认识到这一点，想办法凸显自己的商业价值，用不了几年，就会以“价值补偿”的方式修复最初的问题。

那么，问题就来了，如何凸显自己的商业价值？

如何凸显自己的商业价值

“程序员如何谋划出月薪 3 万元”一节用很大篇幅系统地讲述了如何识别、培养自己的商业价值，以及如何寻找与自己商业价值匹配的环境。

所以，这里只用一句话简单地概括一下：别人的难题，就是你的价值。假如你能通过自己的能力，快速帮企业解决问题，那么你的价值就凸显出来了。

结语

到现在为止先后谈到同工不同酬的背后原因、职业的本质、工资的本质、个人价值发现与凸显等内容，最后用一句话来概括一下吧：

个人的薪酬是由其商业价值与企业需求的匹配程度决定的。如果想获取更高薪酬，就要努力提高自己的商业价值，并且选择与自己整体价值契合度较高的环境。

为何公司愿花更多钱从外面招人

要是你有心留意，可能会发现这样的现象：

公司新招来的程序员，经验、技能、经历、学历都和你差不多，薪水却比一直待在公司的你高。

为什么公司愿意花更多的钱招聘新人也不给老员工加薪？

这是很多人都遇到过的问题。怎么解释呢？

奖励工资的必要性

强烈推荐一本书——《牛奶可乐经济学》。这本书第 3 章，“职场的奥秘”，里面就有一个问题：

为了吸引到想要的高素质员工，或为了吸引到足够数量的员工，为什么雇主有时会提供高出标准的工资？

书里的解释很有道理，直接照搬了：

竞争性劳动力市场理论认为，为吸引到合适的劳动力，雇主只需提供达到必要水平的工资即可。可在不少公司，每一个空缺的职位，都会有无数高素质申请者竞争。难道这些公司不能支付较少的工资以赚取更高利润吗？

一个可能的原因是，提供奖励工资有助于确保员工的诚实行为。只拿到市场价格工资的员工，没什么道理担心失业。毕竟，在竞争充分的劳动力市场，符合市场价格的工作岗位数量基本上是稳定的。可有着奖励工资的工作岗位，却不是随随便便就能找到的。所以，有幸能得到这类工作岗位的员工，有着强烈的经济动机，尽其所能地保住这一岗位。尤其是，相较于拿市场工资的员工，拿奖励工资的员工消极怠工的可能性很小。倘若公司能通过这种方式缓解怠工现象，那么，哪怕需要支付奖励工资，也仍能赚取利润。

这就是《牛奶可乐经济学》里的解释。我觉得很有道理。

我们本节开始的问题，其实还有一点特殊性：为什么不直接用老员工呢？

隐性成本

大多数公司招聘时，采取的是经验主义与行为招聘相结合的方法。这种方法基于两个假设：

- 一个人过去或现在的行为可以预测这个人将来的行为。
- 招募已熟练掌握某项技能的人比选择当下还不懂该技能的人成本低。

所以，当一个公司要开发新产品时，往往会从外部招聘契合产品技术方案的员工。以软件产品为例，就是到兄弟公司去挖人。

通常公司领导会认为时间成本、试错成本、机会成本对产品的成败影响很大。所以，快速找一个门儿清的“老炮儿”和几个熟练使用所需技术的程序员是最保险的选择。这也是上面两个原则之所以存在的基础。所以，根据上面的原则，即便公司里现在有几个人工作量不饱和，让他们现学现卖也是不划算的。

就找那些已熟练掌握某种知识、技术、框架的程序员！

所以，你看到软件公司的招聘信息，都有“精通 C++”“精通 Java”“精通 Hadoop”“精通 Spark”“精通图像处理算法”“熟悉神经网络和深度学习”之类非常具体的要求。

而拥有这些知识、技能和经验的程序员，没事为什么要跳槽？人家最起码拿着市场平均工资，你要是不给出一大截奖励工资来，人家完全没有跳槽的动力。起码提升 50%才能让一个正干得好好的程序员动心。有时甚至需要翻倍。

话说回来，对公司领导来讲，本公司熟悉某种技术的程序员面临的状况也是类似的，只要公司保持他现在的薪资水平和满意度，他就不会随随便便跳槽——得有公司给他高出现在 30%~50%甚至翻倍的薪水啊。在一个稳定的行业里，这种概率不太大，除非本领域有新的土豪玩家入场。

所以，就出现了本节开始说的现象：公司居然会花比现有同级别程序员薪水高一截的代码从外部招人，这种现象，就是由几点原因导致的：

- 不给更多钱不能让外部程序员产生跳槽动力。
- 任用缺乏所需技能的本公司程序员，时间成本、试错成本高。
- 任用缺乏所需技能的本公司程序员，机会成本高。
- 本公司的程序员，因为公司招募拥有不同技能的程序员而愤然跳槽的可能性很低。

培养自己的稀缺性

如果你对前面提到的那种情形不满意，那就自己努力培养稀缺性吧。按我的理解，程序员的稀缺性来自以下几个方面：

- 在某一具有前瞻性的技术上占据先机，先跑一步，卡好位。举个例子，Go、Rust、Scala、Swift 等语言才出世时如果两三周你就门儿清了，那就很有优势，为此你可能需要从 Beta 版、Alpha 版就及时跟进。
- 在某一应用相对广泛的技术上秒杀同行，越资深越有价值，比如 Android 开发者和 iOS 开发者都很多，一抓一大把，但资深的并不太多，你兢兢业业干过五六年，技术阅历又对得起年限，绝对比刚干了一年的新人竞争力强得多。
- 在某一产业领域的积累秒杀同行，越资深越有价值。这是从懂行的角度来看的，你看阿里系创业的那些人，有一大半在做电商，你看腾讯系创业的人，有多少在搞即时通信，想想就明白了。

问答 | 学历差的程序员就该被虐吗

我曾时候到一封标记为“有问有答”（我微信订阅号程序视界的栏目）的邮件：

foruok 先生：

您好！今天终于鼓足勇气给您发邮件，看了您的程序视界，受益良多，也非常感谢您的分享和贡献。

一直以来有一个疑问和困惑在我脑海里盘旋不去——程序员该需要好的文凭和学历吗？我本人专科生，有三年工作经验。在找工作时候也因为学历的问题吃了很多亏，碰了很多灰，总是不能进入自己心仪的公司，甚至在投简历时候因为学历的问题都不能获取面试机会。

不知道您是怎么看待这件事情的？

急盼回复。

万分感谢。

我能从文字里感受到这位不具名的朋友的困惑、焦虑和不甘。看到邮件后面写着“急盼回复”四个字，我把我的想法第一时间回复给他。

我想起东野圭吾的《解忧杂货店》的开篇，翔太、敦也、幸平在“浪矢杂货店”回复“月兔”来信的情景。有时我真不知自己的想法对于别人来讲是否合适，也不知它能否给别人带来安慰，但我想，这样的交互过程，诚如“月兔”所说，其本身也应当是有意义的。除此之外，我还希望，我的回复不要给等待的人带来不便。

由于时间仓促和心情急切，15 号晚上的回复，可能有些地方行文不太通顺，因此我又抽空整理了一下，放在下面：

Hello：

首先感谢信任。

在我们社会的观念里，文凭和学历的问题是实际存在的，是我们当下不可回避的。你自己也感受到了，很多公司在招聘时都会把学历作为一道基本的门槛。我们的社会讲究出身，学历是出身之一种，对学历的重视，在相当长的时间内都不会有大的改观。甚至随着本科教育普及化，会有更多的单位更倾向于招聘具有好大学学历的学生。

在大多数人的观念里，学历和能力成正比，而且也基本符合实际情况。当然，个体总有超越一般规律的可能，这也是逆袭、奇迹令人兴奋的原因。但是，要超越别人通过学历有意无意对你施加的限制，就要付出更多的努力，而不是抱怨。

我个人认为学历能从一个方面反映个人的能力，但不能完全代表能力。企业以学历为门槛，更多地是看重学历背后体现出的自律及为目标所付出的努力。

按照我的经验，对于工作经历在 5 年以内的求职者，学历都是会被参考的一个很重要的因素。有些公司甚至看到非本科（或非一本）的简历直接 PASS 掉，这是常有的事。一个职场人，工作过了 5 年或者更久，工作经历就会慢慢洗掉学历的影响。招聘方也会更看重你做过什么，做得怎么样。

我原来有一个同事，就是专科学历，工作也比较靠谱，甚至比大部分本科学历的程序员做得都好，后来公司让他担任开发团队的 Leader。那时他工作了将近十年了。

我还认识一个朋友，也是大专毕业，年龄和我差不多，在一家面向石油行业提供软件解决方案的公司从事开发工作，现在是部门经理。2015 年年初时他在攻读西安交大的在职硕士。

曾有一个高中毕业的程序员带着自己的作品到我们公司面试，我觉得他很有钻研精神，但最终没有录用他。我的上司也不同意录用他。

前几天和一个朋友在微信上聊天，他在深圳，也遇到了学历方面的问题，说是投了 100 多家公司，都没人理他——他初中毕业，上的软件开发培训课程——还好，他现在已经工

作了。

在没有靠谱的鉴别应聘者能力的工具时，学历是一种约定俗成的、简便的筛选标准。这是客观存在的情况，是我们很难回避的。这两年华为在某些学校校招只招研究生，还有很多知名的企业也只招募研究生，这都是通过学历设置门槛的例子。

当然在这种情况下，也不能说个体就没有超越的机会。**无论在任何情景下，个体都有选择的机会和自由，可以按照自己的想法走自己的路，通过自己的选择和努力来成就自己，让别人对你的偏见不能在你身上证实**。

所以，作为个体，如果你不能也不甘接受外界对学历的偏见以及这种偏见对你职业发展的影响，就要努力冲破这种无形的藩篱。

有两种做法：

- 考取名校本科、硕士学位，抹掉专科的教育经历。这样的转型是大家喜闻乐见的，别人也容易因为你的积极进取而高看你一眼。当你从学校出来时，就有了“重生”的机会。
- 在能找到的职业环境里，努力修炼技术，在某个方向上建立优势，等待机会，进入你心仪的公司。

不管选择哪种做法，都需要付出很多努力，也都有很多成功的先例。以 IBM 公司为例，最近几年校招都只招 211、985 院校的研究生。然而实际上，在公司内部，有一批普通大学本科毕业的员工，而且有的人还是经理，工作风生水起。当他们想要进入 IBM 之类的公司时，在一开始面临的问题和你一样，不过，他们采取了曲线策略，先在小公司锻炼自己，储备进入目标公司必需的技能，比如英语，比如数据挖掘，比如 Hadoop，然后利用社招的机会进去。

你的现在，是你过去的选择和行为造就的。而你的将来，则是你现在的选择决定的。不要抱怨，也不要灰心，找一条路，努力去实现自我，每一个街角都有美好的事物等着你，要相信自己。

因为我本身没有因为学历问题受到困扰，之前我没有仔细考虑过学历的问题，在收到这位朋友来信时才开始认真去对待这个现象。我意识到很多程序员的求职过程都被这个问题影响过。我在招募小伙伴时也曾与不少高中毕业、专科、培训机构的学员聊过，他们面临和前面这位勇敢来问的朋友一样的问题，希望大家都能用自己的方式走出这种基于学历的偏见对自我职业发展的影响。

程序员这样优化简历，一投制胜

为什么你投 10 份简历，只有 1~2 家公司约你？或者为什么你每投一份简历都能获得面试机会？

最根本的原因，就是一方在汲汲渴求，而恰恰另一方呈现出的关键点让其怦然心动。**求者心中有所想，而应者恰恰展现了求者所想的那一面**。这就是个中奥妙。

程序员在找工作时，在一开始有三件事情会对能否获得面试机会至关重要：

- 知识、技能、经历梳理。
- 确立求职目标。
- 简历优化。

知识、技能、经历梳理

知识、技能、经历，这都是一个人能体现出来的商业价值。一家企业招募某个人，一定是因为这个人可以帮助企业在某方面实现价值。而且，正常情况下，个人的贡献一定要大于企业为这个人负担的各种成本（薪水、社保、公积金、个税等）。

所以，作为程序员，我们一定要清楚自己的价值在哪里。个人的商业价值，可以通过下面五大要素分析出来：

知识、技能、经历、天赋、人脉

我们在招聘网站上填写简历时，内容多数情况下就是前三个要素，模板都是差不多的，填下来千人一面。

在最开始的时候，不建议直接到招聘网站上填写简历。**强烈建议先用 Word 或 MarkdownPad 来整理记录你认为你具备的所有有价值的知识、技能、经历**，不论大小，统统记录下来。这是后续优化简历的基础，也是确立求职目标的基础。

1．知识与技能

有必要说一下知识和技能的区别，这是很多人常常混淆的。

知识可以通过语言文字、语音、视频等进行传授，比如像 C++、Java、数学、物理、Qt、Android、设计模式、网络协议等都是知识。

技能是指按照某种规则应用知识和经验完成某种任务的能力。比如使用 Qt 开发桌面客户端软件就是一种技能，使用 Java 和 Android 界面类库开发 APP 也是一种技能。

我的知识大概有这些：

C、C++、Java、Scala、Python、Qt、MFC、WTL、QML、Qt Quick、JavaScript、HTML、CSS、Lua、MySQL、MongoDB、XML、Json、Win32 SDK、Node.js、AngularJS、ffmpeg、VLC、DirectShow、Android、Objective-C、HTTP、P2P、RTMP、RTSP、HLS、P2P、socket、UML、软件开发模型（瀑布、迭代、Scrum 等）、项目管理知识、团队管理知识、微信订阅号管理。

知识会淡忘，一段时间不用就扔掉了，所以在你的知识图谱里，一定有一些是你经常使用的，比如我最熟悉的就是 C、C++和 Qt。

我们需要把自己最熟悉的三种知识标注出来，后面会派上用场。

技能就是对知识的运用，所以一般来讲你有什么知识，就能找到一组对应的技能。比如我可能有下列技能：

- 使用 Qt 开发客户端软件。
- 使用 Qt 开发服务器软件。
- 使用 Java 开发 Android APP。
- 管理项目，制定项目计划，跟踪计划，控制项目进度。
- 团队管理与激励。
- 博客、微信订阅号等自媒体运营。

……

现在我们可以进行知识和技能的梳理工作了。

需要特别注意的是，每个人都有很多知识和技能，一定要找出你擅长的 2~3 种知识，2~3 种技能，这将是你求职时的重要参考。人只有使用最擅长的技能去做事情，才能达到最好的效果。

2．经历

知识和技能可以帮助我们创造商业价值，而知识和技能的积累过程本身也是有价值的。积累知识和技能的过程，就是经历。

程序员的（学习、工作、项目）经历具有非常独特的价值，在求职过程中往往会发挥非常重要的作用。比如你做过视频项目，那么找类似做视频的公司就很容易脱颖而出；如

果你做过图像处理相关的项目，进入美图秀秀之类的公司就相对容易……

在回顾项目经历时，关于你自己的那部分，一定要想明白并记录下来，从下面三点来挖掘你的亮点：

- 你负责的工作内容。
- 用到的知识、技能。
- 你对整个项目的贡献（最好可以量化）。

我的一个项目经历：

【互联网视频卡顿优化】

项目描述：

智能机顶盒上的视频聚合客户端，在播放视频时，高峰时段或热点视频，经常卡顿。针对此问题，进行优化，以使能够对用户提供流畅的观看体验。

业绩：

播放效果大幅度提升，卡顿投诉下降 80%。

职责：

- 作为项目经理，负责项目范围界定、进度跟踪与控制。
- 作为系统设计，选择技术方案，设计加速与优化算法，设计系统结构。
- 作为核心开发人员，负责客户端 http 基础类库的开发和服务端 http server（基于 Qt）的开发。

我没有列出前面所说三点对应的所有内容，个人亮点也没完全写出来，但你的心里一定要清楚，你发挥了什么关键作用，如果有攻克技术难点的经历和明显可以量化的业绩，一定要总结出来，面试时经常会被问到。

天赋和人脉其实也是个人非常重要的商业价值。但在程序员的简历中较难体现出来，在面谈中倒是有较多机会展现。

确立求职目标

不管是从大学走向社会的初次求职，还是在职场摸爬滚打了 *N* 年，找工作时都要忌讳一点：茫无目的，漫天撒网。

求职时，明确目标行业、企业、职位，有针对性地做准备，事半功倍。

1．职业延续性

跳一次槽换一个行业，跳一次槽换一条技术栈，这样极其不利于程序员的商业价值积累（知识、技能、业务等）。假如你不是第一次求职，就有必要考虑职业延续性。

假如你现在在金融行业，那么换工作时，最好还在行业内部。这样你的行业相关的经验就可以积累下来，慢慢形成优势。假如你现在做 iOS 开发，最好换工作时还找 iOS 相关的，这样你的 Objective-C，Swift，Cocoa 才能持续精进……

有一种情况另当别论：你发现自己不喜欢现在的行业和所用的技术。此时跳槽，就要找到新的方向（后面方法可以帮到你）。一旦你找到新的方向，转换过去之后，同样要考虑以后的延续性。每次都说自己不喜欢现在的，更喜欢另一个，这样捣腾几年之后，你很可能会发现，那些闷生不响持续深耕的小伙伴们居然一夜之间都闪闪发光了……

2．成就事件挖掘职业兴趣

在我们总结整理自己的工作、项目经历时，要特别留意那些让你特别有成就感的事件，其中隐藏你的职业兴趣，可以挖掘出你今后乐意从事的职业方向。

我们可以遵循 STAR 原则来回顾成就事件：

- S（Situation），背景情况，包括面临的障碍、限制或困难。
- T（Task），任务，目标，想完成的事情。
- A（Action），行动计划与步骤，如何克服障碍、达成目标。
- R（Result），对结果的描述，重点关注你取得了什么成就。

成就事件列出来后，可以按照下面两点来深入挖掘：

哪一个具体的点让你特别有成就感，比如“达成结果受到领导表彰”“独立克服某个技术难题”“用自己的某种技能帮到了某某某”“成功组织大家齐心协力达到目标”……

通过这样的分析，你就可能找到自己的最佳技能和工作中最在意的点在哪里，就可以根据它们来确立新的职业目标。

3．聚焦行业、企业、职业

聚焦的目的是缩小目标范围，节省时间和精力，深入研究分析，有针对性地对自己的商业价值进行优化组合，提高简历的吸引力，最后提升获取面试机会的概率。

程序员运用编程语言、技术框架、设计模式、算法等开发针对某个领域问题的软件，

软件必然和目标需求和业务密不可分，所以，程序员左手技术，右手业务，假如你对业务内容完全不感兴趣，很难想象你可以把软件做好。因此，当你有了目标产业及目标公司后，还要去了解这家公司做什么产品，是产品导向，项目导向，还是外包为主，选择那家你对它的业务范围感兴趣的公司，不感兴趣的果断筛掉。

基于延续性的考量，以往工作过的行业领域都需要认真对待，但不一定非要继续在之前的方向上做，换一换也可以。但有一个基本的原则，离夕阳产业和走下坡路的企业远点。

以我为例来分析一下。

在技术方面最擅长的语言是 C++，最擅长的框架是 Qt（出过《**Qt on Android 核心编程**》和《**Qt Quick 核心编程**》两本书），最擅长的是客户端软件开发。对 Android 开发有了解也有兴趣。新的职位最好和这两方面相关。

我锁定的第一个目标是高级软件开发工程师，语言是 C++。行业领域方面，互联网、智慧城市、企业服务等都可以。

还有，我之前都在小公司摸爬滚打，现在希望到比较大一点的公司里体验一下不同的工作氛围。

在锁定职位的过程中，还要考虑自己当前的劣势，以便在简历和面试过程中妥善应对。以我为例，如果找软件开发工作，就有两个不好的点：

- 最近一年没做什么开发工作，会被人质疑廉颇老矣尚能饭否。
- 最近几年都在做技术管理工作，别人可能会奇怪你为什么倒回去做开发。

这两点也会影响到目标职位的选择和匹配度，比如有的高级开发工程师职位要求有一定团队管理经验，那么和我的匹配度就高一些。

好了，我的聚焦过程已经完成了，接下来进入简历优化的实操过程。

简历优化实操

知道了自己有什么商业价值，弄清了想到什么样的行业、企业、岗位工作，接下来就该优化简历了。

简历优化有几个要点：

- 分析招聘信息，提取知识、技能、职责当中的关键词。
- 根据招聘信息里的关键词，筛选个人知识、技能、经历。
- 将匹配到的知识、技能、经历重新组织、呈现。

程序员的简历，一般包含下列内容：

- 基本资料（姓名、性别、年龄、婚否、电话、当前在职状态）。
- 求职意向（职位、工作地点）。
- 知识、技能。
- 自我评价。
- 教育经历。
- 工作经历。
- 项目经验。
- 附加信息（比如兴趣爱好、荣誉、博客、开源项目等）。

根据实际情况，有的简历可能没有自我评价、附加信息，有的可能没有可写的教育经历（比如高中生），有的可能没有工作经历和项目经验（如大学生等初次求职者）。

特别提一下程序员的一些个人品牌相关的附加信息（具体参看我的微信订阅号“程序视界”里编号 10080 的文章：这 8 种武器点亮程序员的个人品牌），比如技术博客、github 主页、参与的开源项目、自己做的 APP 等，都能非常好地展现程序员的知识、技能，能为你的简历加分，也能弥补面试时间短、信息交流不充分的问题。要知道，决定你能否通过简历关的，通常是另外一个程序员，而对聪慧内敛的程序员致敬的最好方式，就是“Show me the code”。

1．简历模板

很多程序员都使用招聘网站提供的简历模板，那么要在不同招聘网站上发布简历，可能要重复填写 5、6 次简历，非常耗时。我一般会有一份基础的 Word 格式的简历，根据它在不同的招聘网站创建不同的简历。

接下来介绍我的简历，非常简单，顺次包括下列 5 部分内容：

- 个人信息。
- 求职意向。
- 技能与评价。
- 工作经历。
- 项目经验。

我的简历的前三部分简单如下：

个人信息

基本信息

（包括性别、年龄、毕业院校、专业、学位、电话、邮箱等）

求职意向

高级软件开发工程师，西安。

技能与评价

7 年部门管理经验，丰富的项目、团队管理经验。

7 年嵌入式开发经验，在互联网电视机顶盒、车载娱乐系统、手持娱乐设备（MP3/MP4）等领域有成功经验。

丰富的软件系统架构设计经验。

熟悉常见的设计模式，有丰富的面向对象设计经验。

精通 C/C++，熟悉 Java，Shell，了解 Python，Lua，JavaScript 等。

可熟练在 Android、嵌入式 Linux、Windows CE、Windows、Linux 等平台下进行开发。

熟悉 Android/Qt(E)/MFC/WTL 等 GUI 框架。

熟悉 DirectShow/GStreamer/MPlayer/FFMPEG/VLC/Vitamio 等多媒体框架。

熟悉网络编程，熟悉各种流媒体协议（HTTP/HLS/rtmp/P2P/RTSP 等）。

博客：http://blog.csdn.net/foruok

Github：https://github.com/foruok

著有《Qt on Android 核心编程》和《Qt Quick 核心编程》

我工作年限长，挑最近的几段看：

工作经历

- 2014.12~2015.12，陕西 XX 网络科技有限公司

职位：技术总监。

职责：团队组建、管理、技术方向把握、项目管理、产品管理。

业绩：3 个月组建 10 人团队。

- 2014.11~2014.12，北京 XX 有限公司（西安）

职位：平台组高级软件开发工程师。

职责：跨平台软件开发。

业绩：解决了 Qt 在 Android 和 iOS 两个平台与原生 UI 叠加的问题。

- 2008.01~2014.09，西安 XX 软件科技有限公司

职位：CMC 部门经理。

职责：

1). 部门员工组织、领导、管理、激励，绩效考评。

2). 项目计划、实施、跟踪、管理。

3). 技术路线评估与选择。

4). 系统分析与设计。

5). 关键模块代码实现。

项目经验：

1). 2009 年至 2010 年，负责组建机顶盒开发团队与互联网视频搜索开发团队，并带领团队，了技术积累和产品探索，使得公司成功进入新的行业领域，完成公司的业务转型。

2). 2012—2013 年，负责组建智能机顶盒团队，完成机顶盒产品智能化转型。

3). 2012.07—今，主持智能机顶盒产品研发。

4). 2011.10—2012.06，主持第二代高清互联网机顶盒产品研发。

5). 2009.05—2012.01，负责第一代高清互联网机顶盒产品研发与改进。

好了，我简历的主体框架就是这样。接下来，我会根据我的目标职位——高级软件开发工程师——来找几条招聘信息，**通过分析招聘信息来优化简历的技能与评价、项目经历两个部分**。这两部分也是我们优化简历时的重点。

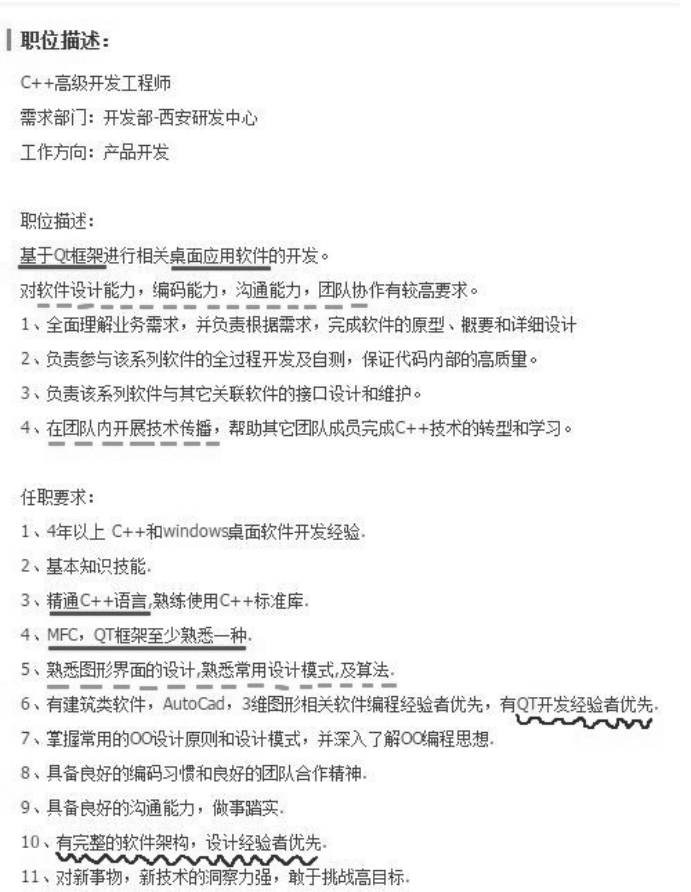
2．简历优化

大多数招聘需求是由我们的同行提供的，假如能从招聘信息里反推出拟定招聘需求的那个软件工程师或经理心中的关键词，那么你的简历优化就有针对性了。

下图是一个招聘需求：

直线标注出来的，是一些关键的基本要求，C++、Qt、Windows 桌面客户端，这些是硬性要求，是应聘该岗位必须要满足的要求。

虚线标注出来的是软性要求，不太容易直接量化和衡量，但面试官可能会在面试中通过交谈来考察。比如软件设计能力、技术传播能力、沟通能力、沟通与团队协作等，通常都是在面试中体现的，而编码能力，则可能会通过笔试或者面试中不断地询问技术细节来检验。



职位描述：

C++高级开发工程师

需求部门：开发部-西安研发中心

工作方向：产品开发

职位描述：

基于Qt框架进行相关桌面应用软件的开发。

对软件设计能力，编码能力，沟通能力，团队协作有较高要求。

1、全面理解业务需求，并负责根据需求，完成软件的原型、概要和详细设计

2、负责参与该系列软件的全过程开发及自测，保证代码内部的高质量。

3、负责该系列软件与其它关联软件的接口设计和维护。

4、在团队内开展技术传播，帮助其它团队成员完成C++技术的转型和学习。

任职要求：

1、4年以上 C++和windows桌面软件开发经验.

2、基本知识技能.

3、精通C++语言,熟练使用C++标准库.

4、MFC，QT框架至少熟悉一种.

5、熟悉图形界面的设计,熟悉常用设计模式,及算法.

6、有建筑类软件，AutoCad，3维图形相关软件编程经验者优先，有QT开发经验者优先.

7、掌握常用的OO设计原则和设计模式，并深入了解OO编程思想.

8、具备良好的编码习惯和良好的团队合作精神.

9、具备良好的沟通能力，做事踏实.

10、有完整的软件架构，设计经验者优先.

11、对新事物，新技术的洞察力强，敢于挑战高目标.

波浪线标注出来的，是优选条件，在有多个候选人的情形下，你具备了这些要素，就会被优先考虑。

一般的企业处理软件开发工程师的简历，是 HR 先挑，然后是技术人员（程序员或经理）再次筛选，通过技术人员的筛选后，就会通知笔试或者面试。

HR 拿到招聘需求后，会和技术人员沟通要关注的关键点，然后根据这些关键点挑选合适的简历。通常在 HR 眼里，关键点就是关键词，所以，我们**修改简历时，尽可能使用招聘信息里出现的字眼，便于 HR 识别**。HR 看一份简历，多则一分钟，少则 20~30 秒。

技术人员筛选简历，通常比 HR 要细一些，大致分两步，先筛关键词，然后看项目经验里是否体现出了他们发布的岗位需要的技能和软实力。**关键词匹配不上，PASS，30 秒不到；能匹配上，继续看项目经验**，整个过程可能会持续几分钟。

在看项目经验时，一方面看应聘者在项目中是否用到了将来需要的相关技能，另一方面看具体项目内容，做行业、产品、业务相关性比对，如果应聘者做的产品与招聘方要做的相关性高，就会被优先考虑。

通过对简历筛选流程的了解，就可以知道关键词的重要性了。接下来我们就要依据从招聘信息中提取的关键点来修改简历了。

在修改简历时，我的做法是尽可能多地在简历中体现企业要求的技能及软实力。一般可以在下面两处反复琢磨：

- 技能与评价。
- 项目经历。

结合我的简历，先看技能与评价部分怎么修改。

因为目标职位是高级软件开发工程师，招聘信息也没有特别体现对管理能力和经验的要求，所以我的管理经验可以淡化或拿掉。然后是突出 C++、Qt、软件设计、技术传播等内容。新的版本如下：

技能与评价

- 9 年 C++开发经验，精通 C++，熟悉 STL。
- 5 年 Qt 开发经验，基于 Qt 开发过 8 个商业项目，代码超过 30 万行。
- 著有《Qt on Android 核心编程》和《Qt Quick 核心编程》。
- 熟悉面向对象编程。
- 熟悉 GoF 设计模式。
- 丰富的软件设计、软件架构经验。
- 熟悉 MFC，熟悉网络编程，熟悉多媒体及各种流媒体协议。
- 热衷于技术传播，推动了 Qt 在团队和公司内的普及。
- 新版本兼顾了硬性要求、软性要求和优选条件，与初始版本有明显不同。

接下来，我们就要重新组织、呈现项目经历，让它更多地体现企业 HR、面试官所关注的关键点。下面是我的一个具体的项目经验。

2009.05—2011.09，跨平台视频点播系统

项目描述：面向电信运营商和零售市场的综合性视频服务产品，具有视频导航、搜索、点播、直播、天气、资讯、股票、教育等功能。产品形态有机顶盒、Windows 客户端、Linux 客户端等。

软件环境：Embedded Linux，Qt Embedded 4.5.1，Qt 4.5.1，C/C++ 硬件环境：全志 F20 芯片方案、PC。

角色：项目经理、核心程序员。

职责与工作内容：

1).软件系统结构设计。

2).基于 Qt GraphicsView 框架设计实现十字菜单、影视信息墙。

3).基于 Qt 的客户端业务流程代码实现。

4).基于 Qt 实现认证模块。

5).团队组建与项目管理

业绩：

1).开发了稳定、性价比高的互联网机顶盒产品，打开了电信市场，销量超过 30 万台。

2).在团队里推广 Qt，形成了技术积累。

3).提供 Windows 及 Linux 客户端，为客服、运维、售后提供了强有力的支撑。

下面这个招聘需求，和我的匹配点主要在技能（C++、Qt）上，接下来主要匹配的则是业务（行业）经验。见下面的图：


职位描述

岗位职责：

1、参与完成音视频信令业务平台的开发、测试、维护工作；

2、参与视频业务系统的开发、测试、维护工作；

3、参与视音频编解码、封装、流媒体处理等相关技术难点攻关；

4、负责项目相关文档的编写。

任职资格：

1、国家重点本科以上学历，计算机或通信等相关专业，2年以上相关开发经验；

2、熟悉C或C++程序设计与开发及各种程序设计模式、模块化思维，并有独立设计经验;

3、有基于Linux系统开发经验，熟悉Linux系统内核、文件系统;

4、精通多线程、多进程开发; 精通网络通信、标准协议开发；

5、有SIP、H.323等信令协议开发或VoIP话机、MCU、代理服务器等产品开发经验优先；

6、有H.265、H.264、H.263+、H.263、G系列音频、AMB、GSM等视音频处理经验优先。

虚线标注出了基本要求：C/C++、Linux。

直线标注的软性要求中，一部分是不太容易量化的技能，比如精通多线程、独立设计经验等；一部分是产品业务相关的，比如音视频编解码、封装、流媒体、网络协议、视频业务系统等。

这则招聘信息中有两个优选条件，不过和我的知识、技能都不匹配。

一个程序员的知识、技能、经历，横看成岭侧成峰，角度很重要。

来看看新修改的技能与评价：

技能与评价

- 9 年 C++开发经验，精通 C++，熟悉 STL。
- 可熟练在 Linux、嵌入式 Linux、Windows 等平台下进行开发。
- 熟悉常见的设计模式，有丰富的独立设计经验。
- 有 6 年视频业务开发经验，设计、开发过点播、直播流媒体处理系统，熟悉 FFMPEG、GStreamer 等开源框架。
- 精通网络通信，自己实现过标准协议 HTTP 协议和私有 P2P 协议，熟悉 HTTP/HLS/RTMP/P2p/RTSP 等各种流媒体协议。
- 热爱技术，著有《Qt on Android 核心编程》和《Qt Quick 核心编程》。

和面向上一个招聘信息的内容又有了较大差异，是不是？

这次我改两个之前展示过的项目经历，方便比较不同。第一个如下。

2009.05~2011.09，跨平台视频点播系统

项目描述：（略）

软件环境：Embedded Linux，Qt Embedded 4.5.1，Qt 4.5.1，C/C++ 硬件环境：全志 F20 芯片方案、PC。

角色：项目经理、核心程序员。

职责与工作内容：

1).软件系统结构设计。

2).独立设计基于 Qt 的 EPG 框架。

3).设计实现音视频解码、解封装流程（参考 ffmpeg）。

4).设计实现机顶盒软件的多线程模型。

5).团队组建与项目管理。

业绩：

1).定制的多媒体框架保障了互联网机顶盒产品的视频处理效果，打开了电信市场，销量超过 30 万台。

2).提供 Linux、Windows 版本客户端，为客服、运维、售后提供了强有力的支撑。

3).坚持自主实现关键的多媒体模块，形成了技术积累。

这次我将重点放在了多媒体、设计等方面，与前面那份简历有了很大不同。

再来看本节最前面提到过的一个互联网视频卡顿的项目经验，新版本如下：

2013.11~2014.06，互联网视频卡顿优化

项目描述：

智能机顶盒上的视频聚合客户端，在播放视频时，高峰时段或热点视频，经常卡顿。针对此问题，进行优化，以使能够对用户提供流畅的观看体验。

职责与工作内容：

1).作为系统设计，选择技术方案，设计加速与优化算法，设计系统结构。

2).C++实现标准 HTTP 协议（含客户端和服务端）。

业绩：

播放效果大幅度提升，卡顿投诉下降 80%。

和原来大不一样了。

需要特别提醒的是，**简历优化是针对程序员知识、技能、经历进行的穿衣打扮，是从不同的视角呈现不同的侧面，可以用心琢磨，但万万不可造假**。

3．检验优化效果

提供一个检验你简历优化水平的方法：

将你钟意的企业和职位分为 A、B、C 三类，A 是最符合你目标的，B 与你最核心的诉求匹配，C 类可能抵触了你的某些核心价值观。

针对 C 类公司的职位，测试你的简历优化技能，找三五家来练手，投递简历，简历过了就去面试，攒面试经验。

C 类之后进行 B 类，最后是 A 类。

这样的过程，能保证你先练简历和面试技术，然后以最好的状态走向你最心仪的公司、最钟意的职位。

如何提高简历投递成功率

根据前面的分析，要想提高简历投递的成功概率，遵循下面的流程将非常有帮助：

- 梳理知识、技能、经历并记录在案，形成基础简历。
- 确立求职目标。
- 筛选招聘信息，选择匹配自己目标的公司和职位。
- 针对每个招聘信息进行分析，提取关键词。
- 根据关键词，结合基础简历，优化技能描述和项目经验，生成一份有针对性的简历。

根据我自己的经验，一天可能只能完成 3 份左右的简历投递。因为针对每一个职位生成一份有针对性的简历，可能会花费 1~2 个小时时间，有时甚至更长。不过，花再多的时间都是值得的，因为这种优化将大大提高你的简历通过的概率。

城市大小对职业选择的影响

2016 年 4 月 19 号广州日报刊登了一则新闻：

夫妻俩都是名校研究生，在深圳有着体面的工作，有房有车有儿有女，白领小鱼儿和高先生夫妇还是选择了卖房、举家离开深圳，回到 1000 公里外的江城武汉，开始另一种生活。2015 年 6 月，高先生在网上投递了简历，凭着深圳多年的工作经验和名校学历傍身，他很快就在武汉找到了一份很不错的工作，薪资比深圳少一点，但是发展前途很不错，对方单位还给了个小的领导职位。自从下定决心回武汉发展，高先生一边在武汉看房、一边将深圳的房产在中介放盘出售。最终，深圳的房子卖了 400 多万元。高先生用这笔钱加上多年的积蓄，于 2015 年 10 月至 11 月在武汉买了 4 套房。4 套房子都有学位，其中 3 套位于汉口金融中心，还有 1 套是华科附近光谷一小的学位房。

小鱼儿感慨说，“终于敢花钱了”。

这是又一则逃离北上广、逃离深圳的例子，在网络上引起热议。

然而有人在离开，还有更多的人正在赶来。前几天，我在 QQ 上和原来的一个同事聊天（他在兰州做服务器维护），他说要北漂去了。有图有真相：

为什么有人选择离开，有人相反？城市大小到底对个人职业选择与发展有何影响？我准备围绕下面四点来说：

- 城市与产业结构。
- 城市大小与公共资源。
- 城市与生活成本。
- 城市节奏与个人性格。

城市与产业结构

地域和产业有密切关系。比如沈阳，机械制造和飞机制造业发达；珠三角，出口加工业发达；东莞，有大量的电子加工、玩具制造企业；深圳，硬件设计和制造发达；上海，软件、金融、教育、公共服务等产业都很发达；北京，软件、教育、政治、医疗等产业优势远超其他城市；榆林，煤炭、石油、天然气等产业有独特优势；大理、丽江，旅游业、服务业发达；……

所以，当我们选择一个城市时，其实更多地是考虑这个城市的产业结构与自己的专业技能是否匹配。我们在这个城市是否有用武之地。假如你的目标职业是硬件设计工程师，可能深圳就是最好的选择。

大城市产业多元化，机会多，这是吸引外来人才的巨大优势，而且这种吸引力具有螺旋增强的趋势：人才越多，相关产业越发达，机会越多，对其他人才的吸引就越强烈。

城市大小与公共资源

很多时候我们选择一个城市，还因为这个城市的公共资源。

比如医疗资源，北京有 83 家三甲医院，协和医院、友谊医院、301 医院等医院的医疗水平全国拔尖。

比如教育资源，全国共有 112 所“211”学校，北京有 24 所居首位，江苏 11 所和上海 9 所分别排在第二、第三位。全国 39 所“985“高校，分布在 18 个省市，排在前三位的：北京 8 所，上海 4 所，陕西 3 所。

再比如体育场馆、赛事资源，也有很强的集中性。

城市与生活成本

城市越发达，人均收入越高。我有一位朋友做 Java 后台开发，2015 年年底，从西安跑到北京，薪水直接翻倍了。然而城市越发达，生活成本也越高。

前面讲的小鱼儿和高先生，他们在深圳，一年税后收入超过 50 万元，但要买个学位房要 1000~1500 万元，压力不可谓不大。房子的压力，已经成为各大城市外来从业人员的首要压力，北京、上海、杭州、广州，个个如此。

在北京工作，一开始可能要租房子，但房租太贵，可能地下室都舍不得租，往往选择与人合租或选择偏远地带。买房子的话，几环就不用说了，好几万一平方米，买不起，最终可能选择在通州、燕郊、昌平等地买，而你的工作地点可能在海淀区中关村，这样一来，通勤成本就很高，一天在上班路上花掉的时间可能在 4 个小时左右。如果你搞软件开发，遇到加班较严重的公司，晚上 10 点 30 后才离开单位，回到住所就后半夜了……

我在西安，午饭一碗面或一份两荤一素的米饭套餐，12 元左右搞定，我到上海培训时，话费大概在 18 元。大城市相比小城市，吃穿用度都要贵一些，这些都是成本。

还有，你在上海谈个对象，成本可能比延安高很多。

城市节奏与个人性格

作为独一无二的个体，你其实是有自己的振动频率的。有的人性子慢，喜欢田园牧歌式的慢生活，放到上海深圳这些地方就不适应。有的人性子快，走路都比别人快三拍，在成都、昆明、烟台这些地方，看着人家摆龙门阵提笼架鸟，会有毋宁死的感觉。

每个人的兴趣、爱好都不同。有的人喜欢文化底蕴厚的地方，在西安、南京、成都等地方就如鱼得水，到深圳、珠海可能就不太适应。有的人深具佛性，偏爱藏传佛教文化，可能拉萨是比较适合他的地方。

职业选择与城市

按照“职业=行业*职能”的定义，选择职业时，行业和职能是两个基础维度，其中行业受城市影响最大，比如你想做软件开发工程师，在漠河恐怕就很难做；而你想进行硬件创业，深圳可能是最合适的地方；如果你想做职业规划师，那么北上广深这些现代化程度高的城市机会更多（因为人面临的选择多、压力大，寻找自己定位的需求更大）……

大城市相关产业发达、机会众多、发展空间大，这就是它对职业人士的吸引力。

我们选择一种职业，除了需要考虑行业，还可能会考虑将来的生活，比如结婚、买房、生孩子、教育、养老、医疗。在大城市虽然机会多，但吃穿用度、教育、医疗、养老等成本也高，可能你在北京月入 1 万元的生活感受还赶不上西安月入 1 万元的。

职业选择是极其个人化的，有很多人宁可在大城市收获忙死的成就感，也不愿去小城市体验闲死的无聊。

史上最全的程序员求职渠道分析

（注：到本书出版时，可能有些招聘渠道的情况已发生变化。）

今天专门分析一下程序员求职渠道，有料是必需的，就算你搜遍互联网深挖全宇宙，也会发现本节内容是史上最全、最强、最有针对性的程序员求职渠道分析。

还记得我 2002 年毕业时曾经在西安跑来跑去参加各种线下的招聘会，投简历等消息，也曾经通过华商报的前程无忧专版查看各个公司的招聘信息和各类招聘会的公告，还曾经辗转于西安的各个人才市场，那样的日子奔波忙碌却被动无奈。还好，互联网的飞速发展解放了我们，现在，我们基本上可以和那些日子说再见了，找工作时只需要坐在电脑前敲几下键盘点几下鼠标就行了。

我们程序员其实是比较特殊的专业人才，求职渠道比其他人群要多得多，让我们逐个来看看。我粗略分为七大类：

招聘网站、专业技术论坛、QQ 群和微信群、内部推荐、猎头、人才竞拍、职场社交

我分析的顺序是先普通后特别，如果你没耐心看招聘网站之类的分析，就跳到最后人才竞拍网站和职场社交渠道吧，惊喜就在那里！

招聘网站

毫无疑问，招聘网站依然是很多程序员找工作的重要选择，也是很多企业招募人员的常规选择。招聘网站多如牛毛，我知道的比较有名的有这些：

- 前程无忧(http://www.51job.com)
- 智联招聘(http://www.zhaopin.com)
- CSDN JOB(http://job.csdn.net)
- 中华英才网(http://www.chinahr.com)
- 大街网(http://www.dajie.com/)
- 中国人才热线(http://www.cjol.com/)
- 中国人才网(http://www.cnjob.com/)
- 猎聘网(http://www.liepin.com/)
- 数字英才网(http://www.01hr.com)

- 若邻(http://www.wealink.com/)
- 拉勾网(http://www.lagou.com)
- 职友集(http://www.jobui.com/)
- 卓博人才网(http://www.jobcn.com/)
- 全才招聘网(http://www.allzhaopin.com/)
- 内推网(http://www.neitui.me/)
- 58 同城(http://www.58.com)
- 赶集(http://www.ganji.com)

从覆盖范围和地域上看，上面这些网站都是全国性的，此外还有很多地方性招聘网站，也是不错的渠道。比如西安高新人才热线(http://www.hrdonline.com/)、西安人才网(http://www.xarc.net/)等就是西安的招聘网站，尤其是西安高新人才网，主要是 IT 类招聘，对于在西安求职的程序员而言是个相当不错的选择。对于一线城市的程序员，基本上不用考虑地方性的招聘网站，全国性的招聘网站已经优先覆盖了这些城市，信息足够丰富了，而对于生活在 *N* 线城市的程序员来讲，地方类的招聘网站往往会有出乎意料的效果。

下面结合我的体验谈谈这些网站。

我使用过的有**前程无忧**，**智联招聘**，**中华英才网**，**猎聘网**，**若邻**，**拉勾网**。

前程无忧是老牌招聘网站了，影响力比较大，最早的模式是集市模式，招人的求职的都到这里来，求职者在海量信息中找职位找企业，企业在海量简历中找应聘者或接受简历，匹配度低，效率低。这个网站有大量的 IT 类职位，是刚出校门或经验较少的程序员的一个通常的选择。我最开始就在上面投简历，不过匹配到的情况比较少。这个网站还提供增值服务，求职者交钱就可以享受到较好的服务，企业交钱就可以在简历库中自主挑选。这种模式的另一个结果就是，不交钱的个人很难找到职位，不交钱的企业很难招到合适的人。

类似的还有智联招聘和中华英才网。我们单位曾在智联招聘上发布过招聘信息，按月、按季度、按年交费用，不过简历质量不怎么样（也可能是我们单位为初创企业不吸引人）。据说智联上面的人才层次要稍高一些。不过在我印象里，它和 51job 是差不多的。

我浏览了中国人才热线、中国人才网、数字英才网、卓博人才网、全才招聘网等，感觉与**前程无忧**差不多。需要提一下的是，这一类网站也都有猎头服务。

接下来要说的是猎聘网，这家招聘网站面向中高级人才，假如你的技能和经验都丰富，可以到上面一试。猎聘网上活跃着很多猎头和公司的 HR，简历提交后程序员与企业的匹配度较高。我的简历好久没更新了，还经常有猎头或公司的招聘负责人联系我。另外它提供的职位推荐也是比较靠谱的，与你的简历和求职意向拟合度比较高。

58 同城和赶集网也都有招聘频道，也有程序员在上面求职，也有 IT 类公司发布职位。这里流量大，鱼龙混杂，涉及行业及职业广泛，不过多以中低端岗位为主，如客服、渠道专员、保洁、家政等。个人感觉不太适合程序员，反正我是从未考虑过。我着手负责人员招募工作后，把公司发布的岗位停掉了。

大街网和若邻网是社交类招聘网站的代表，既然是社交，走人脉路线，找工作自然比较慢，不过还是可以试试的，尤其是 IT 公司的职位还是比较多的，运气好的话，能碰到一些高薪职位。我 2014 年在若邻发布过简历，不过没有收到一家面试通知。

我们公司曾经在拉勾网上面投放过招聘信息，我本人没有在上面发布过简历。拉勾网宣传说专注于互联网职业机会，我们发布了职位后，确实收到了不少简历，而且我们是没有交钱的。

职友集我没用过，不过看起来算是比较独特的一个个例，它从公司评价、打分、薪水等众多层面给公司评价，想帮助你找到合适的职位。它的首页也不同于**前程无忧**、**智联招聘**等网站，不是众多的职位展示，而是一个公司的搜索框。

内推网我没用过。但这个网站也找了一个很棒的切入点：内部推荐。不知道效果怎么样。

CSDN JOB 依托 CSDN 的流量和活跃社群，按说是比较有潜力的，当然还需要找到合适的切入点，目前它的模式和**前程无忧**、**智联招聘**等没什么不同。优点是专门针对 IT 类职位，比较适合程序员和有这方面需求的公司，匹配度会比较高。我们公司之前在这里发布了一个后端的与 Scala 相关的职位，工作地点是西安，结果很快就有一个懂 Scala 的朋友找到了我们。

总算说完了，福利还在后面，继续吧。

专业技术论坛

程序员非常特殊，天生就是互联网动物，经常聚集在各种专业的技术论坛，比如 QTCN(http://qtcn.org/)，eoe Android 社区(http://www.eoeandroid.com/)，Cocoa China(http://www.cocoachina.com/)，PHP100(http://www.php100.com/)，讨论交流各种技术问题。很多 IT 类公司会到程序员出没的地方发布有针对性的招聘信息，所以，那些技术论坛往往开辟有招聘专版。

专业技术论坛上的招聘信息非常有针对性，就是针对这个技术论坛讨论的技术方向的，

比如企业在 QTCN 上发布的招聘信息，就是针对熟悉 Qt 的开发人员的，而企业在 eoe Android 社区发布的招聘信息，就是针对熟悉 Android 开发人员的，其他同理。

几乎每一种技术都有相关论坛和网站，这些网站上也都有企业发布招聘信息，求职方和招聘方往往具有很高的匹配度。这也是专业技术人才找工作的一个重要途径。我曾经在论坛上看到一个招聘信息，然后联系上了，最终进了那家公司，这也是一次神奇的经历。还有朋友告诉我说在看雪(http://bbs.pediy.com/)论坛发了一篇有深度的安全文章，结果就被一家安全方面的公司招去了。

我认为很多在技术论坛发布招聘信息的，都是公司研发类的技术人员，好沟通，比较直接，对岗位的描述和要求也都相对靠谱一些。

QQ 群和微信群

每一种技术都有若干个 QQ 群、微信群存在，很多程序员其实是在 QQ 群里而不是公司上班……

技术类的 QQ 群（微信群）分为两种，一种是热爱技术的大神们建立的，专门交流某一项技术的技术导向群；一种是 HR、猎头或具有相关身份的人或机构设立的，针对某一项或某一类技术的招聘群。这两种群，都有招聘信息发布，我的感觉是，技术导向群里的招聘信息，针对性更强，但频度低，而招聘导向的群，针对性稍差，但频度高。

要想找某一方面的群，只需要在通过 QQ 查找关键字即可。很多公司的 HR 或者开发人员，会潜入到技术主导的 QQ 群里寻找他们需要的程序员，这也是一个不可忽视的求职渠道。

内部推荐

《你好哇，程序员》一书中 “找工作的辟邪剑”一节中提过内部推荐，这也是非常值得一试的方式。内推网看起来应该是从这个点切入的。

很多公司信这个，认为自己的员工不会推荐不靠谱的人。我工作过的公司，如果你推荐的人被录用并且成功通过试用期，你会获得 1000 元的伯乐奖金。据我的了解，IBM 公司有类似的机制，阿里巴巴也有……所以，如果你能找一个目标公司的员工来推荐你，那么你就踏出了成功的一步。

根据六度空间理论，如果你愿意，总是能找到推荐你的人的。比如你曾经共事过的同事、你的同学、你在 QQ 群里认识的志同道合的朋友……只要你愿意挖掘就有。别怕麻烦，虽然我们程序员大多数人脉不够广，不过三五同事、好友还是有的，努力挖掘吧，找工作是大事，求人帮个忙也别张不开嘴。当然反过来，别人找到我们，也别总拒绝。

猎头

谈谈猎头吧。我有一些和猎头打交道的经验。

猎头在人力方面还是有很大作为的，他们和企业关系较好，很多企业会委托猎头来招聘高端职位。所以，如果你有机会结识猎头，一定要留下他们的联系方式，保持联系。当你要换工作时他们就会帮得到你了。

另一方面，猎头拿到职位也是很希望快点找到合适人选的，你可以向他们推荐合适的人，要知道，程序员身边有很多程序员，经常会有人换工作，你推荐给猎头，猎头回头有机会也会照顾你。

很多猎头公司有自己的网站，也可以到网站上联系他们（当然我没这么做过，你可以去看看）。我和科锐国际(http://www.careerintlinc.com/)、大沃猎头(http://www.wodcj.com/)、大瀚人才(http://www.vastsea.com/)打过交道，确实比较专业。

人才竞拍

互联网改变一切，互联网+改变世界。

100 offer(http://www.100offer.com/)，我看到这个名字，点进去一看，立马被它所采用的**人才竞拍**模式震惊了，它颠覆了传统的程序员求职模式。

前程无忧、**智联招聘**这类网站，需要程序员自己找职位、投简历，然后等待回应，海量投，慢慢等，有时等到花也谢了。从求职者和企业的关系来看，这是最传统的模式，也是最不经济、最没效率的模式。

猎聘网引入了猎头，提高了效率。

大街网和**若邻网**希望通过社交、人脉来助力求职，不过效果很慢，需要你慢慢发展人脉，这和你发掘线下身边的关系、加入 QQ 群、技术论坛发展人脉都类似，不知道什么时候会有效果。

而 100 offer 采用的人才竞拍模式，则与上述网站完全不同。它是*互联网+猎头*的方式，你只需要提交简历，审核通过后，就可以坐等竞拍，2 周内就有很多公司主动联系你，你可以选择中意的职位，决定是否进入面试，很快就可以搞定满意的 Offer。你再也不用在浩如烟海的招聘信息里翻找职位了，效率大大提升，省下来的时间、精力可都是你的。

类似 100 offer 这种人才竞拍的模式，彻底改变了传统的求职者巴巴地找职位、投简历、等消息这种不良体验，让你瞬间从草根变身 VIP，让企业来竞拍程序员，到处都是橄榄枝。不过它有一定的门槛，要求具备 2~3 年的经验，会有专业人士审核你的简历，你能不能通得过，就要看你有多少干货了。

做这种人才竞拍模式的还有几家，比如**拉勾网**上线的**一拍**(http://pai.lagou.com/)，牛才拍(http://www.niucaipai.cn/)。这是新出现的模式，我觉得必然会大放异彩。这几家做人才竞拍的，**100 offer** 目前专门针对程序员群体，**牛才拍**针对游戏人才，都值得一试。

职场社交

脉脉(http://maimai.cn/)有一阵子动作很大，电梯间都是广告，江南春、王小川都在给它背书。有网站有 APP，定位职场人脉拓展，上面也有很多找人的，而且这种依托职场关系链的招人方式，看起来更靠谱，求职方和招募方之间更容易建立信任。

Boss 直聘(http://www.bosszhipin.com/)，号称互联网招聘神器，移动端 APP。顾名思义，上面有很多 Boss 在，可以直接联系他们，实时聊天相互了解，提高求职效率。我曾在上面发布过招聘信息，可惜因为在西安，它覆盖的人群以北上广深杭居多。但别人能够直接关注你的招聘信息，还能区分出感兴趣的程度，还能在见面前先聊聊，的确感觉有点不同。

LinkedIn 中国(http://www.linkedin.com/)，LinkedIn 是全球知名职场社交平台，目前已来到中国，很多高端人才都在上面注册了账户，你可能也收到了 LinkedIn 发来的你认识张三认识李四的邮件了，反正我已经收到了无数封，虽然有些烦，但这种对职场关系的挖掘，确实非常有效。

程序员的求职渠道指引

总结一下。对于刚出校门的，或者经验不多的程序员，可以在前程无忧(http://www.51job.com)，智联招聘(http://www.zhaopin.com)、中国人才热线(http://www.cjol.com/)之类的

网站找找机会。

如果你摸爬滚打了两年，有了一些积累，可以考虑 eoe Android 社区(http://www.eoeandroid.com/)，Cocoa China(http://www.cocoachina.com/)等专业的技术论坛、CSDN JOB(http://job.csdn.net)、QQ 群、拉勾网(http://www.lagou.com)、猎聘网(http://www.liepin.com/)等。

如果你觉得自己水平不错，某方面出色，去 100 offer(http://www.100offer.com/)、牛才拍(http://www.niucaipai.cn/)、一拍(http://pai.lagou.com/)等人才竞拍渠道检验一下吧。

如果你相信职场社交的神奇效果和效率，可以使用脉脉(http://maimai.cn/)、Boss 直聘(http://www.bosszhipin.com/)、LinkedIn 中国(http://www.linkedin.com/)等。

程序员跳槽神级攻略

关于程序员跳槽这件事，我打算从三个方面来说：

- 什么时候该跳槽。
- 跳槽前你需要做的准备工作。
- 到哪里找跳槽机会。

什么时候该跳槽

我在《你好哇，程序员不》一书的一节中提供了“周末探视法”让大家分析自己对当前工作的感觉。这个方法很简单，你只需做下面这件事：

在周日的晚上，想着明天要上班了，记录自己此刻的念头和心情。

OK。就是这样。如果你内查到犹豫、恐慌、紧张、担忧、抗拒之类的情绪，很可能你已经需要慎重考虑换工作这件事了。

周末探视内心感觉是一种通用的做法，每个人在使用时都可能找到一些导致自己必须换工作的具体情况。这里是我遇到的和想到的一些情况，列出来大家相互印证一下。

1．产品没有前途，从各方面看都要玩完

经常有公司上马一个产品时行差踏错，做着做着就感觉恐怕是方向搞错、不会有前途

了：要么错过了好时机，要么没有切中刚需，要么这个产品所需要的营销运营能力以及资本都不具备……总之，眼看着事难成，继续做下去只是自我安慰。作为程序员，你需要在一个有希望的行业里做有希望的产品，这样自己才有希望。

2．公司经营不善，面临倒闭风险

互联网时代，公司起来或倒下都很快。稍有不慎，一个公司多年积累就可能轰然崩塌，资金跟不上，亏损，最终面临倒闭。这些都是能看到的，流言四起，议论不绝，周围的人已经纷纷心猿意马四处出击，准备换工作了，你怎么想呢？

3．不受重视，被严重边缘化

有时你自觉有才有能力，却总被安排打下手，深深觉得不受重视，有一种英雄无用武之地的感慨，此时也有必要考虑换个能发挥你长处的地方。士为知己者死，千里马需要伯乐。在不对的地方跟着不对的人，很难做出对的事情，自己也很难由此成长起来。当然，你要真的有才。因为，怀才就像怀孕，只有足够大时别人才看得出来，你才能享受到相应的重视和特别的待遇。往往我们觉得有志难伸，只是我们的才还小，没被别人看到而已。

4．不被尊重

公司不尊重人，领导不尊重人，看不起下属，内心不屑鄙夷，动辄横眉冷眼呼来喝去，甚至给你头上蒙个黑布罩就想让你拉一辈子磨。他只当你是一颗小石子，用得着就捡起来，用不到一脚踢飞。这样严重缺乏基本尊重的环境，不待也罢。

5．没有成长空间

团队里没有高手，连个切磋的人都没有，何等寂寞。遇到问题没人能搞定，何等无奈。

你只要完成上头交待的任务就行了，你的技术之路如何发展、你在公司的职位和等级怎么晋升……从来没有人找你谈过，也没有人真的关心你。

有时公司和产品到一定程度，新东西少，老东西多，维护性东西多，你在技术上已经到顶了，不能再进一步，眼看着没机会再挑战了。技术就如逆水行舟，不进则退。你没有锻炼和挑战的机会，是很难百尺竿头更进一步的。

当然你可能不做技术，而是去做管理。可一个萝卜一个坑，你上面的人不走，你就没

有机会。上面的人还年轻，一时半会儿既不会内退也不会走人，人家干得好好的呢。那你呢，短时间内怕是看不到希望了。

凡此种种，不论怎样你都看不到自己的成长空间在哪里……

6．生活环境发生重大变化

一般普通人的生活路线，在特定时期都会面临谈对象、买房、结婚、生子等问题，一但你到了这个年龄段，各种压力和支出纷至沓来，很可能你原来的收入已经不足以维持生活，而此时放眼单位和工作，一时半会儿升职加薪无望，迫于生活的压力，你是否会选择换一个收入更好的工作呢？

7．与 BOSS 关系紧张无法调和

你和老板或上级发生矛盾，比如经常性地在某些事项上存在严重分歧，经过努力也无法调和，很可能会给你带来比较大的压力和不适，让你心生不公、委屈、厌倦、绝望，感到再也无法待下去了……

8．公司里身边的人都在混日子

“昔孟母，择邻处”其实讲的是外在环境对人成长的影响。在工作上也是一样的，假如你周围的人都浑浑噩噩不思进取，你也很难一枝独秀、勇往直前挽狂澜于既倒。这种环境就像一盆污水，你就是一滴昆仑山矿泉水，滴进去也很快就被同化了，赶紧走人是正事。

9．失去激情，无法投入

你的心已不在所做的事上，无法保持激情，也不能全身心地投入，整天懒洋洋的，干好干不好都不在乎了，这时你该和当下的工作说再见。因为这种状态消磨的是自己的生命。

10．同行业比较，待遇差别太大

大部分人需要和周围的人比较来平衡自己，当你发现同行业同经验的小伙伴们拿的钱比你多出一大截子时，你就很难淡定了。虽然有人说“追求财富不如追求满足，满足才是最大的财富”，可是你的价值也需要通过收入来体现，当收入和自身价值严重脱节时，恐怕满足感、归属感很难再有。

11．薪水倒挂

有的公司领导很“奇葩”，宁愿花更多的钱请新人，也不给老员工涨工资，以至于具备同样的工作经验，新来的人工资居然比老员工高一大截，这叫辛辛苦苦兢兢业业的程序员情何以堪啊。

12．扛不住加班

这也是有的。比如你的公司老是没日没夜地加班，晚上十点走都有人给你白眼，而你媳妇又怀着小宝宝，这怎么受得了……

不管你实际上因为什么原因不能忍受当下的工作，一定要明确具体的因素，列出来，记下来，这样在找新工作时你才能跳过一些陷阱。

跳槽前要准备的 *N*件事

一旦你决定要离职，一定要做充分的准备（除非事发仓促，比如家庭变故之类的）。那么，在跳槽前要做哪些准备工作呢？

其实我觉得从大的方面讲就三件事：

- 分析自己。
- 分析目标行业与公司。
- 撰写简历。

1．分析自己

人贵自知，自知者明。这是有道理的。一旦你决定跳槽，那就要分析自己。

要知道自己有什么：

- 工作几年。
- 待过的公司有什么影响力，产品有什么特色。
- 掌握了哪些技术，程度如何。
- 和周围的人比，自己的长处在哪里。
- 学新东西快不快，举个例子。
- 解决问题能力如何，举几个例子。

- 沟通与协作能力怎样。
- 领导力如何，有没有影响别人促使项目成功的例子。
- 积极性如何，有没有在无压力时自我驱动完成事情的实例。

要知道自己要的是什么：

- 更强的技术能力。
- 更高的薪水。
- 更好的福利（公积金，商业医疗保险，……）。
- 成就感。
- 被认同。
- 从头做一个成功的产品。
- 大公司镀金的经验。
- 跟随可能的机会快速发展。
- 轻松，不加班，可以照顾家人。
- 管理能力提升的机会。
- 股权。

要知道自己能干什么，这也是很重要的，一定要明了自己的能力边界。比如：

- 富有钻研精神，能解决技术难题。
- 有领导力，可以影响、驱动团队。
- 能把握技术，对技术敏感，善于把控技术方向。
- 可以快速、高质量编码。
- 带团队。
- 当尖兵。
- 架构设计。
- 写出还说得过去的代码。
- 会活跃团队气氛。
- 能写一手好文案。
- 懂产品。
- 能带人，能当导师。

很重要的一点是，自己愿意干什么。因为你从一个地方离开，一定是这个地方有什么东西你不能接受，那么你找新工作时，一定要考虑自己愿意干什么，如果考虑不清楚，那么才出虎穴又入狼窝的可能性就会非常大。

最后还要想自己能失去的是什么，千万别小看这一点。这世上并不存在完美的工作，没有哪个公司、哪个岗位、哪个产品是为你量身定做的。就算你走运，碰见了这么一个机会，但随着时间的轮转和周围环境的变化，曾经你觉得百般熨帖的，也会慢慢出现各种不适。所以，无论何时，痛苦和快乐是伴生的，你必须考虑取舍。在找工作时尤其如此，鱼与熊掌不可兼得，这一点必须明白。

举几个简单的例子吧。

- 你老婆怀孕了，需要照顾，那么你可能想找一个轻松的不加班的工作，此时薪水要求就不是那么重要了。
- 你的身体突然变差了，比如腰椎间盘突出，或者颈椎强直……此时你肯定应该哪里轻松去哪里。
- 你想搏一飞冲天的机会，可能就要舍去稳定性；你想稳定，可能就得忍受身边某个人突然某一天扶摇直上九万里。
- 你想要股权，可能工资就会低一些。
- 你想得到经理的待遇和福利，就得能抗住经理需要承受的压力。
- 你想去外企而英语不好，就得自己花钱去培训来提升英语能力。
- 你家在西安人在上海，有一天想离家近点，就得接受两地的收入差距。

类似的太多了，总之我们需要明白什么对自己是最重要的，然后就可以在必要的时候，舍弃一些不那么重要的东西。

2．分析目标行业与公司

我们找工作时，需要结合自己的现状，瞄准特定的行业和公司来做准备，跑到网上见什么职位都投，绝不是最好的方法。

举个例子，比如你之前在做医疗软件，以后还想做这方面的工作，那么你找工作时就要分析自己所在的地区做医疗软件的公司分布情况，每家公司都是什么状况，做了分析之后，锁定目标，再来看哪家公司在招人、有机会。

如何知道有哪些公司在做类似软件呢？你在一个行业里待着，一定要了解这个行业的现状，前景如何，竞争对手都有哪些，竞品优势在哪里……虽然你是技术工作者，但也要了解这些，否则换工作时会手忙脚乱遇到各种状况。如果你自己在日常工作中接触不到，也可以从公司里其他岗位的同事（比如市场、销售、产品等）那里了解到。如果有心，你

一定可以知道。这些和你的工作息息相关，绝对有必要去了解。

一旦你锁定了行业和公司，找工作时就不会那么随意、那么慌张了。凡事预则立不预则废，机遇只青睐有准备的人。

3．撰写简历

分析了自己又分析了公司，接下来就是撰写简历了。网络上有很多谈如何写简历的文章，可以搜搜看看，找一个模板来参考，先根据自己的工作经验写一份基础简历，把你前面对自己的分析、项目经历等都捋清楚写进去。

有了基础简历，接下来就到了非常重要的一环：有针对性的修改。

为每一个公司的每一个岗位准备一份简历，这是非常非常重要的。具体参考“简历优化实操”一节。

招聘信息里的职位描述和任职要求很详尽。

如果你确认了这个公司值得你投递简历，那就需要仔细分析职位描述和任职要求。职位描述会说明你应聘的这个岗位要做什么，不过很多公司填写的也很空泛，都是行业黑话，有效信息不多。所以我们的重点就是任职要求。

有些看起来比较虚的要求，类似“较强的学习能力”这种，可能我们一开始觉得很难针对它来修改简历。其实不然，如果你做的某个产品用到的技术是你第一次接触（一定有这种情况，除非你天生是会各种技术的大神），你就可以把这个经验当作例子，在里面加入你对产品的贡献。比如这样：“在两个星期内完成了 Scala+Play 的学习并顺利接手 10 万代码行的项目”。

每一份有针对性的简历都值得反复琢磨，优化得当你就有很大几率通过简历筛选这一关。一旦通过了简历筛选，就有机会进一步了解公司了，比如可以在面试通知电话里询问要做的产品、团队的大小等，为面试做准备。

到哪里找跳槽机会

“史上最全的程序员求职渠道分析”一节，专门分析程序员的求职渠道，往回翻翻参考一下吧。

入职薪水对你的影响有多大

我们找工作时都想获得满意的薪水，可实际情况是，有相当一部分程序员对自己谈下来的薪水都不是很满意。并且，面试时谈的薪水往往会对自己有意料不到的影响。

这次，我们就来谈谈入职薪水这个问题，看看它会怎样影响我们的工作和生活。也许你会说，一句话就足以概括了：“其实，我的内心几乎是崩溃的。”当然，事情要是有这么简单就好了。

为什么会不满意

程序员跳槽往往有很多原因。诚然，多数时候是因为钱，却也有些时候不是钱的事。我曾经在“**让程序员跳槽的非钱原因**”（收录在《你好哇，程序员》一书中）一节分析过，感兴趣的同学可以去看看（也可以关注我的订阅号“程序视界”，回复 10034 或“让程序员跳槽的非钱原因”来查看这篇文章。）

不管你是奔着“跳槽==加薪”去的，还是别的什么原因让你义无反顾地选择了跳槽，面试谈薪水都是必不可少的环节。而谈之前，你一定是有底线的。多少能接受，多少不能考虑，心里一定明镜似的。

对于多数企业来讲，发布出去的岗位都设定了薪资范围。每个岗位做什么事情，准备给入职人员开多少钱，基本上在发布招聘信息时就定了。而且，是个范围。当你看到招聘信息上写着 4000~12000 时，基本上你要想的是 4000 而不是 12000。这是残酷的概率性结论。

用人单位负责谈薪水的 HR 或技术负责人，会拿薪水范围的低端来和你谈，一定是如此这般的，具体请看“薪资，你是我不能言说的伤”（收录于《你好哇，程序员》一书和我的博客）。

当企业给你的薪水低于或接近你的底线时，你不满意。是的，虽然是你可以接受的，可是不是你能欣然接受的，所以你在没有其他选择的情况下，只能带着小情绪接受。这时，你就是不满意的。

注意，很多人都是这么过来的。比如你从一个城市迁徙到另一个城市时，归心似箭的紧迫感就会导致你产生“差不多了，先找一个过去再说”这样的想法。再比如你对眼下的工作环境实在厌倦到了不能多留一秒钟、不能多忍受一刹那煎熬的境地时，有一个单位对

你唱“来吧来吧一起舞蹈”你就“随风奔跑自由是方向”了，到了之后冷静下来就会反刍出不满意来。比如你被迫离职，如同小小的蜗牛背负着房贷、车贷、奶粉钱，不敢有一日空闲，偏偏找了几家单位都杳无音信，此时有一家忽然伸出橄榄枝，你就会觉得“哇哦，就这家了，正合适”，其实对薪水也不是特别满意。诸如此类，情况很多，不能穷举。

入职薪水水深几许

很多单位在面试时会告诉程序员，一年加薪两次，有年终奖，有项目奖金，有项目奖金，摒弃论资排辈、以能力论英雄、只要你能干，涨薪无极限。其实呢……要知道，我那些曾经同事过的程序员和测试人员里，三年不加薪的都有呢，二年不加薪的很常见呢。

这就是实际情况。说实话我觉得这很不合理，除非经验能力和工作年限成反比或者总是把事情搞砸的程序员，我觉得谁都不该摊上这种实际情况。

假如你相信自己被扔在什么环境里都可以凭努力和才华绽放光芒，那么入职薪水真不是事。我或者身边的朋友经历过的企业，卓越的程序员总是有超高的薪水，这一点不容置疑。假如你不那么自信，那么我的建议是，还是好好准备准备，耐下心来，顶住压力，找一个入职薪水让你满意的公司吧。

不满意的后果很严重

其实离职只是小事情，一般都不会导致特别严重的后果。

当你不满意时，就会产生怀疑、猜忌、不安、不满、不平、抱怨、委屈、愤怒等不良情绪，不良情绪都是穿肠毒药，会严重损伤你的身心，让你不能安静平和地做事，让你觉得整个人都不好了，干什么都带着情绪，那就什么都懒得干，什么都干不好。

不满意怠工玩游戏刷朋友圈，这是最严重的后果。

要知道，你不是为薪水工作，不是为老板工作，是为你自己工作。虽然客观上你在别人的公司里拿着老板的钱为老板做事，但实际上，你做的任何事，收获的任何成就、经验、能力提升，都是你自己的，一旦你懈怠了放纵了，损失的就是自己的提升机会，最终损失的就是你自己的价值。这就是最严重的后果：**一旦你因为对入职薪水不满而无心修炼，你的价值提升就会搁浅**。

怎样跳过入职薪水陷阱

最后谈谈如何跳过入职薪水陷阱。

1．是否接受一个不满意的入职薪水

前面我说在某些客观的实际因素影响下，我们会接受一个不太满意的入职薪水。正本清源，要想不让自己失望，就得耐得住寂寞和压力，等待合适的机会。

想清楚你为什么离职，这是首要的。由此定下你的目标，在整个过程中都要紧盯着你的目标，哪怕持续不断地碰壁，也要坚守初心。

搞明白你的压力所在，看自己能承受多长的换挡期。要相信，属于你的机会、能令你满意的机会终会到来，它就在下一个路口等你。要带着这样的信念，顶住压力，坚持往前走，不要被一时的挫折打败。找工作是一件大事，从一定程度上讲，你找什么样的工作，就决定你过什么样的生活，万万随意不得。

在你入职之前，你都有反悔的机会，而且在入职之前反悔，一定比入职之后要好。

我有几次在论坛上看到有人发帖，问“答应了 A 公司，又遇到自认为更好的 B 公司，该怎么办”，这种问题当然让人纠结，心里可能觉得过意不去，对不住 A 公司，可如果真觉得 B 是你的命运，那就果断地和 A 说再见吧。这对双方都好，一旦等到入职了 A 公司再别别扭扭地说不合适得分手，那对双方都是极大的浪费。所以，我现在也能理解我曾经谈好的那些小伙伴已经答应入职了后来又跳票的行为。我想提一个建议，如果你遇到这种情况，不管你觉得多么难以开口，从礼仪上讲，都要给你即将说再见的公司一个明确的回复，一句话不说，电话不接，短信不回，微信无视，这都是对 A 公司不负责任的，对你自己也是不好的。

如果你特别钟意一个公司，又对它提供的薪水耿耿于怀，那也可以谈，多谈几次，在接受 Offer 之前，都是可以谈的，找工作本来就是双向选择，谈到自己满意为止吧。别不好意思，要学大妈早市上买菜还价的劲头，谈好了再约比什么都强。

在换工作时，我们要明确重点考虑的、决定性的因素，让这些因素指导我们是否接受一个机会。如果出于现实的原因，不得不接受一个自己不满意的入职薪水，也请考虑一下这份新的工作是否还有吸引自己的“薪水之外”的方面。

2．已接受不满意的入职薪水怎么办

有一年我找工作，也不懂行情，也是从一个城市到另一个城市，薪水谈了 4500，后来发现被坑了。怎么办呢？

那时我就说服自己踏踏实实干活，程序员要拿工作结果证明自己。我干了几个月，漂漂亮亮地做了个项目，找老板谈加薪，结果就加薪了。

所以，对于“已经接受了不满意的入职薪水”这种情况，我的建议是，修炼自己的心性，告诉自己，你所做的一切都是为自己的价值提升，不是为了别的，不管怎样都要抓住机会好好做出点成绩来，这才是不辜负自己的正确选择。

这种情况下，能不能让自己淡定的重要一步，就是分析当前的公司是否存在自己在意的薪水之外的因素。比如你很看重它提供的大数据这个技术方向，比如你更看重它给你的 Team Leader 这个锻炼机会，比如工作过程让你感到身心愉悦……说白了，有时钱不是那么重要，如果你能审慎考虑，正确认识到这一点，并能从当前公司找到你愿意为之而接受较低薪水的慰藉，那么薪水暂时就不是问题了，你可以期待加薪，有合适的机会也可以向合适的人表达你对薪水的看法。

如果你真的不能安分工作，那就趁早了断，别折磨自己也别耽误公司。最忌讳的就是浑浑噩噩混日子，浪费自己的生命，浪费公司的资源。

三个因素决定你的薪水高低

下面三个关键因素，决定了作为程序员的你的薪水水平：

- 工作内容。
- 工作表现。
- 被替代的难度。

工作内容

从某个角度来看，我们经常听到的“选择大于努力”这句话，就说明了这一点。也就是，不同的行业的平均薪资水平是不一样的，你选择进入 IT 行业还是选择餐饮业，结果就

很不一样。2000 年左右，我还在上学时，这种差别尤其大，学机械、学能源动力和学计算机学通信，毕业的几年内，工资差别就很大。即便在同一个大的领域内，细分的领域也会导致明显的薪资差异，比如你做手游和做建筑类软件，做 Web 前端和做服务后端，在一定时期内，结果都会有明显差异。

OK，这就是“你的工作”这一点的含义：**工作内容本身的差别，会导致薪资差异**。所以，如果你对薪水敏感，在选择行业及行业内的细分领域时，就有必要研究一下行业薪资报告之类的东西。

工作表现

在同一个领域内，做同样的事情的程序员，薪水也会有差异。这貌似是废话，然而这时的差异，就体现在“你的工作表现”上。

在同样的工作岗位内，干同样的工作内容，你做出来的效果比别人好，表现得比别人积极，你能更及时地反馈，你的薪水就比别人高。这就是所谓的“同岗不同酬”。懒懒散散敷衍了事地干一件事，与积极主动尽善尽美地做一件事，差别就很大。比如张三总是不愿意接受任务，写的代码脏、乱、差、不可读、无逻辑，李四总是欣然接受任务，代码规范、自解释、简洁、逻辑清楚、性能优越，那结果怎么样不言自明了。

被替代的难度

软件开发这个行当流动性很强，程序员跳槽现象很普遍。如果干得差呢，不说也知道结果。工作表现好的人很多，在工作表现好的一堆程序员里，还是有拿钱多和拿钱少的区别。这个时候的区别呢，就和“你被替代的难度”有很大关系。

假如你做的工作是核心，业务复杂、很难掌握，或者技术难度大有高门槛，你的可替代性很差，就是说，除了你别人玩不转，那么你的价值对企业来讲就很高。同样都是以积极的态度认真干活，如果张三做的事随随便便找个人就把他替了，那他就一定没有你拿钱多。这就是“替代难度系数”的作用。

那么，怎样才能提高自己的替代难度系数呢？“程序员保值的五个秘密”一节里已经做了简单分析，感兴趣的可以跳过去看看。在这里分享一个新的秘密：如果你具有区分事情重要程度的能力和快速搞定重要任务的能力，那你就很难被替代。因为具有这种能力的

人，不说百里挑一，也是十不有一。

35 岁程序员的独家面试经历

创业失败后，再找工作。选择了三家（两家上市公司，一家即将上市），都走到了关键的节点。我记录了面试过程中被问到的一些问题，希望对自己将来的面试有帮助，也希望对读者有所启发。

我的经历和现状与被问到的问题息息相关，如下：

2002 年毕业。

2002~2005 做售后技术支持，2005 离职转做软件开发。

2009 开始做项目经理，后来做部门经理，期间还有 50%左右精力在技术上。

2014.10~2014.11，项目总监，偏重管理，脱离技术细节。

2014.11~2014.12，主动回归开发岗位，进入 C3 公司。

2014.12~2015.11，离开 C3，以技术合伙人身份与朋友创业，技术总监。

2015 年 11 月 25 号解散团队，重新开始找工作。

下面把我面试三个公司的关键节点和问题列出来回顾一下。我最终入职的单位将从这三家中产生。如果你没耐心看面试问题，也可以跳到最后看我总结的“如何准备面试”。

第一家，和研发总监面谈

我考虑的第一家公司是安防行业的一家上市公司，叫它 C1 吧，西安有分公司。北京的 HR 直接邀请我考虑的，我了解到岗位是研发总监。

HR 和人力资源总监的两轮面试通过后，和西安这边的研发总监面谈，聊到了下列问题。大体的顺序是我列的这样。

1．自我介绍

这个问题一般都会遇到，技术人员会问，人力资源也会问，一家公司的面试流程走下来，可能会自我介绍 2 次或更多。

我基本是从大学毕业开始，介绍到现在。介绍时会涉及工作经历和关键的项目、产品经历，如果有与目标岗位或公司产品相关的经历，就重点提到。

2．了解这个公司吗，了解哪些产品

C1 的公司官网我浏览过几天，研究了产品，还了解了另外两家安防类上市公司。

年龄大了记性差，被问到这个问题时，居然把看过的产品线的信息给忘了……

3．了解这个岗位吗

HR 告诉我是研发总监，后来研发总监告诉我是产品线负责人，对应研发副总监，有一些偏差。

我表示不太了解。面试官介绍了一下这个岗位的情况，包括负责的产品、职能、基本工作情况等。

4．为什么来应聘这个岗位

我说是 HR 主动找我，没怎么展开。

现在看来我当时的回答很糟糕。以后碰见这类问题，应该结合自己的经历、商业价值和将来定位，再加上对岗位职能的了解，将回答聚焦在个人与公司的匹配性上。

5．谈谈你对这个岗位工作的理解

产品线负责人，会负责研发、测试、UI/UE 这几部分技术团队的管理、产品的研发，这是产品线负责人直接管理的团队。

C1 是矩阵式管理，销售平台可以有选择的销售产品，市场和销售应当被看作是产品研发部门的客户，产品线负责人需要和市场、营销、销售等部分保持密切合作，共同促进产品销售。研发部门和市场、销售部门会共同对年度业绩指标负责。

售后、售前和技术支持团队也是必须协同工作的。

我原来做产品开发工作时干过类似的事情，其理解偏差应该不大。

6．你找工作时考虑哪些行业，为什么

我说了三个，安防、企业应用、互联网。简单从行业发展周期方面谈了理解，还谈了

行业与企业的关系，企业与个人的关系。

在处于上升期的行业里，处于上升期的平台上，个人才有比较大的发展的可能性。

这样的理解应该没错。

7．你的技术积累在 C++方面，我们团队偏重 Java，有没有障碍

我从“技术积累到一定程度是相通的”这个角度回答，我自己也用过 Java。另外从团队管理角度也做了解释，我把控的是与产品相关的技术方向和方案，特别细的技术细节不需要了解。

8．如何管理你的团队

我谈了两点，一是授权，二是参与感。

9．这个岗位需要出差，可以接受吗

我明确表示如果经常出差就不用往下谈了。

然后我们讨论了什么算是经常出差，比如“一年两三次，一次一两个月”“一月两三次，一次两三天”等。

C1 在全国各地将近 20 个办事处，负责销售和技术支撑工作，产品线负责人每年都要拜访所有办事处，维护好各种渠道和关系。频繁出差是不可避免的。

这个问题在对方看来，我的态度显然是不大乐意接受的。

10．如何向上管理

我只是听说过“向上管理”，具体不了解，只从目标管理这一点上谈了谈。

这一点回答得较差。

11．如何避免项目延期

这个问题是上个问题带出来的。

我从目标及交付期设定是否合理、交付期本身是动态变化的需要以迭代观点理解这两方面谈了不少。

复盘时发现，我这种理解从客观的角度看没差，但明显不是对方想要的答案。

12．入职后第一个月你准备怎么开展工作

这可能是经常被问到的问题。我从了解目标、了解团队及周边团队三方面谈了一下。

13．你最擅长的是什么

我工作十几年，各种技术搞过，各种管理工作做过，面对这种问题，有点不好回答。

一般一个人必须对自己有清晰的认知，然后对目标公司和岗位有足够的了解，选择自己擅长的点来展开，这样能提高人职匹配度，加大通过面试的几率。

我选择了技术方面的一个点来说。

这一点表现得不好。

14．你想做客户端开发还是 APP

这个问题应该是上个问题带出来的。我从移动互联网的趋势着眼，谈了几句，告诉对方我偏向 APP。

后来面试我的研发总监补充解释说，问这个问题没别的意思，过一阵子公司可能有这方面的岗位放出。

告别 C1 之后，我心里已然清楚不会有下文了。并且我也明白研发总监的意思——我更适合做开发工作。没错，我对自己分析的结果也是如此。

这是我在 C1 的一次关键面试，历时 90 分钟。面试官是研发总监、西安分公司老大、两个产品线负责人。我在“是否了解公司产品”“能否接受出差”“向上管理”“如何避免项目延期”这个几个问题上表现不好。后来我主动和 HR 说目标岗位要经常出差不适合我，如果有开发岗位可以再谈。

第二家，与技术负责人视频连线

第二家简称 C2 吧，是建筑软件方面的上市公司，西安有分公司。我应聘的是高级软件开发工程师，通过猎头接洽的。

就我的经历来讲，这里面有个非常重要的问题：从管理岗位回到软件开发岗位。面试中一定会被问到。

一面是西安这边目标部门的技术负责人，人挺 nice 的，聊得也比较顺畅。

二面的面试官在北京，是更高级别的技术负责人。我们通过 QQ 视频连线进行。他可以看见我，我看不见他。

谈到的问题，我努力按顺序列在下面。

1．自我介绍

为什么简历上都有，大家还是会让你做自我介绍？假如你反问“简历上都写清楚了，没必要吧”，那你被 PASS 掉的概率是 90%以上。

通过自我介绍，可以看出很多东西，比如你的逻辑思维、表达能力、概括能力、现场把握能力、价值观、你对自己的认知及适应性。当然也有的面试官没来得及看你的简历，会在你做自我介绍时快速翻简历。

我这次介绍时提到了为什么去创业，为后面做了铺垫。我有预期，从管理岗位回来做开发工作，有几个问题一定会被问到。

2．对将来的定位

这是我预期中的问题，别人看到我的工作经历也必然会问，因为我应聘软件开发岗位，从常规角度看不利于职业发展。

我通过一些经历和感受谈到对自己的职业定位，把定位放在“专业技术者”这一点上，走技术路线将是我未来的方向，我会放弃做管理。

3．成就感事件

我谈了几个，比如写作《Qt on Android 核心编程》和《Qt Quick 核心编程》，再如成功研发机顶盒产品。

4．在开发多媒体产品时，遇到过什么样的技术难题

我描述了遇到过的三个问题，没详细讲怎么解决的，这一点不太好。

5．在上一家公司时的薪酬有多少

我参与创业，薪酬不必说了，我把之前在 C3 的薪酬如实说了。

6．你有没有什么问题

这是经常会被问到的问题，也是发挥自己长处的机会。

我问了三个问题：

- 房地产行业下滑对公司的影响。
- 分公司搬到高新区的计划（我在高新区，公司不在，路上时间很长）。
- 公司内部技术人员的上升通道。

对方都一一作了回答。对方对第一个问题的回答很赞，说了三点：

- 国际化，房地产在全球来讲是区域性的。我应聘的也正是国际化部门。
- 从卖软件（Licence）转向服务租用，门槛变低，客户会变多。
- 基于房地产客户的特点，开发金融相关的服务。

面试结束后我被告知 5 个工作日内 HR 会给我反馈，我觉得应该是能通过的。当然，我的感觉有时准有时不准……对方怎么想，其实我们不太能了解到，只能通过分析面试过程中双方的表现来估计。不过，如果自己表现糟糕，相信一定可以感觉到结果。

重回 C3 时的面试经历

我做了一个决定，试试看能不能重回离职创业前所在的那家公司 C3。于是我联系之前的同事，就有了一次机会，也有了下面要谈的这些面试问题。

1．来自 VP 的面试问题

我和 VP 用 C3 的云会议系统远程面谈，他可以看见我，我看不见他。VP 人很 nice，整个面试过程像聊天。

你未来 5 到 8 年的规划是怎样的？

这个问题其实和前面讲过的另外一个问题——将来的定位——类似。我谈到了职业定位，还聊到了技术人员的年龄及未来。

你最擅长的技术方向有哪些？

C++和 Qt。

你给自己的定位是什么？

我对将来的定位是应用技术专家。高深的算法什么的干不了。

你创业时做产品（APP）用的是 Qt 还是 Native？

安卓和 iOS 都采用 Native 方式开发。

为什么没有选择 Qt？

我们的创业产品没有采用 Qt，有两个原因：一是我组建团队时已有一部分开发工作在进行，是直接用 Java 开发 Android APP；二是因为 Qt 的体积大，会提高我们的产品抵达用户的门槛。

在技术上的积累有哪些？

这种开放性问题，每个人都有自己的说法。我围绕具体的技术、技术阅历和经验等方面做了一些介绍。

介绍他管理的四个团队，问我愿意到哪个团队。

这说明 VP 已接受我。因为之前了解到 C3 并无招人计划，我表达了我的看法：如果真的没有招募开发人员的计划，不必特意因我为难。我不知道这样做是好是坏，很自然地这么做了，这跟我个人的性格有关。

2．来自人力资源总监的面试问题

技术 VP 之后过了一天，C3 的 HR 告诉我人力资源总监要和我聊聊，还是以远程视频的方式。还是对方看得见我，我看不见对方。

我回顾了当时被问到的问题，比技术 VP 的问题略显尖锐一些，不过从企业角度来讲，面对我这样的二次回归者，有这些问题非常正常。易地而处，我也会问出类似的问题。所以，我丝毫也没有因为这些问题而感到不爽。

自我介绍。

这个好像技术 VP 也问过。

我简要介绍了个人经历，有两个地方做了较多的描述。一个是我从技术支持转做软件开发时的情形，谈到了当时接受我的企业和上司对我的影响。另一个是之前从 C3 离职创业时，拉我合伙创业的人与我的关系，就是当时转行做软件开发时引我入门的那个人。

我要再回 C3，这些问题不可避免会被问到，我先自己澄清一下应该比较好。

之前为什么选择 C3？

嗯，下个问题一起讲。

为什么离开？

好，这个问题和上个问题其实是相互呼应的。我当时选择 C3 是因为看好 C3 的产品，同时我的技术又能在这里发挥出来。

为什么离开呢，其实自我介绍那里已经提过。这里很自然地展开来讲了一下。

还会因为有朋友拉自己去创业而离开吗？

回答了“为什么离开”，那么这个问题可以说是自然而然会被问到。我谈到自己适合做什么，明确了自己的想法，也给出了回答。

之前待过的时间里，觉得 C3 存在什么样的问题？

我提了一个开发团队异地协同工作、效率较低的问题。

为什么回来？看中了什么？

虽然之前在 C3 只工作了一个月，但在技术、产品、人际等方面感觉挺好。这是我想再次回到 C3 的原因。

我看中的主要是企业的发展和产品的定位。谈了我对行业、企业的看法，仔细聊了对 C3 目前三个主要产品的看法。

创业这一年，有哪些收获？

收获主要是个人的阅历、成长，以及对自我职业定位的澄清。

你这么多年的积累在哪里？

说实话这个问题怎么回答的我已经忘了……应该是围绕技术、经验两点来谈的。

你对自己的定位是什么？C3 可以满足你吗？

我从第一次进入 C3 之前的管理岗位感受讲起，讲到创业时的感受，导出了自己的职能定位——专业技术人员，将来往技术专家方向发展。

经历了 C1 的面试之后，我更确定了这一点，没有丝毫犹疑了。

我从对 C3 这个公司和产品的理解谈了我和 C3 的匹配问题。

你觉得自己适合做哪个岗位？或者你的优势在哪里？

因为我对具体的岗位职责不了解，所以从技术和产品意识两个方面谈了我自己的优势。

你还有没有什么问题？

因为是再次回归，其实我没什么特别的问题了。但对后面的面试流程不太清楚，所以就提了一下流程的问题。

你对薪水的期望？

聊上个问题带出了这个问题，我的回答是：和以前差不多就行。

如何准备面试

虽然我面试过很多人，也被很多人面试过，但我个人不是特别擅长面试。这次求职，因为年龄大了相对慎重，想得稍微多了一些。我留意了面试过程，也做了一些反省，收获了一些个人经验，和大家分享一下。

下面几点对面试很重要。

1．个人的职业定位

看我的经历，每家都逃不掉这方面的问题，如果自我定位不清楚，面试时肯定出问题。

2．个人与企业的匹配度

这方面的准备是必需的，你应聘一个职位，要了解企业对这个职位的要求，要看自己的知识、技能、经历与企业的要求契合的点在哪里。

多数企业招人时首先考虑的是人职匹配。假如你挖掘不出匹配的点，可能连简历关都过不了。

3．了解企业

要尽可能多地了解目标企业，比如它所处行业的状况，它的产品、文化，它的竞争对手……越多越好，别怕花功夫。我面试 C1 时就忘掉了一些产品信息……

面试官会更倾向于接受对自己公司有了解的求职者，这说明你的意愿强，而你愿意了解，了解了之后还来，还说明你和公司的匹配度相对较高。

4．个人形象

颜值不可更改，衣料好坏也不重要，整体上做到干净、整洁即可。记住，没有人有义务透过你凌乱邋遢的外表去发现你的内在，也没人有你爸妈那份耐心。

5．心态调试

心态调试有两个方面。

一方面是对多久找到工作的预期。因为经济或他人的期望带来的压力，求职者容易焦急，失去平常心态，产生胡乱先找一家干着的想法，此时面前有根稻草都可能被当作橄榄枝，很容易导致选择不慎，害人害己。我的想法是要沉下心来，慢慢来，坚信一切都来得及，美好的相遇一定在等着自己。

另一方面是关于企业和求职者关系的。这里没有谁强势谁弱势，企业和求职者是双向选择，不必紧张也不必焦虑，保持从容淡定的心态才能更好地展现自己。

而对于像我这种过了 35 岁的程序员，还有年龄带来的问题需要调试。我相信，经验和阅历同样产生价值，技术专家也可以像老中医那样越老越吃香。

培训机构毕业的程序员被歧视的背后逻辑

现在，像达内、华清远见、国嵌、北大青鸟、传播智客等 IT 培训机构很多，为尚未毕业的大学生、毕业了一时找不到工作的大学生、工作后想转行的再就业者提供了一个掌握新技能的机会，通过三个月或半年或更久的培训，你就可以掌握某一种技能，比如 Android 开发、Java Web 开发、iOS 开发、嵌入式 Linux 开发等。

然后有的机构会推荐你就业，有的机构会放你出去闯荡江湖四处碰壁……形式不一，但一段痛苦的旅程从此开始了，这倒是真的。

很多单位歧视培训机构毕业的学员，你所在的单位是这样吗？或者你从 IT 培训机构毕业后，找工作时被鄙视了吗？

为了弄明白为什么 IT 培训机构出来的程序员在找工作时经常遭遇不平等对待，我们需要弄明白“教育”和“培训”的差别。

教育和培训

大致上讲，我们所说的“教育”，指的是掌握一般性的原理与技巧的过程；而我们所说的“培训”，不过是学会某种特定技能的过程。

上面是温伯格在《程序开发心理学》中说到的，符合大多数人对“教育”和“培训”的理解。

你可以通过培训机构学会理发、做饭、修汽车、写代码、做蛋糕，这没什么稀奇的，当我们接受培训时，就是为了某项技能而去的，不是吗？我去蓝翔，难道不是为了开挖掘机吗？

通常我们认为培训机构（学校）是学习某种不太复杂的谋生技能的摇篮。这种技能还有一个特点，就是相对稳定、变化不是特别频繁，能够在相当长一段时间内保持基本的稳定性。所以，一旦你获得了这项技能，就可以靠它吃饭吃上一阵子，一年半载，三年五年，十年八年，都有可能。

而教育通常被认为是一项基础性的工作，重知识，重原理，周期长，见效慢，与社会脱节，有时还能把人变傻（注意我不是讽刺现行教育制度，也不是诋毁我们伟大的大学教育）。虽然如此，很多 IT 公司的基础部门还是被受过正规大学教育的朋友们占据了重要岗位。因为大家普遍认为，虽然学校教育严重脱离社会现实，但名牌大学的学生智力水平、学习能力，平均来看还是高于未能考上大学的中学生，更适合于从事某种对智力、学习能力有些特别要求的技术岗位。

没错，程序员正是这样的岗位。

然而，教育和培训的差别并不是关键。关键是，出于某种原因，面试官或公司主管对出身“培训机构”的人有偏见。为了说明这一点，我们先要看看程序员需要的特殊能力。

程序员需要的特殊能力

成为一个合格的程序员，需要以下“特殊”能力：

- 自知之明。
- 自我学习。
- 努力。

看起来没什么出奇之处，也许你会觉得一个程序员最重要的能力不是上面三项，没关

系，一千个观众就有一千个哈姆雷特……

大家公认程序员从事的是烧脑性工作，行业发展日新月异，各种新语言、新技术、新框架、新概念层出不穷，需要程序员时刻保持归零的学习心态，持续不断地保持学习维持竞争能力和价值。所以，我也是从这个角度出发，选择了前面提到的三点，实际上这三点指向的是“学习之道”。

1．自知之明

我们要了解自己拥有什么、缺乏什么，然后才能开始学习。

通过不断地总结、回顾自己做过的事情，我们就可以慢慢了解自己的能力边界。哪些事情做好了，好在哪里，为什么好在那里而不是别处，是由你自身的哪种行为、才干、能力决定的？哪些事情做得不好，坏在哪里，为什么坏在那个点而不是其他的点？改善你自身的哪种行为或能力可以改变事情的走向，还是说你没什么能改变的，那就是你的局限？

当你了解自己之后，就能发现自己应该做什么，就能决定自己的学习方向，而不是盲目地把自己交付给别人（学校、老师、家长、培训机构），因为没有什么人可以真的为你负责，能为你负责的只有你自己。

当然，自知是最难的事，也不是一朝一夕的事，需要不断地自省和内视才可以做到。

2．自我学习

知名的教育专家林格有两本非常著名的书，《教育是没有用的》和《学习是不需要教的》。林格有一个非常核心的观点：学习能力是人与生俱来的能力，是人之天赋，是不需要教的，但这种能力会随着年龄的增长和家庭教育、学校教育的误导而萎缩或消失，所以，教育的方向就是营造一个环境，让人自己发现自己的学习能力。

很多人不是天生不会学习，而是在成长过程中在家庭、学校、社会的各种外力撕扯中慢慢丧失了自我学习能力。更有甚者，多数人不自知这种能力的丧失。

然而，程序员尤其需要自我学习能力。

学校会教你操作系统原理、计算机组成原理、算法、C 语言、Java……各种知识都会教你。然而在工作中，你能不能用你学到的知识解决问题，实在是个未知数。

培训机构会教你怎么写 Java 代码，怎么安装某个 IDE，怎么完成一个个人博客或购物车之类的小项目。然而，你是在框好的架子下被动地按照老师的要求“完成”了这些事。在实际工作中，面对陌生的项目，你能不能举一反三灵活运用，充满了未知。

一个程序员能不能自己学会一门技术、能不能自己解决一个问题特别重要。因为很少有主管会手把手地教会你编程和设计的实际技巧，也很少有主管会大发慈悲把你送去研习班学习工作需要的技能，在一个现实的环境里，一切都要靠你自己。假如你自己不能独立习得某项必需的技术，真没有人能帮得上你。假如你自己不能独立解决问题，真没有人能始终拉扯着你。这也正合梁漱溟说的话：“任何一个人的学问成就，都是出于自学。学校教育不过给学生一个开端，使他更容易自学而已。青年于此，不可不勉。”

很多人轻视培训机构，是认为培训机构的老师，多数脱产，没有丰富的一线工程实践，是业余选手，而这些“业余选手”却要通过短短的一期培训来为社会培养“专业选手”，这基本是一个笑话。即便我们能举出不少从培训机构出来的优秀选手，那也只能说明，这个选手本身具有很好的学习能力。

如果一个人意识到了自己还具备自我学习能力，那么他完全没必要去培训机构浪费动辄上万元的学费——他完全可以自己学到必需的知识和技能，假如他真有兴趣的话。

而要检验你是否还有自我学习能力，先不要去培训机构，自己找本讲编程的书，找台电脑，连上网，花一两个月时间就能搞明白自己是否适合做一个程序员。一旦你通过了这种自我学习实验，那时再挑一个培训机构系统地学习某条技术栈不迟。

总而言之，你能不能成为合格的或优秀的程序员，取决于自我学习能力，而不是参加过专业培训。所以，很多公司在招募程序员时，不太愿意考虑培训机构毕业的学员，因为在面试官的心里，觉得如果你有能力，自己就可以学会，完全没必要去培训机构，你接受了培训，他反倒认为你可能缺乏自我学习能力（以及对技术的兴趣），担心你不能胜任将来的工作，他太了解了，你学的那点东西根本不够用，还有很多新东西等着你学，所以，他不愿意考虑你。

而对于知名大学的毕业生，虽然可能和你一样是一张白纸，但别人会以为，能进得了大学当得了学霸，起码学习能力没问题的概率高一些。

3．努力

有一句话是这么说的，“以大多数人的努力程度之低，根本还轮不到拼天赋”。其实，努力也是一种天赋。为什么有的人明知努力可以改变生活，可他还是不努力呢？因为做不到啊！为什么做不到？因为他缺乏“勤奋”“努力”之类的天赋和才干。真的，勤奋、努力的人，多数是生就的，少数是为生活所迫。

程序员白天要上班，晚上偶尔还要加个班，自由时间少，而新技术很多，什么时候去

学呢？你以为实际的项目一定可以让你锻炼新技术吗？要知道，大部分的项目在技术选型时，会考虑技术的成熟度和团队的技术储备，很少有冒险采用大家都不熟悉的技术的，不可控因素太多，风险太大，项目失败的概率很高。那这样的话你在什么时间丰富自己呢？

八小时之内是现在，八小时之外是将来。你可以用的，就是你的业余时间了。你看，人家都在打游戏、看电视，你还要苦哈哈地学习，如果你没有“努力”这种天赋，是很难做到的。

所以，很多面试官在面对培训机构毕业的人选时，也会有诸如“如果你有自我学习能力并且努力，其实没必要上什么培训班，完全可以自己搞定”之类的想法，“你上了培训班，是不是反过来证明你不够努力咧”……然后，他又会想“是不是因为你没自知之明，不知道自己要干啥才被忽悠到培训机构去交学费了呢”……也许，他还会想，“是不是因为你对技术其实不感兴趣只是想谋个事干呢”……

你看，面无表情之下，其实各种想法如同暗流漩涡，澎湃不息……所以，最后，你可能只好“回去等消息”了……

不知道说了这么多，你是否明白了个中原因——面试官会觉得培训出来的学员，可能对技术没有那么浓厚的兴趣、缺乏足够强的自我学习能力，也不能很好地自律和努力。但在我的观念里，其实应该这么看待 IT 培训机构的学员：他只是找了一个类似学校的地方系统学了一些东西，和别人并无什么不同，如果他对技术有兴趣，有自我学习能力，一样可以做一个优秀的程序员。

成 长 之 路

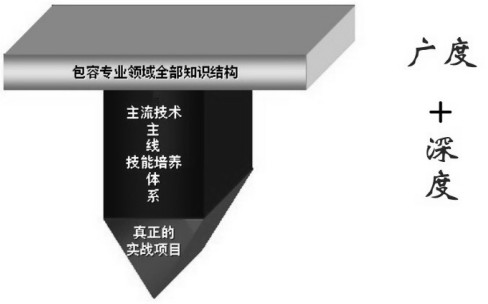
两招让你成为卓越的 T 型人才

有个小伙伴在微信上问我：

我刚工作半年，有时候对于 Java 的发展方向有点迷茫，Java 的范围实在是太广了，我有时候会不知道从哪开始入手，我想问一下，您有什么好的建议吗？

我理解这位朋友的问题是：工作中该如何发现自己要学什么，怎样构建自己的技能树。这是一个非常重要的问题，假如你不知道要学什么，那么一旦你工作不那么忙时就会觉得无聊，只能通过浏览新闻、看小说、打游戏等方式消磨时间消耗自己。这对自己的成长和增值非常不利。所以，下面就来谈谈“工作中学什么”这个话题。

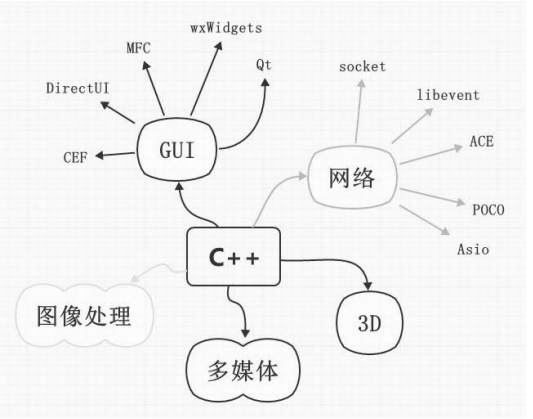
根据我的经验，在开发过程中有两种学习策略：一种是横向拓展，也可以称之为广度学习；一种是纵向深入，也可以称之为深度学习。如果能贯彻这两种学习策略，就会慢慢形成程序员最理想的技能结构——T 型，即在横向上熟悉足够多的知识和技能，在纵向上又精通某一领域。简单说就是“一专多能”。



广度学习

以 C++语言为例，如果你做网络方面的开发，就会遇到选择哪个框架的问题，libevent、ACE、Asio 还是 Qt Network？如果你做 GUI 开发，可能会在 Qt、CEF、MFC、wxWidgets、

DirectUI 之间做个选择。



选择和了解的过程，是个人进行广度学习的过程。你会快速了解每一种框架的优缺点，会搭建环境，会写一些 Demo 来对关键技术点和需求做验证……最终会根据文档和实验结果来做出实际的选择。这个过程非常难得，会快速拓宽个人的知识面。建议把每次了解到的技术框架都记录下来，闲暇时可以进一步学习。

即便你不是预研技术方案的工程师，而是别人定方案你来开发，也可以有意地让自己经历这个选择过程，只需要问自己一个问题即可：为什么用 A 而不是 B 或 C？

其实不单单是大的技术框架选择，小到某个页面要用的某个元素，都会经历选择的过程，有心，就可以用这种以点带面的策略学到更多知识。

深度学习

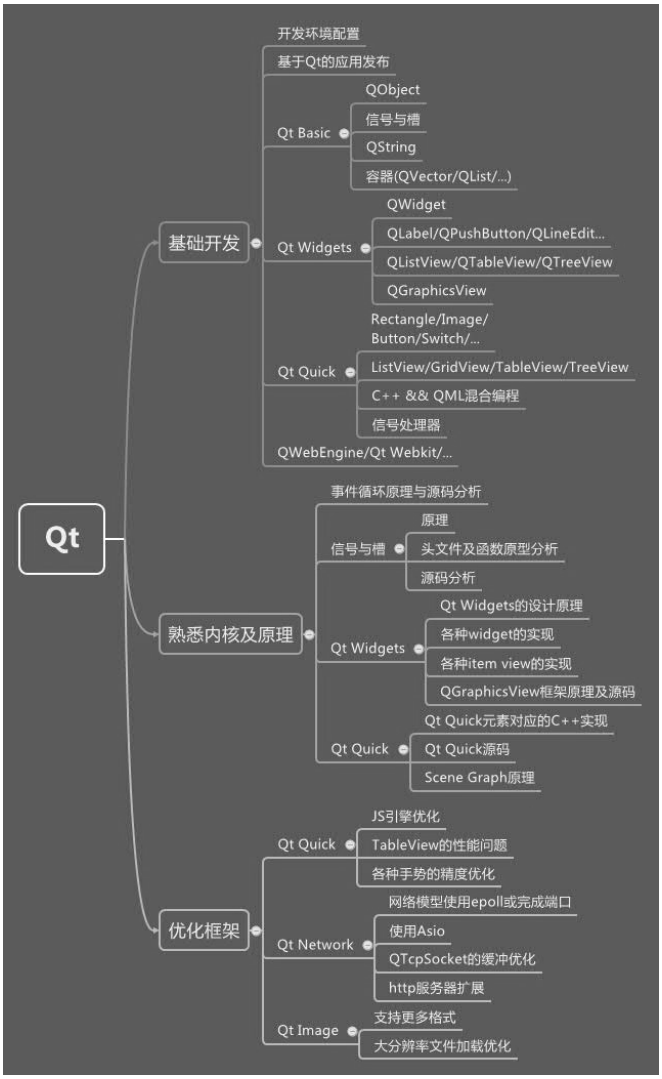
大多数时候我们会使用一门语言和一两种技术框架进行软件产品的开发，此时就是纵向深入学习的好机会。

对技术框架的学习，大体分三个阶段：

- 基础开发，主要是了解 API、基于 API 完成应用。
- 熟悉内核及原理，主要是了解框架的设计原理、阅读源码、洞悉内在机理。
- 优化框架，主要是针对框架的已有功能的不足进行完善、优化，或者使用框架提供的机制扩展框架功能，或者对框架进行定制，让它适合特定情境。

我以 Qt 为例画了张图，供参考：

很多时候我们经历了第一个阶段——能够使用 API 进行简单开发——之后就跑别的地方去了（工作需要），然后就把这个框架搁置不管了，一直停留在那个阶段。建议有时间梳理一下自己用过的技术，挑出当下工作中还在用的项，往深里钻，去熟悉原理及内核，若有可能，也可以优化、扩展或定制。唯其如此，才能真正掌握一个框架，才会有深度，在该项技术上形成自己的价值和竞争力。



小结

不管是广度学习还是深度学习，都是从实际需要出发的，是你工作中要用这门技术而不得不学。明白这一点非常关键——你要用到某项技术时才会有学习的动力。所以，最好的学习时机就是一边开发一边学习，白天上班晚上学习，周内上班周末学习——在有明确目标时最有学习动力，在热度还未退却时学习效果最好。

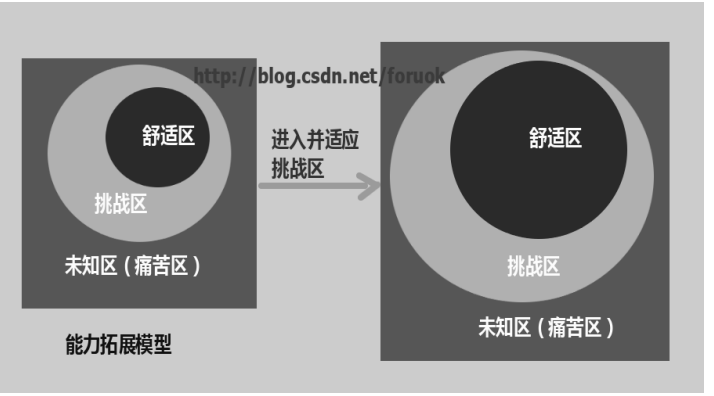
如果你想学习与当下工作内容无关的知识和技能，投资自己的未来价值，最好的办法是给自己设计一个软件产品（或者参与到别人的项目中），设定一个期限，用你想掌握的技术去完成它。用不到的知识等于没有，这就是这种说法背后的原理。

所以，没有需求，要创造需求，没有问题，要创造问题，有了需求和问题，学习才更有方向感和目的性，才会有持续下去的动力。如果你能主动创造目标和需求，那么你很快就可以建立自己的技能树并让这棵树根深叶茂，你就会越来越值钱，也越来越能赚钱。

程序员的能力拓展模型

听人说到一个词，叫作“Comfirtable Zone”，中文是“舒适区”。

这个词让我瞬间联系到程序员的能力边界问题，我画了能力拓展模型图如下：



我觉得这个能力拓展模型适用于一个人的方方面面，但这里我只谈及程序员。

能力拓展模型

前面的图中左侧是我们的现状，最内层的圆形是我们感到舒适的区域，我称之为“舒适区”。

以程序员为例，当程序员的技术能力和解决问题的能力达到一定水平之后，就能够轻松胜任某些开发任务，解决特定实际问题，给用户带来某方面的便利。他的能力与他接触到的问题匹配，此时程序员处于舒适区。这个舒适区的大小是由他解决问题能力的大小定义和界定的。

当问题超出程序员现有技能和经验时，他能看到并了解，但还不能解决。那么这些问题就是这个程序员随时可能面对的挑战区，也就是图中的外侧圆形。

图中大圆之外的区域，对程序员来讲就是未知区域，未知即迷茫，未知即痛苦。这个区域往往是程序员看不清或看不到的，是百慕大三角的一片未知而神秘的区域，贸然跳入，可能折戟沉沙、铩羽而归。

假如一个程序员愿意跳出舒适区，踏入挑战区，接受一定的不适，那么他就有机会拓展他的能力，将自己的舒适区扩展得更大，他的能力模型就变成该图右侧部分，舒适区变大，挑战区变大，痛苦区变大，这也是符合人类认知规律的：知道的越多，不知道的越多。

如果一个程序员连轻微不适都不愿意接受，那么他就会渐渐固步自封，落后于别人，落后于时代，渐渐被这个日新月异的时代所抛弃，成为一个别人眼中没什么用的老家伙。

在开发过程中扩展舒适区

一个程序员的能力是可以通过锻炼不断变强的。就像人的肌肉，在一段时间内让锻炼强度超负荷一点，适应、习惯了，肌肉变得比原来强了，就再超负荷一点，通过这样的螺旋式递进，肌肉就会越来越强。程序员也是一样的，你的学习能力、写代码能力、设计能力、沟通能力、管理能力等，都是可以通过锻炼来加强的（我们得考虑一个人适合做什么，如果他没有某方面的才干，虽然通过锻炼也可以加强，但违背天性的事通常会事倍功半）。

在软件开发过程中，一个程序员会什么语言、懂什么框架、水平如何，自己心里有数，项目经理通过他的表现也认为自己心里有数。那么在有新的项目要做时，通常的做法是，哪个程序员熟悉实现 Tx 任务相关的技术，就让哪个程序员做 Tx 任务，这通常又是出于交付期、生产率、成本等各方面的考虑。

在这种情况下，每个人都做自己驾轻就熟的事情，对整个项目来讲自然是最经济的。可是对程序员自己来讲，却是不经济的。因为你无法接受新的挑战，能力边界的拓展会很慢。所以，合理的情况是，项目经理在划分任务时要对程序员负责，既给一个程序员能轻松完成的任务，也要给他需要费点劲才能完成的任务，通过具有挑战性的任务来锻炼这个程序员，让他更好、更快地成长。但是这样做的管理成本太高，所以，现实当中，很少有公司的项目经理会主动这么做（没合适人手接受某个任务时会被动这么做）。

鉴于这种现实，作为程序员自己，如果想更快地成长，就要表现得勇敢一些，主动走到挑战区域，去抢具有挑战性的任务（如果不好意思主动，也可以在有挑战性的任务落在自己头上时欣然接受）。一旦拿到了对自己来讲具有挑战性的任务，就赚到了。没错，你赚到了。虽然你会为此殚精竭虑，可能为此加班，可能为此在别人看不见的地方付出，但是你拥有了机会和更多的可能性，如果你顺利完成，那么你的舒适区会扩大，接触新挑战的机会也会变大，你就进入了良性循环，越来越强大。

想想看，这是多么美好的事！

所以，某个语言没学过？不是问题。某项技术没搞过？不是问题。软件结构太复杂，一时掌控不了？不是问题。业务不熟悉？不是问题。如果你觉得这些是问题，请坐看云起，等待时光带走一切然后把你丢在原地。

在渴望成就自我的程序员眼里，问题即机会。只有抓住机会，解决问题的能力才会在痛苦的历练中像雪球一样越滚越大。

这 8 种武器点亮程序员的个人品牌

提到段誉，我们会想起凌波微步和六脉神剑；提到乔峰，我们会想起降龙十八掌；提到王语嫣，我们会想到她惊为天人的容貌和熟知各门派功夫的渊博知识；提到欧阳锋，我们会想起蛤蟆功；提到李寻欢，我们会想起小李飞刀……

提到 Linus Torvalds，我们会想起 Linux；提到雷军，我们会想起 WPS 和小米；提到 Bill Gates，我们会想到微软和 Windows；提到王江民，我们会想起江民杀毒软件；提到侯延堂，我们会想起网际快车……

每一个人，不管是小说中的还是现实中的，被提起时，听者都会联想到和这个人相关的一种或几种能代表他身份的东西，这是身份认同，也是品牌效应。虽然身份不等于人，一个人可以有多重身份，身份只是人内在能力的结果，但我们在实际生活中，却往往以身份论人、挑人、交人。

作为程序员，如果能有自己的身份和品牌再好不过，你走到哪里，不管是找工作还是参加活动，都能以某种方式被人记住。所以，个人品牌对于程序员来讲也是非常重要的。

所谓程序员的个人品牌，按我粗浅的理解，就是这个程序员所做的那些让人觉得他有独特价值的事情、产品、活动。互联网创业者面对投资方时，不可避免地都会被问到开发团队的构成、曾经做过的产品等问题，也说明了投资人看待团队的角度。假如你的研发团队成员来自百度、阿里、腾讯、网易等公司，就很容易让投资人生出信任的感觉。其中的道理不言自明。

总结一下，看看程序员该如何经营个人品牌。主要有以下几点：

- 产品
- 所在公司和团队的背景
- 开源项目
- 技术博客
- 出版技术书籍
- 持有专利
- 证书（各种工程师证书、获奖证书、学历证书等）
- 口碑

产品

虽说让程序员纵马江湖的个人英雄主义时代已经过去，一个程序员个体已经很难独自完成重量级的产品，但独具特色的产品依然是一个程序员最好的证明。

- 全球第一个走红的 P2P 音乐交换软件 Napster，它的创始人是肖恩·范宁。
- 知名杀软 McAfee 的创始人，是约翰·麦克菲。

- 最早的 P2P 下载软件 BitTorrent，作者是布莱姆 • 科恩。
- 豪杰超级解码梁肇新，UCDOS 鲍岳桥，江民杀毒王江民，网际快车侯延堂……

这些赫赫有名的程序员，都是因为他们的产品而被人所知。也许他们离普通的程序员太远，但道理是相通的。作为一个普通的程序员，如果你参与过一个知名软件产品的开发，毫无疑问这会给你的履历增添一抹亮色。进而，如果你独立开发了一款颇有影响的 APP，那就更会让人刮目相看了。

所在公司和团队的背景

我们写简历时，都会写上工作经验。招募方挑选简历时，也会浏览工作经验。如果你所在的公司或团队很牛，往往会给招募方带来深刻的印象。有些单位在招人时都是盯着某个公司的研发团队来的，甚至有的直接去目标公司、目标团队挖人。

有一款社交软件叫碰碰，有一阵在微信里也疯传过。碰碰的团队曾通过在微信里投放一些社交小游戏来转化用户，其中有一款小游戏名字叫作“你懂我吗”，你可以发给你的朋友几个个人喜好问题请他回答，看他有多了解你，结果你会发现，你的大部分朋友都回答不出你提供的诸如你的生日、爱玩的游戏、爱吃的水果、爱看的书之类的问题。

是的，人和人之间是很难相互了解的。熟人之间尚且如此，更不用说陌生人了。仅仅靠几分钟或几十分钟的面试，就能真的了解到一个人吗？非也！

正因为人和人之间难以了解，程序员的工作背景才会变得很重要。假如你在 IBM 公司的云计算团队工作过，那么华为就可能找到你把你挖走，而至于你这个人是否真的很牛，其实人家一时半会儿是了解不到的。所以，这个时候，是公司和团队在为你的个人品牌背书，招募方看的是背书者的影响力。

所以，作为程序员，如果你有机会到大牛公司的大牛团队工作，会对你的个人品牌及后续的发展有很大帮助。

开源项目

开源改变世界，这不是一句空话。

很多程序员在开发软件时，都喜欢先搜索一下有没有开源项目可以拿来用。说实话，国内很多大牛公司的大牛软件都是在开源项目的基础上搞起来的，比如做视频的会去用

ffmpeg，再如做视频会议的多数都是从 webrtc 改过来的……

假如你能有一个被人喜爱和传播的开源项目，那么你一定会被别人刮目相看。退一步讲，你自己没有这样的开源项目，但是能参与到一个知名的开源项目中去为其贡献代码，也是值得一说的事情。

Node.js 是开源项目，它的创始人是 Ryan Dahl。后来这个开源项目非常火爆，Joyent 这个公司注意到了 Node.js，决定赞助这个项目。Ryan Dahl 于 2010 年加入该公司，全职负责 Node.js 项目的开发。

关于 Node.js，后来因为 Joyent 的管理问题，几个重要的开发者出走，另立山头，开始了一个叫作 io.js 的开源项目，其社区一度非常活跃。不过，2015 年 9 月 15 日，io.js 和 Node.js 再度合体，Node.js v4.0.0 发布了。

虽然我们不可能像 Ryan Dahl 一样牛，但参与开源确实也是你构建个人品牌的途径之一。

技术博客

其实技术博客和开源是类似的事情，它们都能在一定程度上说明一个程序员的分享精神和技术能力。假如你有一个非常牛的技术博客，别人也会因此而对你的印象加分。

StackOverflow 和 StackExchange，开发人员都知道，它们的创始人，Jeff Atwood，有一个非常著名的博客 Coding Horror，国内的技术大牛、现在爱奇艺的技术总监陆其明还翻译了 Jeff Atwood 在 Coding Horror 上的一些文章，集结成书，其中一本是《高效能程序员的修炼》。

虽然不是每一个人都能像 Jeff Atwood 那样，但再小的个体也有品牌——笔者也有一个技术博客（http://blog.csdn.net/foruok），学习 Qt 的人可能会知道，关注程序员职场提升的人也可能会知道。

维护技术博客是相当有挑战的事情，你需要锤炼你的技术和写作能力，还需要有很好的自控力。不过，一旦你坚持下来，就会收获多多。

出版技术书籍

像我这种半路出家的野和尚，也出了《Qt on Android 核心编程》和《Qt Quick 核心编

程》两本书。所以，我觉得，有相当一部分人都可以出书。以大多数人的努力程度之低，实在还轮不到比拼天赋。所以，出或不出，关键在于你是否努力和坚持。

写一本技术书是非常大的挑战，除了坚持和努力，还要求你对所涉技术有全面、系统的了解，要求你有比较好的书面写作能力，因此，一本认真的技术书籍是一个人能力的一种证明，能够成为你个人品牌的一部分，对你以后的工作和生活都会带来一些便利。

现在很多出版社都通过技术博客寻找技术图书作者，你看看 CSDN 上有多少博主出版了图书就会明白这一点。所以，维护一个高质量的技术博客，能增大你出版图书的机会。

持有技术专利

即便是在小公司，开发人员持有技术专利也是可能的。我之前所在的公司，就有两个人拥有专利，还是技术方面的，让我有高山仰止之感。

其实，专利也不是那么高不可攀的事情。Google 一下图像处理、视频处理，专利多如牛毛，有的专利看起来似乎也不是那么难……

还是那句话，你可以试试的。

证书

我把学校、学位、软考、ACM 程序设计大赛获奖、PMP 等都归类到这里。

知名大学计算机、数学、电子等相关专业的学位，对刚走出学校的程序员是很好的个人品牌，这和有经验的程序员的工作背景类似。但是随着工作经历的增加，学校和学位的影响会逐年降低。

很多人在学校时就参加软考（计算机技术与软件专业技术资格水平考试），拿到了初级、中级或高级证书。这些证书有一定含金量，有一部分公司会看重它，比如有些传统的软件公司或国企。

还有的人参加过 ACM 程序设计大赛或其他的比赛，拿到过奖牌，这当然是极好的。即便有人觉得现在的 ACM 已经注水，可前几名还是有吸引力的。

参加 PMP（项目管理专业人士资格认证）考试要花 3000 多元，熟读 PMP Book 就可以通关。有些企业招聘项目经理会加上一条：有 PMP 证书者优先。

还有很多其他的资格或证书，比如微软的 MVP……

总之，有各种各样的证书可以丰富你的履历，给你的个人品牌锦上添花。有时甚至能当作敲门砖用。

口碑

很可能我们作为一个非著名程序员，前面的哪一条都沾不着边……

那也不必沮丧，这最后一点，真的是一视同仁的，无论你在大公司或小公司、无论你学历高低、无论你颜值高低，都可以通过实践来获取。

大多数程序员都工作在一个团队中，和各种各样的人打交道。你在别人的眼中是什么形象，别人怎么评价你，这很重要。非常重要。

现在的软件公司，人员流动性很高，有些单位的程序员，经常是过两年置换百分之八九十。在这样高流动性的背景下，你能流向哪里，在很大程度上取决于在既往的工作组织里别人对你存有什么印象。也就是说，你的口碑如何。

这个口碑，其实是你自己经营、塑造出来的。

假如你留给别人的印象是认真、能干、学习能力强、各种技能强悍、能独当一面，别人提起来你就会点头，“嗯，这人靠谱”或者“嗯，这人 iOS 开发门儿清，没什么问题他搞不定”，那么，你那些曾经的同事、朋友，在软件江湖里泛舟月下之时，很可能就会想到你，“这事让那谁谁来做肯定没问题”。于是你的各种机会和邀约自然就会多了，即便我们一直在小公司奋斗，即便还没有知名产品开发出来，也一样能够自由自在地飞翔。

而假如别人提到你就摇头或皱眉，心里犯嘀咕，“这家伙太不靠谱了”或“什么都搞不定还狂得很”，那么估计就算他们身边有工作机会，也断然不会邀请你的。

春种一粒粟，秋收万颗子。你今天的负责、努力、上进、能钻研、与人为善、乐于助人、有担当，一定会为你树立良好的个人口碑和形象，一定能在将来帮得到你。即便前面 7 条于我们都是浮云，做好口碑这一点也可以“莫愁前路无知己，天下谁人不识君”。

那些你不愿说给领导的话

来聊聊那些你不愿说给领导的话，讲讲不说的原因，再看看为什么要说，说了又能怎么样。

哪些话你不愿说给领导

尽管 IT 行业的公司文化相比国企、政府已足够简单，然而还是有一些话我们只愿在同级别的“自己人”之间当作牢骚来发一发。这些话虽然经常关乎公司的利益甚至将来的生存，可没人愿意把它们说给领导听。

比如领导决定了一个新项目的交付日期，不容置疑地说一定要在 4 周内完成。而作为开发人员我们觉得这根本不可能，起码需要 10 周时间。可我们不说。作为项目经理，往往也清楚自己的团队不可能在 4 周内交付，可他往往也会答应下来。

比如领导拍板了一个技术方案，程序员和项目经理都觉得不合适，既不是技术上最可行的，也不是最经济的，可没人列个一二三出来跟领导理论。

比如程序员明显能看到产品划分上具有上下游关系的应用产品部和系统部配合不畅，应用产品部总是埋怨系统部提供的中间件或底层服务错漏百出以致不愿升级到新的版本，系统部总是埋怨应用产品部抱着老版本不放、死也不愿意采用解决了众多 Bug 应该更好用的新版中间件。

比如团队里的程序员都觉得王二麻子不靠谱应该踢出去，可尽管王二麻子总是拉低整体的生产效率（1+5 都等于 3 了）让人忍无可忍，也没人和领导说。

比如张三丰明明对领导分派给自己的任务不满意，对所用技术也不赞成，可还是以“哦”“好的”“行”接受下来。

比如明明版本还未经过充分测试，甚至一轮都没测完，领导为了赶上截止日期，愣是要发布，开发人员和测试人员尽管心中充满担忧，可左思右想后还是算了。

比如我们明明觉得发布的产品架构不合理、技术落后、代码脏乱差、隐藏的 Bug 一大把、雷区早晚被引爆，总之欠了很多技术债，可还是没能告诉领导说“咱们先把技术债务清偿一下再往前‘滚’”。

比如程序员明明觉得某个需求不合理，既不符合用户的逻辑，也很难用现有技术实现，简直就是个奇葩，可还是默默地开始做了。

比如张三明明觉得今年的绩效自己应该得 A 而不是 C，可面对落在自己头上的 C，尽管心中不满丛生如蛛网怨气纵横似霜剑，最后还是不了了之了。

不说的千般考虑

王小波有一篇杂文，就叫“沉默的大多数”，还有一本同名的集子，可能很多人听说过。

前面我提到的那些场景和与之相关的话中，大多数具有不那么招人待见、容易让人尴尬、很可能让人不痛快等特点。我们欲说还休终究算了，部分考量如下：

- 多一事不如少一事，少一事则相安无事……
- 领导眼睛不瞎，总会看见我的努力，主动去提反倒让领导面子上挂不住转过来对我有看法。
- 这么明显的问题，领导会看不见吗？肯定是他没办法解决，或者还没想到好的解决办法。或者他觉得没什么大不了的，或者有办法但解决起来复杂系数太高，只好装没看见。
- 说了也没用，领导不会听的。
- 说这些领导会不高兴，觉得自己是对他不满，以后自己升迁就没戏了。
- “努力”而无结果比一开始就下断语说这事肯定不成在政治上更正确。

为什么要说，说了又怎样

你是公司的一部分，也是你的工作环境的一部分，环境和文化是大家创造的，没理由别人整个好环境你坐享其成，你自己不为自己负责，自己不堵破窗户，破窗户就一直在那里，慢慢变大，直到那幢建筑轰然倒塌。

你说了别人才会知道你的想法，你不说，保持沉默，别人就当你没事，就当你同意、被代表、被数字化了。而你越被代表，越没有说话的意愿，就越有可能成为沉默的大多数，即便面对与自身利益相关的事，也没说话的意愿了，你好像一个局外人，看着自己被别人支配，被行货化，麻木，无动于衷，爱谁谁，不行就换个地方……换个地方会好吗？重新开始一轮新的循环……

这就是不说的坏处，人人都不说，就等着《皇帝的新装》里那个天真的小孩出现，等着别人推动事情往好的方向发展然后自己跑过去分一杯汤羹……这样的结果就是死鱼把大家都臭死了，谁也没得玩。

所以，我们要说，说出来领导才会知道还有这么一个人、一群人是在意这些事的，领导才会正视原本想糊弄过去的事，重新开始考虑正经的方案，这样事情才有转机，才可能走上正确的道路，整个环境也才会慢慢变好，大环境好了，处在其中的每个人才能受惠。

当然，说真话有风险，可能会不受领导待见，可能会因此丧失升迁的几率，甚至可能

因此被“冷落”，然而想想不说也是个河臭鱼死的结果，还是说说看更有价值——兴许多你一个人的一句真话，美好的一切就开始了呢。

而即使出现最坏的结果，你因为说真话而发展受阻，也真没什么好沮丧、失望的，如果一个组织不能容纳你这样真正为组织好的声音，不能从内部焕发生机，那就走吧，在一个根本不在乎你的地方浪费生命没有任何意义。

要不要使用新技术

那时，天下人的口音、言语，都是一样的。他们往东边迁移的时候，在示拿地遇见一片平原，就住在那里。他们彼此商量说：“来吧！我们要做砖，把砖烧透了。”他们就拿砖当石头，又拿石漆当灰泥。他们说：“来吧！我们要建造一座城和一座塔，塔顶通天，为要传扬我们的名，免得我们分散在全地上。”耶和华降临，要看看世人所建造的城和塔。

耶和华说：“看哪！他们成为一样的人民，都是一样的言语，如今既作起这事来，以后他们所要作的事，就没有不成就的了。我们下去，在那里变乱他们的口音，使他们的言语彼此不通。”于是，耶和华使他们从那里分散在全地上；他们就停工不造那城了。因为耶和华在那里变乱天下人的言语，使众人分散在全地上，所以那城名叫巴别（就是“变乱”的意思）。

这是《圣经·旧约·创世记》第十一章的故事——巴别塔和变乱口音。我引用这个故事是想说：“言语和沟通是成就事情的一大障碍”。非但建造通天塔如是，对于软件开发，也是一样的道理。

我拿巴别塔比喻软件开发，看起来有点风马牛不相及，实则其致一也，古今同也。

看看当你所用技术有了新发展后可能会发生的事吧。

C++ 11 是一门全新的语言吗

看我博客的朋友们都知道，我最熟悉的语言是 C++，这一点我在“程序员这样优化简历，一投制胜”一节中也曾提及。然而，十来年了，我一直都在用 C++ 98……想起来真是汗颜，C++ 03 我都不了解……更别说 C++ 11 了……那天我女儿听说了这件事，就用冰棍

里面的棍棍编了把扇子送给我遮脸，这事就不再提了。

我决定学习 C++ 11 还是因为我的第一本书《Qt on Android 核心编程》的一个非常认真、非常严谨的读者（致谢），他给我来信，说我这本书有一大半的篇幅在讲 Qt 基础，讲得非常好，比市面上的 Qt 书都好，但有一个缺点：就是没有讲 C++ 11 在 Qt 里的应用。

我看了信很是汗颜，那是 2014 年了吧，我居然只是听说而从未去了解过 C++ 11 到底是几个意思。当时我也不好意思说我不懂，只说改版时会加进去。然而到现在也没改版……我很少用 Qt 了，也没改版的动力了……

不过，我还是下决心了解了一下 C++ 11，决心引入到在做的产品里。然后还准备录一个《C++ 11 in Qt》的视频课程，要给自己个理由去学习新技术，还要把新技术用起来。

C++ 11 里引入了初始化列表、统一的初始化模式、Range-based for loop、线程、自动类型推断（auto 和 decltype）、lambda 表达式等新特性，组合使用它们，可以让你的代码神鬼难读，你可能觉得这是 C++标准委员会和 Bjarne Stroustrup 故意搞的巴别，特意来变乱程序员的语言的。终于和文章开始的“巴别塔和变乱口音”联系上了吧？现在看一段代码：

```
auto it = m_msgHandlers.find(strCommand);
if (it != m msgHandlers.end())
{
    for_each(it->second->begin(), it->second->end(),
            [&message](MessageHandler*&handler){
                    handler->handleMessage(message);
            });
}
```

你觉得上面这段代码好读吗？我觉得不太好读，我的几个同事看了这段代码第一反应也是不理解——这都是什么啊！不过如果你用过 Scala 的函数式编程，就会感到很亲切。

还有一段代码：

```
class XXX
{
public:
    struct UIViewZCompare
    {
        bool operator()(UIView* a, UIView* b) const
        {
            return a->zOrder() < b->zOrder();
        }
```

```
    };
    ...

protected:
    std::multiset<UIView *, UIViewZCompare> m_views;
};
...

void XXX::doPaint()
{
    ...
    for (auto v : m views){ v->paint(m canvas); }
    ...
}
```

也挺难读的吧？

这就是我今天要说的问题：当你想要使用某项技术的新特性时，将会面临“to be or not to be”的问题。

用还是不用

到这里才算进入正题。

我的建议是：让我们排除万难，使用同一种语言。

作为程序员，技术演进必然要跟进，跟进得太慢了你都不好意思和别人说话，人家都用 auto 代替诸如“std::map<std::string, std::list>::iterator”这种又长又难记又难敲的类型了，你还不知道 auto 是干嘛的……

有人说我用了别人看不懂怎么办？别担心，他们看不懂会因为不好意思而去学的。这样也算是帮助别人进步，助人为乐啊。你只要管理好自己，改变自己，就可能影响别人。

甘地曾说：“在这个世界上，你必须成为你想看到的改变。”如果我们不在日常之中实践我们想要的改变，到头来就是被组织和环境改变、异化而无法改变组织和环境。

除此之外，你还可以更进一步，通过在团队里布道，主动带别人玩，比如吃饭时聊、组织分享会、说服你的上级使用新技术……只要你决定去做，总是有各种方法来实现你的目的。想做事的人找方法，不想做事的人找借口，就是这个道理。

程序员为什么热衷于造轮子

搜索一下“造轮子”或者“程序员为什么喜欢造轮子”，会看到很多相关的讨论，这是个老生常谈的话题，很多人谈过了，谈了很多年。不过还是有再谈的必要。

“造轮子”的含义：

明知道你不可能比前辈做得更好，却仍然坚持要做。

就软件开发而言，“造轮子”是指“业界已经有公认的软件或者库了，却还坚持要自己做”。

在软件开发过程中，有时你想造轮子老板却极力反对；有时你不想造轮子老板却坚持要造一个出来。为什么会有这种两极状况？

这篇文章就来讨论“造轮子”这件事，包括下列主题：

- 程序员为什么会重复造轮子？
- 为什么有人不让“造轮子”？
- 什么时候可以造轮子？

为什么会重复造轮子

每个造轮子的程序员都有自己“不得不造”的理由。比如：

- 以为自己的需求独一无二，现有的库在某个点上就是满足不了。
- 老轮子没有规格说明书，或者接口太复杂，不知道怎么用，搞明白太难。
- 需要在老轮子上添加新功能，然而老轮子代码难读又无人可问，不知道何时能弄明白，看不到结果，容易放弃。
- 眼界有限，不知道已有这样的轮子。
- 版权原因无法使用第三方库，比如 Google Android 实现 JVM（Google 曾因为一行代码而和 Oracle 打官司），比如阿里 YunOS 自己实现 JVM。
- 就想锻炼自己，因为造轮子对自己的设计、编码能力有很大好处，对理解业务也有很大好处。
- 自己造轮子，有“控制感”，看得见摸得着，可以一步一步来，通过一个一个小目标迭代出大目标，不断成功的小激励，会带给自己前行的动力。

- 创新成分多（对自己而言），有成就感。
- 不相信老轮子，譬如老轮子可能有后门、漏洞（想想 OpenSSL 的心脏出血漏洞）、后期万一要修改没把握等，反正觉得自己造轮子心里更踏实。
- 不想让自己产品的关键技术掌握在别人手里，也不想让自己的核心用户数据流经别人的系统。
- 别人的轮子不开放，自己就是要赶紧造（山寨）一个出来以便获得话语权或商业利益。

为什么有人不让“造轮子”

有坚持要造轮子的，也有高呼“不要重复造轮子”的。那么为什么有人不让造轮子呢？

- 项目（产品）时间紧张，用第三方库搭积木快，能节约时间。
- 领导（或队友）认为想造轮子的程序员水平就那样，不可能造出比现有库（软件）更好的轮子，显然会漏洞百出、推高维护成本。
- 造轮子是个看上去很美、做起来很复杂的事。复杂一点的轮子，造出来很费劲，道阻且长，很可能骑虎难下或半途而废，导致精力和时间的浪费。
- 待造的轮子不是产品的关键（比如一个字符串类、一个 XML 解析类），不属于核心竞争力，不值得花费人力，要把精力放在最重要的事情上。

结合为什么要造轮子以及为什么不让造轮子，就可以理解本节一开始提到的那种反差极大的状况。

什么样的轮子可以重新造

看现在的软件发展趋势，越来越多的基础服务能够“开箱即用”“拿来用就好”，越来越多的新软件可以通过组合已有类库、服务以搭积木的方式完成。这是趋势，将来不懂开发语言的人都可以通过利用现有软件组件快速构建出能解决实际问题的软件产品。

在这种趋势下，软件（服务）就慢慢演化为两极：

- 满足终端用户的应用类产品。
- 解决软件产品通用问题的基础服务（组件）。

比如你在自己的 APP 中需要即时通信功能，完全可以使用融云、环信、网易云信等服

务快速集成。

比如你想在自己的 APP 中添加支付功能，完全可以使用 Ping++或 Pay++来解决诸多支付渠道的集成问题。

比如你想添加分享功能，ShareSDK、友盟 SDK 可以节省你很多时间。

比如你想做跨平台的游戏，使用 Cocos 2d-x 远比自己在 Android、iOS 上从底层从 OpenGL ES 干起要高效得多。

比如你想让你的网站支持更多用户、更多并发，能够快速部署、迁移、规模复制，那么完全可以借助阿里云、AWS、Azure 等而没必要自己搞。

比如你想推送消息给用户，就可以用腾讯信鸽、极光、个推、百度云推送、友盟等。

……

类似的场景有很多。这种趋势使得一部分厂商集中精力开发基础服务（组件），一部分企业集中精力解决用户需求。对基础服务（组件）厂商来讲，它通过解决更复杂的基础问题为其他厂商带来便利而盈利。对终端软件产品企业来讲，它通过解决用户问题给用户创造价值而盈利，从理论上讲，只要其产品从用户端或第三方获取的价值大于支出给基础服务厂商的价值，生意就可以做下去。

有了这样的认识，什么时候可以造轮子、什么时候最好不重复造轮子就不再是问题了。

对于提供基础服务的软件厂商，很多轮子必须造。因为它要提供服务给其他软件厂商，拿友商的组件换个包装提供给其他软件厂商，没有竞争力。所以你看到在某个软件服务市场上，会有多家企业各自在造轮子，为的就是自己掌握核心科技，有自己的竞争力。比如提供云服务的，有阿里，七牛，百度……提供即时通信服务的，有融云、环信、阿里云信……提供语音服务的，有科大讯飞、百度、OKVoice、Google、微软……

对于开发满足终端用户的应用类产品的公司，很多轮子就没必要造。比如你提供一个健身类的 APP，可能需要引入即时通信功能，用第三方就好。

从公司的角度讲是这样，那么对程序员来讲呢？

对程序员来讲，在一开始的学习成长阶段，造轮子则具有特殊的学习意义，学习别人怎么造，了解内部机理，自己造造看，这是非常好的锻炼。每次学习新技术都可以用这种方式来练习。

当我们掌握了一门技术，可以用于实际产品开发中时，关于造轮子就有了另外的划分：

一些基础的工具类库，比如 String，Xml，Json，HTTP，推送，流媒体协议，重新造的必要性不大。而与业务相关的，可以尝试重构、再造，对理解业务有好处，也能更好地适应新需求。

这样读源码，想不卓越都难

程序员在工作过程中，会遇到很多需要阅读源码的场景，比如技术预研、选择技术框架、接手以前的项目、review 他人的代码、维护老产品等。可以说，阅读源代码是程序员的基本功，这项基本功是否扎实，会在很大程度上影响一个程序员在技术上的成长速度。

2014 年写《Qt on Android 核心编程》和《Qt Quick 核心编程》时，很多内容都是通过分析 Qt 源码搞明白的。这阵子研究 CEF 和 PPAPI，也主要靠研究源代码来搞明白用法。最近工作中要修改已有项目的一个子系统，也是得硬着头皮先读懂代码。

总之在从事开发工作这十多年中，读过太多源码，从源码中学到太多东西，如果不阅读源码，真不知道自己能否成长起来。

写代码是从模仿开始的，提高也是从观摩别人的优秀设计和代码开始的。所以阅读源码至关重要，接下来从下列方面聊聊阅读源码的事。

- 目的。
- 工具。
- 知识准备。
- 运行与开发环境。
- 笔记。
- 实用技巧。
- 心理调试（散步在各个环节）。

目的

当我们阅读面前的源码时，无非有以下几种目的：

- 纯粹学习。
- 添加新功能。
- 重构旧代码。
- 修复他人的 Bug。

目的不同心情会有所不同，会影响到工作的进展，像修复他人的 Bug 这种事情，是很让人反感的，很容易让人拒绝。所以因这种目标而阅读源码，往往是欲拒还迎、欲说还休，

效率较低。然而实际工作中帮别人修复 Bug 这种情形，十有八九会遇到，无可逃避。所以，心理调试很重要。

为了学习去读源码，是最愉快的、最放松的。不过提醒一点，设定可检验的目标才会有收获，否则就会像走到大街上看见美女擦肩而过那样，惊艳一下，过后什么收获也没有。

其他的目的，重构旧代码、添加新功能，因为带有创造性，创造性的活动能给人带来强烈的愉悦感，所以虽然这两种目的也有很多让人不爽的部分，不过想到自己可以让一棵老树焕发青春，不爽也就慢慢弱下去了。

工具

“工欲善其事必先利其器”，这是亘古不变的道理。要很好地完成阅读源码的任务，我们大概需要下列这些工具：

- SourceInsight，最好的源码浏览工具，它能维护符号库，动态显示上下文，还能绘制调用关系图，是最好的，没有之一。
- 纸质笔记本，随时记录心得和疑惑，随时绘制各种图（类图、时序图、框图），比 UML 工具快，也比 Visio 快。
- 中性笔。
- 记事本、Notepad++、有道云笔记、为知笔记等，记录阅读源码过程中的关键点、心得体会、分析过程。
- Visio，用于绘制简单的框图，表述源码的模块划分、层次结构等。
- StartUML，用于最后绘制类图、时序图等，方便交流。
- 扫描全能王（CamScanner），一款可以通过拍照达到扫描效果的 APP，可以用它扫描你在纸质笔记本上写下的文字，绘制的框图，分享给其他人，如果你懒得用软件绘制图标，那么手绘之后扫描成电子档就最适合你了。

知识准备

- 业务基础，每一份有实际意义的源码都离不开业务，必须先对业务有概念。
- 技术基础，这个源码用什么语言，什么框架，什么第三方模块，都需要先有所了解。
- 文档，尽量找到业务、需求、概要、详细等文档，帮助会很大，然而，我们经常面

临的情况是，只有源码，片言只字的文档也无，所以只好坚信——源码是最好的文档。这个心理门槛其实也容易过。

- 人，搞明白哪个程序员维护过这份代码，方便后面不懂时请教，有时人家点拔一下顶你自己瞎琢磨一天。

运行与开发环境

- 配置好开发环境，目的是为了调试，对有些程序员来讲，调试是弄明白软件内部机理的最好方法，按着按 F5、F10、F11、F9 键，一切都搞定了。
- 配置好运行环境，为使用软件、体验软件做准备，从用户角度，从外面看看软件到底是怎么回事，便于揣摩内部逻辑。

笔记

在阅读源码的过程中，做笔记是必需的。我有这样的体会，因为代码不是自己写的，很难很快在脑子里刻下印记，经常是看着这里忘了那里，早上觉得弄懂了数据流向，中午吃个饭就忘了。所以，笔记显得尤为重要。

找到适合你的记录方式，小本本、软件皆可。用软件（Notepad++、有道笔记、为知笔记等）来记录有个坏处——必须切换屏幕，会在形式上中断代码阅读过程。所以我经常在紧张得不能中断时随手用笔在本子上写些断句残章，告一段落时梳理一下，用软件再记录。

- 尽可能详细地做记录，但不必看到什么记录什么，要间隔性地做记录，比如弄明白某个子模块的逻辑、某个类的作用、某些函数的调用关系时再记录，否则记录这个动作本身会打断思考。
- 每天工作结束，记录进度（弄明白的部分），记录疑问，记录第二天要弄明白什么东西，这样你的工作状态就入栈了，第二天来了很容易出栈，快速进入工作状态。
- 记录看到的优秀设计，提高审美，见贤思齐，自我成长。

沧海遗珠

我在漫长的读码生涯里积攒了一些经验供参考：

- 理清某一业务是如何映射在代码执行流程上的，这一点很关键。
- 理清不同模块间的业务关系，代码调用关系，很关键。
- 调试是弄明白代码调用流程的最快方式之一。
- 找出关键代码（代表实际对象的类、衔接不同模块的类、代表业务关键节点的类）。
- 分析日志可以帮助分析代码执行流程和业务流程。
- 先用已有的可运行软件，体验业务，琢磨你点这里一下点那里一下代码可能是怎么做出反应的。
- 阅读应该围绕目的，把实现目标放在第一位，比如修改 Bug，如果有期限，在最后日期前搞定是第一要务，然后有时间就继续读源码或改进 Bug 修复方案，力求没有副作用和后遗症，再有时间就修修别人留下的破窗户（你也可以顺带鄙视一下前任维护者）。
- 千万次地问，还记得前面说要弄明白谁维护过你要读的代码吧？别不好意思，问吧。
- 对着设计文档、接口文档或测试用例看代码。
- 心理调试，勿畏难，别放弃。我有时看代码，看两天也不知道看了什么，一头雾水两眼发花是常有的事，有时真是觉得搞不定了，然而，这要么是你的基础知识没准备好，要么是你找错了入口，要知道，任何一份代码，都有一条隐形的线串着，耐心点，总会找到。这样不行就那样，多换换角度，多换换方法，读不行，就调试，调试不行，就运行，运行不行，就研究日志……总之，你不放弃自己，就没人能放弃你！
- 给自己设置小奖励，弄明白某个逻辑或某个模块的代码后奖励自己休息一下，5~10 分钟，走出办公室转转，或者干脆在网上瞎逛一下，浏览自己喜欢的网站。

读不懂才要读，想不明白才要想，这是进步和成长的开始。那些阻挡你的蹂躏你的而又杀不死你的，终将帮助你成长，让你变得更强大。

十年的老代码，你敢动吗

假定你入职一家新单位，被告知需要维护一个老产品，经理找质管人员给你开通了 SVN 权限，告诉你迁出哪个分支——就是那个十年前已经定型的分支，也就是那个超过 6 代程序员维护过的分支——然后告诉你，就在这个分支上改，添加一个新接口，以便支持 H5

Video。

于是你开始看代码，云山雾罩，各种痛苦，完全搞不懂业务逻辑和代码的关系，也弄不明白这块代码为什么这么写、那块代码是几个意思。你战战兢兢如履薄冰、思前想后寸步难行。

你去问进来 5 个多月还没转正的老同事，他告诉你他也不懂，让你凑合着加个新接口实现了功能就行。

你去问干了快一年的资格更老的同事，他叮嘱你千万别动里面的代码，千万别管里面什么样，就在外面包一层，先交付新功能，其他的有时间再说，里面的逻辑十年没人动过了，没有一个人能说清楚怎么回事，你要是改，一不留神就遍地狼烟。

你怎么办？

关于老代码的禁忌

对程序员来讲，维护老代码连我这种自认为随意、灵活、代码适应性强、没有原则的老程序员也觉得是一种罪。如果你恨一个程序员，就让他去维护年久失修、摇摇欲坠的老代码吧。

然而，老代码啊，我们恨之厌之烦之远之却不可弃之。对一家软件企业来讲，老代码就是资产，是多年积累下来的核心资产和重要竞争力。尤其是软件产品，一份代码被先后几波人维护过是常有的事。

这种时候，老代码就老而成精有了生命，每当有新人进来，它都会用特有的超出我们耳力边界的高频发出声音：警告，警告，一波新程序员正在赶来，快给它们点厉害杀杀它们的士气。

而且，部分熟悉老代码的老程序员也会谆谆告诫我们，这几个文件不要动，这几个类不要动，这里一改就奔溃，那里一改就连不上服务器，雷区标识很多。还有那一进来就因为被告诫而根本就没看过老代码的程序员也会告诫我们，那代码谁也不懂，多少年没人动过了，不动为妙，免惹麻烦……

故事就这样发生了，禁忌就如此这般传承下来。我们知道那里可能有问题，可是没人敢去动它，等最后一个熟悉一部分老代码的程序员悠然远去，此地空余叹息，从此以后，新来的程序员就成了想吃香蕉的猴子。

动，还是不动

老代码成了禁忌，我们往往被迫（没时间或不愿意或害怕或不屑）在漂浮于海面的冰山的尖尖上修修补补，深入了解深层代码成了谁也不愿言说的痛。

然而都不懂看代码都不动老代码，老代码只能越来越老越来越陈腐，越来越没人敢动。越往后动的代价越高越没人敢动。这对公司和团队都不好。裹一层又一层，终将积重难返成为裹脚布，无人问津。

而一旦老代码没人能够把握，这些作为资产的代码实际上已经丢了，不再有价值增长了，原本领先的优势随着同行们百舸争流的追赶渐渐失去了。

这对企业是一种损失。对程序员其实也是一种损失。

为什么这样讲？

情人还是老的好

程序员有个毛病：**自己不写文档却老抱怨别人的代码没文档，而碰见了有文档的代码却又往往弃文档如敝履**。所以，业界流传一句话：代码即文档。所以，业界的代码和文档，少见有匹配的。

所以，我们只能接受这个现实：代码即文档。

所以，对维护老产品的程序员来讲，要想弄明白老产品的逻辑，就只有如下两个办法：

- 找到熟悉产品的前辈，让他给你讲讲。如果你找了产品经理，他只能告诉你产品设计上如何如何。如果你找了程序员，他通常会说就是这样那样，然后说一句高深莫测又拉仇恨的话——看看代码就明白了。
- 自己啃代码。

维护旧产品，弄明白产品设计逻辑和代码实现逻辑是非常重要的。

对于本节一开始提到的问题，其实也可能有在外围包装的办法，比如你封装一个本地的 HTTP Server，用旧 API 拿到数据作为 HTTP 流转发一下就能支持 H5 Video 标签了。也可能很多情况下都存在折中的替代办法。

然而，能弄懂代码是如何实现产品和业务的，还是有非常重要的好处的——对程序员来讲很重要的好处：

文档会过时，代码是不会说谎的，读懂代码，你就真正明白了业务是如何实现的。对

于没人敢动而又核心的老代码，你搞明白就占领了战略要地。

这里我要引用格力空调的一句广告词：掌握核心科技。掌握核心科技才有竞争力，对程序员来讲也是一样，唯有掌握核心业务和代码，才能彰显自己的价值。

除此之外，读代码也是非常重要的学习途径，尤其是经历过线上考验的代码，必然有其过人之处。在阅读的过程中，我们可以学到很多东西，既可以学到业务，也可以学到设计。退一万步讲，即便你认为自己的水平远超一般人，属于一针顶破天的那位，也还是可以从当时、当地的选择中学到东西：见贤可以思齐，见过可以自省。

所以，我的主张是：老代码，动啊，为什么不动！

不能拒绝时就接纳，无须排斥，何时何地都可以修行，只要有心，处处都是成长的机会。最不济，也锻炼了阅读代码的能力，庖丁解牛之技成了，也可以在将来以无刃入有间，发挥用武之地。

至于怎么读代码改、代码，一句话：多读读就好了。当然展开来讲也可以有很多技巧，后面会专门谈。

技术债务可能是这样来的

看我的技术博客的朋友可能会注意到，我更新了一系列与 CEF、PPAPI、Skia 相关的文章。在研究它们的过程中，有一些有意思的经历，非常典型，可以从一个方面解释“技术债务”的由来。

接下来我会讲讲这次经历，并从此展开，看看形成技术债务的原因及应对策略。

选择容易的替代策略

因为业务需要，我得在 PPAPI 插件中显示另一个模块（已有模块，基于 C++代码完成）传递过来的图像数据。那个模块提供的数据，图像格式是 RGBA32，我在 Intel Pentium 主机上编译出的 Skia 库，默认的颜色类型是 kBGRA_8888_SkColorType（kN32_SkColorType），这导致在使用 Skia 绘制之前，必须将收到的 RGBA32 格式的数据转换为 BGRA，否则就什么也画不出来。转换确实可行，我试了试，不过在图像分辨率较高时，耗时会明显增加。

因为需要逐像素转换，一个像素转换至少需要三次赋值和两次索引操作，百万像素的图片就需要 500 万次操作。

当我完成像素格式转换，看到 CEF 中显示出颜色正常的数据后，大大松了口气。此时我有两个想法：

- 就这样吧，功能实现了，PPAPI 插件进程就显示图像也不干别的，慢点没太大影响。
- 研究 Skia，看为什么 SkCanvas 以 RGBA 格式的 SkBitmap 为后端绘图时失败。

那天腊月二十八，马上要过年了，第一个选择所需要的工作仅仅是找理由说服自己，说服领导。这很容易，我们程序员老这么干啊，不是吗？

第二个选择就要困难一些，得做实验，得硬着头皮看 Skia 源码……这且不说，做了努力也不见得就能解决，说不定耽搁了时间和进度，最后还得回到第一个选择上去。总之我发现自己内心有点小拒绝，也能找到各种理由让自己放弃这个相对较难的选择。你遇到过这种情况吗？

我该怎么做呢？

后来过完年，2 月 14 日情人节那天我就去上班了。大部分人还没到岗，我觉得研究一下第二个选择里那个问题很有必要——有时慢就是最大的缺陷啊。于是就做了下面这些事情：

- 组合变换 SkBitmap 的 ColorType、AlphaType。
- 变换 PPAPI 中 Graphics 2D 使用的 PPB_ImageData 的颜色格式。
- 变换 SkCanvas 的创建方式。
- 改变 SkCanvas 使用 PPAPI 创建的 ImageData 的方式。
- 采用 Skia Images 模块的方法完成 RGBA 到 BGRA 的转换以提高性能。

总之折腾了一天，没找到原因！此时又想就这么算了……

选择容易的路，这种想法貌似难以避免，往往是不经意间就冒出来了。而真正去解决问题，则需要鼓足勇气努力说服自己。

第二天我决定研究 SkCanvas 到底是怎样使用 SkBitmap 来绘图的，花了大半天时间做实验、阅读 Skia 的源码，终于发现 SkCanvas 基于 SkBitmap 绘图时会创建一个 SkBitmapDevice，而 SkBitmapDevice 会根据 kN32_SkColorType 来进行分支操作，而 Intel Pentium 主机是小端字节序，Skia 的默认编译选项编译出来的库 kN32_SkColorType 就是 BGRA，所以当我把 SkBitmap 的 ColorType 设置为 kRGBA_8888_SkColorType，再将这个 SkBitmap 传递给 SkCanvas 时，SkCanvas 构建 SkBitmapDevice 来作为绘图的 backend，就注定了失败！

原因明白了，我又有两个选择：

- 认为这是 Skia 的局限，就此打住。
- 研究 Skia 的编译系统，看怎样在小端字节序的主机上编译出默认使用 RGBA 的库。

第一个选择，工作到此就完了，也可以解释给领导和小伙伴了。

第二个选择，得研究 Skia 的构建系统，还得了解控制颜色类型的那些宏定义，还得看 SkBitmapDevice、SkBitmap 到底如何使用颜色类型……就这样还不知道编译出来的库是否有其他副作用（结果证明是有的，后话）……

我又为此纠结了一阵，这天就这么下班了……

第二天上班，我鼓起勇气选择了研究 Skia，嘿，别说，经过对 ninja 脚本、config 头文件以及 SkImageInfo 相关类库的研究，我真了解清楚了怎样才能编译出默认使用 RGBA 的库，并且下班前编译出来了。哇，我做实验时写的小 Demo，使用这个新版本的库，RGBA 的 Bitmap 运作后端正常了！

接下来的一天，我修改了 PPAPI 插件的代码框架，取得了大大的进步：接收到的数据可以不经转换就正常显示了！

心里挺高兴：我战胜了自己从权的想法。

然而问题来了，从本地图片文件加载图片时，Red 和 Blue 通道反了，用到的图标看起来诡异得很，效果全部不对……折腾了三种加载本地图片的方法都不行。可是内存里绘制是正确的……

又来到了十字路口，面临了几个选择：

- 不用 Skia 加载图片，用其他的，解码成 RGBA。
- 用 Skia，加载后获取图像数据，交换 R 和 B（图标都不大，性能损耗可接受）。
- 研究 Skia 从磁盘加载并解码图片的流程，搞明白问题出哪里。

你说我选哪个？

第二个，前面干过了，就是复制点代码，工作量很小，很有诱惑力。

第一个，BITMAP 用 Windows API，PNG、JPEG 也都有解码库，之前也用过其他的图像库，解码出的数据也是 RGBA 的，看起来也没什么难的，就是工作量多一些。

第三个，充满未知和阻力。要知道看别人的代码总是很头疼的，尤其像 Skia 这种优秀的开源库，代码很棒很巧妙，模块间设计的接口挺简洁的，很多东西都抽象、封装了，要理解需要不少时间……还有，缺少文档（好吧，代码就是最好的文档，永不会遭遇文档和实现脱节的问题）……

找了会儿图像库，什么 FreeImage、CxImage，看了会儿 Skia 源码。

我回到家里就琢磨这个问题：为什么我老想选择容易的路？

是啊，为什么总是拈轻怕重？

此时我脑子里就回想之前的工作历程，哪些能 Run、能出结果的代码被迫重构了，哪些临时策略导致了技术债务造成了恶劣影响……

后来我决定选择第三个！

现在问题已经解决了！

程序员出于（我不代表所有程序员）本能都想省事，选择容易的、确定性强的路，这没什么好非议的——假如容易的路能漂亮地解决问题，它就是最好的选择。

然而为了摆脱压力而采用易行的、凑合的、从权的、临时的技术方案，多数时候会带来技术债务。

技术债务是怎么来的

如你所见，我在使用 Skia 在 PPAPI 插件中绘制另一个模块传递过来的 RGBA 图像数据时遇到了问题，在解决问题的过程中，经历了三次选择，每一次都面临容易的妥协方案和较难的、不确定的但能真正解决问题的方案。我在这三个时刻，都倾向于选择容易的、工作量小的、耗时短的方案。

这不仅仅是我个人的情况，而是大多数程序员从某个问题的多个备选解决方案中选择时的习惯性行为。这种习惯性行为，一方面出自人的一种本能——人总是不假思索地放弃挑战，选择那条容易的路。因为挑战伴随着不确定性和自我控制，不如唾手可得的方法来得快速。以小孩子为例，你在他面前放两颗糖，告诉他如果明天吃掉就可以再得到两颗糖，现在吃掉就再也没有了，他很可能稍稍犹豫后抓起那两颗糖吃掉。

我们倾向于即刻的满足，延迟满足是需要经过练习才能形成的思维和行为习惯。这是程序员选择易行解决方案的一个重要缘由。

避免麻烦和烦恼是本性，所以即便没有外界压力，我们还是倾向于选择看起来更省事的方案。然而这并不是程序员做出这种选择的所有原因。程序员这么做，往往还有其他的原因：

- 交付时间的压力。
- 单位时间做得更多绩效就越好的文化导向。
- 能力不足。
- 出了问题再说的心态。

1．交付时间的压力

这是导致选择简单粗暴解决方案的最常见、最直接的原因。

回到我前面说的问题，其实是个小问题，在这样的小问题上花将近一周的时间，在交付时间紧张的情况下是很难被允许的（我是刚好在过完春节这个缓冲期才有可能从容来解决它），自己不允许、领导也不允许。当你感知到交付时间的强大压力时，结果往往就是先实现再说，以后有时间再来优化。然而这是程序员说过的最大的、最堂而皇之的谎言，那些凑合事的实现，往往就那么着了，直到它出问题被用户和领导逼着限时修改，才开始新一轮的循环。

2．多就是好的绩效导向

在很多公司，如果你能更多地并行工作、更快地交付、完成更多的版本，绩效就会更好。在这种导向（制度）下，我们就会更重视数量和速度，忽略质量和可维护性，眼前不出问题就得过且过。

不仅仅是程序员，程序员的领导的绩效往往也是这个导向，所以一线管理者往往会接受中层、高管的不合理交付要求，转身就会给程序员更紧迫的交付时间把压力传递下去。

3．能力不足

这也是一个重要原因，有时程序员遇到的问题他眼下根本解决不了，只能粉饰过去或绕道而行——要从根本上解决，时间太久，谁都无法接受（参考前面两点）。

4．出了问题再说

这也是很多程序员做事时的常见心态：能 Run 就先 Run，不出问题就先这么着，想那么多干嘛，出了问题再说，反正有干不完的活。

这就是典型的干活心态。

过年时和我原来的老板一起吃饭，他说了一句让我印象深刻的话：如果一个公司不能让员工觉得是给自己工作，那么这个公司迟早会完蛋。这其实说的是事业心态。

你觉得自己是在干活还是干事业，是给自己干还是给公司干，在很大程度上决定了你怎么干。

如何避免技术债务

回头来看我前面的经历，有两点是需要特别注意的：

- 我根本没想到会碰到这个问题（意外总是存在）。
- 时间宽松时我才会有彻底解决问题的可能（时间允许时程序员才会向难题发起挑战）。

这也是软件开发过程中避免堆积技术债务时需要参考的非常重要的两点。

关于这两点，做过两年开发的都有体会。然而深有体会的人往往并不掌握决定权，所以，我们还要让管理层和老板们意识到下面两点：

- 对软件开发来讲，没有一条道路是重复的。人家走过的路我们来走，仍然可能走不好，被别人验证过的方案，我们采用时仍然可能遇到各式各样的问题。即便我们开始前做了自以为最充分的评估，还是有考虑不到的问题跳出来碾压我们。
- 软件开发是手艺活儿，必须留够时间让程序员耐心打磨才能出好东西。

只有意识到了这两点，技术氛围、绩效导向才会改变，才有利于形成向技术债务说“No”的文化。

傻瓜才放弃成为指导者的机会

有个朋友问了这么一个问题：

我们经理安排了一个同事带我，想让我快速熟悉工作。可我每次问这个同事问题，他都好像懒得搭理我，很没耐心，一副不耐烦的样子，总是草草敷衍几句，也没给我什么有用的信息。到公司快两周了，现在连基本的业务流程都还不是很清楚，也不好意思总问他。而且这个同事又比较好面子，我要是当着他的面问其他的同事，他又表现得很不爽。我是不是应该向我们经理反映一下，让其他人带我啊？

我觉得这个问题很有意思，也很普遍，我们聊了一会儿，我也没什么好的办法能让那位担任“导师（mentor）”的程序员从敷衍塞责蓦然转变到积极分享，我给的建议总结起来就这么几点：

- 趁对方心情好时提问。
- 斟酌问题，选择容易引起对方兴趣的点作为开始。

- 想办法在提问时让对方有很强的价值感，有很受尊重的感觉。

这里面其实有一些话术可以学习，比如对方觉得自己哪里牛，你就在提问时不经意地夸哪里，交流结束后来一句碰到点子上的赞赏（没事可以看看《蔡康永的说话之道》这本书）。不过这些都不是我今天要说的，我今天要说的是：

成为指导者对程序员是百利而无一害的事，拒绝这样的机会是愚蠢的。

当你是权威人士时，你会怎么做

有时我们会在某方面比别人懂得多一些，了解得深入一些，会因此而成为开发团队里某方面的“专家”，具有一定的“权威地位”。此时你会怎么做呢？

像前面的问题里那位指导者那样，取笑别人、鄙视别人、质疑别人、挫败别人，以此来显示自己的技术有多牛，除了我谁都不能行？

还是对待向我们求助的兄弟姐妹，像春天般的温暖，竭尽所能共享自己的知识，让每个人都成长起来？

有人选择前者，理由是“台上一分钟，台下十年功”，看着一句话就能解决的问题，其实下面花了很多时间和精力，凭什么就这么轻易地告诉你？再说教会徒弟饿死师傅怎么办？还有我自己忙得一塌糊涂，哪有闲工夫帮你提高？

有人选择后者，理由是把自己的知识和技能传递给别人，自己的那份不但不会变少，还可能会变得更多、更扎实。引用一下《高效程序员的 45 个习惯》里的话：

好的想法不会因为被许多人了解而削弱。当我听到你的主意时，我得到了知识，你的主意也还是很棒。同样的道理，如果你用你的蜡烛点燃了我的，我在得到光明的同时，也没有让你的周围变暗。好主意就像火，可以引领这个世界，同时不削弱自己。

想想你的选择吧。

成为指导者的好处

成为指导者，对程序员本身来讲有很多好处。

1．知识与经验可以相互促进

当我们向别人解释自己知道的知识时，可以让自己更深地理解已知的知识，还能促进

自己更系统地思考所解答问题牵涉到的知识。自己明白，能讲得让别人明白，这中间的差距相当大，这种差距，可以靠指导过程来完成。从这个方面讲，指导对别人是帮助，对自己也是巨大的促进。

知识和技能也具有“横看成岭侧成峰”的特点，别人的问题很可能是从不同的角度出发的，甚至可能会让你因此而发现新的技巧。

2．表达能力可以不断得到锻炼

毋庸置疑，你需要思考知识、经验如何连接、梳理，还需要组织语言来将你连缀成的知识图谱表达出来，这个过程是一种挑战，也是极好的锻炼。

3．捕捉关键问题的能力可以不断提升

有经验的人往往会很快判断别人的提问，甚至有时不等别人讲完就说“好，我知道怎么回事了”，然而这种基于认知和经验落差出发的结论，往往是错的。这都是因为我们没有认真聆听别人的问题，在想当然，所以有时会出现偏差，你以为你理解也给出答案了，实际上别人并不是那个意思。

你以为你以为的就是你以为的吗？有时不是。所以，在和别人交流问题时，需要先清空自我，这样才可以捕捉到他人的“点”，他人才会觉得“你是理解我的”。这是一种非常宝贵的能力。

4．建立好的口碑

还没见过哪个热心助人、乐于帮别人解决问题的程序员口碑不好的……

5．提升团队实力

你帮助团队里的伙伴成长，你所在的团队实力就会增强。因为你播下了乐于助人的种子，受惠的小伙伴会继续传播这种习惯，其他人也可能会效仿，于是整个团队的技术讨论氛围都会好起来。这样，不但每个人的能力会不断提升，团队在某一知识领域的积累也会不断丰富、完整、深入。

6．提高团队凝聚力

团队里的大部分人都愿意积极参与问题讨论，都愿意帮助别人，这样的氛围是程序员最喜欢的。“我知道我不是一个人在战斗”“我知道我遇到问题时会有人帮助我”“我愿意在这样的团队里工作”“我乐于和这样一群人一起工作、生活”。团队的凝聚力就这样提高了。

指导别人的途径

程序员的工作中，能够指导别人的机会很多，形式也多种多种，大概有这么几种：

- 别人主动询问
- 小范围的问题讨论
- 内部主题分享
- 吃饭时的讨论
- 知识库（博客、Wiki、各种格式的文档等）

……

任何机会都能提升你的战斗指数和口碑，点亮别人，照耀自己。

设定目标的 SMART 原则

2015 年刚过完年的时候，我写了一篇文章，题目是“咦，你也在混日子啊”（收录在《你好哇，程序员》一书，我的 CSDN 博客和微信订阅号“程序视界”也有），没想到点击量特别高，光 CSDN 一个站点就超过 16 万。还有很多站点转载了这篇文章。后来我又写了一篇题目为“怎么告别混日子”的文章，想提供一些经验，希望能对那些不想混日子的程序员有所帮助，不过很明显，这篇点击量没有之前那篇高，到现在也才不到 3 万。

“怎么告别混日子”一文的核心观点是：设立目标可以告别混日子。文中也简单举了个例子说明如何设定目标，还大概提了如何寻找目标。

时间过去大半年，我又有了新的想法，于是写了一篇文章，“如何快速定位自己热爱的工作”（本书内有，也可以在“程序视界”订阅号内回复 10078 查看），再一次讨论怎样寻找目标。这篇文章我认为比之前的文章更好一些，因为里面提供了可操作的方法，有很强

的可执行性。不过这篇文章还有很多东西没有说清楚，比如什么是有效的目标。这次我想把有效的目标说清楚。

1954 年，德鲁克在《管理实践》一书中提出了一个具有划时代意义的概念——目标管理（Management By Objectives，MBO），它是德鲁克所发明的最重要、最有影响的概念，并已成为当代管理体系的重要组成部分。

串田武则有一本书叫作《目标管理实务手册》，非常详尽地介绍了如何进行目标管理，值得一看。

假如你想系统了解目标管理，仔细研读前面介绍的两本书就可以了。假如你没时间细看，那么往下看看就能帮助你快速了解什么是有效的目标。

SMART 原则

在目标设定中，SMART 原则被普遍运用。

S（Specific）：目标必须是具体的，要对标特定的工作指标，不能笼统；

M（Measurable）：目标必须是可衡量的，衡量的指标是数量化或者行为化的，验证这些指标的数据或者信息是可以获取的；

A（Attainable）：目标必须是可实现的，在付出努力的情况下可以实现；

R（Relevant）：与其他目标有一定的相关性；

T（Time-bound）：目标必须有明确的截止期限。

下面我们来具体了解一下 SMART 原则。

注：此处参考了百度百科。

1．具体的

用具体的语言清楚地说明要达成的行为标准。明确的目标几乎是所有成功人士和成功团队的一致特点。很多团队（人）不成功的重要原因之一就因为目标定得模棱两可，或没有将目标有效得传达给相关人员。

看两个例子：

- 无效的目标：我要提高代码质量。

具体的表达：我要降低 Bug 率（千行代码缺陷率）

- 无效的目标：我要成为一个程序员具体的表达：我要掌握 C++语言。

2．可衡量性

衡量性就是指目标应该是明确的，而不是模糊的。应该有一组明确的数据，作为衡量是否达成目标的依据。如果制定的目标没有办法衡量，就无法判断这个目标是否实现。

比如前面的例子可以进一步细化，让目标可衡量。

目标 1：我要将 Bug 率控制在千分之 2.39 以内（CMMI3 的标准）。

目标 2：我要掌握 C++基本语法、继承、多态、虚函数、STL 中的常见容器类的用法，并完成一个具有 10000 代码行的项目。

3．可实现性

目标是可以通过努力实现、达到的。应该高于现状，但又是跳一下能够得着的。

假如一个还没有做过软件开发的应届毕业生，定下了“我要在一个月内成为 C++语言的专家”这个目标，这可能就是一种不切实际的表达。更接地气的目标是：“我要在三个月内掌握 C++基本语法、继承、多态、虚函数、STL 常见容器类”。

C++并没有想象中那么好学，能在一个半月内掌握基本语法并能够简单应用就非常不错了，像继承、多态、STL 这些内容，能够真的理解并应用，没有三个月以上的时间是相当困难的。

4．相关性

目标的相关性是指实现此目标与其他目标的关联情况。如果实现了这个目标，但对其他的目标完全不相关，或者相关度很低，那么这个目标即使达到了，意义也不是很大。

比如你的中期目标是“三年内成为一个合格的 C++软件开发工程师，能够独立完成模块的设计与开发”，那么短期目标“掌握 C++语言的基本语法”和中期目标就具有很强的相关性，是实现中期目标的一步。

假如你的目标是沿着中轴线参观中山陵，那么每一个台阶都可以看作一个小目标，爬完 392 级台阶，就可以到达中山陵，开始参观。目标也是一样的，一连串的短期目标指向中期目标，若干个中期目标又指向远期目标，指向你工作或生活的愿景。

只有当一个目标和人生愿景相关联时，才有实际的意义。这种关联性，可以通过短期目标与中期目标关联、中期目标与长期目标关联、长期目标与人生愿景关联这种递进的模式来保障。而实际上还有一种自顶向下设计的方法（从愿景分解出长期目标、从长期目标

分解出中期目标、从中期目标分解出短期目标）来从方法上保障关联性。要做到这么志存高远又步步为营对个人要求实在是太高了……不过，你去看那些很成功的人士，基本上都是有清晰的人生目标的；而混得不错的，也基本上有中期目标或长期目标；对于连短期目标都没有的亲们，多数时候是在随波逐流……

5．时限性

目标的时限性就是指目标是有时间限制的。比如我在运营微信订阅号“程序视界”时给自己定的目标是“在 2016 年 5 月 9 日达到 10000 关注”，这里的 2016 年 5 月 9 日就是确定的时间限制。

目标设定举例

前面概述了如何通过 SMART 原则设定有效目标，这里提供几个简单的目标作为示例。

- 我要在三个月内掌握 C++基本语法。
- 微信订阅号“程序视界”在 2016 年 5 月 9 日前要吸引 1 万个关注。
- 我要在 5 年内成为公司 X 研发部门的经理。
- 我要在 5 年内成为后端架构师。
- 我要在 8 年内成为全职专注于 IT 的自由撰稿人。
- 我要在 60 岁之前赚到 2000 万元。

我要在某个期限内怎么着，类似这种目标，还是比较容易设定得满足 SMART 原则的。

有目标只是第一步，接下来的任务更为艰巨：如何分解你的目标形成计划并执行下去。很多人不是没有梦想，而是缺乏找回梦想的勇气。如果你真的有这种勇气，那么什么都不能阻挡你对未来的向往。

怎样新学一门技术

因为公司缺一个 Web 管理系统的开发，我决定挑一个技术栈来学习一下，然后自己来写。我选择了 Node.js+Express+AngulaJS+MongoDB 这一条技术栈，花了将近两周的时间，

做了很多小 demo，写了一系列博文，终于基本熟悉了 Node.js、Express、AngulaJS、UI BootStrap、CSS、HTML、MongoDB、Mongoose 等内容，觉得可以开始写我的 Web 管理系统了。

之前公司来过一个从没做过开发的同事，在学习与实践的路上遇到一些问题。最近有一个来实习的同事，也在学习中。结合我自己的学习过程，感慨良多，遂成此文：怎样新学一门技术。

注意我的用词，是“新学一门技术”，而不是“学一门新技术”。我想强调的是再一次学习这件事，而不是强调技术是新的。因为你能找到东西跟着学，就说明技术其实已经不新了。

选择什么技术栈

每一个技术栈都有存在的理由，都有最适合使用它的场景。某一个技术栈最适合解决某一类问题，你选择它有时是因为它适合解决你的问题，有时是别人觉得它适合你或者你的问题，总之你一定有个理由，也许你只是想用用不同的技术。这些都不重要，重要的是，你终于开始了学习之旅。

对于我学习基于 Node.js 的技术栈，有两个原因：

- 我以为它能轻松胜任写一个 Web 管理系统这类任务，而且后续还可用于后端服务。
- Node.js+Express+AngularJS+Mongoose+MongoDB 这条路，一门 JavaScript 基本就贯通了前端后端。

了解你的问题和技术栈的特点

我们新学一门技术，往往是为了解决用现有技术栈不太容易解决的问题。因此，很有必要了解你面临的问题，看看解决问题的关键在哪里，可能的路径有多少。然后就要了解备选的技术栈能做什么、擅长做什么、有什么公司或什么产品使用了它，这些产品的特点和规模与你面临的问题有没有可比性。

这一点其实和前面的“选择什么技术栈”是相辅相成的，甚至可以合并同类项。

列出待学习的技术点

熟悉了待解决的问题，选择了一个技术栈后，就要静下心来，进一步深入了解技术栈，看看究竟这条路上有多少技术点是必须要学的，把它们列出来，这样才可以一个一个来学，不至于学着学着忘了这个漏了那个。

对每一项技术学到什么程度也应心中有数。当然对于从未有过开发经验的人来讲，怎么讲可能都是心中忐忑，不过没关系，且去学就是了。后面还会讲到。

寻找合适的学习资料

互联网时代，知识盈余，信息过量，你想学什么东西，Google 或百度一下，有关联的主题成千上万，没关联的主题万儿八千，总之信息浩如烟海，而我们却如落水的蚂蚁，实在有点浩淼水面终生难渡之感。

学过 C，学过 C++，学过 Windows 编程，学过 MFC，学过 Python，学过 Qt，学过 JavaScript，学过 Java，学过 Android，学过 SQL，学过 Node.js，学过 Objective C……

对我来讲，我觉得一本好书是最好的开始，我每次新学一个东西，第一件事就是了解有什么书，而且我一定要买纸质的。所以，我的书越积越多。这次学习 NEAM（Node.js+Express+AngularJS+MongoDB），我买了《Node.js+MongoDB+AngularJS Web 开发》这本书，觉得挺不错的。

一本系统讲述某项技术的书，可以让你少走好多弯路，短时间觉得过程漫长，长远看却是捷径。

另外最好的资料就是某项技术的官方 SDK，一般技术都有文档，API，Quick Start，Guide，Tutorial，Demo，Example……这些资料是顶好的。不过我觉得结合书对照着来看效果更好。

如果能在网络上找到前辈们写的系列文章，也是蛮好的。但很多文章就是蜻蜓点水或只涉及某一细节，适合对该技术有了一定了解再去看。不过现在有很多人在分享，质量高的文章确实也很多。但是对新学一门技术来讲，来回寻找、判断、选择，时间成本太高。

网络上还有很多视频教程，也可以一看。

总之各种资料都会有，根据你自己的学习经验，选择最适合你的那种就好。

坦然面对问题，不放弃

新学一门技术不遇到问题是不可能的。

比方说你换了技术树，原来是玩 C 的，现在改玩 J2EE 或者 C#，能不遇到问题吗。比方说你原来玩 HTML 和 CSS，现在改 Swift……

当然也不全是技术问题，还会有心理上的问题。比如你急于看到你的问题能够三两下就解决，一旦过个三五天你还不能用新学的技术解决实际的问题，就开始怀疑、不自信了，就自我否定、打算放弃了，就有换个技术试试的想法了……我想说的是，软件开发这里，没有哪个技术你学个三两天就能用它干出点名堂来，两三周都难。所以心要静，要坐得住，意志要坚定，把一个坑挖出水来再说。

有时可能公司或领导不给你那么多时间，但是这也不是大问题。8 小时之外你还有很多时间，还有周末，总之你要是想坚持，就一定有时间，时间就一定不是问题。

也许你有个阶段会笃定“我实在理解不了、掌握不了这门技术”，真的想放弃，但我想还是坚持多一秒吧，拐过这个路口就有彩虹了。

你不放弃自己，就没人能放弃你。

保持对最终目标的清晰认识

我们最终要做出一个什么东西，具有什么功能，解决什么问题，一定要明确，这样才会有稳定的目标。

宏大的愿景会让人身心澎湃，会带来源源不断的动力。

只有树立明确的目标，才能进一步往下拆分出一系列细小的子目标。最好把大目标分解，列到纸上或记到电脑上，不管怎么样，记下来很重要，而且要放在容易看到的地方，每天都看看、想想，保持敏感、紧迫感、期待感。

不断实践，积累自信

新学一门技术，不断的小练习、持续的实践是非常重要的。每学完一个知识点，都要动手写点代码来看看效果。有时一个 Hello World 都让人激动不已。

每一个小的成就都会传达给你正能量，加强你的信心，都会让我们离目标近一点。

按我的经验，最好一两天就能有小的 demo 完成，这样刺激就会不断。要是你看上十天书才动手写一个 demo，那可能还没到十天就放弃了。

目标太远看不到产出和成长，就非常容易懈怠。即刻的成就感刺激很重要，它能推着你前进。如果你能把每个小 demo 都和前面从大目标拆分出的小目标结合起来，形成每一个小 demo 完成大目标的某一个细分小目标的话，那就太好了！

记笔记

毋庸置疑，学习过程中会遇到各种问题，又因为我们是奔着解决问题而去的，很多知识原本是成块的，而我们只用到某一点，或者某个技术还有很深入的内容而我们只是蜻蜓点水……

总之我们有十二万分的必要记笔记。笔记也不需要那么正式，不一定要遵循什么特定的格式，也不一定要多么美观，只要易懂即可。

笔记可以记录下列内容：

- 学习过程中搜集的资源链接。
- 遇到的未决的疑问。
- 那些很宽而我们用得很窄的知识点。
- 那些能进一步深入而我们浅尝辄止的知识点。
- 某个功能可能存在的其他实现方案的蛛丝马迹（要知道，知道茴香豆的“茴”字有几种写法对技术人员来讲可是极好的）。

待你长发及腰或者胡茬胜草，就可以回头整理你的笔记，回顾并进一步学习。这样对新学的技术的理解就会更进一步。

步步为营，持续推进

新学一门技术，是还走马观花好还是细嚼慢咽好？

前一阵子有个哥们买了我的书《Qt Quick 核心编程》（http://item.jd.com/11587406.html），我们反反复复邮件来往了 *N* 次，他让我非常意外也非常感动。这个哥们每次发邮件都会提出针对书中细节的疑问，附上拍摄的图片，从图片可以看到用手写在书上的标注，还能看到用绘图工具加的标注，我一下子就震惊了，这是多么认真对待新学一门技术这件事的人

呢！这哥们最终指出了书中的 4 处有效谬误，我记录了下来，准备再版时修正过来。他告诉我，他在新学技术时，书里的代码都要照着敲下来验证！

我自己在学习时，也会隔三差五跑一跑书里的代码，但绝对没有这位朋友这么认真，他让我深感惭愧。

对于初学编程的人来讲，我建议向这位朋友学习，步步为营，持续推进，不要怕慢，要求稳求细求扎实。多花些时间是值得的，养成良好的学习习惯会终身受益。

对于有经验的程序员，在新学习一门技术时，也要尽量避免走马观花，要力求落到实处。当然此时我们往往很急切地要奔向目标，也有能力判断哪些知识、技能是解决问题所必需的，可以挑着看，有针对性地学，但是，对于分拣出来的这些点，就要持初学者的虔诚态度，一步一个脚印，稳步前进。

投资自己要放开手脚

程序员最本质的财富是自己，挖掘自己的潜能，让自己不断增值，这是最要紧的事。像学习技术拓宽知识领域这种事情，再怎么花钱都不过分啊。

有的朋友可能因为经济原因，在学东西时不大愿意买书、买视频、买资料、买 VPN（很多技术资料都在国外站点），觉得不值当。

你像 Android 开发的在线文档、AngularJS 的文档，来来回回找啊试啊特别浪费时间，而你买个靠谱的 VPN，一年也就花一两百元，非常划算。

还有就是书，有的影印版的按美元定价，折成人民币好几百元钱，不过也是值得的。知识无价。

在增值的方向上投资自己，才是最明智的。

跨越心理障碍

其实新学一门技术的有两类人：

- 没有开发经验的新手。
- 掌握了某种技术的人。

这两类人在新学一门技术时会有不同的心理感受，也会遇到不同的问题。

对于没有开发经验的新人，进入到一个公司才开始学习开发技术，会有比较大的压力，

因为什么都不懂，有时别人说的词语都理解不了，巨大的未知领域会带来巨大的压力。有些人喜欢挑战，会特别兴奋，每天都自 High 到不行。有些人比较容易自我否定，碰一次壁就收缩一下，碰多了就跟无花果干似的皱得不行毫无生气。

其实呢，“我没学过”“不知道”“不会”绝对是正常现象，不必为此忧虑，有智慧的人也不会因此看不起你，因为大家都是这么过来的，没哪个家伙娘胎里蹦出来就能开发出一个 APP 来。遇到问题少关注“别人怎么看我”“我这么差”“我搞不定”“我好难受”这些负面的东西，多想怎么解决，积极尝试解决问题，这样才会越来越好。

有过项目经历的人在新学一门技术时，通常会有急于求成或错误预期这方面的问题。因为有了一定经验，就会觉得再学别的什么应该手到擒来易如反掌，往往会期待几天就能搞出点成绩来，领导也会有类似的期待，而实际上，这种浮躁的心态往往会导向不好的结局。你想啊，哪一项技术是那么容易的，随便搞搞就能搞定？

我 2014 年其实动过学习 Node.js 技术栈的念头，在网上看过几天资料，没搞定就放弃了，当时就是因为错误地估计了新学一门技术的难度导致心态浮躁，在预期结果没有如期到来时很容易就动摇了。

所以有经验的人，新学一门技术，就要努力放空自己，让自己归零，这样会比较容易学进去，坚持下来。

坚持，坚持，再坚持

重要的话说三遍，坚持很重要，所以要坚持，坚持，再坚持。这是最后的制胜法宝。

给新程序员的 10 点建议

我是一个爱瞎琢磨的程序员，根据我的个人经验和观察别人的结果，我发现，如果一个新手可以坚持并实践这么几点，就能够很快从青涩变成熟，完成最初的角色转变，融入到让你欢喜让你忧的软件开发工作中。

接纳自己是一张白纸这个事实

我觉得这是一个首要的前提。也许你很优秀，有很强的学习能力，有强大的信念，有超强的小宇宙，有百折不挠的韧性……但是，你没做过，你确实是一张白纸。这是一个客观事实，必须要承认。我们所做的一切，都是在接纳现实的前提下展开的，唯有承认这一点，其他美好的事情才有可能。

平静地看待这一点，不要妄自菲薄，也不要好高骛远，从一条线开始，慢慢绘制自己的蓝图，一切可能就会慢慢变成现实。

关注自己能做到什么

虽然我们是一张白纸，但还是可以做很多事情，我们通过学习，将能做更多的事情。我们会遇到各种各样的问题，会产生自己这个不会、那个不会的想法，可能安装一个 IDE 都会出问题，可能 3 行代码都会遇到 10 个编译错误，可能 Run 起来你不知道结果是对是错是否符合预期……这都没关系。

我们要看自己今天做到了什么，明天又能做到什么，一周后能做到什么，一个月后能做到什么。我们关注积极的方面，不要因为各种挫败而将自己陷在不良情绪的泥潭里，自怨自艾、自我否定、羡慕嫉妒恨、失望、沮丧……这些情绪都是我们的敌人，都是有不良居心的魔鬼，它们只会坏我们的事。我们要逃离它们，不给它们机会，我们的对策就是，看自己能做什么，看自己将来能做什么，看自己需要做哪些事情才可以达到明天、下一周、下一个月的目标。这样我们就能积极行动起来，就会把时间花在有价值的地方，我们也会慢慢更有价值。

如饥似渴地学习

我们是一张白纸，就要如饥似渴地学习，抓住一切机会学习，读书，求人，读代码，上班 8 小时，下班还有更多时间，我们可以把一切可以利用的时间都投入到相关知识和技能的学习中来。

当你连一个 DEMO 都不知道如何创建，当你连一个 Hello World 都跑不起来，当你弄不懂变量、操作符、函数、对象、类、实例、控制语句，当你不知道那些被老手们当作常

识的递归、链表、单例、观察者、MVC、CRUD、RBAC 时……你唯一能做的，就是把自己变成一块干燥的海绵，持续不断地去吸收各种知识。

这个过程是快乐、幸福的，你会发现随着你掌握的知识越来越多，随之你能解决的问题越来越多，你会发现今天的自己和昨天的自己不同，这个月的自己和上个月的自己不同，每一天、每一周、每一月都是一种惊喜。就让自己变成一个小孩，收获简单的快乐。

别怕犯错

很多新手怕犯错，其实不必。**犯错不等于失败**，犯错也是一种学习。

我们不必担心别人因为自己犯错而看轻自己，要知道，此时此刻我就是一张白纸啊，我没有经验啊，我什么都不会啊，我不犯错才是奇怪的事情。

就像小孩子蹒跚学步，跌倒是很正常的事情。可是它会在跌倒中收获宝贵的经验，摔几次，摔几天，它就学会了。

犯错一点也不可怕，可怕的是不总结、不归纳、不吸取教训。

迎难而上

我遇到一些新入行的朋友，面对一个任务时会畏缩，说“这个我不会啊”“那个我从来没弄过啊”“我搞不定啊”……

当然会遇到困难！作为新手，你遇不到困难才怪。要把困难当作机会，只有做你眼下搞不定的事情，你才能提升，明天才能搞定更多的烂事。

所以，要迎难而上，把困难当作猎物，当作锻炼的机会，要见猎心喜，用于挑战自己。

记录问题和心得

我觉得这一点非常重要。在初始阶段，记录遇到的问题、学习心得，总结解决问题的经验，将会对后来的进步提供非常大的帮助。我们要模仿牛的反刍，不断从有限的实践经验中总结、消化，获得更多的营养。

同时这也会帮助我们养成良好的工作习惯，良好的习惯，会让我们终生受益。

适时求助

我觉得应该给自己设定一个期限，若在期限到达时还搞不定，就向别人求助。

向别人求助是正常的事情，每个人都可能会求助于别人，一个人不可能搞定所有事情。不要担心别人太忙没时间帮你，也不要担心欠别人的人情。要知道“帮助别人”这件事本身，就是一件快乐的事。在帮助别人的过程中，提供帮助的人会收获快乐。多数程序员都很乐意帮助别人。真的。你获得了帮助，你在遇到别人的求助时也乐于提供帮助，这就够了。别的不用想太多。

提前告知上级你真的不能搞定

没错。就是这样，一旦你经过了种种努力，确信自己不能搞定手头的事情，一定要尽早告知你的上级和小伙伴。

一个软件项目的周期，通常是由最晚结束的那个任务决定的。

假如你是一个新手，碰巧 Take 了一个任务，你当然不希望自己的任务是最晚结束的那个。可实际情况表明，你有很高几率扮演那个角色。我要说的是，你需要清楚地判断自己手上的任务的状况，如果你真的不可能搞定（或者不可能按期搞定），一定要提前告知你的上级，把你的状况、任务的状况都说明白，这样你的上级才可能有时间做必要的调整（比如分配新的资源、延长时间等）来应对可能的项目延期。如果你在最后一刻才告诉别人，那就没有补救的机会了。真的。

别担心别人看轻你，即便经验丰富的优秀程序员，也会碰到搞不定的事情。软件开发的性质就是如此，永远会有超越你能力边界的问题蹦出来。这是很正常的，让别人知道，让大家能想办法补救，是正确的措施。

向优秀的同伴学习

三人行，必有我师。

想让自己变优秀的一个好办法就是和优秀的人走在一起。

你一定要知道（自己观察、别人告诉你、上级告诉你）你所在的团队中哪些程序员在

哪方面比较优秀，比如张三代码简洁能自解释，李四设计能力强，王五总能搞定技术难题，赵六具有卓越的影响力，钱八善于分析问题，赵十能深入浅出地讲解技术问题，诸如此类，如果你能够清楚，一方面可以在自己遇到问题时向正确的人求助，另一方面，你也可以主动学习。

让上级为自己指定导师

如果可能，让你的项目经理或者领导为你指定一个导师（mentor），让导师来帮助你制定阶段目标，并且让他来辅导你达到目标。

有个实践，叫 OJT 培训（On the Job Training）。不过不一定每个公司都运用这种机制。如果你所在的公司没有，那么可以让上级给你找一个导师，或者自己找一个。

有经验的导师，既能引导你少走一些弯路，又不会越俎代庖直接帮你把事情搞定。这样你就会自己走在正确的方向上，在节省不必要的精力浪费的同时，又能自己收获实作的经验。

这 10 个问题去哪啦

我决定花些时间梳理一下我程序员生涯中看到的各种问题，看看它们都去哪儿了。

外科医生剪箭尾

程序员经常这么处理 Bug：找不到 Bug 出现的根源；再花点时间看看能不能搞点沙土增高一下堤坝（想象黄河河堤的形成过程），嘿，真不错，还真找到了，我只要这里塞点代码堵一下那里扔几个判断围一下，这 Bug 貌似就不出现了啊……说实话，这种策略是把自己当作了外科医生，把软件当作了中箭之人。因为外科医生弄不懂内外科的分别也找不到起出箭头的办法，只好通过剪断箭尾来安抚病人，殊不知箭头还在肉里，要是烂在肉中，病人的疼痛只会更加厉害。

我管不着啊

现在我们开发软件通常都是兵团作战，有时我们就会发现嵌入到代码库中的某段代码很烂，比如各种条件分支未覆盖全面、逻辑不严密，比如变量名凌乱导致可读性很差，比如类库设计不合理违反了单一职责原则（Single Responsibility Principle，SRP）……，此时大部分程序员的做法是：算了，这代码是别人的，我管不了那么多，还是先忙我的吧……

有的程序员在开发过程中会感觉到软件系统设计有问题，就会认为这是系统分析师或架构师的责任，因为“我可能没他牛”“我也没有更好的解决方法”“我不大可能说服他采用我的方案”，于是就放弃了，就按“我只要实现我的模块就行了”这种方式往前走了……

还有的时候开发人员觉得产品某处不合理，交互方式不符合用户使用习惯，但因为这是产品经理的责任，就被“算了，我只负责实现”这种想法给俘虏了……

还有的时候阿猿觉着 UI 给的这个商品详情展示界面看着色调不协调，可还是放弃沟通照着做出来了……

类似的情况很多很多……

须知我们这么多人叮叮当当忙活，虽然每个人干的事不同、担的责不同，但其实是在造同一条船，没有谁的工作与别人无关，任何一处出现纰漏，都可能导致我们造出的船无法下水或者倾没深水不能远航。

也许问题不会在用户那里出现

很多软件在测试过程中都会出现一些难缠的 Bug：没有规律，几率极小。就连微软这样的大公司发布的操作系统，也要不断地打补丁。

作为程序员，有时遇到一个概率很小的 Bug，会因为难以重现而决定先不管它，我们会这么说服自己、测试、产品经理：这个 Bug 可能是极端操作或操作不当引起的，用户根本不会那么用，所以在用户那里很可能不会出现，就算出现了，按这种概率，也是极个别极个别的用户才会碰到，而我们还有很多其他功能没实现，不能因为这种小概率的 Bug 影响整体的进度。OK，事就这样成了。可是我们每个人头上都悬了一把达摩克利斯之剑。

要知道，用户环境比测试环境更为复杂，测试环境无法重现的、几率很小的 Bug，很可能就被某个用户硬生生碰上了，对这个用户，发生一次就是百分百啊，对他来讲，根本不存在千分之几万分之几这种鬼东西，他只知道他碰上了，你跟他说这个 Bug 实在是很难

重现真的是小概率事件，可他的痛苦真真切切摆在那里，怎么可能认同你的解释？这个用户失去了，还会有下一个“幸运”用户，最终一定是星星之火燃起燎原之势，你的产品口碑会烂到家。

跳过技术难题，别影响进度

软件开发的一个特点就是随时都会遇见未知的问题。比如开发互联网电视产品，很可能会遭遇音视频失步问题，具体表现是有时一个影片刚开始播放几分钟就失步，有时播放半个多小时才失步，有时超过一个小时也不失步，有时在这台机顶盒上失步，有时在那台机顶盒上失步……怎么办？

有一种做法是，承认这是个技术难题，一时半会儿难以解决，别让它影响进度，先跳过去干别的，以后专门抽时间解决它。

这种做法很常见，它的后果也很常见：项目进度还是无可奈何地被这个难题拖累了。

别人都这样

有时你会碰见这种开发团队：执行力差，产出率不高，忙闲不均。作为一个原本追求上进的社会主义四有青年，你看到这种状况往往在开始时感到不正常，想要去改变它，可是久了之后发现你一个普通程序员改变不了什么，要是你执意去做这个事劝那个人，反倒把关系给搞僵了，种种担心或实践让你意识到，你不是那个可以改变团队面貌的角色，怎么办？

最后的最后，现实的结果往往是：算了，睁一只眼闭一只眼吧，我管好我自己就行了。更让人不能接受的结果是：大家都这样得过且过，我也没必要这么卖力，于是自己也慢慢放松自己得过且过了。

慢慢向生产率下限看齐，这是非常令人伤心无奈的事实。

我们后面会追上进度

软件项目，延期的占大多数。很多项目，干着干着就感觉要延期了。可是很多团队的项目经理、开发人员在这件事情上却表现得充满童话风格：只要解决了某个问题，只要某

某猿加个班往前赶一赶，最终会赶上交付期的……

结果呢……

没奖金、不加薪干个什么劲

不是每个软件公司都能持续盈利，都能大把大把地发奖金，都能隔三差五地加薪。实际情况是，相当一部分公司在艰难地生存，没有办法发可观的奖金，也没有办法按承诺的节奏加薪。在这样的公司里做开发，程序员会怎么想、怎么做？

公司效益不好，与我有什么关系？不发奖金，不加薪水，我干嘛还那么卖命？我何必还像以前那么努力？混混看吧。

这是一部分程序员的真实心态，我也曾经有过。要知道，当我们在一个看不到将来的环境里工作，又没有金钱刺激时，难免心灰意懒进而放纵自己随波逐流。可后来我意识到：你在公司提供的平台上通过为公司做事而修炼自己，你得到的，远不止是薪水，经验、历练、成长都是自己的，并且永远没人能从你身上拿走；时间是自己的，虚度了再也无法追回；无论何时，我们都是在为自己的现在和将来工作，而不是为公司、为老板工作，我们不单单是一个拿薪水帮别人干事的人，如果停留在这种意识上，那么我们一定是在扼杀自己成长的机会，浪费自己宝贵的生命。

还有×××呢

受经验、能力和眼界所限，有时某个难题看起来会超出程序员的能力范围，让我们感到再花精力也是枉然，于是我们就不打算研究下去了，告诉自己：算了，还有技术大拿杨过呢，还有技术经理郭靖呢，不行还有技术副总风清扬，再不行公司会想办法，反正后面还有人……

反正我不是项目经理时有过这种想法，我当了项目经理还是有过这种想法，甚至我做部门经理时也曾经这样想过，就连项目总监，可能也难免有类似的想法……直到我成为一个公司的技术合伙人，我的后面，再也没有人可以接我丢下的烂摊子了……

Sigh……，这是多么令人不适应的情况啊！

现在我要说的是，无论何时，都应该把自己当作最后一道防线，要坚守阵地，有时一夫当关可以万夫莫开，有时一人放弃却也会导致全线崩溃失地千里，我们要想尽一切办法

解决问题，这个问题到自己这里就是要终止，必须破釜沉舟不留退路。

反正不是我的责任

不可避免地，程序员会碰上项目延期这种事。有的人会回顾项目执行过程中的种种问题，包括反省自己的不足、梳理团队成员协作上的问题、琢磨任务安排与进度跟踪是不是出了差错……这是积极的做法，我觉得每个程序员都该这么做，然而实际情况却并非如此，有相当一部分人会这么想：领导安排的事，我该做的都做了，我有什么办法，这是大家伙儿的事，很可能是项目没管理好或者需求老是变化，跟我有啥关系……

算了，换个环境

程序员换工作的频率很高，有人两年换一次，还有人一年换一次，更有人一年换几次，这么频繁地换工作，真实的原因是什么呢？

有时是干得不顺心，觉得自己空有一篮子想法奈何领导不重视；有时是觉得团队的氛围不行，不能让大家安心做事，自己也无法保持积极向上的心态，在技术和职位两方面都无法获得成长；有时是觉得产品没有希望，看不到未来，越干越迷惘；更多的时候是，哎呀，那个公司真有钱啊；当然，还有时就是莫名地觉得该挪挪地方了……

假如不是钱的原因，你还是决定换个环境，你觉得这样也许会好一些……果真如此吗？

你最终决定跳槽，一定是发生了什么问题，导致你无法容忍现状，但一定要问自己：自己现在面临的问题，究竟是公司、团队、产品的问题，还是自己的问题，换个环境是不是不会再出现类似的问题？

也就是说，我们决定跳槽之际，需要深入分析，问问自己的内心，到底是外在的原因导致自己再也无法忍受现状，还是发自内心地觉得自己需要做出改变，去寻找新的方向。我们要弄明白自己的期望，搞清楚自己想要什么，这样才不至于才出泥沼又入火坑。

题外的话

作为程序员，在开发和工作的过程中，难免会遇到各式各样的问题。面对问题，每个人都有自己的处理方式，不管你采取哪种方式，问题都不会自动消失。假如你采取的措施

暗合“希望问题自行消失”这种模式，一定要注意，这是非常低级、非常有破坏性的习惯。用这种模式“解决”问题的次数多了，久而久之，我们自己慢慢就成为问题了：因为这样做事实上是在逃避问题，拒绝成长与成熟。我们希望跳到一个问题很少的环境里开开心心地工作，可这样的环境无异于理想国和桃花源，它们只存在于想象之中，如果我们自己的内在没有发生改变，不能培养出接纳问题、解决问题的能力，将在寻找的路上疲于奔命却永远也找不到这样的地方，最终绝望，最终选择“放弃自己”这一看似最不可能的结果。

既然各种各样的问题不可避免，那么唯一可行的就是：直面问题，解决问题。

软件开发问题多多苦难重重，只有接受这个前提，只有理解这一生活与工作的本质，我们的工作与生活才会变得美好起来。

我们往往在问题面前缺乏耐心，想让问题马上解决，如果东搞一下西搞一下还解决不了，就想丢在那里不管，迈步跨过去或绕道而走，我们总想尽快脱身，尽快缩短与问题接触的时间而不愿花足够的时间来应对这种不舒服的感觉，不愿冷静地分析问题。因为直面问题解决问题，真的是需要绝大的勇气和卓越的耐受力来承担切肤裂心的痛苦。

然而，我们必须面对。如果选择逃避，只会越来越糟，问题会跟着我们走，我们走到哪里它们就撵到哪里。套用香港警匪片里的经典台词：“出来混，迟早要还的”。我们也一样，遇见问题，逃是逃不掉的，越逃问题越多，到后来总有你要还的那一刻。

既然如此，何妨早一日受苦、早一日解决、早一日浴火重生？让我们接受“先苦后甜”的生活模式，坦然面对问题。问题可以开启我们的智慧，激发勇气。为解决问题而努力，我们的思想和心灵就会不断成长，心智就会不断成熟，解决问题的能力就会不断获得提升。我们终将因为直面问题而淬炼成钢，在混乱中成就自己，最终成就独特的自我，找到自己的方式活出个样子。这就是成功了。

程序员三重境界，你在哪一重

古今之成大事业、大学问者，必经过三种之境界：“昨夜西风凋碧树。独上高楼，望尽天涯路。”此第一境也。“衣带渐宽终不悔，为伊消得人憔悴。”此第二境也。“众里寻他千百度，回头蓦见，那人正在灯火阑珊处。”此第三境也。

——王国维《人间词话》

王国维先生从几位大词人的名作中摘出这一段原来写男女之情的名句来概括做学问的

三境界，一语中的，非常透彻。而这三境界，居然也可以用到程序员身上！

第一境界：迷茫前行

“昨夜西风凋碧树。独上高楼，望尽天涯路。”出自北宋著名词人晏殊的《蝶恋花·槛菊愁烟兰泣露》。

王国维借此词描述人生的迷茫。而程序员之路开始之际，我们也有类似的感受。时常有迷惘，不知道什么语言、什么框架、什么行业、什么产品才是我们该选择的，难以了悟前路如何，自然也看不到十年之后我们又在何方……总之各种迷茫，孤独而不知道前路几何。就算有人对自己说“何愁前路无知己，天下谁人不知君”，也不能宽慰心中的彷徨。

然而路终究是要走的，而且是一步一个脚印走出来的，没有目标，那就不断地试探吧，我们永远年轻，总有试错的成本，终会找到自己的灯塔。一旦有了灯塔，我们就进入第二境界了。

第二境界：追逐目标，无怨无悔

“衣带渐宽终不悔，为伊消得人憔悴。”这句出自北宋柳永《蝶恋花·伫倚危楼风细细》，千古名句啊。柳永太牛了，号称“凡有井水饮处，皆能歌柳词”。

这种执着，被王国维拿来譬喻成大事者对事业的不懈追求，恰恰是极妥帖极形象的。同时它也刚刚好可以用来形容程序员之路的第二境界：向着目标前进，九死不悔。

一旦我们渡过了最初的迷茫，在技术上或者产品上找到了目标，我们就有了灯塔，就可以辗转反则思之念之上下求索，就可以渡过水雾迷茫的人生之海。最重要的是，能有一件事为之努力、奋斗，这挥洒汗水的历程，是多么幸福。

第三境界：终有所获

“众里寻他千百度。蓦然回首，那人却在、灯火阑珊处。”出自南宋杰出大词人辛弃疾的《青玉案·元夕》。

这几句几乎人人皆知，王国维用它们比喻经过长期的努力奋斗而无所收获，正值困惑难以解脱之际，突然获得成功的心情。

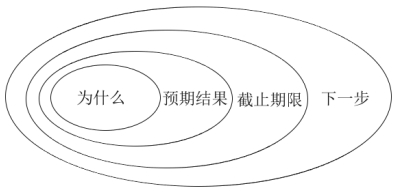
其实，只要我们找到正确的方向，执着地在既定的道路上坚定不移地追求，经过千百劳作，必有所成。“只有经历苦难，才能体会什么是幸福”，这也是程序员的生活写照。

知易行难

王国维先生总结的这三种境界妙到毫巅，程序员生活中符合这三种境界之事俯拾皆是。比如我们从小处着眼来说，学习技术，最初的犹疑迷惘于选择哪一种，就是第一境界；选定之后，一往无前，坚持不懈孜孜以求，就是第二境界；当历尽艰辛终于精通，能够庖丁解牛，自然喜不自禁。

然而这三种境界，却又不是那么容易就能达到的。大部分都可以从容做到第二境界，但要想突破第二境界进入第三境界，就没那么简单了，只有果敢坚韧、不屈不挠的斗士，才可以逾越自我极限，造就与众不同的成功。

效率提升圈



效率提升圈

在工作和生活中，有些事情我们会经常性地拖延下去一直完不成。不知道你有没有想过为什么？不知道你有没有尝试着找一些能让自己轻松自如控制一切的办法来打破这种令人沮丧的糟糕状况？

工作效率低下的原因

大部分时候，我们无法轻松自如地控制一切，没办法完成那些始终萦绕在脑海的事情，是因为：

- 不知道为什么要做。
- 没有确切地认定它们的预期结果是什么。
- 没有明确的截止期限。
- 没有决定你下一步的具体行动到底是什么。
- 如果你的某个任务能够明确上述四点，一定可以高效地完成。

为什么工作效率会倍升

当我们找到了做某件事的意义（为什么要做）后，就会从内在产生动力。

当我们明确了预期结果，就能够准确衡量什么算是完成，就能定义出清晰、明确的目标，而有效的目标又可以让我们以“如果……就……”的方式来展望结果，这样的展望会继续带来激励，产生动力。

当我们的目标有了时间限制后，就会反向给我们施加压力，压力带来紧迫感，进一步会转化为动力。

当我们找到了下一步行动计划后，就可以立刻开始。一旦开始，事情就成功了一半，最终就会形成良性循环，帮助我们完成事情。

这就是明确为什么、预期结果、时间期限、下一步行动会大大提升工作效率的原因。

为了能够让我们更好地规划自己的事情，我设计了一个简单的表格：

为什么要做	给做某件事赋予对你有关的某种意义（“不做老板不高兴”通常不是一个好的理由）
预期结果	预期结果应该能够明确衡量什么是完成
截止期限	最好能够精确到天，下一年、下一月都是无效的期限
下一步计划	下一步计划应该是简单的、可立即开始做的
精力预算	规划你的时间和精力，保证对这件事的投入

希望这个小工具能对你有用，能帮助你提升做事效率。

程序员保值的 5 个秘密

在国内，很多人说程序员是吃青春饭的，一开始说过了 30 岁就得转行，后来又有人说

35 岁是一道坎……看起来好像程序员真的和风尘女子一样，注定有朝一日会年老色衰不再受人青睐。

我要说，不是这样的！

现在就实打实地来看看程序员的价值之路在何方。

应用技术

相当大一部分的程序员都在做应用层面的开发，所做的软件用来解决特定场景的问题，给用户的工作和生活带来方便。

开发一个应用，经常会用到高级语言和框架，比如 C#和.NET，C++和 Qt，J2EE，Ruby on Rails，比如 Python 和 Django，Java 和 Android，Objective-C 和 Cocoa Touch，比如 JavaScript, PHP……太多了，数不胜数。

应用层面的开发技术，很多人觉得门槛低，小年轻儿和富有经验的老资格差别不大，后浪会把前浪拍死在沙滩上，所以当新一茬韭菜长成时，老一茬就黄了被弃之如敝履了。

其实不是的，即便从纯技术的角度来讲，你对一个语言和框架的理解与把握程度，也会严重地影响开发效率和产品质量。

什么样的人会被轻易替代？不求甚解、似懂非懂、干了多少年还看不透所用技术的本质、遇到问题仍然懵懂不解茫然无措，这样的程序员，注定很快被一大波正在赶来的小鲜肉挤下工作岗位。

假如你对一门语言的各种特性都体察入微了悟于心，假如你对一个框架的机理和各种应用场景都有独特的理解和丰富的实践经验，那么你几乎是不可替代的，你已经成了这个语言和框架方面的专家，价值不可估量。你的一句话就可能为一个项目节省几个月的时间，别人一筹莫展的问题到了你这里分分钟就搞定。

所以，不要理会“能 Run 就行”“完成任务就可以了”“用不到学那么深干什么”之类的话，在你用完一项技术解决了一个实际问题、满足了某个需求之后，继续钻进去吧，多学一点，深入一点，日积月累，你必然会与众不同。如果上班时没时间，那就下班以后继续投入。记住，你的学习和研究都是为了自己，不是为了老板，不是为了项目，你唯一的产品就是你自己，而这个产品值得一辈子打磨。

高难技术

有一些技术的门槛是相对较高的，比如汇编语言、操作系统内核、驱动……正因为门槛高，回报也高。比如你熟悉 Windows 内核或 Android 内核，能够熟练撰写各种驱动，那么找个月薪五六万元的工作不成问题。唯其稀缺，所以保值。

算法

大部分程序员其实不懂算法，都是用框架里的模块拼积木。如果你妙悟算法真谛，那你就超越了 90%的程序员了，你的重要性和价值将不可估量。

比如你精通图像处理算法，或者精通视频编解码算法，或者精通搜索推荐相关的算法，或者在模式识别领域有建树……试看谁能挡得住你前进的脚步！

业务

在一个行业内持续积累，对业务的理解到位，积累深厚，你的价值是巨大的。不信你去浏览招聘网站上的岗位需求，99%都要求相关行业背景。所以，选择一个靠谱的、前景好的行业非常重要，只要这个行业能够不断发展、前进，你的积累就是有价值的，你自己就是不断增值的。

特定领域的业务有一定门槛、金融、电力、电商、彩票、考古、医疗……在这些行业里，你是个业务门儿清的程序员，即便技术不是特别优秀，价值也是无限的。

产品意识与思维

究竟什么是产品意识呢？我认为产品意识一般包含商业意识、用户意识、创新意识和团队意识。

——《人人都是产品经理》

互联网时代，产品为王。有产品意识，懂产品思维的程序员，是最受欢迎的人群，也是最能做出好产品的程序员。

通俗地讲，商业意识就是要思考这个产品能不能卖出去，好不好卖。这个意识一定要

有，程序员虽然很少直接接触市场，但是一个产品成功与否，多数时候就是看市场表现，如果你能关注市场，从市场的角度来看待技术实现，接纳“技术为市场服务”的理念，那么更容易把产品做好。

用户意识是最容易理解的，它是说我们要从用户的角度来思考这个产品该如何设计，因为产品好不好用最终是由用户说了算。如果你在实现产品时也能站在用户的角度来思考，斤斤计较一个功能是否符合应用场景、是否与用户的行为特点吻合、是否贴合用户的使用习惯，那么恭喜你，你超越了 90%的程序员——大部分程序员是按产品经理和 UI 设计师的要求来实现产品。

创新是人类发展的源泉，是社会进步的动力，同样也是产品的核心竞争力。但这里所说的创新不一定是那种颠覆性的创新，也许是把众多不被重视的细节做到更好，也许只是把另外一种理念引入到这个产品中去，也许是像海底捞的员工一样给用户不一样的感受，这些都是创新。创新意识是产品经理必须有的，否则，他永远不会有进步，产品永远也做不好。创新意识也是优秀的程序员必须要有的，否则他就不能把一个产品实现得很好，不能把产品的核心竞争力演绎到极致。

当下已不再是孤胆英雄单兵作战的年代，我们要想做好一个产品，多数时候都需要一个团队。团队意识是必不可少的，你是和一帮程序员在一起，你还和产品经理、UI 设计师、业务分析师、项目经理等在一起，如果你能融入团队，并且能影响、促进其他人为共同的目标做出有效的努力，那么，你这样的程序员是无敌的，是国之瑰宝！

说了这么多，究竟你适合在哪个方向下功夫，还需要你自己来做决断。你可能一看便知自己适合做什么，那么无须赘言。也可能你有些茫然，不要担心，我们都还有大把的时间，你可以试着去做，发现不合适了换个方向再来，直到踏上适合你的路。

别被技术绑架

通常我们说程序员需要在某个技术方向上积累到一定的厚度，要能够运用技术有效地解决实际问题。可是当程序员在某一项技术上浸淫长时间之后，却经常会出现另外的问题，那就是：看待问题时受限于自身的技术积累。

下面从几个方面来谈这个话题。

一定有某一项技术最适合解决某个问题

有的人对 C++比较熟悉，在开始一个新产品或新项目时，比如做一个 Android APP，就会考虑怎么用 C++来实现目标，于是就会去找可以用在移动端的 C++框架，比如 Qt，NDK+Native Activity……这是一种受限于现有技术的情况，其实可以跳过 C++的藩篱，直接找 Java 去！

在我看来，技术是用来解决问题的，当我们要解决某个问题、实现某个目标时，技术可能有很多种，可能每种都可以实现，但不同的技术面对的难度、付出的成本一定是不一样的！对于公司、个人来讲，都应当选择综合成本最低的那项技术。

当我们做技术方案选型时，应当问“这项技术适合解决哪类问题？”，而不是问“我掌握的技术该怎么解决这个问题？”。

如果你总是想尽一切办法用已经掌握的技术来解决所有问题，虽然这种担忧未知、害怕变化和尝试的心理可以理解，但我觉得这不是一种健康良好的心态，也不是一个有益的习惯，它其实发出了“你被所学技术奴役”的告警信号。

要知道，我们学技术是用来解决问题的，要能够灵活有效地控制和运用掌握的技术，而不是把自己交给某项技术，反受技术左右。

一字槽口的螺钉就要用一字螺丝刀，内六角的螺丝就要用内六角的扳手。这是很直接、很简单的道理，我们在生活中会下意识地遵循这种规则，而一旦我们面对技术这种相对复杂的事情时，却往往不能回归到事物的本源，这是需要我们思考的。

换工作时拒绝换技术

当程序员熟练掌握了一门技术（比如 Java，C++，Qt，Spark……）后，他在选择新工作时就经常会主动给自己设置路障，不用 Spark 的公司不去，不用 Qt 的岗位不考虑……其实我觉得这是不必要的。

学习一门技术当然有成本，用精通的技术来解决问题会有得心应手、驾轻就熟的感觉，能够事半功倍。这是非常好的。但是，等等！难道你真的打算一辈子把自己局限在某项技术上吗？难道你认为你干了 *N* 年程序员就只获得了这项技术？

不是的。你熟练掌握了某门技术，当然是你非常重要的收获。但是这绝不是最重要的，真正重要的是下面两点：

- 学习能力。
- 解决问题的能力。

对，你没看错，这才是一个程序员在精通一门技术后真正的收获。

你通过掌握、精通一门技术，发现自己的学习模式、确认自己的学习能力，一通百通，再学其他技术就会快很多，因为你获得的有关学习的经验和认知是通用的。所以，我认为发现自己的学习模式、强化自己的学习能力是真正核心、重要的收获之一。

还有，技术只是一把剑，这把剑的威力如何，就看使用它的人怎么样因地制宜审时度势地运用它。攻守之道，妙乎于心。真正的高手，摘叶飞花皆可伤人。这就是解决问题的能力了，同样的太祖长拳，在乔峰手里就有摧枯拉朽的气势和震慑人心的威力，这就是乔峰的本事，这个本事是独立于太祖长拳的。程序员也一样。所以，解决问题的能力是一种真正重要的收获。

如果你确认自己已经收获了学习能力和解决问题的能力，那么具体的技术就已经不重要了，它也不应该成为你选择新工作新产品的绊脚石。不设限，天高海阔任你遨游。

招人时限定精通某种技术

我们会发现，绝大多数公司在招聘开发工程师都会列出诸如“精通 J2EE”“精通 MySQL”“熟悉 Hadoop”等非常细的技术要求。

对公司来讲，招聘拥有相关经验和技术的人才能够大大降低一个产品（项目）的成本，这本无可厚非，因为公司都是成本敏感的。但是对于程序员来讲，有时这是不公平的。

不公平体现在两点。一是没有相关技术经验的程序员会因此而失去学习新技术的机会，对于那些刚毕业不久或意图转换技术方向的人来讲则更是残酷的。二是拥有匹配技术的程序员，自身发展会因为公司的这种倾向而受到限制，甚至裹足不前，他们会觉得，原来你仅仅是想利用我已有的技术和经验来解决你的问题啊，技术人员的价值是在不断战胜新的挑战的过程中提高的，失去面对这种挑战的机会，他们的能力和水平就会原地踏步，甚至回落、下降。

其实在我看来，招聘人员时，拥有相关技术并不是最重要的，一个程序员是否具有学习能力和解决问题的能力才是关键！有了出色的学习能力，他一定能够快速掌握产品需要的各项技能；有解决问题的能力，他一定能够解决新产品开发中不时跳出的各种意外。也就是说，选择程序员时，公司更应该从一个人的才干和能力出发，而不是盯在某项具体的技术上，合适的人会带给公司更多，唯有把正确的事情交给正确的人，成功才可预期。

总之，不论是程序员还是需要程序员的公司，都应该从人才的核心出发，发现关键才干和能力，不要被具体的技术蒙蔽，不要让已有的技术成为包袱，不要让自己的视野受到不必要的限制。心不设限，将来就没有终点。

程序员接私活的玄机

很久以前，我曾经接过几次私活（在“千奇百怪的程序员”一节中我列出了全局变量控的一段代码，就是我帮别人干活时碰到的代码），现在不断有人想让我帮他们干点私活，但我都拒绝了。

为什么我曾经接私活现在又坚决不沾呢？

为什么接私活

有人可能说：还不是为了钱嘛！对，钱，是很多程序员接私活时主要考虑的方面。但也不全是，还有相当一部分人愿意接私活是出于其他原因。

先说说我的三次私活吧。

我曾经教过一个人学习 Qt，在线的，一对一辅导，帮他熟悉 Qt，协助他完成一个测试工具的开发。他人在美国，先通过西联汇款给我打了一百多美元，然后我们就开始了……后来因为我总是想按我的方式教他，而他老嫌我让他自己先学习这个学习那个，闹崩了……

这次经历对我是一次尝试，我当时其实是想了解一对一的在线辅导是否可行，事实证明这里面可能存在很多坑，一是沟通问题，二是教学适配问题，一不注意就会不欢而散。所以，现在这种事情我也不尝试了。我刚在订阅号“程序视界”后台拒绝了一个类似的请求，见下图：

我的第二次私活经历，是帮一位买了《Qt on Android 核心编程》一书的老师的忙，修改一个潮汐预测软件，从已有的软件代码里将预测功能提取出来封装成一个动态库。虽然那个软件是开源的，写软件的人也很牛，但那份代码真的是满满的全局变量，所以我在“千奇百怪的程序员”一节中专门展示了部分代码，并戏称这类程序员为“全局变量控”。

我还接过一次私活，帮人搭建一个客户端软件的框架，中间需求变了几次，有一次我和朋友带孩子一起去秦镇吃凉皮，在河道里遛弯时还接到对方沟通需求的电话。不管怎样，拒绝了一部分需求变化，接受了一部分，反复几次，后来总算交活了。

这是我作为程序员的三次私活经历，它们带给我很多的思考，也让我能够结合个人体验，客观地来解释“程序员为什么接私活”这个问题。接私活一般有下面几种原因：

- 缺钱，就想多赚点钱。
- 业余时间闲得慌，没事干，把时间共享出去赚点外快（这也是共享经济的一个基础）。
- 因为信息差或减少了中间渠道，私活拿到的报酬按时薪计算，比工作多很多，很有诱惑力。
- 想在实战中培养自己感兴趣的技能。

如果你接私活，会是哪种原因？

接下来，再聊聊我现在为什么不接私活了。

私活与成长

我打算换个角度，从干私活与个人成长的角度再来聊聊，这个角度也可以给我们接不接私活、接什么样的私活提供更多的参考。

接私活有两种情况：

- 赚更多的钱（复制已有经验，使用已有技能，通过付出更多的时间赚更多的钱，单位时间赚钱能力基本不变。此时很容易被高昂的沟通成本和反复的需求变化搞得身心交瘁，工作、家庭都受到影响，整个人的状态都不好了……）。
- 让自己更值钱（通过精进现有的主要技能或者培养感兴趣的新技能，让自己更值钱，单位时间赚到更多的钱），眼下报酬多少不计较。

从这个维度来看，我个人更认可第二种。因为它是对自己的价值投资，会让自己将来更值钱，比如时薪从 20 元提升到 200 元，比你干多少复制经验和技能的私活都更有成就感。这也是我对私活的终极看法：

最好的私活，能在经历、技能、方向、未来可能性上使你更丰富，让你更值钱。

这也是我在“月薪 3 万的程序员都避开了哪些坑”“程序员如何谋划出月薪 3 万元”“大龄程序员的未来在何方”等章节中一贯坚持的观点：让自己更值钱比当下赚更多钱更重要。

当你开始考虑如何让自己更值钱时，你就会主动发现自己、丰富自己，业余时间做私活就成了让自己更值钱的一种途径，你就超越了“私活”与快钱的概念，你的未来也就有了更多的可能性。

小结，共享经济与私活

现在有很多产品，比如实现、程序员客栈、解放号、我帮你、极客邦、啦啦私活、兼职猫等，从共享经济的角度来宣传，以共享程序员的时间和技能为出发点，能让你在工作之外赚到更多的钱。

但你要不要做呢？先问自己下面三个问题：

- 缺了这笔钱生活会无以为继吗？
- 会影响到我当下的生活质量吗？
- 能让我更值钱吗？
- 相信你可以做出更好的选择。

假如你想成为全栈工程师

让我来发挥一下剪报君的特长，下面是百度百科对全栈工程师的说明：

全栈工程师，也叫全端工程师，英文 Full Stack developer，是指掌握多种技能，并能利用多种技能独立完成产品的人。

上面的定义基本上已经比较直白了，再举两个例子就更明白了。

假如你是一个 Web 开发者，如果你既能做前端（需要熟悉 HTML、CSS、JavaScript、H5 以及 Bootstrap、EasyUI 等各种前端框架），又能做后端（需要熟悉 Java 或 ASP.net 或 PHP 或 Node.js 或 Go，选项太多就不一一列举了），可以独自完成一个类似电子商务网站的产品的开发，那你就算是全栈工程师了。

假如你是一个 APP 开发者，既能开发 Android 应用（需要熟悉 Java 和 Android 框架），又能开发 iOS 应用（需要熟悉 Objective-C 或 Swift 以及 Cocoa for iOS），还可以开发应用需要的后台（需要熟悉 Node.js 或 Java 或 Go……），总之你可以独自开发出一个覆盖 Android 和 iOS 的且有业务后台的 APP，那你也就算是全栈工程师了。

现在明白全栈工程师是什么了。那么让我们来看看，假如你想成为全栈工程师，都该了解些什么……比如怎么成为全栈工程师，全栈的好与坏，选择哪条技术栈来贯通……

全栈 ABC

关于全栈工程师，有一些周边信息是你必须了解的。

1．对自己产品的渴望

注意，我放在第一位的，不是对技术的渴望，而是“对自己产品的渴望”。就像一个男人，渴望要一个自己的孩子，99%会选择走“谈对象、结婚、做爱、生娃、养娃”这样一条路。作为一个工程师，如果你对产品的渴望就像那些思子若狂见了人家孩子就想抱走的男人或女人，那你就可能会愿意成为全栈工程师。

一个真正的全栈工程师，会从生活中发现问题，洞察需求，设计解决方案并迫不及待地实现产品。而为了实现产品，他愿意去学习任何领域的知识和技能。注意，他们学习某个领域的知识和技能，并不是为了成为那个领域的专家，而是为了完成自己的目标。他们视野开阔心存高远，不会拘泥于技术，假如挥一挥手，就有产品、设计、开发人员蜂拥而至为他们开发想要的产品，那么他们丝毫也不介意扔掉所有的技术。换句话说，只有在一个人既对产品有迫切的渴望、又没人帮他实现时，他才会走上全栈之路。所以，全栈只是实现目标过程中的副产品，目标才是首要的。

2．时势造全栈

前面说了，当一个人渴望自己的产品又没人帮他实现时，他如果不能放下灼心的渴望，就可能走上全栈之路。这其实是形势逼迫。类似的还有另外一种情况，也可以逼着一个人成为全栈工程师。那就是：加入一个缺人的创业型公司。

大公司人员充沛，一个萝卜一个坑，个个都是螺丝钉，让你一个人搞定所有事情的概率很小。而小公司、创业型公司则不同，人员往往极度匮乏，一个人得顶几个人用。你是

搞前端的，如果后端没人你也得顶上。你是搞 Android 开发的，如果 iOS 没人你也得顶上，后台没人说不定也得顶上。就这样你很快就全栈了。所以，有一种全栈工程师，是被别人养成的。

3．思维方式和学习能力

有的人，拿刀逼着也成不了全栈工程师；有的人，把他扔到没电没网络的荒漠，他也可以走上全栈之路。

这其间的差别，就是思维方式和学习能力。

从思维上讲，要想全栈，你就不能给自己设限。把自己定位在前端工程师或 iOS 工程师上，任尔东西南北风，咬定青山不放松。这样是不行的。应该心随好猫意纵天高或者鹤舞白沙我心飞翔，总之因时而变、因势而变，需要用什么就学什么，服务器没人搞？我来搞；Android APP 没人写？我来写……这样打破了自我设限，就具备了成为全栈工程师的基础。

除了这种自我设限的思维模式需要破除，还有一种定势要破，那就是精通每一项技术。对于热爱技术的人来讲，搞精每一项技术是很强的诱惑，简直比门口走过的妙龄女郎的吸引力还大。这种思想要破除，因为门门技术都精通耗时必然会很长，影响你实现自己的产品，所以，只要你学到的那部分能够顺利帮你实现目标，就可以挥一挥衣袖，继续前行了。

一旦思维上破除了定势，具备了成为全栈工程师的基础，如果你有很强的自我学习能力，那就真的可以顺利走上全栈之路了。而假如你学习能力稍差，面对新技术总是寻寻觅觅寻不到入门的路，那么恐怕自己不会被逼成全栈，说不定被逼到白头。

全栈的好与坏

全栈工程师的好处就是涉猎技术很广，能够很快运用他所了解的技术开发出产品原型。所以，很多全栈工程师后来走上了创业之路，成了创业者；或者进了创业公司，成了技术合伙人。他们视野开阔，思维活跃，对技术和产品都很敏感，是创业期不可或缺的核心。

然而，正因为全栈工程师的技术是横向发展的，广博有余而精深不足，所以你提到什么，他都能侃上半天，但你要问一些基础的知识点，他可能答不上来。假如全栈工程师不去创业或不加入创业型公司，而是应聘某一个技术方向的岗位，那么在面试时就会比较吃亏，因为他用到的大部分技术，细问起来，别人可能都会觉得“有了解但不深入”。所以，这可能会影响他的求职。不过，如果先一专再多能，然后有意识地选择目标职业，就可以

避免这种情况。

选择哪条技术栈

对于全栈工程师来讲，往往是没有选择的。比如你做 APP，你要全栈，你基本上就没多少选择余地，Java、Objective-C 都是必需的。所以，很多时候是产品和形势选择你成为全栈工程师，是做着做着成了全栈。而不是为了全栈而全栈。比如选择 MEAN（MongoDB+Express+AngularJS+Node.js），比如 J2SE+SSH+Android +Objective-C + Cocoa……

假如你的目标就是成为全栈工程师，你会发现好多东西要学。那假如你要做一个产品，比如因为你酷爱炒股，要做一个简单的股票 APP：跟踪自选股行情，支持自定义提醒，界面要清爽干净，平台要支持 Android、iOS 和 Web。那么你在做这样的产品的过程中，就会被逼成全栈工程师而不自觉。因为，做你喜欢的事是不会觉得累的。

最后，娱乐一下，送给全栈工程师一个来自 Twitter 的段子：

刚来这家公司面试的时候，老板语重心长的对我说：“虽然工资不高，但是你可以在这里获得快速的成长，这对做 IT 人来说是最重要的。”

现在，两年过去了，老板没有骗我。

我看起来已经像是 60 岁的人了。

——From. Scswga（https://twitter.com/Scswga/status/548684273717215232）

10 分钟搞定工作周报

有不少老板要求下属写周报，希望通过周报了解部门每个人及整体的工作情况，但很多人在写周报时往往遇到很多问题，比如：

- 花费很多时间回忆一周做的事情。
- 不知道写什么。
- 不知道用什么格式写。

我也给老板写周报，一般花费时间不超过 10 分钟。总结了一下，只需要做到如下两点，就能解决上面提到的三个问题，达到 10 分钟搞定周报的效果：

- 每天记录工作笔记。
- 固定周报格式，汇总工作笔记形成周报。
- Scrum 有个环节——每日站会，团队所有成员参加，同步自己的状态，回答三个问题：昨天完成了什么、遇到哪些问题、今天准备做什么。

这是一个非常棒的形式，能让我们每天都知道自己要做什么，让我们每天都比昨天前进一点点。我把这个应用到日常工作中，就解决了周报、日报的问题。

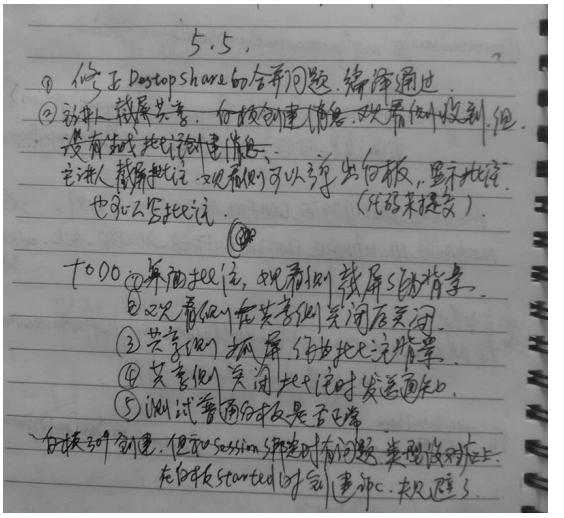
每天记录工作笔记

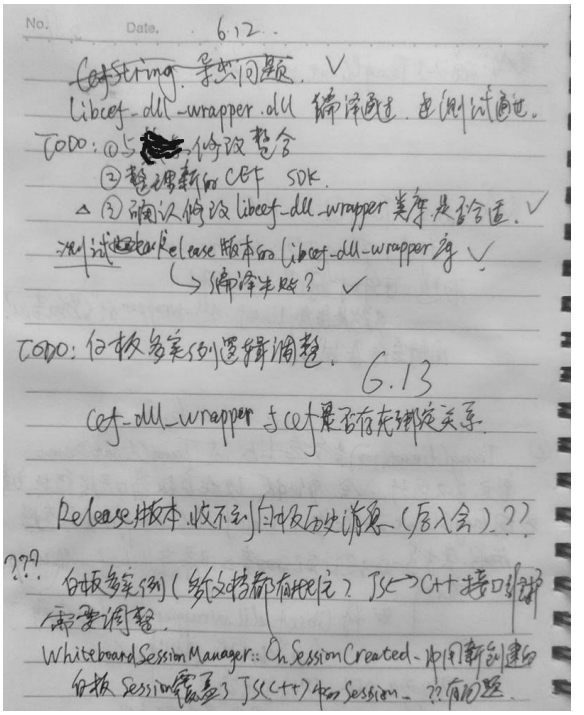
我使用软抄本来记录笔记，比较有成就感，拿在手上翻看，也很有质感。需要的时候，可以拍照或用全能扫描王之类的软件转化为数字版，也很方便引用。

工作笔记的内容，我参考每日站会的问题，略略做了调整。一般包含下列几点：

- 今天完成了什么。
- 今天遇到了什么问题。
- 明天要做什么。

下面是我的工作笔记摘录：





每天下班前，我会把今天完成了什么标注出来（在 TODO 后面打对勾或者文字记录），把遇到的问题记录下来，最后，会认真地琢磨 TODO List。

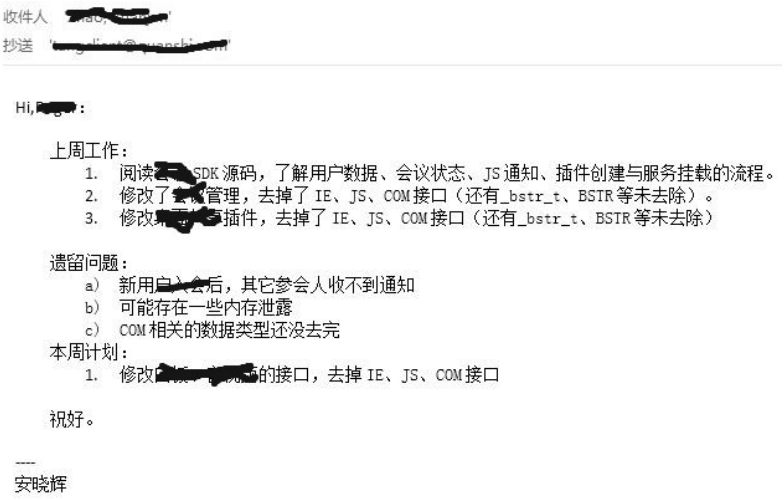
TODO List 很重要，很管用。一方面可以让你放下今天的工作，更快地完成个人状态切换，另一方面，也为明天的工作目标奠定了基础。有目标才有效率。

每天早上我会翻看笔记本，看看昨天列的 TODO List，想想是否有变化，有的话就加进去，然后开始一天的工作。

我很喜欢有道云笔记的 Slogan：记录，成为更好的自己。一旦记录成为习惯，你就拥有了持续向前的力量，每天都会更好一点。

10 分钟写周报

当我们有了每日工作笔记后，周报的材料就有了，只要固定一下周报格式，撰写起来就几乎不用费脑子了。下面是我的周报：



收件人 [redacted]
抄送 [redacted]

Hi, [redacted]：

上周工作：
1. 阅读[redacted] SDK 源码，了解用户数据、会议状态、JS 通知、插件创建与服务挂载的流程。
2. 修改了[redacted]管理，去掉了 IE、JS、COM 接口（还有_bstr_t、BSTR 等未去除）。
3. 修改[redacted]插件，去掉了 IE、JS、COM 接口（还有_bstr_t、BSTR 等未去除）

遗留问题：
a) 新用[redacted]后，其它参会人收不到通知
b) 可能存在一些内存泄露
c) COM 相关的数据类型还没去完

本周计划：
1. 修改[redacted]的接口，去掉 IE、JS、COM 接口

祝好。

安晓辉

由上图可以看到，我的周报也参考了每日站会的问题，包括下列三方面内容：

- 上周完成的工作。
- 遇到的问题。
- 本周工作计划。

因为我们周一开例会，周一发周报，所以周报中我用了上周、本周这样的词，如果你周五发，可以是本周、下周。

每个老板对周报的要求不一样，格式、内容可能都不同，但有了工作笔记，汇总一下，就能适用于各种格式，完全不是问题。

习惯的力量

养成记录工作笔记的习惯，我们每天的工作会更有效率。有了工作笔记，每周的工作总结就再也不是问题了。当你养成每天记录、每周记录、定期回顾总结的工作习惯后，你的成长，每天、每周都能看得见。

管 理 迷 思

混日子不是你的错，根源在这里

程序员会经常干着干着就没干劲了，有时一两天觉得没意思，有时十天八天甚至几个月都觉得提不起精神，上班开了电脑不知道干什么，浏览浏览新闻，翻翻微信朋友圈，然后就下班了。

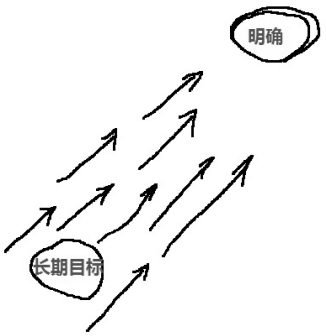
为什么？

团队没有真正明确的目标

我待过各种各样的团队，也混过日子，就算现在，有时也不免把上班的几个小时稀里糊涂混过去。我极端讨厌消磨时间混日子，所以一旦我发现自己在混日子，就会琢磨：为什么我懒洋洋的什么也不想做？其实答案没那么复杂，大家都知道。

如果一个团队的多数人都浑浑噩噩，最重要的原因就是：团队缺乏真正明确的目标和角色定位。

研发人员一旦对团队目标和自身角色定位缺乏明确的认识，就会感到困惑不解、压力重重、焦虑不安，进而无所事事，通过在毫无意义的事情上虚掷光阴耗费精力来对抗无聊的上班时间；而当他们拥有明确的团队目标和清晰的自身定位时，就会积极振作起来，自行朝着目标前进。



那么，什么才是真正明确的目标呢？

我在“设定有效目标的 SMART 原则”一节中曾有简要介绍，有效目标具有下列五大特征：

- 具体（Specific）。
- 可衡量（Measurable）。
- 可实现（Attainable）。
- 相关性（Relevant）。
- 时间限制（Time-bound）。

简单说，有效的目标，既鼓舞人心又明确具体，既充满意义又容易衡量。团队成员一看到这样的目标，就知道“什么样算是成功”，这样就会让人自觉向着目标前进——每个人都渴望成功，当努力一下就可以成功达到时，人会更愿意付诸行动。

有效的团队目标

很多研发团队，年初的目标通常类似于下面这样：

- 改善代码质量。
- 随时响应客户需求。
- 添加产品团队提出的新功能。
- 预研 XXX 技术，做一个 DEMO。

这些目标一点都不具体，也不能用来指导研发人员的行动。假如你拿到一个任务，领导说，“先做做看吧，具体做成什么样子还没定”，你会有干劲吗？反正我是没干劲，我都不知道做成什么样算好我怎么做？

而下面的目标则好一些：

- 团队产品的 Bug 率控制在千分之 2.39 以内（CMMI3 的标准）。
- 5 月 31 日发布三方视频通话功能。
- 预研 H.265，实现视频编码 DEMO，720p 和 1080p 的视频，编码后，码流大小相比 H.264 要降低 30%。

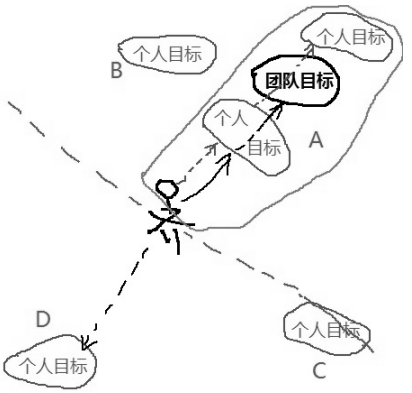
但这些依然不是最好的目标。一个真正的好目标，能够指导团队和个人获得更明确的目标意识，帮助团队找到通往未来的路径。

比如 Google（谷歌）有一个热气球网络计划，其目标是“让全世界每一个角落都能连

接网络”。这是一个很棒的愿景，如果真能实现，世界上那些欠发达地区、偏远的农村、落后山区的人们就能用低廉的代价接入互联网，与时代同步。然而从有效目标的角度来看，“让全世界每一个角落都能连接网络”是一个伟大的愿景，还应该细化，比如，针对中国，修改为“2020 年让中国的每一个角落都能连接网络”就是一个很棒的目标了。

要想让团队持续前进，就要不断提出新目标，一个目标接着一个目标，形成一个不断上升的目标阶梯。

个人目标与团队目标



根据个人目标与团队目标的关系，A、B、C、D 代表了一个团队成员的四种典型状态。

A 代表自燃态，此时个人目标与团队目标一致，团队和个人契合度最高，个人最容易斗志昂扬积极前进，不用任何敦促个人就会自发奋进。

B 代表可燃态，个人目标与团队目标有所偏离，但实现团队目标对个人目标也会有贡献，反之亦然。此时个人也愿意做事，如果有适当激励或监督，个人能把工作完成得很好。

C 代表不燃态，此时团队目标的实现对个人目标没有任何帮助，个人完全没有动力工作，在高压之下可能会应付了事。

D 所代表的情况则更坏，团队目标和个人目标背道而驰，此时个人处于阻燃态，他非但不会为团队出力，还可能为了实现自己的个人目标而主动做出损害团队目标的事情。

作为一个团队的管理者，应该能够识别处于不燃态和阻燃态的成员，探查原因，能改善则留，不能改善就果断辞退。

对于个人来讲，如果发现自己的目标与团队背道而驰或相互没有帮助，最好换个团队，让自己处于 A 或 B 的状态，这样对自己、对团队都有好处。

假如一个人没有明确的个人目标，那么他最好以团队目标为个人目标，这样既可以分享到团队的成长，也可以让自己不那么无聊，甚至可以慢慢发现自己的目标。

团队目标缺失时，个人怎么办

假如你所在团队，经常处于目标缺失的状态，导致你不知道干什么，整天无所事事，怎么破？

其实，如果你有个人目标，这就不是个太严峻的问题了。因为无论个人目标还是团队目标，都能激发人努力前进。甚至个人目标更容易让人废寝忘食积极投入工作，这也是可燃态员工整天打了鸡血似的拼命工作的原因——个人目标与团队目标完全一致。

有了个人目标，就可以自顶向下分解出执行计划，一旦团队目标实现或者团队没什么目标，你有大把时间不知道如何度过时，就可以把当下的个人目标与计划拉出来执行，这样你就不会因为团队的目标问题而荒废自己。

如果团队是因为业务或产品原因，间歇性地没有明确目标，而你又有个人目标可以充分利用时间，那倒也是挺好的事。

如果团队常年没什么明确目标，那就走吧——待在这样的团队里只是消耗自己，对你的将来没任何帮助。

既没团队目标，又没个人目标

对于这种状况，我有一个非常恰当的比方——浮萍飘在雨后的小水坑里。这种状况的最终结果显而易见：团队和个人都完蛋。

假如你不想就这么浑浑噩噩地在缺乏目标的团队里凄凉地等待死亡，就应当问自己几个问题：

- 我想成为什么样的人？
- 我最想要的是什么？
- 我的职业目标到底是什么？
- 假如我只能在一件事情上做到出类拔萃，这件事情是什么？

然后，然后你可以经常性地找人聊聊这个问题，找到自己的方向就好了。

缺这两点的 Scrum 注定失败

很多软件公司，在遇到产品交付延期、开发周期长、产品质量低下、运维成本高、响应需求慢等问题时，会尝试引入敏捷来改善。然而，大多数公司的老板和管理者，从一开始就是错的，注定要失败——因为他们本末倒置，忽略了人的能力和责任心，只期望从过程和控制上来改善。

个人或团队绩效低的原因

一个人做不好工作，主要有两个原因：

- 自我认知不清。
- 缺乏责任心。

自我认知不清指一个人不知道自己想干什么、能干什么、能把什么干好，简单说就是没有自知之明，不知道自己的位置。

缺乏责任心是指一个人认为手上的事不是他的，是别人的（比如老板的、公司的、经理的），干好干坏和他没什么关系，所以这件事唤不起他的责任心，做起来就没动力，结果自然就是做不好。

一个公司或团队，要想有好绩效，就应该在这两方面努力：

- 找到有自我认知的成员。
- 给成员选择和做决定的权利，让其产生责任感。

搞明白了这个关键，接下来我们就结合 Scrum 最重要的一个阶段——冲刺启动会议——来看看为什么大多数 Scrum 实践会失败。

启动会议的四个关键点

Scrum 中的 Sprint 启动会议（计划会议）必须开，而且 PO（Product Owner），SM（Scrum

Master），ST（Scrum Team）都要参加。

一般来讲，Sprint 启动会议要做下列事情：

- 进行需求讲解。
- 挑选 PBI（Product Backlog Item），也就是由客户选出重要的 PBI。
- 拆解 PBI 到 SBI（Sprint Backlog Item）。
- 讨论每个 SBI 的工时。
- ST 挑选任务。

很多团队会把启动会议分解，1、2 两件事放一起做，3、4、5 三件事放一起做。

当我们单独来做需求讲解和 PBI 时，就会很容易走回老路上去：只有所谓的“核心人员”参与讨论。

很多团队在做 Scrum 实践时，会嫌启动会议中的需求讲解和 PBI 挑选耽误事，或者觉得没必要大家都参加，只要产品经理、项目经理、技术大拿参加就可以了，需求讨论好了再让其他团队成员参加。

这就在最开始犯了严重的错误。这样做，那些没参加需求讨论的成员，肯定没什么积极性！因为，这种形式和之前的瀑布或迭代，没有任何差别啊，都是一小撮人决定需求，然后告知或安排给其他成员做。告知、命令、安排，这些传统的控制手段，非但不能产生积极性，还大大挫伤和打击团队成员的积极性。

参与感非常重要，哪个成员没有参与决策的过程，他就没有积极性，也没有责任感，潜力和效率就出不来。所以，不要怕一开始花费的时间长，只要拿时间盒限制一下，不要离谱就可以了。

这是 Sprint 的起点，也是第一个关键点：Scrum TEAM 都要参与需求确认。这一步如果做错，对后面的影响很大。

第二个关键点就是从 PBI 到 SBI，和第一个关键点一样，要由 PO、SM、ST 共同完成。这个冲刺是 ST 在做，所以 ST 一定要对 SBI 认同，他们需要“我决定了我们这个冲刺做什么”的感觉，这种感觉带来掌控感，带来积极性。否则，就回到老路上：你让我干啥我就干啥呗，还能怎么样……注意到了吗，这是非常消极的做法。

当一个人不能决定他要做什么时，就会失去责任心，就会消极对待（或者无法主动积极）。

这一点下面的环节还会谈到。

第三个关键点就是讨论工时。合理的工时，才能让人心甘情愿地接受。

假如一个 SBI 需要 32 个工时，SM 非要压到 10 个，那么谁拿了这个 SBI 都会不情不

愿，结果一定是花费比 32 个工时更多的时间。

讨论工时需要注意一点：不要让技术能力最强的人来决定工时。显而易见，技术水平高的开发人员估出来的工时，要比其他人少很多，因为他会以自己的能力和效率为基准。如果某个 SBI 比较复杂，只有技术和业务能力强的人才能把控，那就把他们估出来的工时乘一个系数（建议乘以 2 或者 1.5）。

最后，对讨论出的所有工时，再乘以 1.2 或 1.5 的系数。这样最终的工时才是比较靠谱的。

最后一个关键点就是挑选任务。

注意“挑选任务”中的“挑选”二字，正是它们带来了参与感、自主性、积极性。

假如 SBI 分解完毕，工时估算就绪，然后 SM 说，张三你做这个李四你做那个，那么前面的功夫基本上就白费了。而这一点恰恰是很多 SM 常犯的错误。当我们被安排时，非但很难有动力去把一件事情完成得很好，甚至还会消极对待，任务差不多就不愿再花心思了。

一个人愿意选择需要他学习新技能的任务去做，这是非常棒的事情，说明他愿意挑战自己，渴望成长。如果你给他当头一盆冷水，就伤着他了……

Scrum Master 面临的挑战

很多 SM 陷入安排 SBI 的局面，往往也是因为一些现实的困难，比如：

- 技能与任务的匹配度。
- 进度压力。
- 关键任务给以往绩效差的成员会带来风险。

SM 对进度压力和关键任务的担心，其实也可以归结到能力与任务不匹配这一点上。技能与任务不匹配，短期有压力，长期是向好的。这一点需要 SM 去平衡，比如一个人领了有挑战的任务，可以再给它几个匹配度高的任务（匹配度高的 SBI 的工时可以调整得少一些，或者不乘以前面提到的系数），这样结合起来，对进度的影响就会小一些，而这位团队成员，仍然可以做他愿意做的事，还会有积极性。

只有 SM 允许 ST 自己选择任务，ST 才有积极性，否则，某人 Qt 做得好，十年都让他做 GUI，他也觉得没意思；另一个人熟悉 Android GUI，就做一辈子，人家想搞搞 iOS 都没机会，也会带来消极情绪。不要把人限制在某个围墙内，要允许、鼓励、帮助他成长。

这是 SM 要考虑的事情，也是他需要承担的压力和风险。

不管 SM 面临多大压力和多少风险，谨记：有选择才有责任感，有选择才有积极性。否则，就回到了告知、命令、安排、压制的老路上，Scrum 注定会失败。

小结

Scrum 不仅仅是一套模型，它的内在是“以人为本”，只有团队成员充分地参与和选择，才能带来积极性和责任心，Scrum 实践才可能成功。

70%的人离职只因领导有这四宗罪

很多人慕企业之名而来，却常常因为直接领导而离职，为什么呢？

紧盯 10%的错误

我在订阅号“程序视界”内讲了 F 的故事（回复“10142”查看），F 的领导的做法，完美演示了一个技术领导如何通过紧盯小错误来“完美挫伤”程序员，让其再也提不起干劲与活力去做事情。

摘录一部分：

有位朋友 F，第一次找我聊天是两个月前，他说部门换了经理，新经理特别强势，对他横挑鼻子竖挑眼，总是碾压他，他想到上班就很痛苦，上班时间也不能专心工作，老担心经理突然挑他的不是。

他还说，每次技术讨论或项目启动会议，别人说什么都没关系，哪怕与项目无关的话题、与正讨论的技术主题无关的话题都没关系，经理都面带笑容地倾听，就算别人说的技术见解明显是不对的，经理也会包容，顶多转换话题。而对他就不同，只要他一说话，经理要么说“这个不用说了，我都知道，还有没有别的？”，他压住受打击的情绪，挖空脑袋说点“别的”，经理往往在他话一出口时就不耐烦地打断他——“别说了，毫无意义，这种情况根本不会发生”……

他还说，每次发周报或者过项目状态，经理不是说他这个词写错了、那个问题还没解决，就是说任务的实现方法不好，为什么不用另外一种更好的，或者说周四发出去的那封信思路和措辞都不对，上面的领导和别的团队的人看了会有想法等，总是指出缺点和批评他，对他列出来的 1、2、3、4、5、6 那么多工作成果看都不看。而别的同事呢，经理都不这样对他们。

总之，F 觉得领导看不上他，对他有意见，有意针对他，想收拾他，让他知道厉害。他觉得很委屈，自己上班早下班晚，领导随便一句话都当命令去执行，工作一丝不苟，搞下来反倒没那些迟到早退偷奸耍滑的同事受待见。他说他想不通，最近压力很大，想到要上班面对经理就别扭、发怵、恶心，问我怎么办。

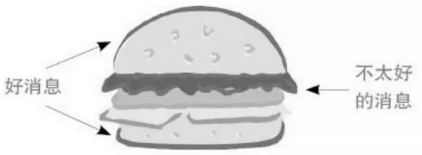
F 郁闷到不能正常工作，仅仅是因为领导对他“横挑鼻子竖挑眼”。在我看来，这里面有很大一部分原因，是这个技术经理犯了一个严重的错误：

从来不夸奖下属，但下属只要有一点闪失，就会被放大，并进行检视与责备。

看看 F 的情感诉求，我们很容易就能明白，假如经理在批评之前，能够针对 F 表现良好的部分给予鼓励与认同，就能有效提升 F 的干劲和活力。

然而大多数上司都像 F 的领导那般，将下属 90%的正确行为视作“理所当然”而直接忽视，只着眼于那仅占 10%的错误行为，管理变成了指责与批评，最后搞得下属一个个像霜打的茄子，没有建设性也没有战斗力。

关于这一点，汉堡原理可以参考：



指责与否定下属

作为一个从一线开发岗位晋升而来的经理，常常是因为技术能力突出才被提拔的，然而当他被提拔为经理后，他出色的技术能力很多时候会成为他转变为一个好的管理角色的障碍——因为他很容易以自己的技术能力为参考点来要求别人、指责别人。

“不对！不是这样！”

“错了，应该这样！”

“算啦，你别弄啦，让 A 来做！”

“这么明显的 Bug 你都没发现？！”……

这是技术经理们经常说的话，你是不是常常听到这样的话？

一边指责对方不成熟，一边否定对方，这就是我们的技术经理作为指导者的角色常犯的错误。

不是说下属犯错不能批评，而是要尽量优先考虑不挫伤他人、不剥夺他人机会与勇气的方式。对方之所以做不到，是因为现阶段的能力不足，但是能力不足只是一时的，将来还是有提高的可能性，而一旦你挫伤了对方的勇气，向想挑战困难的下属泼了冷水，往往会让对方感受到自己的无能并心生自卑，最终导致其失去克服困难的动力，甩手不干，不再成长。

顺带说一句，有些技术经理认为通过指责别人会彰显出自己的优秀，从中感受优越感，甚至会以为只有彰显了自己技术上的优秀才能体现出自己作为领导的称职性。这是一个误区，我在“程序员，这 12 个问题让经理比你痛苦多了”一文中提到了这一点，温伯格的《成为技术领导者》一书里也有详细介绍。

经理们可能会感到委屈：“打也不行骂也不行，到底怎么办才行？”

其实并不是说我们不能批评或指责下属的错误，而是要对自己的指导方式是否能有效提升别人克服困难的勇气、增强别人提升自己的动力做考量。因为只有团队成员都成长、能独当一面了，整个团队的战斗力才会提升。要知道，只有经理技术能力最强的团队是失败的团队！

要避免本节开始的那种指责与否定方式，其实也不难，只要在你脱口而出之前问自己一句话——“对方透过这种体验能学到什么？”——就行了。

害怕别人失败影响自己，不愿放手

人只能透过失败来学习，借由失败的经验，守护自己“想要改变”的决心。

然而作为一个软件开发团队的管理者，却往往是害怕失败和错误的，总因为担心一犯错就产生质量低劣的软件、延期而不愿意给某些下属独立解决问题的机会。

一部分技术能力强的经理，往往不懂得放手，忍不住伸出手来亲力亲为，结果只会导致下属产生依赖性，不能独当一面；另一部分不信任下属的技术经理，则采取能者多劳的

策略，拼命把重要的、关键的、难度高的任务都压到某一个核心开发人员身上，一方面导致其他人没机会成长，另一方面这位优秀的开发者很快就会因难以承受这么大的压力而选择退却或离职。

这样看来，“每个人都只干自己熟悉的事”这种司空见惯的分配任务行为其实是不科学的。我们不能因为一个程序员不熟悉某个任务所需的技术而剥夺他接触新领域的机会，也不能因为害怕某个程序员做不好而越俎代庖替他解决问题。

每个人都有自己的课题，必须自己来完成，也唯有自己完成才能获得成长。作为上司，应该放手，让下属有独立解决问题的机会，培养他克服困难的勇气，这样他才能成长。而你所要做的就是支持，以及容许一点点错误的代价。

谨记：不是因为做得到而放手，而是因为放手才做得到。

不聚焦如何解决问题

程序员可能对下面的场景很熟悉：

3 月 15 号要上线新版本，结果 14 号下午 5 点测试阿美曝出王二一个 Bug：APP 上的轮播海报，播到第三张时，一点击海报应用就崩溃。

经理焦大听了阿美的反馈，快步来到王二工位，满脸不快地喊道：“为什么老是到上线前出问题？到底怎么回事？”

……

王二听了焦大的话，第一反应就是：用得着这样让我丢人吗！然后，心里肯定不快，接下来的交流肯定是各自带着气的。比如，王二可能忍了焦大的气，转身去找阿美：“阿美为什么你们老是到上线前才测出这种 Bug!”而阿美可能又很委屈，反唇相讥：“为什么你们送测前都不自测一下呢？我们测试的时间就不是时间啊!”

到这里结束吧。我们来分析焦大和王二说的话，如果你仔细回味你所在的工作环境和生活环境，就会发现这其实是我们多数人遇到问题时的习惯性反应：先指责错误追问原因，然后再去解决问题。

这种方式往往会滋生嫌隙和怨恨，不利于解决问题。我们应当直接将焦点放在如何有效解决问题上。比如阿美报了那个 Bug，焦大可以省略指出问题与探究原因的步骤，直接问王二：“咱们该怎么查这个 Bug？”接着给出建议：“你看咱们从 APP 本地的 Activity 调用和服务器的轮播海报配置两方面来查，怎么样？”这样就可以将兴师问罪的行为转变成

解决问题的行动，同时也能让王二更有勇气和责任感投入到问题的解决上。

一味指责对方的错误、打破砂锅问到底，对解决问题没什么实际帮助，反倒会伤害下属，挫伤其补救与解决问题的勇气，更不利于其从错误中汲取经验自我成长。聚焦于如何有效地解决问题，一边给建议，一边唤起犯错人补救的积极性和勇气，会更好一些。

作为开始的结束

好啦，到这里我们列举了经理们经常会犯的与员工成长增值有关的四宗罪：

- 总盯着下属 10%的不足，忽略 90%的成绩，把管理变成粗暴的指责与批评。
- 用批评与否定来指导下属，不考虑这样做是否能帮助下属成长。
- 不懂如何放手与授权，分不清课题到底属于谁，剥夺下属自我成长的机会。
- 遇到问题先指责对方，之后才去解决，挫伤下属解决错误的勇气，降低下属借由错误成长的可能性。

正是因为领导们常常犯这四宗罪，导致员工当下很难愉快地成长，将来也看不到成长的机会，担心自己越来越不值钱和没竞争力，最终悻悻而去。如果一线经理们能够创造一个帮助员工成长的环境，相信很多人会反过来——因为领导而留下。

现在，请你想想吧：

作为经理，你犯了几宗？

作为一线员工，你的经理，又在使用哪种方式扼杀你成长的可能性？

有人离职时项目经理的反应

和 HR 聊天，发现今年（2015）的软件公司人员流动特别大。这也可能是我的错觉——也许年年都是这样，以我最近两年了解的几家公司来讲，人员流动率基本都在 30%以上，有的甚至更高，50%以上。更夸张一点的，有时一个开发团队会在很短的时间内十去其八。

作为一个开发团队的直接负责人，项目经理或部门经理，当有人离职时，会有什么反应？我在这里根据自己的经验再加上事后诸葛式的臆想来分析一下。

这家伙可算走了

有时经理巴不得某人离开呢，只是苦于公司政策或情面上过不去没办法请他离开，所以，当这个人主动提出离职时，经理内心如释重负，其实是欢喜的：哎呀，这家伙可算走了。然而，他还是要惺惺作态挽留一下。

他为什么要走

当经理没有预料到眼前的这个人要走时，离职的消息就显得突然，此时经理一般都会有这种反应：他为什么要走？这是人面对突发事件时的直接反应：怎么会这样？为什么？这是一时不能接受事实的表现。

当人不明白真相时，就会想办法了解真相，所以，接下来的反应就是臆想各种可能导致员工离职的原因，自己和自己对话，把每种可能性都琢磨一下，看看是否有挽回的可能性。

面谈，了解离职原因

经理寻思了一遍可能的离职原因后，就会以最快的速度找要离职的员工面谈，企图了解他为什么离职。尽管经理对各种可能的原因都自以为是地准备了劝说的理由，然而对于斟酌再三才提出离职的程序员来讲并没有什么用。除非这位员工就是想通过离职来胁迫经理给自己加薪。

离职原因林林总总，从大的方面看，无非两种：心，受委屈了；钱，没给到位。然而具体到某个人，原因总是扑朔迷离，有时经理很难搞明白这个人为什么要离职——因为他通常已经想好了云山雾罩的离职借口，做好了打死也不实话实说的准备。

当一个程序员离职时，最容易说给经理的理由其实是钱没给够：本人要谈女朋友要买房要养爸妈要养车要养娃……生活压力很大，刚好有家公司给的钱太多，根本没办法拒绝……类似这种，其实还是比较靠谱的。

而另外一种，因为心受委屈而离职，具体的原因，对经理来讲，就确确实实难以弄明白了。因为此时经理和他已貌合神离甚至分道扬镳了，立场不同了，他会觉得说什么都已无所谓，何必说出那些可能让还在这个单位的经理感到不爽的话呢。

考虑招人

一旦经理意识到不能挽回要离职的程序员的心，就会接受这个事实，转而考虑该不该招人、招什么样的人、怎么招人、花多少钱招人这些具体问题。

如果离职的人很重要，是团队技术链条中不可或缺的一环，那么经理就会立马启动招人计划，并和待离职的程序员商量交接时间；如果目前团队没什么事可干，经理可能会准备招人的计划，但没那么急切。

思考这个人离职的影响

当一个程序员离开一个研发团队时，经理对影响的担心是非常多的，如果他是一个负责任的经理，那么很可能会寝食难安。大概会有下列问题轮番摧残这个人：

- 是不是我管理有问题，是不是很多人都受不了我了？
- 大家会不会都觉得产品没前途了？
- 大家会不会觉得公司不行了？
- 是不是加班太多了，大家都快受不了了？
- 是不是我们给的钱低于行业薪资平均水平太多了，大家会不会一哄而散？
- 他有没有扩散他要离职的消息？
- 他有没有向别人透露他新东家给他的薪水？
- 他有没有鼓动其他人离职？
- 其他人怎么看待他的离职？可能会采取什么行动？
- 我要怎么消除他的离职带给团队的不良影响？
- 我什么时间告诉大家？以什么方式和态度告诉大家？
- 他手上的工作交接给哪位？那个人能否接受？
- 交接会不会影响整个项目的进度？
- 对部门的 KPI 有什么影响？
- 招人的话，对部门预算有什么影响？
- 招不到人怎么办？
- 如何向老板汇报？
- 老板不给人员、不让补招怎么办？这个人身上的任务怎么才能顺畅地分给别人？

有责任心、想解决问题的经理的日子其实也没那么舒坦，一次看起来普普通通的人员离职事件就很可能让他痛苦很长一段时间。

征求待离职人员的改进建议

有时经理自己也会感到团队的工作和管理中存在问题，他可能觉得要离职的人会更坦白一些，能够据实相告，所以，他很可能会问离职人员下列问题：

- 你觉得我们的团队有什么问题？
- 你觉得我在管理方面有什么问题？
- 我们在做的项目有什么问题？
- 对项目、对我个人、对整个团队，你有哪些建议？

有的人会实话实说，比如告诉你团队整体技术水平太差劲又没有大牛，告诉你大家都觉得在做的产品没什么前途，告诉你他不喜欢每天开会，告诉你某人太奇葩完全无法和他共事……

有的人会打哈哈，告诉你都挺好的没什么问题，他离职纯粹是个人原因。

思考自己的去留

在一个风雨飘摇的公司，这种情况极有可能发生。即便在一个看似稳定的公司，经理也可能早就厌倦了，正在等待机会。而下属员工的离职，有时会成为一种催化剂，加速经理思考有没有继续待下去的必要。

我的建议

其实，“铁打的营盘流水的兵”，没有人员不流动的团队，有人离职很自然，有新人进来也很自然。自自然然地看待，当某人要离职时，这基本上已经是事实，对于已经发生的事，再怎么着也是无可挽回了，所以，先要接受这件事，承认它发生了，然后再接纳自己的各种担忧和不爽，再然后，要相信美好的事情总会到来，一段缘分结束，还会有另一段到来。

至于有人离职对项目、团队、公司的影响，其实很多时候我们也只能猜测着做一些工作，然后不断调整，不断接受这种影响。这就好比有人往湖里扔了一块石头，一开始会溅起水花，然后是一圈圈的涟漪微澜，再后来，就只是你曾记得有过这样一个刺激发生过。

“包干到户”是最好的项目管理方式

先想想这几个问题：

- “包干到户”是什么概念？
- “包干到户”与软件项目管理有嘛关系？
- “包干到户”如何应用到软件项目管理上，它的挑战是什么？

时间在此停留 15 秒，你有答案了吗？

“包干到户”的特点

“包干到户”，又称“大包干”。承包合同中不规定生产费用限额和产量指标，由承包者自行安排生产活动，产品除向国家交纳农业税、向集体交纳公共提留以外，完全归承包者所有。即“交够国家的，留够集体的，剩下都是自己的”。

我们来看看包干到户的特点：

- 自由（生产费用、生产资料、粮食种类，都是自己说了算）。
- 自主安排，自行生产，自我负责。
- 多劳多得。

到现在农村种地还是这个模式，在适应环境的前提下，想种什么种什么，想怎么种怎么种，种得多种得好收获就多。这种模式看起来很落后，实际上却极大地提高了积极性，解放了生产力，创造了个人、集体、国家的多赢。

软件项目管理的现状

来看看咱们软件项目的实情，很多公司、很多团队，采用了各种各样的开发模式（瀑布、迭代、XP、结对、TDD、精益）和绩效方法（360 度考评、OKR、KPI、MBO），效果却往往比“包干到户”差远了。

为什么？

作为程序员，想想你是否遇到过下面的情景：

- 项目经理隔上一会儿就来问问你什么时候搞完。
- 你拿到开发任务时都不知道它是怎么来的，为什么要做这个。
- 大家准备了技术方案，开了 N 次会，老板一句话就用了他刚提的那个。
- 交付日期都是领导上安排的。
- 你想用 C++11，你想用 Scala，你想用 RabbitMQ，你想用 Swift，你做不了主……
- 你干得很好张三干得很差，可你们奖金、工资、福利差不多。
- 想申请个 MacBook 要经过层层审批，发一堆邮件。
- 你想申请个 VPN 上 Google，还得说明你访问哪些网站干什么用用多长时间，还得几个部门批。
- 想请个假得走一堆 OA 之类的系统。
- 有点意见没处说，给项目经理说，项目经理哦噢喔，给经理的经理说，又见不到，给职能部门说，又不拿事。
- 代码好坏谁知了，权力把人管制了，部门边界把人挡了，目标和人没关系了，谁在意？
- 干什么、干不干都不重要，每天必须在单位待够 10 个小时。

我们会发现，软件项目管理，尽管有各种各样的模型、方法、理论，但实际执行下来，并不是以人为本的。这是最大的问题！

流程把人绑架了，还有什么搞头呢？

人、人的积极投入、人的责任感、人的自由感、人的价值感，是一切软件项目成功的基础。

我都不能决定我干什么、不能用什么工具、用什么语言什么框架，我都不能决定和谁搭档、要个资源都要不到、不高兴都没处说、干多了干好了自己也得不到更多，我凭什么

好好干啊？。

当一个人依附于其他人、事事被安排的时候，他的积极性和创造力是无法发挥出来的。

所以，一切的组织管理与行为活动，都应该是辅助性的、服务性的、不用时隐身用时召之即来（就跟阿拉丁的灯神那样式的）的，以创造一个能够激发程序员的积极性、创造性，能让程序员感到有归属感价值感认同感的环境为目的，只有这样的环境就绪了，生产力才会上来。

包干到户与软件项目管理

再回头来看看“包干到户”多么犀利吧。

包干到户创造了自由的环境，每一户都可以自行安排生产，种什么，怎么种，中间怎么管理，都是农民自己说了算，也因此每个农民都必须为自己的选择承担责任，他做得好、干得多、工具得当、力气使用到位就收获得多，所有收获里扣掉国家税收和集体提留，剩下的就是自己的。这是一种对自己的选择和人生负责的环境，是一种多劳多得、按劳分配的文化。正是这样的环境和文化，推动了小岗村的大发展，燃起了农村改革的燎原之火。

对于软件项目管理来讲，包干到户意味着项目管理人员提出目标和结果测量标准，放心、放手让程序员用自己的方式走向目标。一天干两小时还是二十个小时，根本不重要，重要的是最终交付共同协定的结果，达到公司、团队、个人沟通好的目标。

这样就给了程序员最大的自由，最大的自主权。他能决定用什么技术、怎么设计、怎么编码、何时休息何时投入工作，能决定是否加班。这样的方式，基于信任的前提，把人本身放在首位是对人的尊重，相信每个人都能自我负责，这样就能够激发出程序员的自我实现欲望，让程序员主动投入，专注于解决问题、完成任务，最终能够取得比其他方式更好的效果。

对开发人员来讲，提前完成工作，就可以收获更多自由支配的时间，可以选择做一些提升自己的事情，或者选择承担更多的任务来获取更好的成长和绩效。

而项目管理和职能管理层面需要做的事情就是服务，没资源你给我资源，没设备你给我设备，没工具你给我工具。

所谓制度、管理、流程、环境和文化，都应该是催化剂，是点干柴的火，以点燃参与

项目开发的人员的热情和干劲为目的。

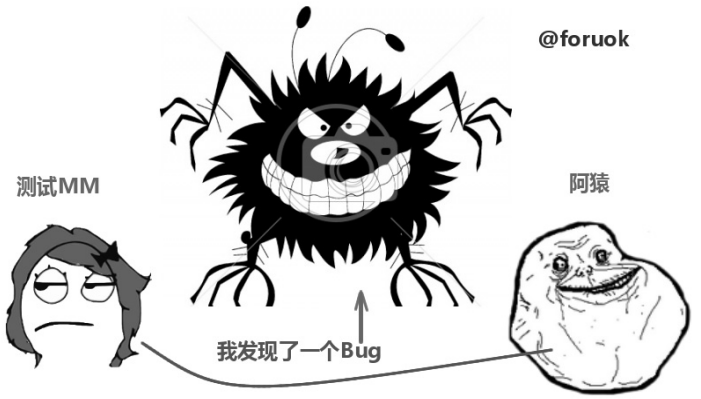
看起来包干到户用到项目管理上就这么简单。然而，这里又充满挑战：

- 必须对产品做恰当的顶层设计。
- 必须对开发任务做合理的拆分。
- 必须识别每个人的长处，给予适合他的工作内容。
- 必须放弃对权力的渴望和通过控制他人验证自我权力之效力的欲望。
- 必须放弃通过控制他人消除自己的不安全感的想法和做法。
- 每个人都必须承担自己的那份责任。

正是因为这些挑战，项目经理和技术经理必须有热切的改变意识和成就欲望并有承担责任的决心和行动。

如果研发团队的管理人员和开发人员愿意为自己的选择和行动负责，那么“包干到户”就是最好的软件项目管理模式，团队就能焕发巨大的活力，效率就能得到爆发式的提升。

为什么开发与测试老掐架呢



让我们思考几个常见的问题：

- 软件测试的目的是什么？

- 开发人员能否构建出没有 Bug 的完美软件？
- 测人人员和开发人员之间是什么关系？
- 软件测试能否保证软件质量？

计算机先驱 Maurice Wikes 回忆起 1949 年他在英国剑桥工作的情形，在拖着打孔纸带上楼给雏形计算机 EDASC 装载程序时，他看到了自己的未来：

我强烈的意识到，生命中剩下的好日子，都将耗费在给自己的程序找错误上头。

Maurice Wikes 告诉我们，没有完美的软件。

温伯格在《颠覆完美软件》中几乎讨论了所有常见的与软件测试相关的概念、问题和指导思想，所以，在这里，我将从以下几方面列一些常见的现象，希望能引起大家的思考。

- 测试和开发的关系。
- 流程与标准。
- 资源。
- 态度。

测试和开发的关系

测试和开发是对立的吗？

从处理 Bug 的角度看，似乎可以这么说。开发人员既生产代码，也生产 Bug。因为开发人员不可避免地会生产 Bug，所以测试人员必须存在，以便在软件交付之前尽可能多地检出 Bug，保证交付给客户的软件质量更好一些。一个产 Bug，一个挑 Bug，看起来似乎是对立的。

在现实中，很多测试团队和开发团队也正是因为这一点而搞得关系不和，甚至真的对立起来。请回想一下你周围发生的与开发和测试相关的事，看看有没有遇到过下面的情景：

- 开发人员说，测试人员净找麻烦，客户根本不可能像他们那样使用软件。
- 测试人员说，问题总是会在看似极端的条件下产生，用户总是会不经意触碰到看似极端的不可能出现的条件。
- 开发人员说，测试人员花在异常情况下的精力比测试主流程还多，不知道轻重缓急。
- 测试人员说，开发人员从来不考虑测试的感受，连测都不测就扔给我们。
- 开发人员说，我都测了，还要测试人员干什么。
- 测试人员说，这么明显的问题你们都不测一下，把我们测试人员当垃圾桶啊。

……

许许多多类似的问题，让开发人员和测试人员的关系从扑朔迷离、相爱相杀走向对立。我见过开发人员和测试人员搞冷战互相遇见侧脸而过，也见过测试经理和开发经理打架，还见过高层领导故意让测试团队和开发团队关系紧张，因为他以为这样可以提高测试效率，也能给开发人员压力，最终会产出更高质量的软件……

实际上，测试人员和开发人员拥有同一个目的：让软件更完美。测试和开发的关系，是一个问题的两面，应该是相辅相成和平共处的。测试人员不是为了挑刺，他提出的问题也不针对生产软件的开发人员，而仅仅是在努力想让开发人员的产出物看起来更好用。只要开发人员不将测试人员提 Bug 这个行为看成针对个人的行为，一切就有了美好的前提。

否定软件，并不是否定开发软件的人。这是开发人员和测试人员都需要明确的一个原则和前提。

还有的人认为开发人员和测试人员之关系类似皮与毛，皮之不存毛将焉附？所以有的开发也会因此而有优越感：没我们写软件，你们测试人员早下岗了。可是，开发人员不写软件，开发人员也下岗了！

感谢开发人员的不完美，让测试人员可以有事可做并练就慧眼。

感谢测试人员的认真细致和耐心体贴，让开发人员可以发现自己的不完美并有机会提升自己——那些说我的软件不好的，都是为了我好。

资源

别动我们测试的服务器，你们自己搭一个！

我们没环境，不用你们的用谁的？

谁把我们的测试手机拿走了？你们申请一个嘛，老来占我们设备。

谁在用我们的账号？招呼都不打！我要用，赶紧退出来！

有时开发人员和测试人员之间也会有资源上的冲突，要努力有创造性地解决（我可以负责任地说，装黑苹果不是好办法），不要让大家的工作卡在环境上，这是管理者要解决的基本问题。我见过很多非常棒的一线经理，在现实制约下，主动把自己的手机、iPad 都贡献出来当作测试设备，这也是解决资源问题的一种办法。

流程与标准

你身边的人员会这么抱怨吗：

- 开发人员根本不看我们的测试用例，评审邮件从来就不回复。
- 我们一报 Bug，开发人员就说用户根本不可能这么用，还说不知道我们怎么会这么测。
- 送测单里根本不写测试范围或者只有寥寥几句跟没写一样。
- 开发人员调整设计从来也不告诉我们。
- 为什么产品经理和 UI 设计师只和开发人员讨论需求变更？
- 为什么发布计划里不给测试人员预留测试时间？
- 为什么开发人员写完代码测都不测就扔给我们？
- 为什么客户那里发现了问题老问是谁测的、为什么没测出来？
- 测试人员老是一声不吭就把 Bug 优先级设置为 Major。
- 测试人员总是把大量时间花在用户根本不可能用到的功能上。
- 测试人员分不清什么是重点，你对他说他还老是一堆道理。
- 测试人员提的 Bug，现象描述也不准确，重现步骤也没有，有的根本就知道是不是误操作。
- 测试人员老来打断我，根本没办法专注开发。
- jira 上的 Bug 重复率太高，一个问题提 *N* 遍，难道就不能合并一下？
- 测试人员发现 Bug，一声招呼都不打就直接告诉老板了，搞得我很被动。
- 测试人员就是专门挑刺的，有劲不往正地儿使，你倒是测测用户常用的功能啊。
- 那么简单的 Bug 都能流出到用户那里，真不知道测试是怎么测的。
- 开发人员老嫌测试报告数据不漂亮，逼着我们调整。

如果你身边的开发人员和测试人员从来没有过类似的问题，那么很好，恭喜你，看来你们的团队人好，协作也很顺畅。

假如你身边充斥着这样嘈杂的抱怨，说明什么呢？是开发、测试、发布这一套流程有问题？还是团队缺乏明确的指向来引导大家向积极、有效的行为靠近？

流程和标准总是有待解释的，再好的规则，歪嘴和尚也能把它念斜……

我们随便挑一个问题吧：“为什么开发人员写完代码测都不测就扔给我们？”这个问题普遍存在，它反映出的是程序员和测试人员的工作边界难以界定的矛盾。

程序员会说，我都测一遍，还要你们测试人员做什么？

测试人员会说，你测都不测，冒烟都过不了，有没有责任心？

程序员说，要我写测试用例，搭各种环境，遍历各种正常、异常逻辑，我还有没有时间写代码？

测试人员会说，我们测试人员是垃圾桶吗，什么都直接扔给我们，我们的时间就那么不值钱？

开发人员会说，测试人员本来就是干这个的，你不测谁测？

……

像这样的问题，是否能制定一个标准，说明什么样的逻辑开发要自测，覆盖什么样的逻辑，可以交给测试来测？能画一条三八线吗？

不能。所以，这个时候，靠谱的一线管理者就显得很重要了。如何创造性地发现适合团队的方法来让大家顺畅地协同工作，比标准、制度更重要，这往往依赖于技术管理者的能力和团队成员的意识。没有普适的方法，只有适合这个组织的、此时此地的策略，加油吧，在战斗中摸索出最适合当下的道路。

那么什么是靠谱的一线管理者呢？

温伯格《成为技术领导者》一书中对领导职责的定义如下：领导的职责就是创造这样一个环境，每个人都能在其中发挥出更多的能力。

如果是一个技术领导带领的团队，大部分人都能专心做与其能力适配的事情，而不用整天泡在与本节前面所列类似的问题里，那么他基本上就算是比较靠谱的人。

至于像给测试人员预留多长的测试周期、调整设计要不要通知测试人员、需求调整要不要测试人员参与等问题，合理的流程和标准可以起到很大的辅助作用，技术领导者只要依据合理的制度，引导大家有效参与，就可以化解。

态度

场景一：

测试 MM 对阿猿（程序员）说发现了一个 Bug。

阿猿矢口否认：不可能，绝对不可能！

MM：真的有 Bug，你过来看一下！

阿猿：不用看，在我这儿好好儿的。

MM：你来看一下嘛……

阿猿：看什么看，肯定是你的环境有问题，动什么东西了吗？重启了吗？

场景二：

测试 MM 想在 jira 上提个 Bug，先在 QQ 上对阿猿说：有个 Bug，你过来看一下？

阿猿：忙着呢，焦头烂额的。

MM：一分钟都用不了，你来看一下吧。

阿猿：思路一打断就不好恢复了，等会儿！

MM：你不看我提到 jira 上了啊。

阿猿：随便，你不就是爱提 Bug 嘛。

场景三：

测试 MM 呼叫阿猿：阿猿阿猿，程序又崩溃了，快来看看！

阿猿慢腾腾地起身过来，鼠标点几下：看不出来什么问题，你是怎么操作的？

MM：这样点一下，那样，这样，……回车……。

阿猿：重现不了啊，你想办法重现，重现了再叫我，我忙着呢。

MM：……

我曾经画过一张暴漫，以“她发现了一个 Bug”为题发布在微信订阅号“程序视界”里，再现类似的场景，感兴趣的可以在订阅号内回复 10019 查看。

开发和测试的日常工作中，上面的情景不断上演，这其中有一部分原因来自态度。我们有时还能听到类似下面的话：

- Bug 里的现象描述根本没用。
- 你根本就没理解这个逻辑，给你说不清楚。
- 测试什么都不懂……
- 你听我的，我让你怎么测你就怎么测。
- 你这种测法，再好的软件都经不起你折腾。
- 用户根本不可能这样用，你们整来整去净瞎耽误工夫。
- 一轮都没测完，你们就给老板说可以按期交付没问题？
- 你们安排计划时根本不考虑测试，三天，三天怎么可能测得完！

……

有时，有一些开发人员会用技术优势藐视测试，认为测试工作技术含量低，内心认为测试是附属没地位，说话就不太客气……测试人员会感觉到，反过来也会对开发人员有意

见……就这样，从相敬如宾开始走向嫌怨丛生……

有个朋友的 QQ 签名档是：没有自我，只有大道。我认为这句话放在软件项目里也挺适用的。

其实，开发人员和测试人员拥有共同的目的：生产高质量软件。具体说，每一个产品、项目、版本都有明确的目标，这些目标是属于开发人员和测试人员的，是大家的。我们把共同的目标牢记在心，摆在首位，还要想着别人所做的一切，都是针对软件本身的，都是在为目标而努力，这样就心平气和多了，就容易从当下的泥沼中超脱出来，求同存异共同前进。

为何你深陷故障驱动式开发

（图片来自网络）

我坐在自己的工位上，盯着电脑荧屏，手抚键盘写代码，耳朵里不断回响着下面这些话：

- “张三，快，服务启动不了了”。
- “李四，客户反馈说保存按钮连续点两次软件就会崩溃”。
- “王五，新版本在 VPN 下连不上服务器了”。
- “赵六，你提交代码后应用开发组的客户端一运行就 Crash”。
- “秦八，客户说他一看别人分享的视频盒子就黑屏”。
- “黄九，1.3.9 版本的共享桌面功能没法用了”。

我觉得我应该带上耳塞式耳机，边听 Katie Melua 边写代码，这样才能屏蔽掉这些嘈杂

的有关故障的申诉和对话。然而我不能，有时我也是被呼唤的那位程序员。

这种被 Bug 和故障抽着被迫火急火燎地旋转的开发过程，我称之为“故障驱动开发”。是的，故障驱动开发。和著名的“测试驱动开发（TDD）”类似，然而却具有明显的无奈和消极。

对于故障驱动开发，我有两个问题：

- 我们是怎样陷入故障驱动开发窘境的？
- 怎么走出故障驱动开发的泥沼？

你喜欢故障驱动开发这种工作模式吗？要是你感到只有这样自己才被需要、才能彰显自己的价值和重要性，那就此打住，别往下看了。

我打算从三个方面来谈谈我们是怎样陷入故障驱动开发的：

- 开发能力失配。
- 绩效导向。
- 有问题再说的思想。

一边谈原因一边给出应对策略，不一定对，抛砖引玉吧。

开发能力失配

很早之前我在另一篇文章“无 Bug 不生活”（收录在《你好哇，程序员》一书中）中说过一句话：“程序员在生产软件，也在生产 Bug”。这是程序员的宿命，再牛的程序员，也注定终生与 Bug 共患难，不死不休。

然而这个残酷的定律并不一定会导致故障驱动开发，导致故障驱动开发的，是另一个残酷的真相：

大部分程序员的能力都配不上他所做的工作。

举个简单的例子吧。假如公司的软件是用 CEF（the Chromium Embedded Framework）+Web 的形式开发的，开发团队里就会有这样的基础分工：

- 搞 JS 的程序员。
- 搞 C++的程序员。

搞 JS 的程序员用 HTML、CSS、JS 等写前端界面。

搞 C++的程序员基于 CEF 做框架开发，还用 C++实现一些核心的业务，比如私有数据传输协议、音视频编解码等。

那么 JS 代码一定会调用底层的 C++代码，C++代码里的有些状态也一定会需要反馈到 JS 中再展示给用户。

那么问题来了，6 个写 JS 的，5 个写 C++的，这其中有几个能融会贯通 CEF、JS、C++的？一个？两个？还是只有半个？

假如有 10 个人能贯通 JS、CEF、C++，那么这个团队的技术能力杠杆的啊！JS 调用 C++出的问题，JS 程序员可以搞定；C++调用 JS 出的问题，C++程序员可以搞定；万一两者各自搞不定，交流一下也搞定了——那种你不会 JS 我不会 C++鸡同鸭讲的事根本就不会发生。

上面的情况有点儿极端和理想化，但我觉得这样的团队，起码有 2 到 3 个人能打通 JS、CEF、C++这三层，才能保障项目的顺利进行。实际情况呢？就一个，而且还是半桶水水准现状呢……

大部分 JS 程序员觉得自己无须了解 CEF 是怎么回事，也没必要知道 C++怎么暴露接口给他，那都是 C++程序员的事。大部分 C++程序员觉得上面有 JS，自己把接口做好导出到 V8 Context 里就好了。所以，到后来，大部分的 Bug 将会聚集在 JS 和 C++交互这一块或间接由这一块引起。

于是，因为我们的能力不够，接下来就会发生很多有趣的事情：

- JS 遇到解决不了的问题就说是 C++的接口不好。
- C++接到问题就会说 JS 根本就不理解接口，调都调错了。
- JS 说 C++代码不稳定，有新版本也不能上。
- C++说 JS 老抱着陈旧的版本不愿意更新，Bug 迟迟无法解决，新特性也没办法应用。
- 技术经理说 JS 和 C++之间的分层不清晰，需要不断开会讨论制定更清晰的接口。
- 高层领导说整个团队的能力不行，得找个大牛项目经理来盯着。
- 大牛项目经理来了，发布的产品还是问题多多，高层领导说得找个大牛测试经理来把关。
- 大牛测试经理来了，发布给用户的软件还是 Bug 多多，高层领导说得找个大牛运维经理来盯着。
- 大牛运维经理来了，被线上系统弄得焦头烂额，说以后测试测过的软件要给我们先内部试用，试用后发布出去还是问题多多……

于是，高层领导、运维经理、项目经理、测试经理、技术经理、质量管理经理统统都过来盯了，两天一小会三天一大会，就不信这么多人盯着你还能把活给干坏了！

于是，壮观的景象出现了：一堆做支持性工作的人员盯着几个能力不匹配的开发人员挖坑。下图是非常形象的说明：

然而这没什么用！程序员照样可以在你眼皮底下搞出 Bug 来，原因很简单——臣妾做不到啊！

【应对策略】

说起来比较简单，找几个大牛程序员，把那些做支持性工作的人都赶走（留一个搞搞服务，需要设备给设备需要安慰给安慰），这样基本就 OK 了。

假如招人很难，那管理者就要注意创造宽松、积极的环境，让我们的程序员们愿意抛开不合理的基于技能的分工，把自己培养成一专多能的“猿中之王”。

3 个能力与需求匹配的程序员的生产率，超过错配的 10 个人。

绩效导向

你知道技术经理、项目经理、部门经理的绩效是怎么评估的吗？你知道程序员的绩效是怎么评估的吗？里面都有什么问题？建议看看我之前在微信订阅号“程序视界”发布的文章——“绩效/加薪/年终奖，虐你如初恋”。

对于技术管理、项目管理类的一线管理者，他们所带的队伍干的活越多，并行的工作越多，发布的版本越多，交付得越快，他们的绩效就越好。

由于这样的绩效导向，很多团队的技术经理、项目经理实际上就容易重视数量和速度，

忽视质量和可维护性。先满足领导的时间要求再说。

所以，技术方案选择，差不多就行了。架构设计，快点定，赶紧开始写代码吧。开发进度，今天 20%，明天 50%，后天就 90%了。当一个程序员忧心忡忡地表示技术方案不合理、架构设计存在缺陷、代码写得太快、又脏、又乱、深海潜雷又多时，得到的答案往往是“来不及了，后面有时间再重构再完善吧”。

这要不出问题，就真奇怪了。

所以，后面你就看吧，拆东墙补西墙，这边贴膏药，那边打补丁，服务不稳定就再写个监控服务管着它，内存泄漏经常把服务器搞死机就定期重启，今天 Hotfix，明天紧急修复。作为程序员，你要不被折腾走那就是有人烧香保佑你了。

【解决方案】

要从管理层就贯彻下面的原则：

在一段时间内，做好做精一件事。

要用数据让管理层明白：

匆匆上马的软件产品的维护成本远远大于（通常数倍于）开发成本，求快反慢，求廉反贵！

调整绩效指标，引导绩效指向：

要把软件发布后的运行情况作为绩效考核的一个重要参考因素。

有问题再说的思想

你有没有过这样的经历：

- 明知道代码有潜在问题，也懒得花时间改。
- 明知道贴块膏药打个补丁只能暂时规避问题，可还是那么做了。
- 明知道张三的代码逻辑上有个漏洞，还是睁一只眼闭一只眼算了。
- 明明有时间把某个模块的代码梳理一下、重构一下，想想没什么外在回报就放弃了。
- 明明看着需求莫名其妙，还是接受了。
- 明明某个界面的 UI 设计反人类，还是从了。
- 写完代码编译通过就扔给测试去测。

这都是很常见的现象，很多程序员都遇到过，可往往都想想算了先这样吧，有问题再说，反正有的是理由：

- 时间紧任务急，考虑不了那么细。
- 别人定的，我管不了那么多，干好我自己的就成了。
- 出力不讨好，说不定还被谁不待见。
- 反正回头还会 Bug 回归，先往前赶进度要紧。
- 对得起老板给我的这么点钱就行了。

一件事你不想做、不想做好，总是找得到理由的。然而，在软件开发这件事上，你总得有一个环节需要认真，而且这个环节越靠前越好，越往后付出的代价越大、效率越差、效果也差。

要么你在需求分析时认真，要么你在设计和编码时认真，要么你在测试时认真，要么你在运维时认证，要么你在处理故障时认真。你总需要在一个地方认真，假如你什么地方都不认真，那就只好认真找工作了……

最后，我想要说的是，对技术要有一颗严谨和敬畏的心，想清楚再干，干好再给别人用。对技术负责，对产品负责，就是对你自己负责。

加薪、绩效、年终奖，虐你如初恋

年底了，经理们忙着做绩效评价，忙着为年底调薪做准备，心里忐忑，反复思量……

亲爱的程序员，你对经理们这么努力、这么费心熬制出来的年终绩效大汤还满意吗？

绩效评估与调薪，有多少不得不说又欲说还休的事？看看我们的 18 个问题，有几个能砸中你。

注：本文将绩效评价结果分为 A、B、C、D 四档。A 为优秀，能出色完成工作，超预期；B 为可以胜任工作，能够及时完成工作；C 为少数时候不能完成工作；D 为多数时候不能完成工作。

有的公司可能不是这样的，用 1、2、3 等级来分，或者用百分制，大同小异，不必深究。

加班多的程序员绩效好

小白白天工作很投入，孜孜不倦码代码，下班时间一到就走人。

小黑九点半到单位，看新闻刷微信，到微博广场遛两圈，QQ 上与人寒暄几句，中午睡上一觉就三点了，玩儿会游戏就四点半了，下了班开始干活，晚上九点半后离开公司。

结果呢，领导觉得小黑老加班，小白工作态度不好，绩效评估时，小黑的结果是 A，小白的是 C。

加班多绩效结果就好，加班多的人调薪机会多、调薪幅度大，是一个很普遍的现象，很多经理都在这么做。然而，这是典型的错误！

绩效评估的目的是客观评价程序员的成就、业绩，应该用结果说话，而不是用在单位停留的时长来衡量。注意，我的说法，是“在单位停留的时长”，不是“工作时长”，因为很多人晚上或周末在单位待着，其实不工作，就是侃侃大山看看电视剧。我曾经一时冲动，总结了一下加班的“作用”，在微信订阅号“程序视界”发布了题为“加班到底有什么用”的文章（订阅号内回复 10068 查看），里面体现了我对加班的看法。

高级开发工程师的绩效总是比初级的好

领导说，高级开发工程师是核心，是骨干。

经理说，高级开发能力强，我们不能伤了他们的心。

高级开发人员说，要是我的绩效比那些初级或者助理还差，当什么高级软件开发工程师……

初级开发人员说，年年得 A 的都是高级开发人员，我们还干个什么劲儿……

这其实是个不大不小的难题，我个人理解，一个人的绩效，应该更少地与职级相关，更多地与具体工作表现和结果相关。假如一个初级开发人员总是能提前完成任务，一个高

级开发人员老是延期，那么初级开发人员的绩效就应该比高级的好。

要是有高级开发人员进来了并且看到上面的话，可能会不满，我要说的是：薪水差异已经体现职级差别了。

Bug 多的程序员反倒绩效好

我觉得：那些老是看起来很轻松的开发人员，很少为 Bug 焦头烂额的开发人员，更应该有好绩效。

为什么这么说呢？开发软件的目的是为了解决生产生活中的问题，给用户带来方便。其中有两点很重要：一是满足需求；二是不给用户找麻烦。你老生产 Bug，老搞 Hot Fix，其实是在给用户惹麻烦，给公司形象抹黑，你这种“忙”，不客气地说，就是自找的，凭什么还要给你好绩效呢？不给差评就不错了亲，委屈啊，自己受着吧……

不过，这种怪现象还会长期存在，一是因为出问题领导上才能注意到你，二是领导上要表彰“工作积极、态度好、肯忘我、愿投入”。绩效评估不好做，态度很重要，也很容易看到。

代码量大的程序员绩效好

王小波有一篇小说，提到王二当程序员的一段生活，说老板看代码量发工资，写 100 行代码就比 1 行拿的钱多，于是他就挖空心思，把 1 行代码解决的问题，用 100 行来解决……

很有意思。

代码量本身没有任何意义！看结果，1 行代码解决了问题，一定比 100 行高明，而且隐藏 Bug 的概率也比 100 行小，后期维护成本更比 100 行小。你想想啊，本来 10 万行代码规模的产品，被变成了 1000 万行，要是有新成员进入这个产品组，那该是多么让人担心啊。

你肯定什么行为，就会强化什么行为。这是绩效评估不得不考虑的问题，那些不该肯定的行为，压根就不应该肯定它们，一旦看见这种小火苗，就要用一盆冷水无情地浇灭。

负责核心功能开发的程序员绩效好

这是应该的吗？我也有点疑惑……

经理让张三负责核心模块，是因为他能力强经验老道。那么张三多半应该职级高。那职级高薪水多半也高。那薪水差异多半已经体现了职级差异……

OK，我的结论不言自明。

三年不涨工资的程序员比刚涨过的绩效要好

大锅饭、平均主义、乱讲人情等。

经理在面对老不涨工资的开发人员时，其实是矛盾的：涨吧，总觉得这人工作表现和结果没到那份上；老不涨吧，又怕他有意见，担心影响队伍稳定性。

如果你是经理，你会怎么办？会不会因为王二麻子三年没涨工资，今年从情面上看，给他一个好的绩效结果，让他也有机会涨涨？

公司效益不好，研发团队绩效能不能好

你所在的公司也可能会受到某些影响，盈利大幅下滑或者由盈转亏。那么作为研发团队，作为程序员，你的绩效能不能有个好结果？该不该有个好结果？

有一个经理，姑且叫他阿郎，阿郎给他部门的人做绩效评价，2 个 A，5 个 B，报给他的老板贾总，贾总说，你们这个产品线今年销售状况很差，你觉得你给的绩效合适吗？大老板看了会是什么结果？

那么，问题来了：贾总的想法对吗？

绩效是对员工过去一个绩效周期内的工作表现、结果的综合评价。绩效评价的结果通常会有下面几种用途：

- 作为发放奖金的依据。
- 作为调薪的参考。
- 作为升职的参考。

所以，我们通常会这么想：绩效评价结果好就应该奖金多、就应该调薪、升职。然而，这还需要进一步解释。

绩效评价结果，其实是分两个阶段、两个层面的：

首先，绩效评价结果是对员工工作的肯定，如果结果是 A，这在精神上是对过往表现的积极肯定。

其次，绩效评价结果可能导致利益再分配，让绩效好的人获得更多的物质回报。然而这种物质回报，是建立在企业盈利的基础上的。

所以，在大环境不好、企业没有营收甚至亏损时，研发团队成员依然可以有好的绩效评价结果，有精神上的肯定。但是，因为企业没有营收，程序员绩效评价为 A 也可能没有奖金可拿，这也是合理的。盈利是任何一个企业的重要目的，企业不盈利，员工就不可能有薪酬之外的更多物质回报。

所以，我觉得贾总的做法是不合适的。他应该从绩效评价结果的两个层面来理解，并据此和阿郎做有效沟通。

产品销售好，开发没事干也拿的钱多

假定一家企业有两个产品研发部门，A 部门研发的产品，市场销售状况很好，产品也已稳定，研发人员基本上没什么事干，相当清闲。可是因为销售业绩好，A 部门的程序员大把大把发奖金。

B 部门产品刚刚上市，软件不太稳定，客户不断有问题反馈过来，新版本还在不断迭代中，程序员们忙得昏天暗地动不动就加班。可是因为还处在导入期，销售业绩没起来。B 部门的程序员虽然很忙，可是却拿不到奖金。

你觉得这种状况是否合理？

反正根据我的经历，B 部门的程序员心中是有诸多怨言的……

然而，我要说，这种结果，虽然未必合情，但却是合理的。因为人家也是从二万五千里长征走过来的，咱不能光见贼吃肉不见贼挨揍。

我们部门的绩效结果不能比别的部门差

前面我们提到阿郎和贾总。假定阿郎是 A 部门，贾总还管了另外一个部门 B，经理是阿虎。我们还假定贾总负责的这两个部门研发出的产品，销售情况都不乐观。贾总还是之前的表现，对阿郎和阿虎上报的绩效评价结果都不满意，示意他们根据销售和营收状况调整。那么，问题来了。

阿郎不知道贾总是否也向阿虎传达了相同的指示，他也不知道阿虎会不会根据贾总的指示来下调程序员们的绩效评价结果。根据阿郎的了解，他更倾向于认为阿虎会忽略贾总的指示不做调整。这样的话，阿郎服从安排做了调整，阿虎作风硬朗不调整，最终，是不

是阿虎部门的程序员，绩效评价结果要相对比阿郎部门的程序员高？问题又来了，阿郎的小伙伴们，是否会因此对阿郎心生怨言？阿郎如何向小伙伴们交待？

这里面牵涉到了企业内部的政治问题。这是经理必须面对的。有些经理认为开发人员忙活了一年，必须在绩效上有所交待；有些经理认为老板的话必须听。

到底哪种好一些呢？阿郎听了贾总的话，那么他在公司内的政治机会是否多一些？阿虎坚持自己的做法，他是否丧失了一些政治机会？这恐怕要看贾总的做事风格……

阿郎是否会失去程序员们的信任，阿虎是否会被人叫好？这恐怕不好说，结果怎样，不但要看阿郎阿虎如何向团队的程序员们解释，也要看程序员们对类似事情的看法……

一种选择是否合理，既受环境影响，又受个人价值观影响，还受到相关干系人的利益影响，这经理，也是蛮难当的。像我这样的人，就受不了这么些情况，所以，我回归简单生活，继续做开发。

绩效管理是彰显权力的工具吗

企业虽然不是官场，但有些经理、领导却利用信息差和资源差来强化自己的权柄，把每年一次的绩效评估作为彰显权力的工具，给张三 A，给李四 D，都没什么依据可讲，全凭一己好恶。我就有“权力”这么做，怎么着吧……

与领导关系近的人绩效好

你有没有遇到这样的情况：与领导关系近的人绩效好，不听话的人绩效就差？

这种情况自古有之。

企业也一样，有一些经理、领导做不到任贤选能，不可避免地走上任人唯亲的老路子。

关于这一点，我其实没什么好说的，你准备做一个什么样的人，全在自己的选择。对于程序员，真实的价值和领导给的绩效无关，具体参看“程序员保值的五个秘密”和“大龄程序员的未来在何方”。

大家绩效都差不多

努力干活与混日子，绩效结果没什么差别，平均主义大锅饭，领导们谁也不想得罪……

然而这样的结果是，工作效率高开发软件质量好的优秀程序员未能获得肯定，滥竽充数的南郭先生也不能被否定，干和不干一个样，那谁还愿意干？长此以往，必然是团队战斗力下降，甚至涣散……

所以，作为一线经理，该奖的一定要奖，不该奖的千万不要因为其他因素而照顾，只有在一个公平、公正的环境里，程序员们才能专注于工作。有态度，有原则，伤几个人不重要，重要的是别伤了积极高效的中坚力量的心。

今年我的绩效是 A，却没加薪

一般来讲有几种可能：

- 公司效益不好，整个公司都没加薪。
- 公司的加薪机制和绩效评估没有对应关系。加薪可能是与职级评定相关，根据职级设计薪资宽带，而绩效评估对应的是奖金（项目奖金、季度奖金、年终奖）。
- 你的薪水已达公司该职位的上限，不可能增加了，只能发奖金、发荣誉、发股份、发分红、发好人卡。
- 你被黑了……

如果你觉得不平衡，也可以找领导问问清楚，或者等领导找你面谈。领导会找你面谈吗？看最后一个问题。

不论怎样，这事的影响应该让它尽快过去，要看前方，你工作是为了自己的成长和成就，钱只是一方面，为这个怄气一年半载磨上几个月洋工把自己耽搁了损失更大。

绩效评价结果一样，张三加薪 5000 元，李四加薪 50 元

张三和李四在同一个部门，今年的绩效评价结果也一样，可是张三涨薪 5000 元，李四只加 50 元。

可能是什么原因呢？姑且来猜一猜：张三在入职时薪水没谈好，本来能拿 20000 元，结果硬被压低到了 15000 元，而他的经理觉得这样不合理，应该给人家合乎市场水平和公司相应职级薪资宽带的水平。李四的薪水已经在薪资宽带范围内了，还是相对较高的……

这是有可能的。

大领导说经理的绩效结果不合理

我一直以为，一线经理对绩效评估和薪资调整最有发言权，然而理想很丰满，现实很骨感，实际情况总是和人的理想有出入。在很多公司，一线经理其实并不实际掌握绩效评估和加薪的权限，这些代表企业中最重要的“管理权力”的东西被远离工作场所的高层们掌控着，所以，程序员一定要理解，有时，你们的绩效评价结果、奖金额度、加薪幅度和绝对值，其实是被你们不认识的那个人决定的……

比如前面提到的阿郎，他给孙二娘的绩效是 A，人家能干，销量好。给扈三娘的绩效是 C，因为她漂亮却不会女红，发挥不出作用。阿郎把结果上报给贾总时，贾总说不合适，给对调了一下。

这种情况很常见啊。有时大领导不说话，一线经理也要揣摩上层的意思。所以，绩效往往不是一线经理说了算的。你以为他们掌握了这些至关重要的资源了吗？他们是做给你看的，让你以为是那样的。

所以，有时你云山雾罩，不知道为什么忽然你的绩效是 A 忽然是 D，也许仅仅是素未谋面的高层领导看到你的名字，觉得读起来铿锵悦耳或柔媚动人，仅此而已。

但这是不合理的。高层就应该把绩效评估、薪资调整等权力切实授权给一线经理，只有他们才真的了解生产环境，知道哪个开发人员出活，哪个开发人员不出活。

你知道你的绩效结果是怎么来的吗

到底什么样的情况绩效应该是 A，什么样的情况绩效应该是 D，很多公司其实没有明确的标准。也有一些公司有标准，但标准就像历史，是个任人打扮的小姑娘，最终解释权归某领导某部门……

然后，我们就面临了黑箱操作：年终了，领导都没找你谈过你的绩效就定了，你也不知道为什么你是 D，明明应该是 B 嘛。你想找领导谈谈，可领导不会给你个列表来说清楚的——这事做得好那事做得差，哪有那么多明明白白的逻辑可讲！

所以，关于绩效评估这个事，一定要有明确的、具有导向性的、公开的标准，让这些整天辛苦酿蜜的程序员们知道提升的方向在哪里，这样干起工作来才有奔头。假如一个程序员处在一个不知道怎么干才好的组织中，那他肯定干不长——他讲逻辑啊，而组织不讲逻辑不讲道理，时间久了谁受得了。

经理会跟你面谈吗

有一年我们公司实行绩效制度改革，管理部、质管部、财务部、几个研发部门的经理们一起讨论新的绩效制度，公司决定给我们研发部门授权，把绩效评估和薪资调整全权交给经理负责。我觉得这样挺好，早该如此。虽然我的工作量会大很多（我负责的部门有将近 20 个人），但我还是觉得这事值得做。

然而作为经理，这里面有几个很有挑战的问题：

- 你准备公开你的评估标准吗？
- 在给出结果之前，你会和你的员工讨论吗？
- 在结果确定之后，奖金、调薪确认之后，你会和你的员工面谈吗？
- 你只和结果是 A、B 的人谈，还是也和结果是 C、D 的人谈？

从管理角度讲，我觉得：

- 绩效评估标准需要公开，要有指导性，让开发知道哪个方向是被鼓励的，哪个方向是应该回避的，这样大家心里才有谱，我知道我怎么做会升职怎么做会加薪做到什么程度会被肯定。
- 经理需要和员工讨论你对他的评估以及他的自评，一定要在结果报批备案之前。
- 调薪、奖金确认之后，要一对一面谈。
- 和绩效好的面谈，绩效差的也面谈。

我是这么想的，可是我没能做到……

当我想到要面对绩效为 C、D 的员工，还要和他们面谈，我就不由自主地想逃避……谈什么呢，你要告诉人家噩耗，还要坦然、真诚、掏心掏肺地给人说虽然今年不好可是明年就有希望明年就给人家 A？都不是小孩子，哄谁呢！他要是拒绝接受闹得不愉快甚至吵起来该怎么办？

有的经理说，结果不好的就不用谈，不谈没事，一谈就有事。

可是要谈呢，你还能让绩效很差的程序员开开心心地投入工作？我是没那种又能透视人心又能舌灿莲花的能力……

那，绩效 A、B 的谈什么呢，告诉他为什么给他 A，告诉他哪里做的好，哪些还可以提高。当然这是极好的。然而他也许会不认同，你说的他的好，可能他觉得不是最好的，可能他觉得你根本没看到他做的最好的；你说的他的不好，可能他觉得你根本不了解实际情况也没能真的理解他……

要知道，每年的这个时候，要评估绩效、要找人面谈，都是我压力最大的时候，表面

上我若无其事，其实内心的煎熬——茶饭不思、食不知味、午夜梦回是常有的事。你可以不管这些，可以谁也不谈，就这么混过去，然而如果你有责任心、心系他人，就不会让自己长久这样下去。

另一面，你还要和你的老板谈，谈什么呢，他会对你有什么期待，你有没有把团队带好？你的目标有没有实现？你还有没有前进的空间？你自己应该在什么方面提升？又或者，你的老板是不是根本就不 care 你，你的感受，你的成长，你的将来，你的痛苦，到底有谁会关注你？如果老板不关注，不支持你，你做自己认为对的事情老板却兜头一棒，你会怎么想，怎么化解这种晕头转脑的愤怒、委屈与悲伤？

行路难，不在水，不在山，只在人心反覆间。假如你不是游戏人心的，那你就不能从中获得乐趣、认同、成就感，那你就会痛苦多于充实，那你就会没有价值感，那你就干不长久。

然而不管怎样，我觉得面谈是非常有必要的，作为经理，一定要和程序员讨论出正确的导向，哪些事做得好哪些事做得不好，实事求是，用结果说话，哪里需要改进，怎么改进等。这样程序员才会感到自己真的被关注了，才可能继续投入情感和热情去干去拼。

至于怎么谈，你可以看看《关键对话》这本书……

说真的，还有希望吗

各种不完美、不合理、不公正可能是现实存在的，是我们所处生活环境的一部分，然而正如莎士比亚所说：“错不在生活，而在我们自己”。生活不会欺骗我们，也不会让我们屈服，是我们自己对生活的解读和反应反过来影响了自己的生活。所以，你选择以什么样的方式来解读我们提到的这些问题，最终将导引你变成什么样子。

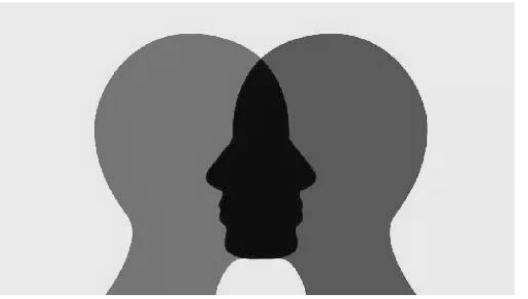
程序员可以选择怎样解读遇到的问题，也可以选择自己喜欢的环境和工作方式，现在可以，将来可以，一直都可以。如果你抱怨现在的环境却迟迟不做出改变，那么也许意味着其实你根本没想明白自己到底想要什么，也没准备好承担自主选择的后果。

你的现在，是你过去选择的结果，你的将来，则由你现在的选择决定。

不能共情你还当什么领导

很多做领导的，以为自己在组织中拥有了权势，能够左右下属的薪水、奖金，能够对

下属进行赏罚，能够名正言顺地指示下属，就觉得自己有了领导力、影响力和管理能力，做起事情来就有些想当然，不考虑别人的感受，结果往往会发现自己一心为公、努力想把事情做好结果却四面树敌、事与愿违。这其中非常重要的一点就是因为管理者换了鞋子走路，不再能够理解一线员工，失去了共情能力。



逼走面临困境的员工

我做项目经理时遇到过一件事，给了我比较大的触动，一直觉得有些愧疚。现在我回归开发岗位了，重新体会一线员工的处境，想起那件事来就更觉得遗憾。

有一个伙计，我曾经给过他比较高的评价，推荐他参与公司的创新奖评选。可是有一段时间，他做什么事都没什么激情，手上的任务几天不见一点进展，问一下，动一下，不问，不动，有问题也不说。

我对他这种状况不满意，找他聊过几次，他消极被动，话不多说一句，看出来也不大配合。我问有没有什么问题，他会说没什么问题。几次下来，我失望了，对他也没有掩饰我的失望。

绩效评估时我给了他很低的分数，按这个分数，会降薪，会被公司劝退。后来我的上级问我怎么回事，为什么曾经看好的人现在表现这么差，我说不上来原因，于是我的领导就找这个伙计聊天，想把他放到另一个项目组中做一些事情。领导后来告诉我，他了解到他面临买房、结婚等问题，而薪水又难以负担，以至于被生活压得喘不过气来，不知道该怎么办，陷入了颓唐窘境无法自拔。而我的做法，无异于踩了他一脚。

最终这个伙计离职了，然而我一点也没有释然的感觉，心里一直是很惭愧的：我为什么没发现这个小伙伴的实际状况呢？我在那个时候的做法真是雪上加霜、让人绝望啊！

以自我为中心是我们的默认设置

美国作家大卫·福斯特·华莱士在 2005 年为肯扬学院做过一次毕业演说，这次演说被评为美国最具影响力的十大毕业典礼演讲，后来有一本书，名字叫作《生命中最简单又最困难的事》，根据他这次演说整理而来，影响深远。

上面提到，自我中心意识是我们自出生起就存在的默认配置。

是的，自我中心意识是我们的默认设置，我们在日常的工作、生活中无不是自动按照这种模式来看待、理解我们周围的人、事、物的。

举几个小例子来看看。

我有个朋友在对我谈起她的一个怀孕的下属时就曾经说，“走一步要好几秒钟，有那么夸张吗？她隔三差五就要请假，说不舒服要休息，都不知道真的假的，我当年怀孕时还天天加班呢！”

还有个朋友嫌弃他的一个员工不加班，说那家伙每天 5 点 28 分关机收拾桌子，5 点 30 分准时打卡下班，根本不管工作进度。暗示了那伙计几次都不管用，气得人不知道怎么办。

我还看见过一些经理最爱当着大家伙的面训斥下属，觉得这样一方面可以彰显自己的权势，一方面又能让当事人长记性。

有没有觉得，这些经理都是以自我为中心在处理事情呢？他们的做法归根结底一定是在这一点上：怎样让下属不影响自己的绩效。

然而这种以自我为中心（缺乏共情）的做法往往是收不到效果的，因为谁都能感觉到你不是真的在意他，而只是担心他的行为影响你的目标才不得不做出关心的样子。

所谓共情

共情是指一种能进入到对方内心世界并体验对方内心感受的能力，然后将对对方感受的理解用自己的言语表达出来，使对方感受到被理解被接纳。共情是从对方的角度而不是从自身的参照体系出发去理解对方的能力。

回到前面我经历的那件事情，当时之所以那么做，有很大一部分原因是缺乏共情能力又自以为有正当理由，没能考虑员工感受，主观上也不愿花费时间去了解其消极行为背后的真实原因。

团队成员有异常表现时，背后一定有你想不到的实际情况导致了他的工作表现。如果

你只想解决问题，而不想去了解问题的缘由，问题往往是解决不掉的，那只是外科医生剪箭尾。

有些问题只有你真的关心了这个人才可能圆满解决，如果你不关心却装作关心，其实是很容易被感觉到的，彼此之间的信任就很难建立，没有信任别人就很难对你敞开心扉。这也是我找那位伙计谈话几次他都不愿意说出实情的原因——我从内心深处并不关心他，我只是从自己的角度出发，把他当作一个必须解决的问题来对待，我表现出的关心、忧虑都是浮于表面的气泡，一戳即破，只会让他感到厌烦而采取疏离我的策略。

作为管理者，要关心员工，要有共情能力，要能够体会下属的处境和心情，才能够更好的管理组织实现目标。让我们再来看看前面举的几个小例子，管理者以共情的方式处理会是什么样子。

如果那位抱怨下属怀孕后做事节奏缓慢又常请假的经理具有共情能力，她可能就会想："也许她胎盘位置偏低，坐一会儿就肚子疼呢？也许医生告诉她必须卧床保胎呢？"然后这位经理可能会根据下属的情况调整其工作任务，让她做一些不那么重要、不在关键路径上的工作。

如果那位嫌弃下属准点下班的经理愿意穿着下属的鞋子走两圈，就可能会想到："也许他要接孩子，也许他家里有卧病在床的老人需要照顾，也许他晚上有约会，也许他正在读在职研究生……"然后这位经理就可能会去了解实际情况，真的关心这位同事，然后采取适合这位同事的方法：给他明确的工作任务，只设定结果而不关心实现路径和工作时间。这样对双方都好。

如果那位爱当众训斥下属的员工把自己放在被训斥的场景下体会一番，也许就能明白这样做对谁都没好处，就能理解让别人出丑就是让自己出丑，就是给自己设置障碍，就会尽可能地当众表扬、私下批评。

激励他人工作的根本

所谓管理者，就是通过他人完成事情的人。所有的管理，都以人际关系为基础，想要脱离人而仅仅管理事务并转动组织取得成就，难比登天。

所以，管理工作中的烦恼，也大都是人际关系的烦恼。当你在管理上遇到麻烦时，就应该回到人际关系上来，看看你是否真的关心别人，看看你是否只是想敦促着别人尽快完成任务，好让自己的目标实现，而不关心他人的目标和情绪，看看你是否拥有共情的能力、

能设身处地站在他人的立场去考虑问题，是否尊重了别人，是否给了别人选择的机会，是否为别人创造了成长的条件，是否拥有和谐的上下级关系……

识别喜欢开发的程序员

识别一个程序员是否喜欢开发，在你遇到下面的情境时特别重要：

- 要招募小伙伴。
- 要选择结对的开发人员。
- 想变得更好、更强大（与优秀的人在一起事半功倍）。
- 研发新产品，你要从公司内部抽调人手组建团队。
- 想判断当前的团队（公司）值不值得继续待下去。
- 公司裁员，你要从几个候选人中挑一个。

我在“如何快速定位自己热爱的工作”“做自己想做的工作”等章节中介绍过一些判断自己是否喜欢一份工作的方法，也谈了一些寻找喜欢的工作的策略。那些文章多数是从“自我分析”的角度来谈的，而这次我们的角度变了，要来判断别人了，得掌握一些新的方法。

自己说喜欢算不算

当事人自己的说法极具参考价值，当你要判断别人是否喜欢他的工作时，听听他自己怎么说很重要。

你可以直接问他“你喜欢自己的工作吗？喜欢哪些方面？”，也可以在和他聊天的过程中收集他随意说出的话。

在说“不喜欢”会带来不良后果时，一个人被问及“是否喜欢”时，可能会言不由衷。比如张三应聘 Android 开发岗位，你问他是否喜欢编程，他多半会说自己喜欢。那么到底是真喜欢还是为了面试通过而迎合你的提问和想法，就较难判断了。所以面试时这么直接地问应聘者这种问题，得到的答案很难说具有多强的参考意义。当然如果你的眼睛很毒，能抓住应聘者听到问题、回答问题时的反应（表情、语气、语调、肢体）并探查到其内心真实想法，那么用这种问题作为试探也是极好的。

相比直接询问，在聊天时通过聆听捕捉一个人对自己工作的态度可能更为可靠。一个人不假思索冲口而出的话，要么是其内心想法的流露，要么是淤积已久的情绪宣泄。无论哪种，都有很强的参考意义。

所以，有时你需要设置一些话题，或者巧妙地询问，让应聘者多说，你多听，多观察。

产出物的质量

对程序员来讲，交付的代码的质量可以从以下几方面来判断：

- 与需求的匹配度，是部分实现了需求、完整实现了需求还是超越了需求。
- 代码本身，比如逻辑是否清晰，比如风格是否良好而一致，比如是否简洁，是否在恰当之处运用了恰当的算法，是否合理运用了设计模式。
- 运行情况，比如测试期 Bug 率，交付给用户后的 Bug 情况。

对于喜欢软件开发的程序员来讲，他会自觉地让自己的代码看起来更美好，因为他觉得这是他的脸面，是其个人价值的体现。而对于另外一些人来讲，好与坏都是无所谓的事，认为能 Run 能交差就行了，想那么多干嘛！

喜欢与不喜欢，一定会导致结果上的差异。如果想让一个团队出成绩，就要找一批喜欢开发、有追求的程序员来。妄图通过完善的管理制度，借助约束和强迫让貌合神离军心涣散的队伍生产出优秀的软件来是不可能的——因为制度只能在一定程度上保证下限，不设限的惊喜与美好，永远来自热爱这份职业的人的自觉追求。

工具选择

“工欲善其事必先利其器。”

对软件开发来讲，很多语言和框架都存在多种 IDE 及周边工具，这些工具里，有一些效率高，另一些效率低，有一些公认很好，有些不那么好。好的工具会提升某些基础工作的效率（如框架代码自动生成、代码补全、代码片自动生成、语法错误提示等），让程序员有更多时间来做那些缓慢的工作（如设计、创新、性能优化等）。喜欢软件开发的朋友会主动尝试新的工具集，追求好的工具集，因为他自发地想提高自己的工作效率。

所以，看一个程序员都了解什么与工作相关的工具及都使用什么软件，也能在一定程度上判断他是否喜欢开发工作。

当他聊起开发时是什么样子

一个人一天要工作 8~10 个小时，工作成了个人生活中非常重要的一部分，工作中发生的事情、产生的思索、累积的情绪不可避免地会延续到工作之外。因此个人也常常会在非工作时间谈论工作内容，而且这时更容易流露出真正的喜好倾向。

聊起技术时，一个程序员眼睛发亮神采飞扬，另一个面露鄙夷冷嘲热讽，任谁都能看出哪个喜欢开发。

看不起自己所做的工作，就是看不起自己。看不起自己所做的工作，又不能跳出那个环境，还鄙视自己、鄙视工作伙伴、鄙视公司，这样的人通常是没什么自信又没什么追求的，在不满现状时往往期待着外部环境改变带来机遇，绝难主动寻求突破。

会不会主动提升自己

假如一个人喜欢自己的工作，就会自发地提升工作技能，主动去追求自我完善。这样的人会合理利用自己的时间，主动安排学习计划，尽量让自己变得出类拔萃。比如暂时没有上级安排的开发任务，他就可能会自己学个新框架，或者写个新框架，或者总结一下过去这段时间的经验。比如下了班，别人打游戏、看电影、侃大山，他就可能会看书、学习、参加培训，构建自己的知识图谱。

是否愿意分享

当你喜欢一件东西时，会愿意分享你的感受。当你喜欢某项技术时，会愿意去“安利”别人。因为物以类聚人以群分，一个人总是乐意找到志同道合的小伙伴，当身边没有时，他就会想用自己的力量去影响、吸引他人，看能不能转化几个过来。当然我的说法简单粗暴，有些人分享时可能不是这么想的，人家可能想的是“我这么牛，不让别人知道一下多亏啊”。

愿意分享自己的技术而非束之高阁敝帚自珍，往往能说明这个人是喜欢他分享的东西的（受命分享或直接利益驱动的分享例外）。

那么怎么发现一个人是否愿意分享呢？很简单，观察他身上是否有下列几种现象：

- 别人经常找他问问题。

- 回答别人问题很热情，别人有技术问题乐意向他请教。
- 技术讨论时常常愿意说出自己对某个问题的看法。
- 写博客分享技术。
- 参与开源项目。
- 在技术沙龙上进行分享。

不是总结

《论语》曰：“不患人之不己知，患不知人也。”

一份职业要做的事情很多，有些是你喜欢的，有些是你不喜欢的。有的喜欢很强烈，会盖过别的不喜欢，让你觉得自己喜欢这个职业。有的喜欢力量很小，会被别的不喜欢淹没，让你觉得无法忍受眼下的工作。

当你喜欢工作时，会有各种各样的表现；当你不喜欢你的工作时，也会有各种各样的表现。这就和你喜不喜欢一个姑娘是一样的。

所以，我们可以通过一个人的外部行为表现来识别他是否喜欢他的工作，具有这么几个方面：

- （自然的）无意识的情感流露。
- 产出物的质量。
- 工具的选择。
- 是否主动学习。
- 乐意分享与传播。

当我们了解一个人是否喜欢他的工作后，就能获取比较准确的人职匹配度，为将来的合作或交往提供参考。

2015 年我组建创业团队时，招募小伙伴，不设笔试，就天马行空地聊，在聊天中观察，根据上面提到的点来判断，后来找到的小伙伴，都是喜欢开发工作本身的，个顶个地能干。

说“这是领导决定的”很扯

假如你是一个执行人员（比如软件开发工程师、测试工程师等），你和你所在的团队，

经常会被要求做这个做那个，你的经理可能会说：“这是领导决定的”“领导想要我们这么做，我们也没什么好办法”“我们就按照领导的要求做吧”。

那你有没有想过：

- 经理为什么会这么说？
- 他应该怎么说？

第一个问题，可以从管理者的影响力上来分析。第二个问题，可以从管理者的责任这个点来分析。

管理者影响力的三个方面

作为团队的直接管理者，他的影响力来自三个方面：

- 行政权力。
- 专业技术能力。
- 个人魅力。

1．行政权力

有相当一部分经理是这么想的：我是领导，你是下属，你得听我的，我有命令你做事的权力。所以，他们有时就会把命令当作领导力；或者，实在说服不了下属，就耍一下无赖，用权力压制下属来做事，或者用某种形式威胁下属做事，比如“如果你能做好，加薪时我会优先考虑你”或者“如果你不做的话，将来的发展可能会受到影响”……

然而这并没有什么用……命令不能产生领导力，也不能让团队产生高绩效，甚至会有损于团队目标、公司目标的达成……

但为什么经理们会想要依赖权力来领导下属？

因为它简单、直接、粗暴，因为经理不知道怎么样才能建立真正的影响力和领导力。当他没什么可以依赖时，就会向权力寻求慰藉。然而，这就像在口渴时喝盐水止渴一样，非但不能解决实际问题，还会越喝越渴……

2．专业技术能力

当你在某方面的专业能力出类拔萃时，你的专业能力能帮助你建立影响力，因为大部分人都信奉权威，都钦佩做事比自己更出色的人。所以，如果你能在某个专业方向上建立

优势，会给你带来影响力。很多公司里也是因为某个人业务能力出色才提拔他成为管理者，这就是“技而优则仕”。所以，你会发现，国内很少见到有超牛的软件开发工程师干了十来年了还在一线写代码，因为做得好的话，五年左右就被拐到管理路线上了。

从技术岗位走上管理岗位的技术管理者，容易对自己的专业技术能力产生依赖，以为只要自己技术过硬，就能有效领导别人。然而，一个人很难把团队需要的技能都淬炼到比别人领先，你总有某个方面不如你团队里的某位成员。所以，单纯靠技术优势来领导，也是有瓶颈的。

这种做法还有一个问题，就是：精于执行，疏于管理。对管理者来讲，这是非常不可取的。要知道，一个管理者最重要的是组织、领导、管理团队成员，让大家为共同的目标而努力。在他人身上花费一分精力产生的效果，相当于亲自下手去干事花费四到五分精力产生的效果。当你能激发团队的积极性时，协力产生的效果，远高于你一个人。

管理者可能会有所担心：假如我自己做不好，团队成员就会看轻我，认为我不足以领导他们。没错，执行者经常这么看待管理者——“啥都不会，凭什么领导我”。

想想西游记里的唐僧，你就明白了。

3．个人魅力

我们经常会信任某个人，愿意响应他提出的号召，愿意跟着他做事，而他并不是经理、总监，他和我们一样，没有高阶职位和行政权力。为什么？

这就是个人魅力。

比如这个人提出看法时，会让你觉得只要是他的看法，就是你想说、想要的，正中你心；他提出行动计划时，你会觉得这就是你心里想的计划；他请求你做事时，你觉得你很乐意帮忙，甚至你会觉得帮他做事义不容辞；当你们的观点产生分歧时，你会觉得虽然意见不一致，但他好像为你考虑了你的所有难处，照顾了你的所有情绪……

我们都愿意和这样的人一起共事，这样的人具有强大的影响力。即便没有领导职位，没有碾压他人的专业技术能力，也一样可以把事做成。

现在，我们可以回答第一个问题了：当一个基层管理者不能从专业技术层面或者个人魅力层面影响团队朝着某个目标前进时，就会求助于权力这种粗暴的机制。然而，这样做非但没什么用，还凸显了管理者的无能和失败……

管理者的责任

管理者的主要职责有两个：

- 完成工作任务。
- 培养下属。

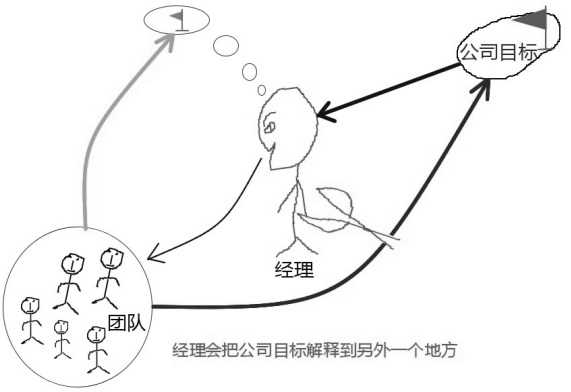
不管用哪个职责，都需要在实现公司目标的前提下进行。从这一点上说，管理者最重要的责任就是：让团队形成协同力，完成公司目标。

而要做到这一点，管理者最需要做的事就是：向团队解释公司目标，在公司目标和团队成员个人目标之间找到平衡点。只有这样，团队才能正确前进。

而现实情况是，中层或一线管理者会根据自己的理解来向团队成员解释公司目标，经过解释后的目标，往往会和公司目标产生偏差（很多经理连 70 分都达不到，如果层级多，偏差会极大），这将导致团队在错误的方向上前进，越努力，距离实现公司目标越远。

意识到这一点，管理者就要时刻提醒自己：不要用自己的双手，剪断执行者和公司目标之间的联系。要有意地、反复地确认自己是否将公司目标拐向了别处。

管理者要争取做到：自己所理解的目标，与公司经营者所理解的目标一致，自己向团队解释、强化的目标，与公司目标没有偏差，团队成员最终理解的目标，与公司目标一致。唯其如此，团队才能沿着前图中的灰色线条努力，最终实现公司目标和愿景。



管理者应该用自己的理解和话语积极阐释公司目标，用更容易被一线执行者接受的方式解释公司目标，这样才可能达到目标传播不失真的状态。如果管理者常常以“这是领导决定的”“领导这么要求了我们就这么做吧”之类的话来要求（压制）下属，那么他实际上

就放弃了自己最重要的责任，表现出了放弃的消极情绪，这种情绪会被执行者放大，产生严重的不良影响，直接或间接导致人力、时间、经济等各方面的极大浪费。所以，老说这样的话，其实是管理者无能的表现……

正确的做法是：管理者要与经营者反复讨论、确认，直到自己与经营者的目标统合之后，再向执行团队解释目标，确保自己的传播不失真，传播之后，还要搜集反馈，确保执行团队对目标的理解没偏差。只有这样，团队才能形成协力，朝着统一的目标前进。

只有这样，管理者的第一个职责“完成工作任务”才可能达成。而要想统合执行团队成员的目标与公司目标，还要求管理者能够关注、了解团队成员，对他们每个人的个人职业目标都心中有数，将其个人发展与公司目标的实现关联起来，这样既能够完成工作任务，又能够在完成工作任务的过程中培养下属。

新任技术领导会遇到哪些问题

《论语·子张》中，子夏曰：“仕而优则学，学而优则仕”。

后半句“学而优则仕”更为人熟知，按我浅薄而世俗的理解，这话的意思是，学问大了就能当官。比如苏东坡、柳宗元、诸遂良，比如孔子、李斯、苏秦……这种情况，在古代实在是数不胜数。

学而优则仕这种传统，在软件开发领域也有体现：很多人会因为技术工作做得好而走上管理岗位。然而，这样走来的技术领导，在刚晋升时往往会面临很多问题，经历痛苦的转换期。这和那些“学而优则仕”的文人才子们的遭遇是一样的，比如范仲淹屡被贬谪，比杜甫总不得志……

现在就来看看，新任技术领导都会遇到哪些问题，怎么破。

以为任命产生领导力

带队伍和当小兵是完全不同的，技术领导需要组织、领导、激励其他人为目标而工作。然而其他人会不会听你的，会不会阳奉阴违，会不会积极主动地干活，当别人与自己意见分歧时怎么办，怎么样让别人接受自己分配的任务，怎么样让别人接受你为其设定的目

标……这些都是问题，需要有影响力和领导力才能顺畅做下去。

然而新任的技术领导刚到经理岗位，对领导力可能还没有体会，很可能不知道怎么做，会错误地用行政权力来强硬地要求（命令）别人做某些事。这是一种误区，任命可以赋予一个技术领导行政权力，但不能产生领导力。来自权力的压力短时间内貌似有效，但实际上会严重损害一个经理的领导力和影响力，假如一个经理频频采用官大一级压死人的策略来推动项目和项目中的人前进，往往最后会适得其反，招致大家的厌烦，在团队中失去威信。

害怕别人不干活

从普通工程师晋升为经理后，开始管理其他程序员，开始管理项目，开始为整个团队或部门的工作进展负责，此时就会产生各种担忧，其中之一就是：要是别人不好好干活或不干活怎么办。

因为刚担任技术领导，对经理的角色还没有适应，不太了解一个项目的人员怎么运转，想当然地以为每个人都应该工作量饱和、工作积极，项目进展才能保证，因此对每个成员是否努力积极工作就会特别在意。同时也可能由己推人，如果自己是特别努力、积极做出了成绩才晋升，可能觉得别人也应该和自己一致；如果自己曾经因为种种原因有怠工的行为，会担心别人找各种借口不好好做事而影响进度……

其实这种担心是很正常的，但从客观上讲没太大必要，你相信大家都会积极完成工作，结果就一定会朝这个方向演进。因为多数团队原本就形成了某种节奏，可以度过领导更替的动荡期，然后继续有效运转。信任是一切的基石。

总想亲自下场

因技术而晋升的技术领导，通常在技术方面有较强的能力，甚至是出类拔萃的。这种技术能力的优势在作为普通员工时可能会给一个人带来显而易见的影响力，然而当这个人成为经理后，有时反倒可能成为他做好领导工作的障碍。因为他可能经常拿自己的技术水平衡量团队的其他人，觉得这个任务张三很难处理好，那个任务李四铁定犯错误，于是不放心把事情交给别人来做，或者交给别人做了又因为看到要出错，忍不住自己伸手去做，把分给团队成员的任务再拿回来自己做。

当一个技术领导因为担心下属会出错或不能按自己预期完成任务而收回这个任务自己

做时，要么会让下属自己觉得自己无能（或者让下属猜测领导认为自己无能），要么会让下属觉得这个领导越俎代庖不干他该干的事，这就会产生严重的不良影响，不利于团队成员自己成长、自己解决问题。同时，这位技术领导也会因为过分关注技术细节而忽略其他的组织、领导工作，导致只见树木不见森林，严重影响整个团队的效率和生产率。

担心丢掉技术，失去竞争力

有些技术领导刚刚开始带团队时，往往还停留在过去的角色里，认为技术是唯一的立身之本，担心放弃了技术细节后，自己会丧失竞争力，会贬值。比如会担心万一自己从这个经理岗位离开，就可能既找不到管理岗位的工作，又因为技术生疏了而找不到技术工作。所以，他们会陷入纠结中，一方面想提升整个团队的工作效率而不得不做很多的组织、激励、领导、协调等工作，花费大量精力；另一方面，这些非技术方面的工作会占用他们的大部分精力，导致无暇深研技术而产生焦虑。

其实，此时更重要的是视野。可能对技术细节了解得少了，但对技术方案选择、技术类别、技术的影响力等可能了解得更多，会形成更为广阔的视野，这足以弥补你在技术深度上的欠缺。而且，其实你之前达到的技术深度仍然存在，甚至会发酵，反过来滋养你的技术视野，因为如果你之前在技术上达到了一定深度，一定在学习上摸索到了适合你的规律，这种学习模式会帮助你更快地了解更多技术，让你从广度上来丰富自己，这即使不能保证让你在技术方面更有竞争力，也会帮助你将技术竞争力维持在某个水平。

最重要的是，除了技术，你在管理岗位上的锻炼，将来一定会带给你更深层次的变化：要么你培育了组织能力、领导能力；要么你认识到自己更适合做什么，对自己的才干和能力边界有更为清晰的认知，而一旦有了这种认知，再做其他事就会得心顺手——因为你会更容易找到自己喜欢做的事情，并带着热忱义无反顾地投入进去。

不理解岗位职责

很多从一线晋升的技术领导，一开始不理解经理这个岗位的职责，不知道具体要做什么、怎么做，公司对该岗位的考核指标、上级领导对这个岗位的期望，这些都是问题。虽然有些公司明确规定项目经理、部门经理等的岗位职责，然而没做过，看那些毫无生气的官方描述也是挺头疼的，看着都是汉字，每个字都认识，但看了就是不知道、不明白什么

意思，和没看差不多。更何况，很多公司其实并没有这个描述，或者根本就是从网上抄来的，是否适用都没人管。

比方说你看到项目经理的职责里写了这么一条：

确保项目目标的实现，领导项目团队准时、优质地完成全部工作。

对你有实质性帮助吗？再比如下面这条：

与客户沟通，了解项目的整体需求。并与客户保持一定的联系，即时反馈阶段性的成果，和即时更改客户提出的合理需求。

对你有实质性帮助吗？

即便你通过公司的文档了解了岗位职责，对工作范畴有了大概的认识，仍然还是会迷惘：

具体我该做哪些？做到什么程度有没有标准？哪些轻？哪些重？哪些是考核的内容？哪些对我的绩效考核影响大？

问题太多了。你知道作为经理要和客户沟通，然而这并没有什么用，并不能将你眼前的铺天盖地的未知揭开，你只有慢慢去试才会知道水有多深，你是从一个工兵的角色忽然就变成了排长，以前的经验几乎没用了，你还没有掌握新的关于项目管理和人员管理的经验，就必须面对那些事情了，这是一个“负位”[①]的过程，你得自己慢慢摸着石头过河去适应。

你需要一个可以伴你成长的同级或高级同事来充任你的 Mentor（导师），帮助你尽快熟悉工作中的各种事情，帮你答疑解惑，必要时为你指点方向。

怕犯错

因为对岗位职责不甚了了，眼前一片茫然，这个时候就会担心犯错，担心一不小心搞错了什么事不受领导待见，又因为对上级不了解而很难明了他是什么行事风格、如何要求下属，自然也担心如果自己的风格和领导不匹配，是否会让领导对自己的错误有过激反应。

还有，也可能会担心领导对自己评价不好——因为你在负位过程中，很多事情做起来没那么得心应手。但你有这种担心的时候，就会愈发想把事情做好，然后，要么迟迟不能决策，要么劲儿用过了把事情搞错，最后反倒真的不好了。

“人非圣贤，孰能无过”。犯错也是一种成长，没有犯错就很难成长，不用怕，错误也

① 注：人们的实际能力，往往低于他所坐的位置，也就是说，有负于他所坐的位置。这种现象被称为负位。

是一种财富。

担心下属议论自己

新晋升的技术领导，往往会因为以前没有做过而特别在意自己是否做好了，既会担心领导对自己的评价，也会担心下属对自己的看法。这个阶段，风吹草动都会让人浮想联翩。心思较多比较敏感的人，还可能会因过于忧虑而导致神经紧张。

其实，大风吹倒梧桐树，自有别人论短长。无论你做什么事情，都不可能符合所有人利益，总是会有人议论的，战战兢兢实无必要，还是信奉这句话吧：走自己的路，让别人说去吧。

不知道怎样培育领导力

别人为什么听你的？你怎么样影响别人使得别人朝着某个目标努力？

这是一个又大又难的问题。对于新任的技术领导而言，有些人会错误地以为任命产生领导力，但多数人慢慢会意识到，领导力和任命没什么直接关系。那么，领导力从何而来？

当你作为一个程序员时，相对他人的技术优势可能让你说话更有力量；当一个团队的各个成员技术水平相当时，技术对领导力的贡献就几乎可以忽略，相互之间的关系会更多地影响一个人的领导力；当一个程序员走上管理岗位后，他的技术能力很可能对领导力没什么特别的贡献，甚至可能会损害他的影响力——假如他事事亲为的话。

温伯格的技术三部曲之一《成为技术领导者》中对这一点有详细的论述，感兴趣的可以参考。我在微信订阅号“程序视界”中推荐过这本书，它是每一个想成为技术领导的技术人员都应该阅读的，它是你成为技术领导者之路上的明灯。它这么定义领导的职责：

领导的职责就是创造这样一个环境，每个人都能在其中发挥出更多的能力。

如果你能理解这一点，对培育你的领导力会有相当的助益。以此为目的，技术领导应该是一个公仆的角色，为团队成员服务，有人需要资源就给协调资源，有人不明白目标就帮助他明确目标并制定其个人目标，不同的模块间接口无法确认就组织相关人员讨论，张三任务完成得好就给予明确肯定，李四对自己所从事的技术方向感到迷惘就协助他找到发展方向，有人忽然情绪低落效率低下就及时发现背后的原因并在必要时提供支持……大家和你同甘共苦完成了一件事，并且都看到并认可你的努力，你就具有领导力了……总之，

你做一切事，创造一个让大家各尽其职各展所长的环境，让这个组织运转正常，让目标得以实现，那么你的领导力就形成了。

不能接受绩效比当普通员工时差

前面我们说从技术岗位晋升到管理岗位后，这个新上来的技术领导往往是负位的，所以，在上任后的那个绩效评估周期内，他所得到的评分如无意外，肯定比他做普通员工时差。

这种结果往往会让这位同志不能接受。你想啊，我当普通一兵时次次得 A，现在当了经理，人都累成马了心都操成渣了，结果却是 C！有情绪很正常，但其实应该换个角度想想，在技术领导岗位上，其实你是从 0 开始的，有一个爬坡曲线也是符合逻辑的。

特定的事情可能会带来挫败感

有时我们也会碰见一些具体的事情，不知道怎么做。

比如公司是结合职级评定和年终绩效决定一个人是否升职加薪，那么张三没有参加职级评定但工作结果很好，从各方面看都应该加薪，此时你怎么操作？传说公司有人操作过非正常流程给员工升职加薪，可是你不知道怎么做，该问谁？怎么说服你的老板支持你这么做？怎样绕过公司的常规流程？

比如你们每个月都要向高层汇报工作，需要写 PPT，而你只有一个管理部门给的 PPT 模板，模板里除了封面和封底只有一页正文，写了几句不痛不痒的话，你要怎么根据这个模板写出你的第一份月度汇报材料？

比如公司规定要定期和下属一对一面谈，你从未有过这方面的经验，根本不知道目标是什么、谈什么、怎么谈、如何应对可能出现的问题，想起来就会担心、想逃避，谁来拯救你？

你会面临各种对你来说是头一遭的事，而多数情况下既没有人告诉你怎么做，也没有机会去演练，你只能凭着感觉战战兢兢如履薄冰地往前走，一不留神没搞好，就可能被老板批评、被同级嗤笑、被下属鄙夷，这样的挫败感可能让你很难接受……

耻于下问

有些新做技术领导的，遇到问题不好意思找人问也怕找领导求助，怕露出自己的无知、

短板，别人瞧不起，显得自己不老练……

其实不必，谁是生而知之的呢？大家都是从不断的学习、不断的实践中培育自己的能力的，很多事情你没经历过就是不知道、没做过就是没体会，求助于别人是再自然不过的，一般别人也乐于帮助你，不会因为你不懂某个技术、某个规则而看不起你，你显露自己在某方面的无知也不说明你这个人能力有问题，人因为不那么高大全而更真实、更有魅力。同时，一个人也只有先正视自己的无知，才能更快地进步。

“士别三日当刮目相待”，无须多虑，尽管前行即可，你明天的成就并不会因为你昨天的懵懂而掉一分成色，相反，你还会获得快速成长的美誉。

不知道怎么应对变化的关系

当一个程序员成为技术领导后，还会面临关系的变化：原来同级的伙伴将成为下属，原来很难见到的高层成了领导。

面对原来同级的小伙伴，是该故作威严拿起官腔，还是依旧嘻嘻哈哈不分彼此？打官腔公事公办会不会让人指指戳戳认为自己小人得志？不分彼此一团和气又会不会妨碍任务的分配和执行，导致最终什么都很难推动？这都是不大不小的问题。但一般来讲不必刻意端着，就事论事公私分明即可。

你的领导往往水平较高，看待问题的角度和切入点与刚晋升为经理的你之间有很大不同。通常的情况是，他看结果不看过程，而你往往还在经理角色的转换和负位过程中，所以经常会收到批评、否定的信息。怎么办？其实这是一个必经的过程，虽然老板邮件里嘴巴上说的都是你的不完美，但很少有老板有闲心专门针对你（很遗憾你没你想的那么重要），一般来讲就事论事接纳即可（不要一看到老板的批评就认为他在否认你这个人）。要相信自己正在变好，随着你熟悉规则，随着角色转换过程的演进，美好的事情很快就会到来。

怎样有效激励一个人积极工作

很多人上班得过且过，拿多少钱干多少事，甚至拿了钱也不干什么事，不但没有积极性，而且牢骚满腹，端起饭碗吃肉，放下筷子骂娘。这到底是为什么？怎样才能有效激励一个人积极投入工作？

传统的经济刺激理论

1976 年，经济学家迈克尔·詹森和威廉·麦克林提出“经济刺激是主要激励因子”，之后这一理论被反复引用，也被管理者广泛应用于企业管理，很快形成了社会共识。它认为经济刺激能有效调节甚至决定人的行为模式，因为人们是“拿多少钱办多少事”的。

过去实际情况也表明，很多员工不断追求加薪，不断提出加薪请求，不断跳槽谋求更高薪水，这似乎也印证了“经济刺激能够激励员工积极投入工作”的激励理论。

然而如果你在公司内留心观察，就会发现：

- 薪水较低的员工，加薪、发奖金后能够持续一到两个月比以往更努力地工作，两个月之后就会恢复原来的样子。
- 薪水较高的员工，加薪、发奖金后只有两周甚至更短的时间（几天）表现出积极性和努力程度上的波动。

为什么？

大棒

“要使驴子前进，就得在它前面放一个胡萝卜或者用一根棒子在后面赶它”。这就是经典的“胡萝卜加大棒”理论。

经济刺激属于胡萝卜，前面说过了，这里单说大棒。

大棒政策源于这样一个假设：只有当人们的生存受到威胁的时候，才会集中精力、激发思维、提高效率。

所以，很多管理者以为，大棒会给员工带来一种恐惧感，员工会在惩罚的威胁下努力工作。

美国哈佛大学克莱默教授的一项研究表明，很多人喜欢给比较凶恶的和比较严厉的管理者做事情。这似乎也证实了大棒政策的有效性。

然而现在社会与以往已经不同了，经济高度发达，物质极大丰富，工作机会到处都是，个体切换工作的成本很低，对大棒已经不敏感了。如果你留心观察就会发现，严厉的惩罚措施只能让员工表面上低头沉默，转身就会采取看不见的报复策略在水面下积极对抗。

胡萝卜不行，大棒也有问题，怎么办？

动因理论

有一个有趣的实验：

把一群孩子分为两组，分开来玩拼图游戏。第一组孩子每拼完一幅拼图，就给他们一美元。另一组就是让他们玩拼图，想怎么玩怎么玩，不给钱。

三个小时后，同时向两个组的孩子宣布活动结束，有趣的现象发生了，第一组的孩子扔下拼图就走了，而第二组的孩子则大部分留下来继续玩。

为什么会发生这样的现象？

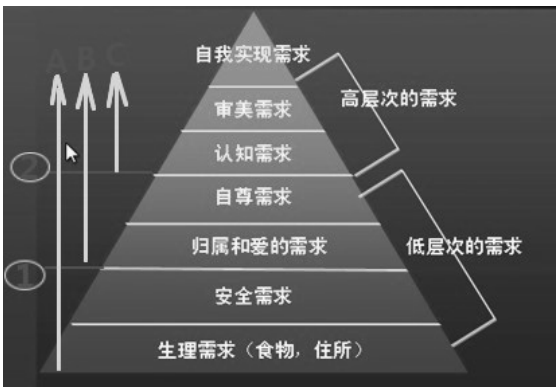
答案很简单：第二组的孩子就是觉得玩拼图这件事本身很有趣，就是想玩。

这个实验隐含了动因理论。

以弗雷德里克·赫茨伯格为代表的动因论者认为：人们需要报酬，但物质激励不是真正的“动因”，人们做某件事的动因是发自内心地想做。

动因理论指出了选择工作的两种不同因素：基础因素和动力因素。基础因素即保障因素，包括地位、薪水、安全保障、工作条件等。保障因素不好，会让人不满，但只有好的保障因素，还不足以让你爱上工作。真正让人们满意并爱上工作的，是动力因素（激励因素），它包括：有挑战性、获得认可、责任感、成就、个人成长。

下图是马斯洛的需求层次理论：



当底层的生理需求（薪水能保障食物、住所）、安全需求得到满足后，人就会产生较高层次的需求，沿着金字塔从底层往上层进化，经历归属与爱、自尊、认知、审美，最终抵达自我实现。

然而这只是常规的理解，很多人忽略了马斯洛需求理论中两个特别重要的点：

假如一个人根本没有高层次的需求，只有低层次的需求，那他就会陷入对底层需求的无穷无尽的追求中。很多人感受不到工作、生活的高层意义，没有价值感，只好一味追求低层次的需求。

假如一个人在高层需求（认知、审美、自我实现等）上得到了满足，他可以忽略掉底层需求。比如焦裕禄就是这方面的典型，为人民服务让他体会到了尊重和自我实现，所以他可以忍受清贫甚至穷困的生活。这一点也可以解释“不为五斗米折腰”“安能摧眉折腰事权贵”等现象。

动因理论里所说的保障因素，和马斯洛需求层次理论中的低层次需求基本上一致；而它所说的激励因素，又恰好可以和马斯洛需求层次理论中的高层需求对应。

结合动因理论和马斯洛需求层次理论，我们可以推断：**超过一定的临界点，改善保障因素带来的激励效用是递减的**。这正好可以解释经济刺激和大棒政策为什么会失效。

工作的隐性价值

薪水是我们找工作时的保障因素，是显性的，那些激励因素（有挑战性、获得认可、责任感、成就、个人成长）则是隐性的。保障因素到达临界点后，就不再能有效激励员工，此时动力因素就显得更为重要，它就决定了员工对工作的投入程度。

1．有挑战性

当一个人的工作无法让他发挥自己的潜力，他觉得这份工作不能发挥他的才能时，就很难对这样的工作产生兴趣，他会觉得这样的工作一点儿也不重要，是对人生的浪费。一旦产生了这样的想法，他就会对工作失去热情，很难投入进去，就会讨厌这份工作，敷衍了事。

十几岁的孩子不喜欢婴幼儿的玩具，员工的工作必须对其本人有挑战性。

2．获得认可

同事和领导对我们的看法会影响我们对自己的看法。如果每个人都认为自己的工作能

够赢得别人的尊重，获得别人的认可，那么我们会变得更加快乐，工作也会更加努力。

3．责任感

当一个人对他要做的工作产生责任感后，就会主动投入时间和精力，努力把工作做到最好。想想看一个父亲或母亲是怎么对待他们的孩子的：不需要任何利益驱使，也不需要任何人敦促，他们会自发自愿地尽自己的最大努力来培养孩子。因为生养孩子是他们自己做出的选择，孩子是他们自己的，他们对孩子负有责任。

工作也一样，当我们能够自由选择工作内容、工作方法、工作目标时，就会对工作产生更强的自主意识，就会觉得自己能掌控工作的局面，产生强烈的责任感，从而更加努力地工作。

有选择，才有责任感。

4．成就

我们想知道自己的努力是有效果的，需要看到自己努力的成果，不断地通过成就获得正向反馈与激励。

当我们能够看到、触碰到、测量、计算出我们的劳动成果时，就会获得更大的满足感，就会在一个成就之后追求另一个成就，形成正向的循环。

5．个人成长

一个人想从事具有挑战性的工作、想获得成就、想获得认可，其实都是对自己个人的成长有追求。每个人都希望自己有机会提高自己的能力，把擅长的事情做好。

假如一个人所做的工作就是挖坑然后填平，不断循环，那么他很快就会厌倦，因为他从这样的工作中无法获得个人的成长。

每个人都希望能够在工作的同时自己也能获得成长，每个人都希望自己变得越来越有价值。

如果一个人能够在工作中体会到上述的隐性价值（哪怕只有一点或两点），那么他就会更加积极地投入工作。

管理者如何创造隐性价值

管理者应该努力为人们提供具有隐性价值的工作，同时也要在工作方法和流程上让员工体会到隐性价值。

具体来讲，可以从以下几个方面进行改善。

1．了解员工为什么在这里工作

员工每周要工作 40 个小时，这 40 个小时是其生活中可支配时间的一大半，当他选择把这 40 个小时投资到眼下的公司、工作中时，他一定期望获得收益。他期望的收益是什么？包含哪些方面？管理者需要搞明白，只有搞明白员工为什么在这里工作（而不是换一家公司），你才可能在工作中改进方法、流程、任务分配模式等来契合员工的需求，只有员工的需求与公司的目标、文化、制度能整合在一起时，双方才能共赢。

2．让每个人都有参与感

管理者可以请大家共同制定团队目标和个人目标。即便是上层领导强制分配下来的目标，管理者也可以和下属一起讨论，让员工参与到上层目标分解到团队目标、团队目标分解到个人目标的过程。

参与感可以产生责任感，责任感会产生动力。

如果人们参与了目标的制定，他们就会努力实现这些目标。

如果一个人参与了一项计划的制定，他就会努力实现这个计划。

如果团队的全体成员共同参与了总结以往表现、制定改革计划的会议，那么他们更有可能接受新的工作方案。

当一个人参与到目标、计划、工作方案的制定过程中时，他就会：

- 觉得自己能够在关键事务上发表意见，获得了认可。
- 感觉自己受到了尊重。
- 觉得这些东西是自己的，产生责任感。
- 因为自己掌控了某些局面并产出了东西（目标、计划、方法等）而产生成就感。

所以，只要让员工参与进来，就可以让他感受到至少 3 种隐性价值，他就会对工作更加投入。

3．改善分配工作的方式

很多管理者日常都在摊派工作，自己分好了工作任务丢给下属。下属是被动接受方，没有参与感，所以积极性不高，不能努力投入工作。

我们在分配工作时，这往往会有下面的习惯：

- 谁做这种事情最擅长就给谁做。
- 谁离我近就给谁做。
- 谁最努力就给谁做。
- 谁不抱怨就给谁做。

第一个习惯具有多重杀伤力：最擅长做某类事情的人会被限制在这类事情上，没有机会从事更具挑战性的工作，很难获得进一步的个人成长；想做这类事情的其他人员没有机会做，也无法获得锻炼和成长。

第二个习惯往往是因为某个人刚好在管理者身旁经过或者工位离管理者近，管理者顺手就会把有些任务交待给他：“张三你帮我们预订一下会议室”“张三你问一下李四他的 PPT 准备好了没”“张三你汇总一下大家的工作进度，下午开会前发邮件给我”……就是这么随意。

至于第三、第四个习惯，此处可以省略 300 字……

总之这些习惯和方法忽略了执行任务的个体，让管理者的任务分配变得不那么有效，甚至低效。

分配工作有三个目标：

- 每个人都有能力完成他所分配的工作任务。
- 每个人的任务都有足够的挑战性。
- 每个人都尽可能地投入工作中。

要达到这些目标，必须让每个人都参与到工作任务的分配过程中。

每个人最了解自己所拥有的能力，每个人最了解哪种工作最能激发自己的工作积极性。如果管理者在**分配任务时能征求每个人的意见**，那么任务就有可能到达最适合做的那个人那里。

为确保需要完成的工作都能得到完成，管理者可以**先把所有任务列出来，然后进行分配**，确保每个任务都有人负责。

上面的两点结合，就会形成“**每个人从任务池中挑选适合自己的、自己想做的任务**”的氛围，而团队的成员之间也会考虑彼此，考虑每个任务都有人负责的原则，不会只挑自己最想做的，还会妥协，接受一些自己不那么想做的。

当然有时管理者需要承受一些压力，比如某项技能还不熟练的员工想挑他目前做起来比较吃力的工作（有其他员工可以熟练地处理这个任务）来锻炼自己，那么这个任务的完成周期就会较长，可能带来交付上的压力。不过这是管理者必须承担的，因为管理者的职责有两块：**完成工作**，**培养下属**。假如一个管理者只愿意驱动下属完成工作，而不愿意给下属成长的机会，那么他就不是一个合理的管理者，最终也会承受这样做的恶果：团队成员的整体业务能力停滞不前、人心涣散、没有协力。

所以，为了改善工作分配模式，管理者必须要有责任感并能承受压力。而这样做绝对是值得的：一旦实践了这样的任务分配过程，每个员工的意见都得到了尊重，每个员工都有了选择的机会，每个员工都有可能从事有挑战性的工作，每个员工都有可能在自己想要的方向上获得成长，那么整个团队的凝聚力和战斗力就会越来越强。

从执行者转向管理者的挑战

中国古代有个习惯，“学而优则仕”。这个习惯延续到职场中，叫“技而优则管”，指因技术（执行）做得好而被提拔为管理者这种现象，除了这种被动成为管理者的情况，实际上很多人做技术到一定年限，自己也希望能转向管理岗位，因为管理者具有更高的经济回报和社会地位，符合主流社会价值观。

但是，从执行者转向管理者，有很多问题会对执行者形成挑战，决定他们能否顺利完成角色转变、能否成为一个称职的管理者。

意识转变

执行者自己做事，管理者通过他人完成事情。

这是执行者和管理者最本质的不同。但很多技术人员（执行者）在初任管理职位时，往往转不过这个弯来。他们会继续紧抓技术，企图用自己的技术优势来奠定自己的领导力，凡事都要亲力亲为亲自执行，反倒是对领导、组织、激励这些与人相关的事情能躲则躲能拖就拖，往往到了不得不做时才被迫去做。

这样的结果往往是团队没带好，领导也不满意，而你觉得自己这么认真的在做事，付

出了这么大的努力，却没好结果，心里感到委屈。

其实这是角色意识转换的问题。作为管理者，要把更多的精力放在人身上，通过他人来完成事情。你在人身上花费一分精力取得的成效，相当于在事情上花费四分精力取得的成效。

共情

领导者不能把人当作完成任务的机器，一定要尊重下属，把下属作为活生生的、独特的个体来看待。作为一个鲜活的个体，他有倏忽来去的情绪，有自己的想法和看法，有自己喜欢与讨厌的东西，有特定的家庭环境，所有这些，会让他和你不一样、和你预期的不一样。他一定不是你期望的那个样子，总会有这方面或者那方面和你的想法不一致，这些不一致，一定会体现到工作上来。

每一个下属都是独一无二的鲜活个体，都有独一无二的个人情况，作为领导者，一定要体谅下属，能够理解下属的处境，这样才能和下属建立相互信任的关系，工作才能在情理通畅的背景下进行。

而要体谅下属，领导就要具备共情的能力。

所谓共情，是指一种能进入到对方内心世界并体验对方内心感受的能力，然后将对对方感受的理解用自己的言语表达出来，使对方感受到被理解、被接纳。共情是从对方的角度而不是从自身的参照体系出发去理解对方的能力。

简单说，共情就是换双鞋子走路，你要时常穿着下属的鞋子走走看，这样你就能体会到他的想法、问题和难处，就能理解他为什么最近工作消极效率低下，为什么三不五时地要请假，为什么早上老迟到，为什么不按照你说的方法去做事……

假如你忽略下属的独特性，不能共情，就只能看到办事不力、态度不好、结果不符合预期、没有责任感等问题，而看不到背后的原因，就不能有效地激励下属改善自己的工作方法，取得更好的生产效率。

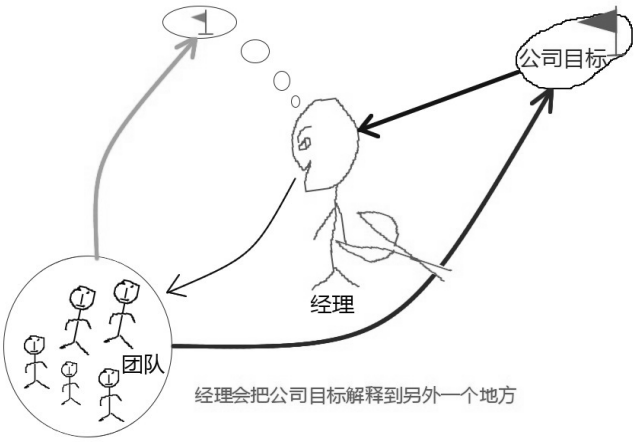
目标整合

前面曾提到管理者必须具备的一项能力：目标整合。

作为公司，一定有自己的愿景和目标；作为团队，也有团队的目标；而每一个团队成员，又有自己的目标与诉求。只有这三者一致时，才能产生协力。

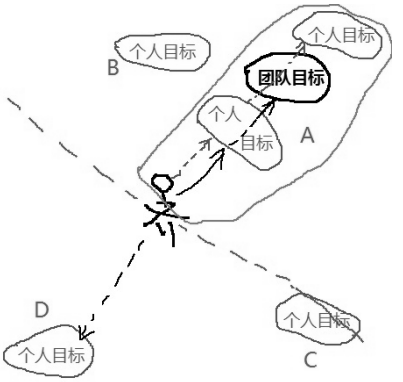
目标在从公司高层传递到一线员工的过程中，各个层级的领导者拥有至关重要的作用。他们必须充当合格的解释官，把上一层的目标合理地转化为自己团队的目标，清晰明确地解释给团队成员。唯其如此，团队成员才能明了团队为什么要这么做、他自己又该怎么做。而现实中，有相当多的管理者把信息视为资源，把信息差视为权力的基础，从主观上阻碍信息的透明流动，公司的愿景与目标，往往在经过他们的过滤和解释之后，就发生了微妙但巨大的偏移。

如果经过的层级较多（三层以上，很多大公司有五层或更多），公司目标到达员工时，往往会从爬衡山变成登恒山，此时团队成员越努力实现他们的目标，距离公司的期望就越远。



所以，管理者首先是一个目标分解者，要具备与他的领导沟通目标的能力，能够做到将上级目标没有误差地转化为自己团队的目标。只有自己准确理解了公司目标，从公司目标合理地分解出团队目标，才可能指导后续的团队执行。

而团队在实现自己的目标时，每个成员的个人目标又会与团队目标产生相互作用。我画了一张图，形象地描述了个人目标与团队目标之间常见的几种关系：



如该图所示，只有个人目标与团队目标一致时，个人才有最大的意愿去实现团队目标。所以，作为领导者，一定要考虑：实现了团队目标，对个人意味着什么。

团队目标对每个个体的意义都不同。所以，领导者要有共情的能力，要能理解下属的痛苦、欢乐、希冀、拒绝，才能针对每个具体的个体来考虑团队目标对于他个人的意义。

所以，领导者的另外一个关键角色就是：目标解释者。他一方面要具备解释公司目标、团队目标的能力，能够将团队目标准确地描述出来，另一方面，他又要能共情，能理解每个团队成员的诉求。只有这样，他才能找到团队目标对于每个个体的意义，才能激励每个个体积极实现目标，才能完成目标统合，最终实现公司目标。

而实际情况是，很多管理者往往不加分辨地接受上级丢过来的目标，然后不加解释地压向下级，公司目标在各个层级间都没有得到合理的整合，最后根本不可能实现。

所以，作为管理者，一定要确保团队目标与公司目标一致、个人目标与团队目标一致，只有这样，才能产生协力，有效实现预期。

反馈

你的领导会因为你努力工作而对你表达谢意吗？你的领导会因为你工作出色表扬你吗？你的领导会周期性地与你沟通，帮助你改善工作方法吗？

恐怕很多人面对这几个问题都给不出肯定的回答。这也是我们面对的现状。

刚从执行者转过来的管理者，在有效反馈这一点上，经常做不好，要么因为“自己在组织中拥有了权势，能够左右下属的薪水、奖金，能够对下属进行赏罚，能够名正言顺地指示下属了”而想端着架子维持领导范儿不愿意轻易给别人好脸色，要么不知道怎么做好

反馈，老表扬怕下属骄上，老批评怕下属不满。

所以，最后的情况往往演变为：只要求下属完成任务，不给予及时反馈，等评估绩效时见。所以，很多得到好绩效拿到奖金的员工其实不知道自己哪里做得好，而很多评级较差的员工则不知道自己哪里有待改善、往哪个方向改善。

每个人都是鲜活的，都有情感上的诉求，他在一家公司工作，绝不仅仅是要充当一部机器干点活拿一份工资，他一定希望得到别人的尊重与肯定，一定希望自己能变得更好。从这一点来讲，管理者应当掌握有效反馈的技能，让员工感受到“温度”和“情感”。

当员工为了工作付出通宵加班的辛劳时，哪怕没有结果，也要适时地表达感谢，抚慰其情绪。当员工出色地完成了工作后，一定要慷慨后表扬，肯定其做得出色的部分。如果员工把事情做砸了，一定要就事论事，不要评判其人品，要和他一起找到改善的方法，让他知道怎样提升自己。

总是，你要能共情，把员工当作活生生的个体来看待，理解其情感诉求，适时正面地反馈，满足其情感诉求，他才能充满激情地投入工作。

要做好反馈，可以从感谢、建议、评估这三个方面入手（参见“横向领导力”一节）：

“感谢”是把你对他人努力工作的感激和赞许之情表达出来。这是一种情感上的表达，目的是满足对方情感上的需要。

“建议”（或“指导”）是指出你认为对方的哪些具体行为应该坚持，哪些应该改变。此时你的关注点是评价工作而不是评价人。

“评估”是根据一组明确或默认的标准及其他人的表现对对方的表现做出评价。

教练式管理

你晋升为管理者后，就拥有了职位附带的一些势力，比如：

- 报酬势力，比如决定下属调薪，奖金发放等。
- 强制势力，比如能够对下属进行赏罚，能够决定下属的职位升迁。
- 正当势力，比如具有指示下属、命令下属的正当权利。

下属则可能因为畏惧领导的这些势力，而在表面上表现出服从的样子，这会给领导带来一种控制感，当这位管理者不能通过目标整合、共情、反馈等方式让团队成员投入工作时，就会祭出权力的大棒，强制下属执行。

然而，命令和权力是危险的，它们既不能产生信服，也无法形成激励，相反，过度依

赖它们还会在团队中形成一股反作用，使得效率和质量下降，阻碍目标的真正实现。

蔡振华命令刘国梁战胜瓦尔德内尔，刘国梁就能战胜吗？

高洪波命令国足夺取世界杯，国足就能夺取吗？

下属和团队能否实现目标，不在于命令和强迫，而在于他实现团队目标对于他而言意味着什么，他自己有没有要实现目标的自发愿望，他有没有自发的行动计划。

当一个人自己找到了目标，导出了行动计划，自发去执行的时候，达成目标的可能性就大大提升了。所以，管理者要做的就是寻找恰当的方式整合团队目标与个人目标，让下属觉得自己做的事情是有价值、有意义、符合他的目标的，让下属觉得他自己也是有价值、被尊重、有选择权的。

为了达到上述的效果，管理者应该尝试放下告知、命令、强迫的管理方式，转向教练式管理。

教练是一种技术，它的目的是激发自信发掘潜力，帮助下属成长。

教练的方式，通过提问来让团队成员自己明确目标，让他自己导出行动计划，让他用自己的方式达成目标。这样他就感到被尊重，就有选择权、自主性、责任感、动力。

《高绩效教练》一书详细讨论了教练技术，并提出了一个简单易行的 GROW 模型：

目标设定（goal），本次教练对话的目标，以及教练的短期目标和长期目标。

现状分析（reality），探索当前的情况。

方案选择（options），可供选择的策略或行动方案。

该做什么（what），何时（when），谁做（who），意愿（will）。

练习 GROW 模型，可以先从提问开始。《高绩效教练》也提供了一组有帮助的问题，可以让我们参考：

“还有什么”，在大多数回答之后使用，会激发更多思考。

“如果你知道答案，它会是什么？”，它让客户越过障碍向前看。

“它对于你或是他人造成的结果/影响是什么？”

“你使用的是什么标准？”

“对你而言，这件事情最难/具挑战的部分是什么？”

“如果你的朋友面临你现在的处境，你会给他什么建议？”

“想象你和你认识或者想象中最智慧的人对话，你认为他会告诉你该怎么做？”

“我不知道下一步该怎么办，如果是你，你会怎么办？”

“如果有人对你说/做了这些……你会有怎样的感受/想法/行动？”

在管理中运用教练技术时，要放下评判，多共情，多反馈，多提开放式问题。

当你采用教练技术后，不但能让下属自己成长，还能让下属高效完成工作，你就能较好地完成管理者的两项任务：完成工作、培养下属。

选择

当你成了管理者之后，会面临更复杂的情况，既要接受公司的目标、领导的管理，还要管理下属实现团队目标，也要管理好自己的目标和工作，有时还要支持相关部门和人员的各种需求。

你会发现，突然多了很多事情，突然要应对很多不同层次的人，你没有自我支配的时间了。

假如你不能区分什么人重要、什么任务重要、什么事情紧急，来者不拒，那么你就会自陷泥沼，无力挣扎，整天一团忙乱，但总是不出成果，甚至还会在关键时刻掉链子。你付出了更多时间和精力，领导对你却不满意，下属对你充满抱怨，同级也指责你配合不力……

这个时候，选择的能力就变得尤其重要。你要围绕着目标进行选择，识别关键要务，遵循“要事优先”的原则。

要识别关键要务，还得回到前面谈过的“目标整合”上来，只有你明确了公司、团队、自己的目标，才能围绕着目标进行选择——那些有助于你实现关键目标的事情，就是你的关键要务。

识别了关键要务后，就要勇敢地选择它们，投入人力、时间、精力，优先保证关键要务的执行。与此同时，你就拥有了拒绝的勇气，能够识别出哪些事情该拒绝，哪些事情可以加入排期，哪些事情可以授权给下属去做。

承担责任与压力

选择需要勇气，还需要担当——你要能够并且愿意承担你的选择带来的后果。比如有的部门会抱怨你支持不及时配合不力，领导可能会因为你拒绝他的一些临时性安排而认为你不够听话。

所以，你要牢记目标，承担选择的后果，承受来自领导、同级、下属、客户等各方关系人施加给你的压力。

假如你只想享受权力和回报而不愿承担责任，那么你就不配做一个管理者——承担责任和压力是管理者的基本要求。

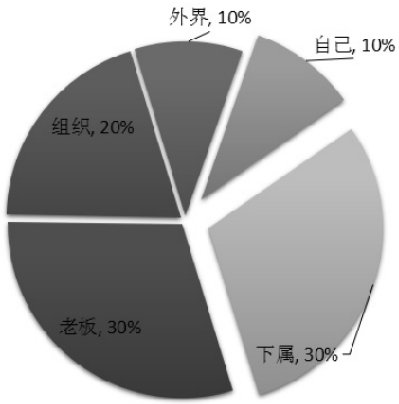
只有你顶住压力，做好团队的隔离墙，团队才能集中精力处理关键要务，才可能更有效地实现目标。

时间管理

管理者的时间会被切分成 5 部分：

- **老板占用的时间**（用来完成老板所要求的工作）。
- **组织占用的时间**（平级部门之间相互协调支持）。
- **外界占用的时间**（客户、供应商、投资者等占用的时间）。
- 下属占用的时间。
- 自己可支配的时间。

用饼图来表示的话是这样的（比例仅供参考）：



老板、组织、外界占用的时间都很难改变，管理者可以改变的，就是下属占用的时间。管理者如果不能有效管理下属占用的时间，很可能就没有自己可支配的时间，不能完成自己要做的事情，完全被他人支配。所以，时间管理的重点就是：**管理下属占用的时间**。

管理下属占用的时间，重点是交办与授权。很多管理者一方面不放心把事情交给下属去做，总担心他们做不好；另一方面又觉得下属总是不成长，什么事都得靠自己亲力亲为。其实你不必等到对他们有信心时才让他们做，你只要大胆地、放心地把事情交给他们去做，

他们就能想办法完成，并且能够在做超越自己当下能力的事情中快速成长起来。

当你把事情委托给下属去做时，就要学会授权。哪些事情他们可以自己做决定（不用来问你），就让他们自己决定，你只需检查结果即可；哪些事情必须经过你的允许才能继续，就说明白，让他们在关键节点来找你讨论，讨论之后的下一步，依然要落实到他们身上去执行。

能做到交办和授权，下属既可以成长，又可以少占用管理者的时间，管理者就有更多的可支配时间，用来完成自己的事情，比如优化团队目标、配置资源、创造更好的环境、改善工作流程、制定相关政策、琢磨每个员工的特点制定相应的培养计划等，而且这些才是管理者应该做的工作内容，也是让未来超越现在的关键。

向上管理

管理者不仅要与下属、同级打交道，更要有效的与老板配合，高效、高质地完成老板要求的工作——如果完不成老板交办的事情，可能立刻就会受到惩罚。所以，从执行者晋升来的管理者，尤其要明白这一点。

通常来讲，你需要做到这些：

- 了解老板的处境，为他排除万难，助他一臂之力，让你的老板工作更有成效。道理很简单，老板升迁你才能升迁，老板被重视，你才可能受重视。所以，你要让老板知道，他所重视的就是你重视的，你会全力协助他达成。
- 把老板当成鲜活的个体，摸透老板偏好的工作习惯，向老板的习惯靠拢。因为你很难为自己量身定做一个老板，你的工作不是去改造老板，而是在认识到老板在性格与偏好方面与你存在差异的前提下，设法找出彼此磨合的工作方式。
- 了解老板的强项和弱点，代替老板完成他不擅长、无法照顾到的或者不愿意做的工作，让老板没有后顾之忧，这样他才可以全心达成他最重要的目标。在老板的弱项上主动承担责任，是最容易让老板知道你的贡献的做法。
- 让老板知道你打算做什么、不打算做什么、正在忙什么、目前处于什么状态。你要主动反馈，这样老板才能掌握你的工作状况，你才不会给老板带来意外，老板才会更信任你，你才可能有机会得到更多的授权。
- 珍惜老板的时间和资源，不要用琐事耗尽老板的时间和精力。你的脑子应该有两张时间表，一张是自己的，一张是老板的。你要清楚老板什么时候有空听你汇报工作，也要清楚哪些工作只要汇报结果，哪些工作需要提供更多的过程信息。该短就短，当长则长，不要让老板觉得你在浪费他的时间。

我真的适合管理职位吗

下面我在微信上和一位朋友的聊天：

ZX：晓辉大哥好，最近很苦恼，干了十年安全相关的开发工作，现在被转到管理岗位了，带着 10 多号人，比较头大，想转回去做开发了。

我：可以啊，开发挺好。我现在也不做管理了，做开发。

肯定是因为你技术做得好，所以被转管理喽。

领导可能以为这样的晋升是对你的激励。你可以找领导谈谈，沟通一下你的倾向。

ZX：其实以前也聊过，说自己不太适合带团队，上面说我没问题。只是我现在干得比较不爽，找不到成就感，活派下去了，我闲着了，不知该干啥了。而且这种带人的事情干得比较别扭，不习惯安排别人做事。

这位朋友很明确他喜欢做技术，喜欢专业带来的成就感，所以他在想着如何逃离安排给他的管理工作。

还有一位朋友在分答上问了我一个问题：

工作 9 年了，现在做 Android 高级开发，目前想转型做管理，感觉发展遇到了瓶颈，请问我是继续做高级开发还是坚持转型管理？

我当时是这么回答他的：

你具体遇到了什么瓶颈？是继续做开发还是转型做管理，只有你自己才能做出选择。我这里提供几个问题，你可以琢磨一下：

- 你工作的成就感来自哪里？是你自己执行完成一件事情，还是安排别人完成了一件事情？
- 想象一下你的最佳工作情景是什么样的？
- 你是否很看重管理职位附加的权力和社会地位？
- 你想一想，假如你老了，你希望别人怎么评价你的一生？

这也是我要对想从技术（执行）岗位转向管理岗位的朋友说的话：确认你的成就感在哪里，确认你在最极端的情况下也不愿意放弃的东西，那就是你的职业锚，它就对应了最适合你的职业。

如果你真的确认自己要做管理，那么前面的那 9 个问题与挑战，对你来讲就是一种淬炼，抱着在世上修行、在工作中磨炼的心，你很快就能做到 80 分。