第 1 章
程序员的职业规划

本章适合所有读者，尤其适合面临 IT 领域的择业者。不论你是一名刚毕业的应届生，还是一名从其他行业转入软件行业有一定经验的人士，都可以读一读。

对于正在纠结“测试还能干多久”的 IT 边缘领域的人员也非常合适。

IT 从业人员的职位介绍

很多同学在入行之前，实际上处于一个懵懵懂懂的状态。看到别人做 Java，自己也去做 Java。看到别人说 PHP 是世界最好的语言，于是自己也投 PHP 职位的简历。这样是不行的。可能浪费几年时间之后才发现这个职位不适合自己，到时候回头就比较晚了。在软件开发的世界当中，目前有下面几种职业新人可以直接上手。

开发人员

程序员是所有职业方向当中对于技术最看重的职业。它的入门门槛不高，但是想做好的话门槛又特别高。

对人最大的要求就是：

1. 脑子要灵活。
2. 英语要好，起码有 CET4 水平。

3. 要有一定的沟通能力。

这个职业可以保证你未来十年左右拥有比较体面的工作，可以坐在上档次的5A级写字楼跟团队谈论一些看起来很高大上的东西，做得好的话还很受人尊重。

这个职业带来的缺点也很明显：回报率看似高，实际低。如果在一线城市工作，拿到的工资虽高，但用于租房、买房等方面的开销是巨大的，甚至很可能工作十年也没有办法在北京、上海这样的城市买到房。

在手艺方面：跟医生律师一样越老越吃香。不同之处在于：中年程序员在中国是要失业的——各个软件公司招聘的基层职位，不会找十年以上工作经验的人做程序员。

程序员可以根据语言分类，也可以根据前端和后端分类。所以，大家先要了解这个职业的分类。

1. 移动前端：用于移动设备或者浏览器上的编程语言，例如Android设备上的Java、iOS设备上的Object C、Swift，以及H5页面上的JavaScript、CSS等。

2. 移动后端：用于服务器端运行的程序，例如C/C++、Java、Python、PHP、Ruby、C#、Erlang、Scala、Node等。

3. 桌面应用：例如C、Java、QT、TCL、VB、Delphi等。

对于现在的形势来说，做互联网相关的开发机会更多，很多人在从事移动前端和移动后端的开发工作。

根据我们之前的统计，在北京2014年到2016年大约每年都会由培训机构提供一万名以上的前端开发工程师。他们的开发语言几乎都是Android或者iOS。对于一些大公司（有自己的产品），往往喜欢招移动后端或者桌面应用方向的人，例如C、QT，这样的工作不太好换。

对于一线城市的程序员，为一个雇主工作的时间不会超过一年半，换工作时最

大的忌讳就是转行（例如从后端转型到运维），或者是更换编程语言（例如从 Java 更换到 C#）。如果一个人之前做前端，他想跳槽去做后端，那么在新公司的工资就会大打折扣，因为他并没有这方面的工作经验。

所以大家在选择自己职业方向的时候，一定要用三五天的时间去了解这门语言，看一下是否适合自己：

- 喜欢 Mac 设备的同学，可以去做 iOS 开发。
- 喜欢安卓产品的同学，或者对手机品牌没有什么要求，但就喜欢摆弄这些小设备的同学，可以去做安卓开发或者后端开发。
- 喜欢在大公司里一直做下去，追求稳定的同学，可以先投递大公司的简历，然后由公司决定，安排什么工作就做什么工作。

如果你什么都不了解，就想找一个很好就业的工作，建议去做移动后端的开发，这是软件行业的万金油。

测试人员

测试分成自动化测试和手动测试，这是门槛最低的职位。可能会有一定的偏见，不过在互联网公司做测试，每天的工作内容就是摆弄手机。

测试人员的日常工作是测试别人写好的程序，保证系统稳定运行，保证能够对某个软件的情况有清晰的了解。这个职位一般在大公司里才会有高级测试人员。在一般的互联网公司，测试就是最普通的人肉测试职位。

我非常不建议大家从事这个职位，因为不长久。这个职位往往干两三年之后，如果不升职成测试经理、不接触更高级的测试工具和方法，很容易就会面临失业。

如果希望在测试行业深入发展，就要多掌握一些自动化测试的技能和工具，早日脱离人肉测试。如果可以掌握一些性能测试、集成测试等工具就更好了。

很多公司都不会有长期的职位来招聘测试人员，往往都是外包职位，特别是外

企，这样的情况很多。

若已经入了这个坑的测试同学，要认识到自己的危机，下班后抓紧时间自学，提高自己的实力，转头向产品经理或者程序员方向发展。

产品经理

产品经理负责把公司的某个想法做成精准实现的软件产品。

这个职位产生于老板的思维： 在一家公司里，老板是最大的产品经理，但是老板往往没有足够的时间去设计某一个产品的各种细节。另外，传统企业的老板往往对互联网完全不了解，这个时候就需要一个人能够：

1. 代替老板来梳理细碎的需求。
2. 懂一些互联网的技术，可以及时给出合理的建议。
3. 跟整个团队沟通，搭建起老板和技术团队之间的桥梁。

产品经理一方面要深刻领悟老板的思路，另一方面又要可以把老板的思路转换成程序员可以理解的原型图，同时跟程序员团队一起工作，把产品开发出来。优秀的产品经理还要管理项目的进度，每日做必要的测试任务。

所以，在国内我看到的现状是：好的产品经理，具备做老板的潜质，因为他在工作的过程当中，既要涉及团队管理，又要涉及公司的战略定制，还要涉及产品线的整体把握，对人的要求很高，能得到非常好的锻炼。

如果你想从事软件行业，很想在互联网的方向做出一些事情，但是对程序不太感兴趣，又不想做一些低端的事情，可以选择这个职位。

这个职位需要跟程序员做大量的沟通，而程序员又是不好沟通的。当某个职业素养不高的程序员理论不过产品经理的时候，往往会用一些技术术语来搪塞，所以在国内的企业中产品经理要懂技术。

很多工作了两三年的人说自己要转型做产品经理，实践中他们也知道这个职位

是干什么的。产品经理是我比较看好的非编程职位，有志于做老板的同学可以考虑。

UI 设计师

每一个公司的产品，有一半的灵魂取决于 UI 设计师。这个职位对于公司非常重要，好的 UI 设计师可遇而不可求，比程序员大牛还难找。

如果你对美术设计很感兴趣，对于 App 界面美感很有自己的想法，又是一名科班出身的艺术院校毕业生，那么欢迎你从事这个职位，做得好的话，你会有很大的发展空间。

这个职位跟程序员一样，需要经过多年的磨练才能出师。一旦出师，就很容易自立门户。

这个职位的职责是根据原型图做出具体的界面设计，可能需要考虑如下问题：

- 这一行的文字用雅黑 15 号好不好？颜色用#eaeaea？
- 这个区域的矩形是不是应该加个边角线？边缘是 3px 的圆角？

工作的时候 UI 可能是最容易感觉到委屈的，可能他设计的作品，公司里的一号老板满意，2 号不满意，3 号满意，4 号又不满意，众说不一，可以说是非常虐心，特别是在有人提出“我想要色彩斑斓的黑色”时。

这个职位的需求数量，大约跟产品经理一样。它比较适合：

1. 艺术院校科班出身的毕业生。
2. UI 设计的狂热爱好者。

运维人员

运维这个职位门槛不高，比较偏门。

在过去的硬件环境下，传统语言的程序员没有太多的精力去掌握某个操作系统或者服务器，这个时候大公司为了维持系统日常工作，需要专门雇一批人来做这个

事情，例如管理系统、上传文件、简单的指令操作、优化服务器的性能等。

这个职位需要具备的技能主要是操作系统、数据库和网络知识，几乎用不到Web开发的知识。

随着操作系统越来越简单，可能只有大公司才会专门配置一个团队，每个人负责一百个服务器。在中小型软件公司是不需要这个职位的。

工作内容就是部署代码，做7×24小时的响应，随时查看服务器的状态。由于要通宵，因此这个职位不适合女孩子。在软件公司，能不做就不要做，工资不高，也没有太好的发展。

我认识的做运维的同学，如果几年之后没有当上部门经理，基本都转型了，但往往转型也有些来不及，不好找工作。

用户体验师（UE/UX）

用户体验师负责设计良好的用户体验，也属于流程改进人员，只有大公司才有这个职位。工作内容是改进用户体验，让某个流程从点三下鼠标变成点两下鼠标，让用户觉得某个产品好用。

让产品经理或者UI设计师来兼职是最好的，不要专门雇一个人来做这个事，浪费资源。

很多时候，产品经理或者UI设计师都可以兼任这个职位。不建议大家考虑这个职位，除非条件很优厚。

技术经理

技术经理职位是笔者大力推荐的。每个上进的程序员做三年以后都应该带团队，独立负担起几个项目，成为公司的顶梁柱。

往往这个职位是为工作三到五年的人准备的，工作职责是：

1. 负责整个项目，保证项目的顺利交付。

2. 负责管理团队，培养新人。

3. 在技术层面可以解决任何问题。

这个职位的待遇比基层程序员有明显的提高。未来的职业发展方向是 CTO 或者技术合伙人。

架构师

架构师是五年以上的老兵才能胜任的职位。工作职责如下：

1. 对某个项目做顶层设计。

2. 确定使用哪些第三方包。

3. 划分前端、后端。

4. 做系统不同部分的耦合和关联。

架构师在十年以前就注定要淘汰了。在十年后的今天，很少有人说自己的职位就是架构师。只负责架构，不写代码，这是完全不合理的。

这个职位在日本外包项目中很流行，往往是由日本的架构师把大体的架构都设计出来，甚至包括某个按钮上的文字都要一丝不苟地画出来，然后把设计图交给国内的软件公司来做。

这就存在一个巨大的矛盾之处：软件项目非常复杂，在系统架构中的粗粒度层面完全看不到细节中潜在的问题。很多问题只有在代码写了一大半之后才会凸现出来，这时能做出正确决策的人，只有可以接触到代码的一线基层员工。

现在，随着编程技术的发展，编程的门槛越来越低，跟架构相关的很多工作都可以交给一线程序员或者技术经理去完成。目前日本的企业也逐渐把架构师的工作交给团队的核心开发者去做，架构设计也不太细了。

如果架构师想跳槽，建议直接去做技术经理或者 CTO。从公司的层面来看，没有必要安排这个职位。

如何选择编程语言

C 和 Java 这样的语言属于编译型语言，它们的特点是：语法复杂，比较底层。C 和 Java 语言都涉及：

1. 指针。
2. 内存回收。
3. 性能的优化问题。

掌握这种传统语言的时间往往在三年以上。我的大学班主任说她做了十年 C 开发，才觉得自己对 C 算是掌握得不错了。

Python、Ruby、Node 这样的脚本语言不需要编译，可以立刻运行。这样的语言语法简洁，掌握的时间更快一些，开发效率很高，特别是 Rails 的 Web 开发效率，是已知 Web 框架中最高的。

对于 iOS 平台，只能用 Object C 或者 Swift，这种情况就无法选择。

对于 Windows 系统的平台，现在一般使用 C#做开发。

做 Web 后端开发建议选择 Ruby

往往代码少的语言好入门，代码复杂的语言不好入门、更不好提高。

Ruby 的语法是已知语言中最简单优雅、对程序员最友好的。其他语言十行代码才能搞定的问题，用 Ruby 语言一行代码就可以了。

Ruby 开发效率非常高，运行效率在绝大部分的 Web 场景也不差。著名的 Javaeye 论坛代表了国内最高的 Java 论坛水平，就是由范凯用 Ruby on Rails 开发出来的。

做 Web 前端（H5）建议使用 Vue.js、React

做 Web 前端，单页应用（Single Page App，SPA）是目前的开发主流。跟传统

应用相比，单页应用的优点是：

1. 开发效率不慢。

2. 门槛不高，传统的 Web 前端开发者可以快速上手 Vue.js、React 框架。

3. 运行速度极快，用户体验极好。

目前 Vue.js 和 React 应用得非常广泛，建议使用。

做移动前端（App）建议使用原生语言和 React Native

移动前端（App 应用）的建议是：

1. 必须掌握一种原生语言（Android 或 iOS）。

无论怎样，原生开发是绕不开的。无论是使用 PhoneGap、Titanium 还是 React Native，进入高级阶段都需要原生组件的开发，这时如果看不懂原生代码，就完全无法进行下一步的工作。对于早期的 PhoneGap 甚至有用户评价：“一天编码，一个月填坑。”

Android 开发使用的是 Java 语言：优点是 Java 入门简单；缺点是过于底层，很多组件封装得不好，门槛较高。

iOS 开发使用 Object C 或者 Swift：优点是 iOS 的开发环境比较友好（比 Android 的开发和调试更加方便一些）；缺点是语言过于底层，门槛也较高。

这两种语言，每种至少需要你经过 2~3 个项目来磨练，才能说得上是基本掌握。

2. 必须掌握 React Native。

（1）在大部分情况下，React Native 可以很好地支持移动端开发，运行速度跟原生语言几乎一样。

（2）React Native 的开发效率极高，一套代码适用于两端。

（3）很多第三方组件都有 React Native 的实现，例如支付、分享、地图等。

（4）目前大部分公司在招聘原生开发者时都会把 React Native 作为必要条件或者加分项。

3. 不要使用 PhoneGap、Titanium 框架，这些技术都不成熟。笔者曾经踩过坑，它们的缺点是：

（1）社区不成熟。缺乏第三方组件的支持，出了问题不知道该找谁。

（2）技术不成熟。PhoneGap 彻底不行了，运行速度极低。Titanium 性能很好，曾经是 React Native 的有力竞争者，但是社区太小，核心功能是闭源的，导致开发进展很慢，甚至很多问题需要使用原生技术才能解决。

（3）国内招不到人，想要新力量只能自行培训。

理想的职业发展路线

程序员的成长路线特点是周期长、见效慢、赚钱相对不多，好处是步调稳健。这个思路适合于：

1. 软件技术发达的一线城市。
2. 学习能力强的同学。

我们用下面这个小李同学作为例子。

第一阶段：新手

小李同学在 2005 年毕业的时候来到北京，开始新手程序员生活，拿的工资不高，属于技术底层。

在工作的第 1 年，学会了基本的网页后端技术，可以完成领导交给的基层任务。

在工作的第 2 年，学会了如何分析需求，可以解决一些中高级难度的问题。

第二阶段：熟手

在工作的第 3 年，小李同学知道了一些软件的高级知识：如何排查性能问题，如何做重构，如何做一些自动化的单元测试、持续集成。

在工作的第 4 年，不但会做后端，还会做一些前端的工作，前端的 JavaScript 框架用得有模有样。

这个阶段最明显的标志是：可以单独扛起一个项目的大旗，老板越来越重视小李。

第三阶段：技术经理

在工作的第 5 年，小李同学负责的项目越来越多，于是开始带团队。这个阶段小李不但自己要承担起一个项目，还肩负着培养新人的任务，日子过得特别辛苦：加班多，活儿都自己干，还要分出时间教小弟。

带的小弟越来越多，小李慢慢也掌握了一套自己的办法，他把办法整理成文档，一套培训新人的教材就出来了。

第四阶段：创业公司 CTO 或大公司技术顶层

如果技术做得好、做得全面，很容易做到这一步。这个时候往往是工作了 7 年以上，要做的事儿很杂。

（1）30%用来开会、做决策。

（2）30%用来管理团队、管理项目。

（3）剩下的时间做一些自己的工作，做不完就要加班。

这种职位很忙碌、很累人，但成长得也非常快。

程序员的基本门槛

做程序员是有门槛的，不是每个人都可以做程序员。在我看来，程序员的门槛

其实特别高。先对自己有一个清晰的认识再入行，会让你的人生少走很多弯路。

英语必须好

对于编程人员来说，可以在入门的时候读一些中文书。但是想进一步发展，会发现到处都需要用到英文。

例如，想用 Java 来读某个文件，如果把文件名称写错了，程序就会报错：

```
Java.io.FileNotFoundException: ……
```

英语不好的同学，可能会卡上半天。英语好的同学就完全没有障碍，看到这个报错可以立刻找到问题所在。

对于编程的初级阶段还好，各种简单的问题都可以通过“百度”这个中文引擎来找到答案；但是一旦进入开发的高级阶段，就会发现到处都是英文，利用百度还搜不出来。

这时必须依靠英文的搜索引擎（Google 等）来搜索，或者直接阅读英文文档。例如，市面上几乎没有公开文档的股票交易组件 TradingView，它的官方文档是纯英文的（如图 1-1 所示）。

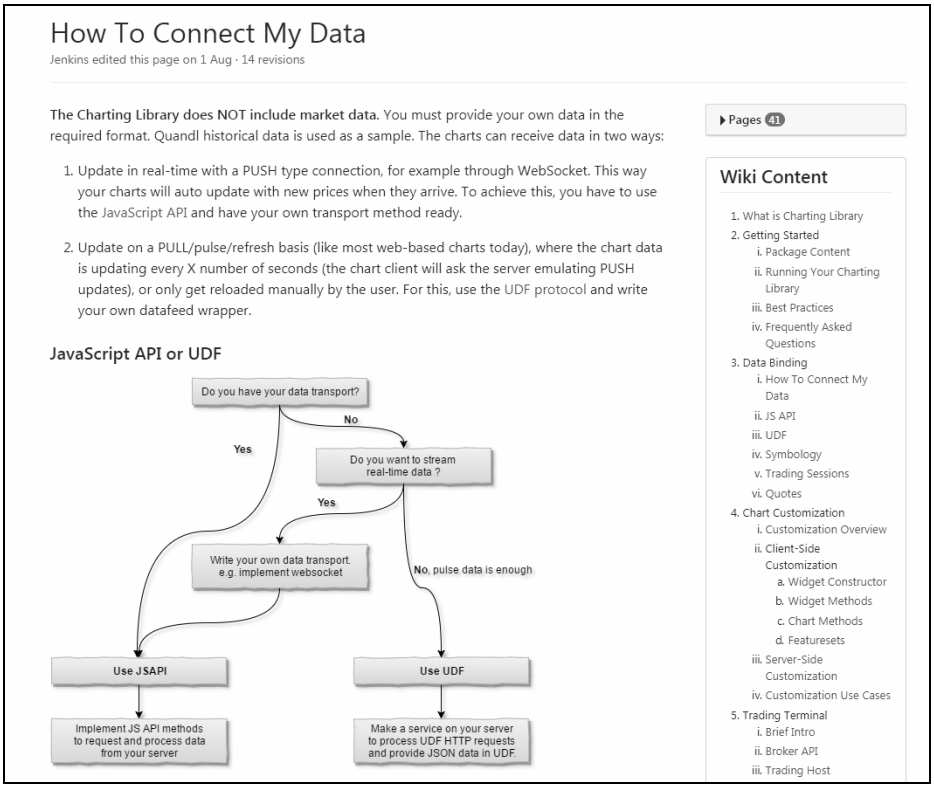


图 1-1　TradingView 的官方文档页

对于一个英语六级水平的人，没有 6 个小时的阅读是无法通读完毕的。对于没有过四级的人，靠翻译完全行不通。

所以，英语决定了一个程序员可以走多远。

思维清晰、反应敏捷

有的同学听课时，思路跟不上老师。有的同学不但跟得上老师，还往往能发现老师出现的错误，提醒老师下一步的思路。

好的程序员，就是后一种人。

思维清晰、反应敏捷的程序员，做事情几乎都是一点就通。给他一个方向，剩下的事他都能自己办完。

思维糊涂的人，让他做事就会让人特别痛苦。一个人做不了事，需求要交代几次才明白。

判断一个人思路是否敏捷清晰，有一个方法很好鉴别：口齿是否清晰，表述能力是否足够好。

如果某个人最多用三句话就可以把一件事说明白，那么这个人就是一个思路清晰的人。如果某个人啰啰唆唆地半天还没说明白一件事，那么这个人就是脑子迷糊。

表达沟通能力强

表达和沟通能力强是非常重要的因素。优秀的程序员可能会用60%的时间来沟通需求，剩下40%的时间用来做事。一个软件项目能否做好，完全取决于参与人员的沟通。

在软件开发过程中，绝大部分出现的问题，我们都要与人商量、跟人沟通。有沟通恐惧症的人是无法胜任软件开发的。“话痨”型程序员特别难得。

十年前我曾经跟ThoughtWorks的一些朋友沟通得比较多，这个公司中的员工都是非常擅于沟通、口才特别棒的人。

对于不擅长沟通的人，往往工作做不好，不受别人的待见。时间一长，这样的人就容易恶性循环，越不擅长沟通越不敢跟人沟通。

下面是一个优秀的程序员与人沟通的例子：

老板：我需要加个新功能，咱们的餐饮软件可以帮顾客点餐！

程序员：啊？“帮顾客点餐”是什么意思呢？是说可以向顾客推荐某个热门菜品吗？

老板：啊，对！

程序员：那么，这个热门菜品是由酒店的工作人员在后台设置的吗？

老板：啊，是的！需要他们设置一下！

程序员：好的。那么，现在后台有两个角色，分别是“收银员”和“领班”。是否他们都可以设置“热门菜品”呢？

老板：不，不，应该只有“领班”才能设置菜品。

程序员：好的，我还想问一下，顾客是在 iPad 上点餐的时候会专门看到一个标签页，叫“热门菜品”吗？

老板：先把推荐菜品做成一个醒目的标志吧，打开 iPad 的点餐 App 就可以看到！

程序员：好的老板，我明白了。

在这个例子中，老板先是提出了一个“一句话”需求，程序员根据这个需求来逐层推进，最终梳理出了自己需要的各种信息。

程序员的进阶门槛

程序员一定要考虑自己的前途和发展方向，不能一直做基层。向上的职位应该是技术经理。这个角色做大了，就是技术总监或者 CTO。

具备领导气质

一个人的能力是有限的。一个拥有十年经验的优秀工程师，在做普通难度的编码方面，也不如两三个普通人。

通常，一个项目中 70%左右的代码都是“普通难度”的代码，所以团队的力量就凸显出来了。你会发现一个 5 人精英团队做的事，比一个独行侠要多得多。

所以，要具备领导气质。因为一旦上级发现这个程序员是核心骨干，就会委以重任。最直接的就是让你做小组长。恭喜你，程序员的晋升之路开始了。把握好这

个机会，努力培养自己的带队能力，你会发现自己的成就更多了。

如何获得领导的赏识呢？不要混日子，努力工作，让自己永远成为队伍的核心。

技术过硬

技术人员的世界观中没有“老资历”，实力是获得尊重的唯一标准。一旦你当上了 Team Leader，就必须具备远超他人的技术实力，例如：

- 对语言的高级特性掌握得清楚。
- 能够及时处理其他人遇到的编程难题。
- Linux 技巧出众，能够轻松化解服务器的压力。

只有这样，才能让你的团队成员服气，团队才能在你的带领之下成长。否则，一旦队伍里其他人发现你的大部分技能实力还不如他们，队伍就不好带了。

IT 从业人员的去路

这个问题没有精准答案，因为几乎无法统计。下面笔者根据自己圈子中的经历和知乎上广大网友的经历总结一下。

随着年龄的增长，IT 从业人员总共约有三种出路：

- 继续干 IT 相关领域。
- 小幅转行。
- 大幅转行。

继续做 IT

1. 做得好的可以继续在企业中打工，成为技术经理、技术总监，比例大约为 1/10。

2. 创业，做技术合伙人、CTO，大约 20 个人中会有一个。

3. 做得普通的就继续划水、做基层员工。这样的基层程序员，年纪不会超过 35 岁，之后就会被裁员。这种人基本就是混日子的，身边的那些油腻大叔，工作起来没有激情，准点下班回家买菜接孩子的人，领导都可能会小他几岁。

小幅转行

虽然人还在 IT 行业，但是做的事情跟之前的经验没有任何关系。

1. 从前端转岗做产品经理、测试或者销售。
2. 做培训讲师。

大幅转行

这种情况算是彻底离开了 IT 行业。

1. 读书，读 MBA 或者考研。
2. 考公务员，全职开淘宝店，或者回老家种地。
3. 移民出国。

不看好的职业：测试、运维、架构师

测试、运维、架构师，这三种职位的共同特点是：一般只存在于大型的软件公司（例如百人以上）。

如果你是一名职场新人，不建议你投上面任何一个职位的简历。如果你是老鸟，可以去做架构师。

测试

测试的基本功是人肉测试。好的测试人员需要熟练运用自动化测试工具。

可惜的是，在大型外企、国企和国内龙头互联网企业中的绝大部分测试人员都是刚毕业的年轻人，没有接触过自动化测试工具，也没有专业的测试素养。

所有的工作都是人肉来做，在项目上线之前通宵加班。他们的工作内容也很简单：用鼠标或者手指点点点。这个事情是不是可以换成初中生来做?

这样的工作是没有技术含量的，而且特别容易造成与程序员的摩擦。曾经有一个朋友，是一个技术好手，测试人员一天给他提了 200 个不是 Bug 的 Bug。沟通无果后这位朋友离职了，原因很简单：工作没法干。

虽然测试人员的工作是找出系统的 Bug，但是从实际效果上看，测试跟开发是冤家，两者和谐共处的不多。

在我看来，一个好的项目经理外加一个懂得测试的程序员完全可以承担传统测试人员的工作。而且由产品经理来把握需求或者 Bug 的优先级是更加合适的。

测试同学可以在大公司里养老，但是一旦离开大公司就肯定找不到工作。创业公司不会有钱雇用测试人员。笔者十年前做测试的朋友基本都转行了，还在做测试的不到十分之一。

评判一个测试人员的实力其实很简单：看看自己会不会 Selenium 、Appium、Load Runner、JMeter 这类工具。如果都不会的话，赶紧转行。

如果会的话也没什么，程序员掌握这些工具比测试同学上手更快、学得更深、用得更好。

运维

运维的工作包括：

1. 管理服务器和域名。
2. 分配各种账号。
3. 部署最新代码。
4. 维护 wiki、防火墙，解决宕机问题。
5. 需要 7 × 24 小时值班。

6. 优化 Nginx、Tomcat、MySQL 等服务器。

其实这些工作中，很多都是对程序员和服务器之间的阻碍，直接导致程序员的工作效率降低和出错时的各种推诿，没有什么技术含量，基本都是体力活。

在 BAT 这样的大公司会比较有用，但是在其他公司（日访问量 100 万以下）基本没有用武之地。

上面这些工作内容，熟悉 Linux 的程序员都会做，而且做部署、优化服务器的工作由程序员做会更合适，因为代码就是程序员写的，一旦发生问题，程序员可以分析日志，第一时间知道问题出在哪里。

加上短信报警等自动化的工具，也不需要 7×24 小时值班。

在工作中，运维人员虽然把 Shell、MySQL、Linux 命令弄的比较熟，好一些的略懂编程，绝大部分都不懂开发，出了问题除了看 CPU、网络、硬盘和进程之外，完全不会从代码层级入手解决问题。

最大的尴尬是运维人员除了 BAT 这样的大公司之外无处可去。职业没有出路、前景一片灰暗。

一个非常常见的情况是，出了问题之后运维人员需要为开发人员“背锅”。运维人员的尴尬如图 1-2 所示。


①
雇你们就是为了管好服务器，结果黄金时间服务器挂了！
老板，对不起！
②
昨天晚上到底是怎么回事？我看CPU，内存，网络都没问题啊！python进程看起来也正常。
是外部的一个Server API挂掉了，导致进程一直处于等待状态，这个看CPU怎么看的出来呢？
③
……

图 1-2　运维人员的尴尬

架构师

不要做只做架构的架构师，因为你无法知道第一线工程师面临的问题。

在十几年前，我们听到最多的是日本的软件公司，它们专门有一个职位叫架构师。收到需求之后，会对需求一个模块一个模块地做分析，然后设计，从框架到伪代码，甚至到某个按钮的名字，都会一一设计出来。

不要做这样的架构师。能把技术的大方向定下来就可以了，千万不要去做干预第一线程序员的事情，因为：

1. 不参与第一线的工作就无法准确判断出开发中面临的问题。没有调查就没有发言权，特别是某个项目会用到新的技术组件的时候，问题会特别明显。

2. 不准确的预判会导致不合理的架构，到最后会给基层员工的工作带来各种障碍。

3. 由第一线的程序员来写代码最合适。

4. 好的 CTO 或者技术经理可以把这些工作做得很好。

近年来貌似日本公司也不会有这样的专职架构师了，都是由基层的程序员老兵来做，而且不会设计得特别细致。

软件培训机构

在北京 2014、2015 年有非官方统计数据表明：每年至少有一万名程序员新人出自培训机构。培训机构一方面培养了不少行业人才，但是另一方面也存在很多让人诟病之处。笔者特意针对培训机构将自己的看法讲给有需要的读者，特别是从其他专业毕业转入希望从事 IT 行业的读者。

确实能改变少部分人的命运

培训机构确实可以帮助一部分学生改变命运，特别是这些人本来比较懵懂，经

过培训机构的正确引导，可以把握住机会，进行某个方向的学习，最后进入并留在大公司工作的学生。

每年每个培训机构都会有学员成为榜样，被下一届师弟敬仰，也都真实存在很多从二、三流学校出来的学生进入到 BAT 等顶级公司工作的案例。

在一定程度上推进了国内技术的发展

培训机构每年的毕业生非常多，大部分毕业生学习的都是 Java Web、Java Android、PHP、iOS 和 Web 前端这些开发语言。哪个语言热门就培训哪个。

对企业主来说，如果本公司用的语言是小语种，就几乎找不到人，只能自己培养。但是一旦开发语言用了上面的任意一个，招聘帖子一挂出去，每天都会收到很多简历。笔者曾经在 2016 年初发过招聘帖，一个月收到 1000 封简历：500 个 iOS、300 多个 Android、100 多个 UI 设计师，90%都是培训机构出来找工作的新人。

所以，培训机构在一定程度上可以抓取市场的需求，另一方面，也影响着软件开发技术的进展。

培训机构之痛

培训机构的名声早期还可以，现在则不太好。HR 在查看候选人简历的时候，往往会直接过滤掉培训机构学生的简历，因为现在的培训机构存在以下问题：

1. 苗子不够好。很多培训机构要跟大学合办，采用 3+1 模式：大一到大三在本校学习，大四来到北上广深大城市。江西有很多高校采取这种模式。其实这个模式很不错，但是要求人才具有学习能力强的特点。好苗子学习能力都强，自学就可以达到很好的效果了，为什么要来培训呢？所以培训机构的学生大部分都是底子差的，看到培训机构说可以解决就业、“上届学生月薪 9K”这样的广告词，对大三的孩子有足够的吸引力。

2. 老师不够专业。行业内的软件高手完全可以找到很高大上的企业，获得很

好的职位。水平一般的程序员也是每日幻想着把自己变成技术大牛，然后进入国内外一流公司。只有不入流的、水平没那么好但是又希望有所改变的程序员才会想着去做培训学校的老师。

3. 甚至有刚毕业的学员会被返聘成为培训机构的助教，代替老师讲课，烂是一定的。不过没关系，培训机构的玩法之一就是：就算学生听不懂也没关系，学生会以为自己笨，不会认为老师讲得不好。

好的老师：把复杂的问题简单化，往往一句话就可以说到点子上。差的老师：把简单的问题复杂化，往往会把问题说得云山雾罩的。刚刚入行的程序员往往更认为第二种老师权威。

4. 简历容易造假。往往培训机构毕业生的简历是一个模子出来的。很多时候我们见到某个年龄应该是应届生的学生，简历上写了 2~3 年的项目经验。面试的时候问起，会回答说“因为我从大二开始就做项目”（正常的工作经验都是从毕业后开始算起的）。实际这些都是套路，也都是培训机构的老师教的，下面列出来常见的套路：

（1）简历有好几页，项目经验 2~3 年。简历很精致，往往会有一张不错的职业肖像照。

（2）简历中任职的公司往往是某某实训基地。

（3）很多同学的项目都是雷同的：某某 Android 小游戏，某某 Android 播放器。

（4）期望工资都不低，很多都是大互联网公司的标准。

5. 就业率越来越低。进大企业的都是少数，笔者知道的很多培训机构都是大比例的同学找不到工作。在北京培训了大半年，毕业季的时候找不到工作，最后打包离开北京，甚至去做传销。

所以，请大家在进入培训机构学习之前，务必想明白利弊。笔者的建议是：

如果抱着试试看的态度，那么千脆不要来。因为你的同学很可能都是差等生，

不会在人生和学习上给你激励。

已经工作了两年以上的人不要考虑。本来培训经历就会给你减分，如果再让HR看到你不是应届生，连电话面试的机会都不会有的，更不用说下一步的现场面试了。

只适合踏踏实实的学员。培训机构不能化腐朽为神奇，高中大学都没改变的差等生在培训机构中一样是不学习的差等生。只有那些可以沉下心来的同学才能有收获，最后找到工作。

第 2 章
程序员的职业成长建议

务必有技术博客

程序员一定要把每天的收获、心得和教训都记录下来，记在自己的个人技术站点或者博客上。收获之大超乎你的想象（记得只写技术，不写私生活）。

表达能力得到极大提高

程序员的表达能力特别重要。

表达能力好的，都是写文章比较多的人。因为他在写自己文章的时候是第一个读者。哪些句子不通顺，哪些语法有错误，马上知道，就可以慢慢改正。

于是这个人的表达能力和概括能力得到提高了。这样的人做程序员没问题，做项目经理也没问题，总之与人沟通非常顺畅。

那种啰啰唆唆半天也说不明白话的程序员，绝大多数都不会写技术博客。

技术可以得到积累

昨天做了一个 MySQL 数据库的优化，前天重装了 MongoDB 等都要记录下来！哪怕只有两百字，哪怕只有寥寥几行。

好记性不如烂笔头，特别是在计算机行业中，会遇到大量不符合人脑记忆逻辑

的内容，例如：

1. 搜索最大的 N 个文件的命令：

```
$ du -a /var | sort -n -r | head -n 10
```

2. Redis 缓存服务器删掉某些 Key：

```
（redis 2.6.0 的办法）
EVAL "return redis.call ( 'del', unpack ( redis.call ( 'keys', ARGV[1]) ) ) " 0 your_keys_pattern*
（redis 3.2.8 的办法）
redis-cli -n 1 keys your_keys* | xargs redis-cli DEL
```

这些内容只靠人脑记忆肯定是记不下来的，一定要记录在博客中。

笔者的个人博客中好多文章都很短，但是只要记录下来，下次拿起来就可以用，五分钟解决问题，再也不需要花时间去回忆了。

个人博客是一张好名片

我们在外面交流时，称呼头衔根本不重要。大家都是李工、张工，大家都是 CTO，大家都是张总、王总。名片也没有任何作用。只有个人博客才是好招牌。我们无论是外出交流，还是求职，个人的技术博客会为你带来数量级的加分，可以带来各种想不到的好机会，让更多的人赏识你。

我的个人博客直接改变了自己平庸的职业生涯，所有的合伙人看到我的技术博客，一眼就确定了。

不要敝帚自珍

有的朋友不喜欢写博客，总喜欢把一些资料放在某个文件夹中，这个效果不好。

1. 不会锻炼自己的写作能力。这个在技术层面特别重要，要写出合格的技术文章，需要具备很好的逻辑能力和抽象能力。

2. 怕被人偷艺吗？在成为大牛之前没有人会关注你的。

3. 把资料放到文件夹中不好查找。很多文件是不方便分级的，而且当文章超过 100 篇之后整个屏幕都会很乱，难以查找。放在文件夹中也基本无法有效搜索。

4. 把资料放在文件夹中远不如在浏览器中搜索 Google 或者 Baidu 来得快。

5. 把资料放在文件夹中无法快速分享给其他人。试想一下，当某个新人问你问题的时候，直接甩给他一个链接，“这个问题我去年记录过了，来这里看吧！”多有高手风范！

要会与人和睦相处，不要任性

程序员是一个典型的学究性格，特别是入行两三年时，会觉得自己不是一个菜鸟了，做项目也有一定经验了，这个时候比较容易滋生这种情绪。

听说过一种说法：

- 上等人：能力大，没脾气。
- 中等人：能力大，脾气大。
- 下等人：没能力，脾气大。

一个聪明人一定是很好的情绪控制者。尽管我们每天的工作压力很大，但是一定要注意：不要傲慢，收起个性，谦逊做人。

控制好自己的脾气

永远不要跟团队的成员乱发脾气，例如觉得产品经理提出不合理的需求，觉得 UI 做出不合理的设计，要理性地提出自己的意见。

程序员：永远不要跟伙伴因为技术问题拌嘴。PHP 永远是世界上最好的语言，Spring 永远是 Java Web 最好的框架，高手永远具有谦逊而温和的性格。

越牛就越谦逊

从业十几年，发现一个很奇怪的现象：

- 技术大牛可以非常坦然、非常容易地说出“这个技术我不懂”。
- 新入行的菜鸟，对于“我不懂”这个词几乎不会说，怕说了之后被人笑话。

沟通能力是立足社会之本

沟通能力很重要

“沟通能力比技术能力重要”这句话在绝大多数情况下成立。这个无论是对于程序员、测试人员还是产品经理都特别重要。沟通能力是智商和情商的代表，沟通能力强的人，智商一定不会差。

另外，软件项目是由多个功能点组成的。技术能力决定了某个功能能否做好；沟通能力直接决定了某个功能要不要做，也就是说沟通能力会更多地影响项目的进展。

在公司里，小王的技术实力一般，但沟通能力很强，一旦某个需求有疑问，就会第一时间找到需求方，进行全方位的询问，然后再行动：

发现这个问题不是很难，马上开始动手，于是很快就做出来了。

发现这个问题很难做，跟对方沟通，发现对方想要的就是很简单的功能，只是在字面上看起来很复杂。于是小王跟对方一起纠正了需求，很快就做出来了。

发现这个问题确实很难做，几乎无法实现。小王及时提出了问题所在，跟对方约定好换另外一种解决方案。

可以看出，无论是哪种情况，小王都可以处理得很好。

小李的技术很强，但是沟通能力不行，很多时候在遇到问题时很少及时跟对方沟通，往往是按照自己的理解去做，结果导致：

- 花了时间。
- 事情用很复杂的方式做出来。
- 需求方还不认可。

几乎每个项目中都有这样的情况。所以，务必把沟通放在第一位！无论你的技术强与弱都要牢牢地记住这一点。

而且甲方永远是喜欢干活儿的乙方（程序员或者产品经理）跟他沟通的，来沟通才说明事情在继续推进。甲方最怕乙方好久没消息，也不来沟通，这基本是乙方没有推进项目的表现。

千万不要性格内向

我们生存在这个社会中，每天都要与别人打交道性格。内向是打交道的大敌，它直接让人处于打交道的下风。我们每天跟人打交道不是靠“脑电波”，而是需要靠语言、行为、表情来把心中的想法传递给对方。比如说下面的情况：

1. 这个需求优先级一般，有空的时候帮我做了吧。
2. 这个需求很着急，一定要加快！周末做不出来项目就会整个失败。

性格外向的人可以很好地表达出上面两个意思，而性格内向的人不一定能表达出来。下面是笔者从业十几年发现的规律：

- 内向的人容易有玻璃心。
- 内向的人谈不了需求。
- 内向的人控制不了进度。
- 内向的人远不如外向的人好管理。

任务没能力完成要勇敢地说出来

做不了的事，千万不要硬扛。硬扛的话往往拖到最后事情没做出来，时间还耽误了。

一般来说，每个老板是能容忍下属某件事做不了的，因为他很清楚每个下属的能力。但是，时间是宝贵的，机遇是要抓住的。

这件事小王做不了，老板可以交给小张、小李，如果他们还做不了，老板可以把它外包出去，交给能做的人来做，但是不要耽误事儿。

老板最怕的是那种接了任务却完成不了的人。

所以，要勇敢地把不能完成的任务说出来，越早越好！老板不会骂你，反而会觉得你很职业。

如果已经晚了，那就现在说！千万不要做那种喊口号最厉害，两个月后项目还做不出来的人。有两次，老板就不敢用这个人了。

小心程序员的膨胀期

每个程序员的心态都会在入行前后发生转变。入行前是乖乖男，入行后就开始世故了。膨胀期一般出现在入行两三年之后，这个时期程序员会认为自己能力上来了。

能干项目了，看不起刚入行的菜鸟。

没见过高手，看不到自己跟高手的差距。几乎没有 Github，几乎没有个人技术博客。

开始喜欢泡论坛，对各种技术指点江山。

这个时期的过渡很关键。笔者见到过一些不错的苗子，本来可以发展得不错，但因为内心膨胀而损失了大好前途：本该有的晋升丢了，要么离职，要么继续混日子。

不要因为被上家公司坑过就对下家公司抱有成见

小王在上家公司干活时兢兢业业，但是后来公司的资金链断了，老板跑路，两个月的工资没发。于是到了下家公司就会提防老板也是这样的人，HR 都是老板的帮凶，每天上班不开心，工作也做不好，稍微有点儿事情就往坏处想。这样很快就会由于工作不努力、态度不认真导致离职。

虽然小王在上一家公司的境遇很无辜，但这跟下家公司没有任何关系。下家公司能给小王这个工作机会，是对小王能力的一种肯定。所以，小王不应该单纯地认为所有的软件公司都是垃圾公司，他不能把对上家公司的情绪带到下一家。

有些程序员能力还可以，就是由于这种狭隘的思想而导致自己错失了很多机会。

另外，互联网公司（特别是北上广深一线技术城市）的底线往往是很高的。公司规模越大、越有规范的 HR 部门，人文关怀和员工关怀越好。

不要论战

论战都是发生在论坛上，往往容易在不同语言之间产生争论。这个说 Java 好，那个说 PHP 好。同一语言的框架之间也容易有争论，这个说 Struts 好，那个说 iBatis 好。其实论战是没有意义的，不同语言之间的论战、同一语言（JavaScript）中不同框架的论战，如图 2-1、图 2-2 所示。

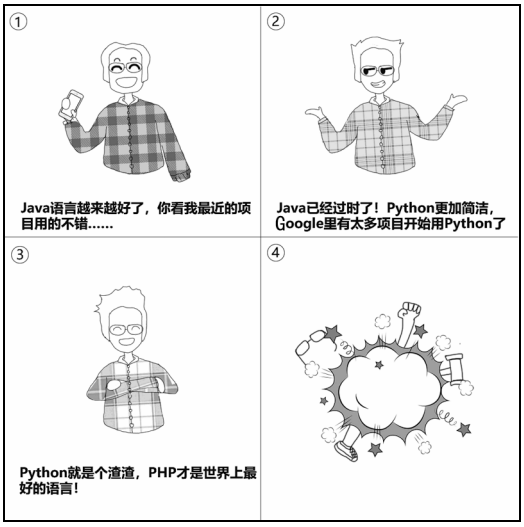


图 2-1　不同语言之间的论战

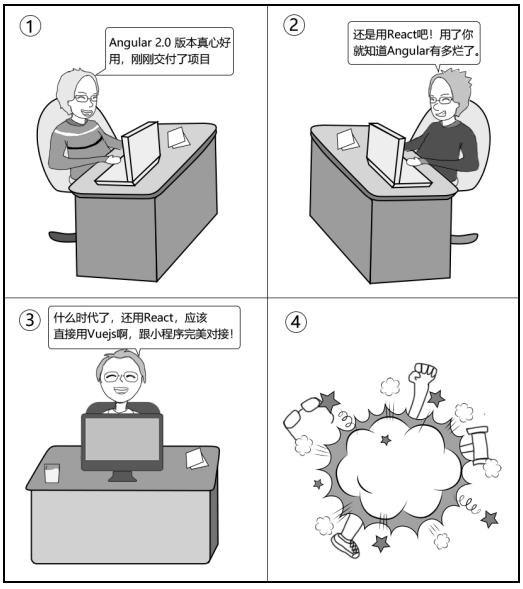


图 2-2　同一语言（JavaScript）中不同框架的论战

我们一定不要参与到这种无意义的论战中去，因为：

- 浪费时间。你永远无法战胜对方，对方也永远无法战胜你，大家都是任性的键盘侠。
- 会使你的脾气变坏。因为越是发火，脾气就越差；脾气越差，人就越容易发火。最终导致恶性循环，无论在职场还是在生活中都会产生影响。

论战特别容易出现在刚入行两三年的人身上。这些人一般是团队的中坚力量，见过一些世面，有过一些经历，特别容易自我膨胀。其实让他们再继续多见一些高手就好了。

使用传统编程语言的人特别容易心态不好

在笔者接触到的朋友中，使用传统编程语言做开发的人中有一半以上是心态不太好的，这是职业病（字面意义上的病痛，入行的朋友务必小心）。

高发诱因 1：过于底层的语言

传统语言编程包括 Java、Object C、PHP 等，这些语言的共同特点是：过于底层。使用 JavaScript、Ruby、Python 做开发时从来不用关心变量的类型，不必考虑这个变量应该是 int 还是 double int，完全不用考虑计算机的感受，可以把全部精力都放在业务逻辑上，解决实际问题就好了。传统语言则是写任意一行代码都要考虑编译器的感受。编译器就是一个爱哭的孩子，总是把你的注意力从业务逻辑吸引到对编程语言的照顾上。

下面是一段读文件的对比：

Java 代码	Ruby 代码

File file = new
File("target_file.txt");
InputStream in = null;
try {
in = new
FileInputStream(file);
int tempbyte;
while ((tempbyte =
in.read()) != -1) {
 	
System.out.write(tempbyte);
}
in.close();
} catch (IOException e) {
e.printStackTrace();
return;
}
	File.read("target_file.txt")

可以看出，左侧的 Java（传统语言的代表）编程是多么的麻烦，右侧的 Ruby（新兴语言的代表）是多么的简洁优雅。哪怕读者不懂编程只懂英语，都可以很好地了解这段代码的作用。

高发诱因 2：开发人群的职业年龄是 2~4 年

因为这个人群的特点是：

1. 基本还处于编码的阶段，每天以敲代码为生。

2. 虽然对一门语言基本精通了，但发现要做任何事情都很麻烦、很复杂。因为传统语言特别麻烦复杂，做项目容易有挫败感。

3. 这个年龄的人往往混迹于各种论坛，有一定的办公室习气，容易脾气不好。

不要踢皮球

踢皮球（如图 2-3 所示）是一个俗称，特指把本应自己做的任务转手让其他同事做。这种行为看似让自己轻松，实则让自己损失很多。

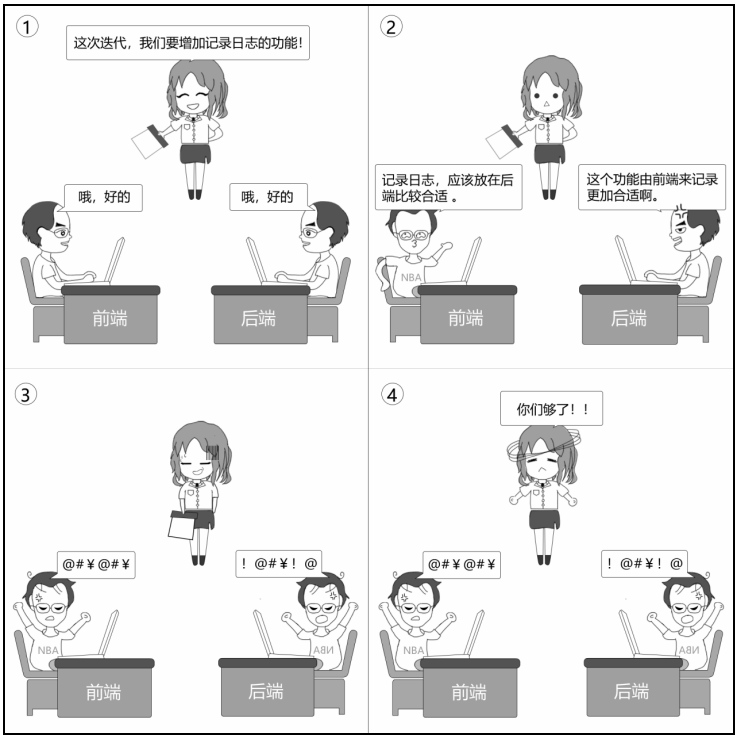


图 2-3　踢皮球

会错失机会

越是难做的任务，做完之后收获就越大。谁越愿意啃硬骨头，谁的成长就越快！小王在工作中经常踢皮球，把工作踢给别人后，自己每天很清闲，非常开心，

工作时间逛淘宝，刷微信、微博。

小李在工作中勤勤恳恳、责任心强，遇到事情他第一个上，出了问题他敢于站出来担责，一边鞠躬道歉一边加班工作。

几年之后，小王能力没提高，看到身边的同事都升职加薪，就他没有。小李在工作中虽然拿的钱不多，工作又很辛苦，但是积累了很多的经验，不到一年就成为项目的核心骨干。老板开始把各种重任都交给他。慢慢地，小李一个人干不过来了，老板就给他配下属，让他成为技术经理。小李业务能力强，带领团队的能力也提高了，很快就成为公司离不开的顶梁柱了。

会使人缘变差

踢皮球的人会直接被认为不靠谱、不担责、油头滑脑、精明而不聪明，口碑会变差。

- 口碑好的人，机会就多。因为大家都喜欢口碑好的人，有相关的事儿一定会喜欢找他做，有相关的机会也会第一时间推荐给口碑好、靠谱的朋友。
- 口碑不好的人，人缘差，朋友就少，遇到事情没人帮忙。

口碑和人缘，会给你的职业发展带来非常持久的影响。

会使人平庸

如上面的小王，踢皮球不干事儿，人都待闲散了，三十出头就会遇到职场危机，遇到公司裁员的话第一个候选人就是他。

要抓住一切机会带团队

笔者认为世界上有两种人：一种是领导，另一种是基层。两种人拥有两种不同

的人生。从父母辈的经历来看：

- 基层的人往往容易任性，做事简单粗暴，方方面面考虑不够，坎坷多一些。
- 做过领导的人往往处理问题周全，有胸怀，容易成功。

一个人做不成事情

这句话听无数人说过，最开始的时候我不理解。一个技术大牛不就能抵得上一堆菜鸟么？

后来慢慢地懂了，一个人的精力是有限的。当我每天要处理的事情过多的时候会发现，很多比较基础的工作应该交给下属去做，因为：

1. 一个人每天的精力有限，特别是领导、公司的老板，每天都会考虑公司的生死存亡问题，睡得晚起得早，往往精神状态不好。

2. 让老兵做新手的事儿是对优质资源的浪费。

3. 失去了对新人的培养机会。新人都是需要锻炼的，刚好很多基层的事情就应该交给新人去做，做好了新人得到成长，做差了也有老手来把关。

带团队能让人开阔眼界

每个人的智商都差不多，为什么有的人能做出正确的判断，而有的人就不行呢？就是因为信息量不同。

如果我们接触到的信息量变大了，就会发现：

“哦，原来这件事我之前以为他的处理方法很荒谬，现在看起来好有道理。”

“哦，原来那件事我不了解，当初我做得不对。”

在公司中管理者的信息量是比基层员工大很多的，每个管理者都会接触到公司的战略方向、下个季度的目标、本次项目的中心等。久而久之，管理者的情商和思考问题的方式也会发生变化。

我们开玩笑时会喜欢说“某某在下很大的一盘棋”，其实这句话对于管理者来说完全是褒义，这种“大局观”是只有管理者才会具备的智慧。

而基层员工则只能机械地处理一些底层问题，不会得到锻炼。

具有带团队的经验能让人更好地在社会中生存

一旦带领了团队，这个人就要思考如何去管理，就要去学习管理学，眼界也会随之开阔，情商也会大幅提高。

1. 知道如何与人相处，化解矛盾

这种能力非常重要。我们日常会看到好多新闻，往往最初是很小的事儿，由于双方都不会处理， 鲁莽行事，导致最后闹到派出所。这种很“鲁莽任性”的人，往往都在基层。

而做过管理的人，往往都会很好地把握对方的情绪和思路，化干戈为玉帛。

2. 知道如何站在老板的角度考虑问题

管理者往往既要管理下属又要面对上级，所以每天都会变换角色思考问题，在这种角色的变换过程中，更加会换位思考，更加知道如何站在老板的角度考虑问题。这对于工作的顺利开展是绝对有好处的。

带团队是职业生涯注定的方向

不要相信四五十岁还能编程，不要崇拜国外的大叔级程序员。本人在 2005 年毕业时，欢送会上学院一位德高望重的博士生导师给大家致辞的第一句就是“你们将来都是要做管理的！”

可惜当时她没有掰开、揉碎了解释这句话，所以在毕业之后的十年里，笔者崇拜的都是国外不带团队、直接向盖茨汇报的那种技术大牛：基本没人管，也不用管理其他人，多么自在潇洒！

后来发现这条路行不通，原因有如下二条：

1. 只要你不踢皮球、勤勤恳恳地干上三年，一定会成为团队的核心人物！只要老板不是傻子，他绝对会让你带领团队。

2. 国内的环境使然。私企（互联网公司和创业公司）的员工平均年龄是 20~28 岁，外企员工平均年龄是 20~35 岁，国企的员工年龄能到 50 多岁。如果你进不了国企，在国内的私企和外企中，30 岁出头就要被裁掉了。

笔者经历了 Motorola 的裁员，眼睁睁看着很多三四十岁的同事被裁掉，老板带着绿卡飞回美国。

笔者也目睹了私企中的各种人员流动，超过 30 岁的平庸人员渐渐没有领导愿意要，进来的都是刚毕业的孩子，有的比自己小十岁，有的跟自己一个属相、年龄却比自己小一轮，危机感扑面而来。

所以，一定要做管理者，一路升到职业天花板再说，否则一直平庸的话就只能给人打工，三十岁以后的日子很难有什么希望。

要有良好的心态

每天都要学习

因为学习是一切工作的王道。

人的一生都需要不断地接受新的知识：

1. 谈恋爱的时候需要了解异性心理，知道如何更容易地追求到喜欢的人。

2. 工作时需要学习基本的社交常识，知道如何待人接物、如何与同事很好地相处。

3. 带团队时需要学习管理学，知道如何做一个能力强、有胸怀的领导者。

对于 IT 从业人员来说，自身职业的知识也非常重要，以程序员为例：

1. 初级：语言的基础知识，起码学通几个框架，熟知 API 文档，提高英语水平，需要 1~3 年。

2. 中级：自动化部署，自动化测试，重构，设计模式，提高英语水平，需要 2~3 年。

3. 高级：带团队，啃技术硬骨头，培养新人。

所以，既然走上了 IT 行业，就一定要记得：每天都要学习！不学习就要有负罪感。

对于程序员来说，不学习就会平庸。技术一直在发展，不学习就跟不上。今天出来 Angular，明天出来 React，后天是 Vue.js，每个都要看、都要学习跟进。

队伍难带，小弟不好管理，怎么办？学习怎么管理啊！

移动端开发，技术不行，怎么办？Android、iOS 每个都要学。

服务器不行了，怎么办？上网搜，翻文档。

不要沦于平庸

我们的格局一定要大，内心一定要有一个远大的目标，例如：

我要三年内成为技术大牛，熟练掌握 Ruby on Rails 和 JavaScript 的 Vue.js 框架。

我要今年内背上 3000 个英语单词，不然每天的文档中太多生词看不懂，太难受了。

我要今年把自己的体重减一减，再不减该突破 200 斤了。

绝对不要想着：

哎呀，还有 30 分钟下班了，先去市场买点儿白菜吧。

听说 XX 超市的 XX 东西要打折了，我是不是可以薅一把羊毛？

工作就是最好的学习机会

对于拥有工作的同学，一定要换一个角度、积极乐观地看待自己的工作。积极的工作心态可以让人更加开心，如图 2-4 所示。

图 2-4　积极的工作心态可以让人更加开心

千万不要认为“工作是万恶的资本家对于我的剥削”，要看成：

1. 我每天在用公司的项目练手，做砸了就会辜负公司对我的信任，做好了就能提升自身的价值。

2. 公司的薪水是我的奖学金。

办公室没有政治

大学的时候，回看高中时期的竞争没有意思；入职之后，看大学同学之间的寝室斗争很没意思；等创业之后，就会觉得打工时期的各种办公室政治完全没意思。

所谓的“张总”“王总”仅仅是一个称呼。大家都是打工的，而老板则是给整个公司的员工打工。大家在人格上都是平等的。

所以千万不要把办公室政治看得很重。如果某天发现办公室政治已经影响到了自己，那么有两种解决办法：

1. 让自己的直接领导来解决问题，肃清干扰。

2. 如果自己的直接领导和稀泥，那么赶快跳槽，离开这个小池塘，找到可以让自己“海阔凭鱼跃”的大公司。

绝对不要让自己慢慢习惯各种办公室斗争，不要立山头，不要站队伍，否则时间长了自己也会慢慢变成自己讨厌的人。

不要参与公司的八卦

每个公司多少都会有流言蜚语：

公司的老板跟秘书似乎关系很神秘。

财务小张据说是老板娘的表妹。

项目组的老王似乎在甲方关系很硬。

不要参与，我们要完全对事不对人，默默地把自己的事情做好，绝对不要被公司的八卦分散精力。

正确面对公司的裁员

据美国《财富》杂志报道，我国的中小企业平均寿命是 2.5 年，集团企业的平均寿命不到 8 年。所以每个公司都有可能倒闭，对于行业新人可能会非常敏感和八卦，务必忍住，要有一个正确的态度来面对，如图 2-5 所示。

图 2-5　正确面对公司的传言

要意识到下面几点：

1. 在事实出来之前，任何猜测都是不负责任的。

2. 树挪死，人挪活。安于现状只能让自己的生存能力退化，增强忧患意识才

能让自己的能力不断成长。

3. 裁员不一定会裁到你的头上。一方面自己要继续完成分配的工作任务，另一方面要让自己每天充实起来，不断成长。如果哪天公司真的要裁员了，确保自己的能力足够强大，可以随时跳槽。

敏捷方法论

敏捷开发不是一个新鲜的词汇，10 年前就有很多类似的书，请大家找几本来详细阅读。敏捷方法论是一组最佳实践，跟截拳道的思想很像：务实、灵活、不死板，什么方式好用就用什么。

下面是对敏捷开发的一些实践。实践证明这些方法非常有效，能非常好地避免项目死掉。

频繁交付、小步快跑

这个方法论的目的是：尽快看到工作成效。

1. 不要做项目时半年才交付一次。每个项目理想的情况是每天都要交付。
2. 下班之前做个部署，把每天的代码都放上去。
3. 手机 App 的项目，每天晚上下班前发一个包。

这样的好处有很多：

- 增强了双方的沟通和信任。
- 每天都能看到项目在进步，大家都对项目有信心。
- 遇到问题也都能知道原因出在哪里。
- 需求方有变动的话可以以最小的代价来实现。

能自动化的都自动化

自动化包括：

1. 自动化部署。
2. 自动化的单元测试。
3. 自动化的集成测试。
4. 其他自动化的任务。

总之，如果你的工作当中存在很多人肉操作可以替换成自动化的工具，那么赶紧动手改进。

必要的测试

软件项目是非常复杂的，很多时候老板问起“这个项目怎么样了？能发布吗？”的时候，我们不应该答不上来，不应该说“老板，我不清楚”，而是应该很潇洒地运行一下测试命令，跑通所有的单元测试，然后告诉老板：“当前项目有 100 个测试，跑通了 92 个，问题不大，我把剩下的测试修改好，估计今晚就能上线了”。

ThoughtWorks 的员工曾经提到过一种非常好的实践方法：在饮水机旁边放一个红绿灯，哪个成员提交了代码，会触发持续集成服务器自动运行所有的测试，都通过的话给出绿灯，否则就是红灯。

这个红绿灯时刻反映出当前项目的健康度，每个人打水的时候都可以瞄一眼，然后立刻知道当前项目的情况。

一个实际情况是：国内的人慢慢意识到了单元测试的重要性，不过敢于实施的项目团队还是非常少的，可以认为几乎没有。

每日例会

一般放在每天早上，大家站着围成一圈，每个人用一分钟说三件事儿：

1. 昨天做了什么。

2. 遇到了哪些困难。

3. 明确今天要做什么。如果自己不知道的话可以直接询问上级。

上级只需要给大家解决问题和困难，再告诉大家要做的新任务就可以了。

之所以要站着开会，是因为全体站久了都会累，这个时候大家都明白该结束了，不会拖沓。另外，站立会让大家更容易集中注意力。

这样做的另一个隐含概念是让所有人都知道队伍中没有闲人，每个人都在努力地干活，提高队伍的工作效率。

要培养成学习型团队

例如，把近期出现的相关领域的新技术列出来，成为一个“技术雷达”，然后整个团队制定一个学习计划，详细分配到人，大家依次来给团队做培训。

一些新技术也可以谨慎地使用。

每个人每天都在进步，你今天学得慢，明天就会被队友赶超，整个学习型团队的学习氛围特别浓，相互激励，相互学习。

良好的程序员工作习惯

现在绝大部分程序员(也包括其他行业的从业人员)的生活规律都非常不健康，违背了自然之道。该睡觉的时候不睡觉，该起床的时候不起床，该运动的时候不运动，一坐就是一天。健康和不健康的生活作息如图 2-6 所示。

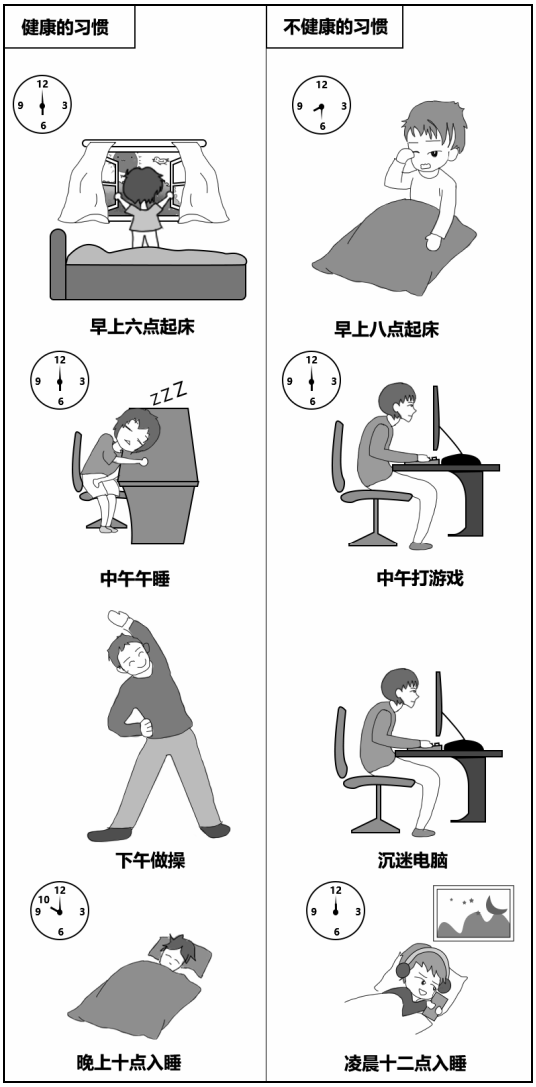


图 2-6　健康和不健康的生活作息

晚上十点前睡觉

最好是晚上九点入睡，早上自然醒（最好在早上五点），但是我知道大家不可能

做到。

尽量不要熬夜。熬夜是非常伤身的事情。很多程序员特别喜欢看凌晨三四点的城市，最喜欢晚上写代码，因为那个时间段安静，没人打扰。这个习惯特别不好。

会导致：

- 秃顶。
- 肾虚，身体弱。
- 心脏不好。
- 脸色黑，皮肤差，长痘痘（这种痘痘挤了就是陨石坑）。
- 记忆力减退。
- 五脏六腑机能都不好。

好的程序员可以很好地安排个人的作息，让自己的职业生涯持久、高效。

健康问题：不要总低头弓背

如果公司不给配外置显示器、只给笔记本的话，赶紧自己买一个。如果你的个子太高、坐在电脑前面都是低头弓背的话，赶紧把显示器垫起来！总之，低头会引起颈椎问题和驼背。

正确和错误的坐姿如图 2-7 所示。

图 2-7　正确和错误的坐姿

离开显示器和手机才是休息

如果干了好长一会儿，务必出去走一走，不要坐在电脑前面。

有的同学认为，我听一会儿歌，看看新闻，也是休息。其实不是的。你坐在电脑前面，整个人都没有得到休息。务必站起来出去走一走，哪怕是上个厕所、拿个杯子接些水，也能改善身体的血液循环。

1. 每用 1 小时左右的电脑，就站起来活动会儿身体。 手头放个秒表很有效果（可惜自己创业后就很难这样了）。

2. 必须有一个杯子，觉得疲劳的时候，站起来喝点水。

3. 找个地方活动下颈椎和脊柱，对于一些办公楼，可以在楼道间慢跑，做做操。

4. 不要吃零食，伤脾胃。

5. 不要抽烟。

不要沙发椅，要坐硬板凳

50 块的那种硬板凳绝对比老板转椅好太多。伤腰的转椅和护腰的硬板凳如图 2-8 所示。

图 2-8　伤腰的转椅和护腰的硬板凳

- 很好地避免腰酸背疼。特别是把它反着坐的时候。因为反着坐是可以强迫人使用挺直后背的坐姿的。
- 当屁股觉得疼的时候，就是它提醒你已经连续坐了两个小时，该休息、该站起来走一走了。

显示器要有护目屏

护目屏不要买几十元的，在淘宝上挑个几百元的才好。现在大部分 27 寸显示器都不好，颗粒太大，炫光太强烈。炫光会直接对眼睛造成伤害，使用护目屏后效果还是很明显的。

推荐使用特种玻璃做的护目屏，27 寸的话，价格一般在 400 元左右。玻璃材质的护目屏很有效果，如图 2-9 所示。

图 2-9　玻璃材质的护目屏很有效果

请记住：眼睛是心灵的窗户，所有的 IT 职业都离不开电脑和显示器，眼睛坏了这辈子就全完了。笔者 2005 年入行的时候没把这句话当成事儿，现在眼睛有飞蚊症、葡萄膜水肿、眼压高。

有的同学会说“休息一下眼睛不就好了？”

说得对，可是作为工作离不开电脑的人你如何休息眼睛？除非辞职。

我们每天的生活可能八个多小时都要使用眼睛：看书，看手机，看电视，看电影，看视频。所以要多出去走走，让眼睛看看自然光会好太多，现代医学表明太阳光对于眼睛的健康有直接影响。

程序员的工作组成

程序员的工作不是一直在写程序

一个程序员工作的时间分配大概是这样的：

30% 的时间敲键盘写代码。

20%~30%有可能是在郁闷、愣神、看技术文档、学习或者思考。

20% 的时间与人沟通。沟通需求，排查 Bug。

10%~20%的时间做其他的事情，例如运维、处理服务器、Git 操作、招聘、写文档（开会纪要、事故的处理情况等）。

技术经理

技术经理的大部分时间是管理团队、培养新人。

10%~30%的时间写代码。

30%的时间谈项目，整理需求。把新人听不懂的需求和任务做划分。

25%~50%的时间培养新人。

25%~50%的时间开会沟通、搞管理。

10%~20% 的时间要资源（要招聘新人）、申请部门经费、申请显示器等设备、写文档、处理团队成员的请假和提薪要求、管理服务器、处理项目硬骨头等。

程序员要走出去

程序员这个职业有个最大的先天不足就是，会让人性格变得内向、敏感。

性格内向

程序员在编程时，其实都是在和机器打交道。很多程序员喜欢深夜工作，与人沟通很少，自然容易性格内向。具体表现是：

1. 工作中，性格内向的程序员永远不会主动搭理人。
2. 聚餐时，性格内向的程序员永远不说话。

过分细腻

可以认为，过分细腻是程序员的职业病。现代流行的任何编程语言都会要求变量不能差一点儿，student_name 跟 student-name 完全是两个不同的变量。

在 Bash 编程中更加严格，多个空格就会引起错误。

所以程序员的性格弱点是容易较真、钻牛角尖，认为事物不是 true 就是 false；程序员的认知是黑白分明的，不存在灰色的地带。

笔者在 2015~2016 年曾经跟体力劳动者一起共事过。当时做互联网家装的 CTO，经常下工地，发现跟这些工人打交道很容易，和他们在一起吃饭喝酒、嬉笑吵闹，性格都很粗犷、豪爽，往往抽根烟就是好朋友，说话直入主题，交流起来特别放松。

跟程序员交流则需要小心谨慎，一个需求没表达明白就是一个白眼，或者遇到问题也不能立刻说是 Bug。

互联网公司中的产品经理跟周围的程序员同事沟通较多，应该深有体会。

容易自傲自大

程序员在性格上容易两极分化。

在自身实力弱小的时候，往往什么都觉得是老鸟说得对。在团队中最有威信的人，是可以搞定其他人搞不定的 Bug 或者功能的人，往往一般会被冠以“大神”的头衔。

一个小菜鸟，经过两年左右的入行时间或者经历过若干项目的磨练之后，可以独当一面时就容易成为这种“大神”。这个时候程序员的心态容易变化，觉得自己技术特别强。

在很多程序员的论坛中，这样的人特别多，都是键盘侠，喜欢打嘴炮，觉得自己天下技术第一，自己用的才是世界上最好的编程语言。

这个时候难免自傲。

不要坐井观天，要多看看外面的世界

程序员的自大自傲，或者水平的止步不前，跟自身不愿意与高手接触有很大关系。很多程序员就是宅男，特别是单身的程序员。

其实跟外界接触的方式有很多。

1. 多参加程序员世界的聚会，例如北京的 Ruby 圈子中就有这些活动。

Coding Girls：教妹子学编程。Coding Girls 的活动现场如图 2-10 所示。

图 2-10　Coding Girls 的活动现场

- Ruby Tuesday：来自台湾地区的一个习惯，大家会在周二晚上聚在一起聊关于 Ruby 的话题。
- Beijing Open Party：北京比较高端程序员的聚会，2016 年以前大约 2 个月一次，地点往往在 ThoughtWorks 的北京东直门办公室。

更多聚会可以在对应的论坛上看到（例如，活动行）。在这些聚会上可以面对面地跟大牛交流，对于程序员的发展特别有好处：

- 可以知道自己未来的发展方向。
- 可以知道行业的前景。
- 可以让大牛传道授业解惑。
- 可以认识更多人脉。
- 可以更进一步地激发自己的学习动力。

2. 多参加开源项目。Github 的账号是必须有的。

- 编程能力强的，看到自己平时用的哪个 jar 或者 Rubygem 有 Bug，刚好自己昨天在工作中给解决了，就可以贡献代码了。
- 编程能力弱的，看到某个项目特别好用，就可以贡献翻译。
- 对于自己感兴趣的事情，也可以建立一个项目，然后吸引更多的人来一起合作。

在 Github 上跟人交流特别重要，特别是在跟老外合作的时候，它可以让人深刻地意识到自己跟其他优秀程序员的差距。从代码质量到命名风格、单元测试，再到沟通技巧，哪怕只参与两三天都可以有很大的收获。

规划好业余生活

每个程序员都有业余生活，我这里总结几点，希望对大家有帮助。

不要爱上旅游

我身边有不少“九零后”IT 从业人员，最喜欢干的事是在朋友圈晒旅游照片。这个周末海南，下个周末长白山。美其名曰“在最美好的年纪对得起自己的青春”。

笔者认为年轻应该是用来奋斗的，用来流血流汗的，不是用来玩耍的。作为年轻人要承担起责任，爸妈退休了你要不要供养？跟女朋友周末看电影吃饭是不是要开销？结婚了买房钱还没有，怎么办？有了娃，奶粉钱够不够？

心中一定要有责任感，特别是不要一分钱不攒，每个月做月光族。当你想外出旅游的时候，想想家中舍不得花钱而省吃俭用的父母吧！

不要接私活

这个问题是所有程序员都会面临的问题。很多程序员认为自己的工作很清闲，赚点外块是很好的事。

小王在某 500 强公司工作时，刚好赶上该公司有人事变动，于是小王每天的工作非常清闲，每天工作 3 小时足够了。于是接了一个私活，参与一个国外项目。

结果，接了私活之后每天的生活质量直线下降：之前每天工作 3 小时，接私活之后每天工作 10 小时，晚上 10 点还要开会，周末还要拿出来干一天，完全没有休息。平时接私活儿打电话要小心翼翼，很多时候开会不方便接，发私活的老板很不理解。

非常辛苦，每个月的报酬大约是自己工资的一半，很多时候觉得不值。

只要目前工资可以满足开销，建议不要接私活。

99%的公司都会对工作时间做私活儿的员工提出惩罚或者开除，风险很大。

如果时间足够的话，可以多学习，多为自己做一些事情，而不是贱卖劳动力。

利用业余时间做教学

程序员用业余时间做教学是性价比最高的事情。教学分成两种：文字教材和视频教学。

某互联网公司小陈的本职工作是运维，工作很苦，每周至少做一次通宵值班。入职第一年刚好是单身，于是周六周日开始写教程，录制视频，只要工作中遇到的技术都会录制，《Nginx 最佳实践》《Linux 从入门到进阶》《MySQL 深层次分析》……结果一年下来，视频录制了不少，自身的实力也在备课中不断提高。

后来他被评为某课程教学网站的知名教师，离开老东家后被人约课不断，收入颇丰。

中国 IT 公司的特点

中国的 IT 公司跟美国的 IT 公司很不一样。

技术实力层面

中国的 IT 公司一般比较弱，根源在于国人的英语水平太差。

如果不考虑英文只考虑算法，国人是一点儿都不差的，可惜互联网开发对于技术的要求是有层层依赖要求的：

1. Web 应用的开发依赖于“应用技术的框架”。这个框架指的是类似于 Spring、Android、Rails、Django 这样的技术。

2. 应用技术依赖于编程语言等底层技术，包括 Web 底层、Android 底层、JVM 等各种编程语言的虚拟机等。

3. 应用技术的底层技术依赖于操作系统和硬件。

所以，从上面的三点看下来，大家会发现：

1. 国内几乎没有好的操作系统，Windows 是美国的，Mac OS 是美国的，Linux 是开源的（世界的）。国内没有操作系统，修改 Linux 的不算。

2. 国内几乎没有好的编程语言，例如 Java。各种编程语言，几乎没有国人发明的。

3. 框架级的技术，国内也极少创造。

所以，面对一水儿的外文资料，英语不好的人完全是发蒙的状态。欧美人在这方面则具有巨大的语言优势。

人员的年纪差距

国内的程序员往往特别年轻，项目中 90%的代码可能是由工作 1~3 年的程序员编写的。工作 4 年以上的程序员，基本都是技术经理。

在美国公司中，很容易看到大叔级的基层程序员。

虽然我不清楚美国的工资标准，但在国内，员工的工资跟年龄没太大关系，而是跟职位挂钩的。基层程序员技术再牛，工资也不如绝大部分的领导层。

35 岁开始失业

只要不是在企事业单位，35 岁就开始失业。在小规模的私企，这个年龄是 30 岁。企业更加倾向于新人的原因如下：

- 有拼劲，可以随时加班。
- 身体好，通宵很轻松。
- 不用上下班接孩子。
- 不用买菜做饭。
- 没有办公室习气，好管理。

所以，只要一名员工超过 35 岁，自身的实力又比较平庸的话，一定会被毫不

犹豫地砍掉。

技术高层不懂技术细节

技术高层（如 CTO、技术总监）虽然在早期为公司做过巨大贡献，但是随着业务的发展和自身职位的提高，往往不懂基层的技术细节，因为代码都是别人写的。

项目出问题时，技术高层不是冲向第一线的人，冲上去的往往是基层做了好多年的人，如基层技术经理、基层程序员。

在一个大公司中，技术高层多年脱离基层，很多自己公司的代码都没有摸过，所以出了问题一定要靠下面的基层程序员来解决问题。

管理更加严格

一些外企和硅谷海归精英创业者创办的公司特点是：

- 推崇人人平等，每个员工都觉得自己很了不起。
- 特别随意随性，喜欢牛仔裤拖鞋。
- 组织性纪律性不强，以自我为中心。

国内的企业则管理严格一些，工作环境更加容易沉闷。

另外，从管理层看，国内公司的管理手段更加严厉一些，不如外企那么活泛。

国内软件岗位的地域特点：北上广深是绝对主力

表 2-1 中的数据来自于国内最大的招聘网站智联招聘中的搜索结果。竖列表示用不同的语言作为关键字，横列表示不同城市的搜索结果。

例如，搜索“Java”这个关键字，在“北京“中出现了 327 页搜索结果（每页包含 60 条数据）。

表 2-1　国内城市和软件岗位统计表

	北京	上海	深圳	广州	天津	成都	杭州	武汉	大连	沈阳	西安
Java	327	156	128	73	34	89	75	68	37	25	46
PHP	69	30	30	24	8	22	16	13	8	6	10
Python	160	74	53	27	10	25	29	15	6	4	14
Android	72	41	42	22	8	21	19	15	8	6	12
iOS	58	31	26	19	6	16	14	11	5	4	7

从表 2-1 可以看出：

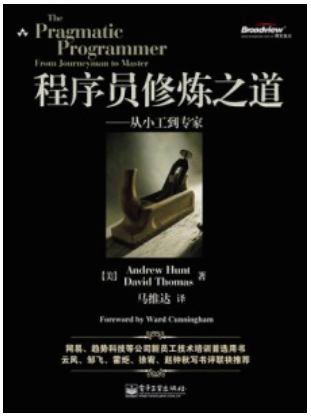
- 北京是全国的软件龙头，对于人员的需求大约是上海、深圳和广州之和。
- 北京的程序员职位大约占表中职位总和的 30%~40%，根据语言有所不同。目前我们熟知的大部分互联网公司都注册在北京。北京作为全国的政治和经济中心，很多政策都对创业者有利。
- 上海和深圳大约是北京职位的一半，位列第二梯队。这个跟国内的城市排名基本一致。
- 广州、成都、武汉和杭州的程序员职位基本一致，大约是上海或深圳的一半，位列第三梯队。杭州的发展应该得益于阿里巴巴这家公司。
- 其他城市，基本没有软件市场。

另外说一下台湾地区的软件特点：一方面他们跟美国接触得比较多，英语底子好一些，技术基本功扎实一些；另一方面由于人口太少，互联网完全不如大陆发达，到现在也没有一个知名网站。而大陆的知名网站随便抓一个就用户上亿，这对于程序员技术实力的锻炼完全是天上地下。

读书清单

本节中推荐的都是软件工程方面的书籍，这些书都是经典。不过也要注意：国外的软件环境比国内踏实不少，所以国内的程序员要注意分辨、活学活用。

《程序员修炼之道——从小工到专家》



这本书无论是行业老兵还是准备入行的新人，都非常适用。每次翻看都会有新的收获，很多现实中的问题都可以在本书中找到答案。这是笔者的入行引导书。在 2005 年的秋天，笔者当时还是一个刚毕业的毛头小子，做项目就是一通乱写，丝毫没有章法，也没有人指导。在书店里的某个角落看到这本书，结果拿起来就放不下了。

曾经试图为它写总结，很快发现这本书没法提炼：干货太多、字字珠玑，整本书一个字一个字地读才是正确的打开姿势。

书中主要包含两部分内容：

1. 程序员的技艺，对新手是非常好的引导。
2. 很多实用的方法论。这些方法论能给人启发，具备增智开慧的功效。

本书内容精炼、句句干货，适合反复翻阅，是笔者少有的几本一读再读的书。另外，中文翻译的质量很好，所有对软件工程感兴趣的朋友都很适合阅读。

《软件工程的事实与谬误》

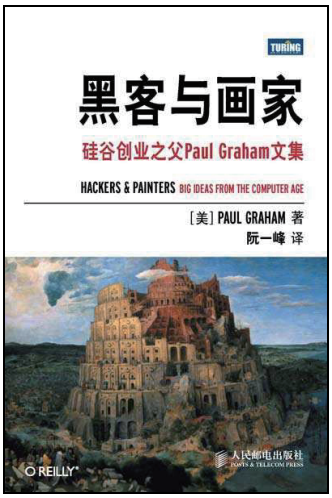
这是一本奇书。笔者在工作的第三年碰到它，书中的内容却在未来的十年中不断地激荡着笔者的思维。正是因为这本书的引导，笔者在 2014 年创业之后不断地思考国内的软件环境和行业痛点。

这本书没有涉及太多的技术细节，但是给读者打开了一个完全不同的软件行业，从中大家可以发现自己很多对软件行业的认识都是错误的，例如：

- 软件项目都是由不懂技术的高层或者营销人员来估算时间，所以估算的时间都是错的。
- 对于项目可行性的调研回答几乎都是“可行”。
- 问题的复杂度每增加 25%，解决方案的复杂度就增加 100%。

它的内容非常精彩，干货太多，很难总结。

《黑客与画家》

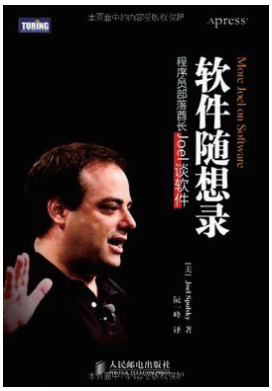


本书作者 Paul 以自身的经历（一个黑客是如何用自己的高超技术干掉竞争对手、实现财富自由，并且进一步创办了硅谷的 TOP 1 投资基金 Y-Combinator）给读者展示了一个合格的程序员的世界：程序员并不是书呆子，反而是极度聪明，知道如何利用自己的长处实现梦想。

书中阐述了大量的方法论、跟技术相关的细节和一些解决问题的思路，特别适合入行几年、以“高级工程师”自居的读者。读的时候把作者跟自己做一个对比，会发现自己的未来出路。

书中对于自己使用 Lisp 语言跟对手竞争，以及对于编程语言的能力论述（比较 C、Java、 Perl、Python、Ruby、Lisp）的章节非常精彩。

《软件随想录》

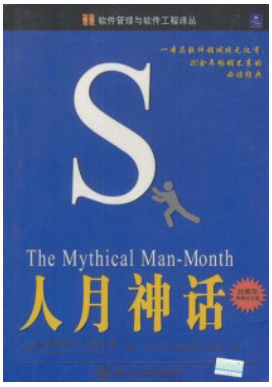


本书是 Stack Overflow 的创始人 Joel 的博客汇总，里面写了软件开发涉及的方方面面，从入职到成长，从管理团队到经营公司。虽然给出的经验都是国外的经验，出版也十多年了，跟国内的形式略有出入，但是非常具有参考价值。

另外，本书用大量的篇幅阐述了作者思考的深层次技术问题，例如，如何识别优秀的程序员，对于美国程序员的分类和 Stack Overflow 的相关内容。

大家阅读的时候不要思考字面的意思，要带有自己的思索，最后才能获得启发。

《人月神话》

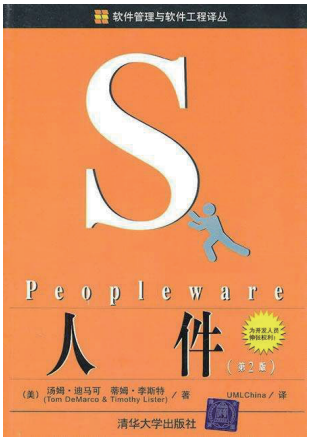


这是一本软件开发圣经，第一版出版于 1975 年，距今已经 40 年，不过里面80%的内容还是为行业外的人所不知。书中非常深刻地阐述了为什么不要用传统的眼光来看待软件开发，以及相关的一系列问题。

本书由于内容过于专业，部分章节可以略过，但是“人月神话”和“没有银弹”的相关章节务必仔细阅读，可以帮助我们识别一些项目陷阱。

本书适合所有人，特别是跟程序员打交道的项目经理、创业公司的老板。

《人件》



本书想要表达的观点是“人”是软件开发的根本，跟其他因素（管理、工具、组织架构等）无关。

强烈建议所有人阅读。有意思的是，从本书中也可以窥见一系列的程序员心理活动、管理者的期望等。

职业前辈的博客

职业前辈的一句话，往往可以让人少走几年弯路。他们的今天，就是新手的明天。每个领域都有自己的前辈，在 Java 开发界，有一些大牛的博客一定要多看。

看他们博客的时候，不要只关注技术，而是要多做全盘思考。

从技术到生活，再到习惯、情怀。大牛的养成，是全方位的。

下面的大牛，是笔者之前打工的时候看得比较多的。

- 范凯，Javaeye 的创始人。一个讨论 Java 的论坛，用 Ruby on Rails 来写，而且还写得非常成功。可在微信中查找公众号“肉饼铺子”或者“robbinthoughts”。
- 熊节，曾任多年 ThoughtWorks 的首席咨询师，也是《重构》《与熊共舞》等若干软件工程畅销书的译者，翻译质量很高，个性鲜明直率。除了个人博客，他在 Javaeye 的历史发言也非常精彩。
- 阮一峰，很博学的人，翻译了《黑客与画家》《软件随想录》等畅销书，很有开源精神。