13 | 特别放送：选择比努力更重要

2019-10-09 四火

全栈工程师修炼指南 进入课程 >

讲述：四火
时长 16:51 大小 13.51M

你好，我是四火。

又到了一章的末尾，特别放送时间。专栏上线后的这几周，我在留言区回答了一些问题，有一些是技术上的问题，也有一些是非技术上的问题。尽管在 [开篇词] 和 [学习路径] 中我已经介绍了全栈工程师的角色、重要性和学习方法，但是依然见到不少困惑和疑问，**其中一个问题反复出现，那就是面对那么多的软件技术，总有一种“学不过来”的感觉，为此感到焦虑和担忧。**尤其是对于全栈工程师而言，这个话题更是被放大了。

颇为遗憾的是，这几年来，我见到了一些相当有经验的做着 Web 全栈开发的程序员，他们还依然走在一条埋头苦干，不断堆积知识，单纯靠量取胜的路上。可是，我认为，学习是需要选择的，并且，选择比努力更重要。

今天的特别放送，我就来聊一聊，我是怎么认识这个问题的，希望能给你带来一点参考意义。

两个小故事

第一个故事，微软的测试团队改革。陆奇是一个程序员从技术做起，进而翻身的典范。最初他入职雅虎的时候只是一个普通的工程师，十多年后，他以执行副总裁的身份，不但牵头打造了 Bing 搜索，还完成了几项意义深远的改革，其中一项，就是合并开发和测试这两个原本独立的部门，大幅裁剪专职测试人员，让工程师做更多的事。

这样一来，有一些擅长使用内部测试工具进行测试的工程师，就慢慢丢掉工作了，原因很简单，他们更多的只是熟练工，而缺乏技术上的竞争力。**听起来，这似乎是微软内部组织架构变动和工具、技术栈封闭的锅。**

其实，合并开发和测试团队，对于许多软件公司，特别是互联网公司来说，是一个基本不需要讨论的事情。除了一些面向互联网用户的、交互较为复杂的团队，还配有专职测试（即便是这样的测试，为了节省成本，也常常外包给合同工去做了），绝大多数团队，都已经变成了软件工程师一肩挑的局面了，而测试只是被合并的一部分专职角色，其实还有线上运维等等。对于大型互联网企业来说，一个事实就是，工程师的全面性越来越强，这是一个不可辩驳的趋势。

第二个故事，是我在几个月前，参与面试一位有着超过 15 年 Web 开发经验的工程师，这件事就发生在五轮面试之后的讨论会上（这里涉及到的面试流程我曾在 [第 06 讲] 中介绍过）。

对于这样一个比较资深的工程师岗，五个面试官，在讨论会上产生了巨大的分歧。招聘经理，也就是负责吃饭聊天那一轮面试的人，反馈非常正面，说此人经验丰富，谈吐得当，leadship 层面也没有问题；Bartender（技术负责人）重点考察了项目经验和系统设计，反馈也颇为不错，经验丰富，而且能够从很多角度去分析和思考问题；可其余的三轮面试，面试官反映却大相径庭，主要问题出在白板代码部分，总的来说，代码生疏，书写也缺乏条理，其中有一轮都没有写完代码，写完的两轮也是 bug 满地，问题频出。最后，在争论之后达成一致，可以通过，但是只能勉强给一个比预期职位级别低一级的 offer。

作为一个技术岗位的工程师，编码能力是一个硬指标，因此这也是照章办事，不得已而为之的。只不过，我们私下里讨论，应聘者很可能是有足够的编码技巧的，只不过有一段时间没有写代码了，确实有些可惜。但是，猜想也只是猜想而已。

其实，故事有很多，但是这两个故事是我精心挑选的。二者分别对应的启发是，工程师在技术学习的时候，需要遵循的两个主要思路。

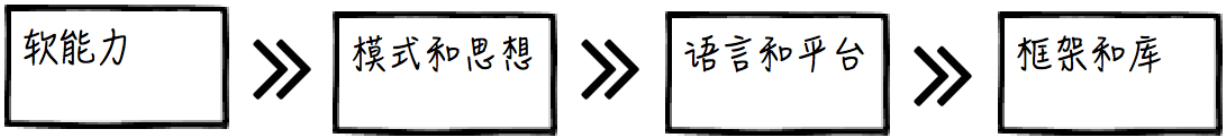
第一，技术是分级的，具体说，是分短命和长寿的，也是分表面和本质的。我们要学习各种技术，但是我们要把足够的精力放到长寿的技术以及技术本质上。这就是第一个故事带来的启发。

第二，基础知识和能力的训练需要长期坚持，无论是在工作中，还是工作以外。这就是第二个故事带来的启发。

技术的分级

下面我展开来讲第一个思路，技术分级。

我们都听过“技术无贵贱”的说法，但这并不代表我们要“无区别”地学习技术。工程师，说白了最重要的就是“工程能力”，就是应用工程化的思想和技术，去解决实际的问题。我把工程能力粗略地分成这样四个级别，请注意，**它们都必不可少**，这个分级也不严密，但是我们能从中看到一个大致的趋势：



比如说，稳定性上，从左到右逐渐降低，越靠右往往寿命越短。

比如说，学习难度上，不一定，但是总体来说从左到右逐渐降低。

比如说，针对性上，往往从左到右逐渐增加，即越往右就越是针对具体的问题。

下面我来具体说说图中的这几种能力。

软能力，比如我们对程序员要求最多的沟通能力、学习能力和抽象能力。哪怕再过几十年，这部分基本也是不会变的。

模式和思想，比如分而治之的思想，模块化的思维，客户端 / 服务端模式。这部分也相当稳定，可能随着技术的革新，也会有一些以往不受重视的模式或思想被列为重点，举例来说，十年前，我们可能很少谈论分布式的思想。

语言和平台，比如 Java 语言，JVM 平台。这部分其实变化就非常大了，有的语言已经几十年了，有的则是几年热度之后就消退了。普遍来说，语言平台要比语言本身稳定一些。

框架和库，这部分差异也非常大，但是总体来说，基本就是最短命的。如果要花费大量的时间和心血去深究某一个框架和库，那么你会希望它要么相当有代表性和典型意义，要么有足够长远的未来。

对于这几种能力，还有几点我要强调说明一下。

首先，请不要误解我的意思。每一级的能力都是有各自价值的，**我们在解决实际问题的时候，需要使用到每一级的技术。**举例来说，要设计实现一个网站，全栈工程师需要发挥沟通能力、合作能力，需要利用在模式和思想方面的经验和认识，选择合适的语言和平台，最后再选择相应的框架和库来完成它。

其次，我们当然希望图中靠左侧的“长寿”的能力得到进步，但每一级能力的培养，都需要通过对具体技术、业务的学习积累来实现，也就是说，**这些分级之间是不冲突的，反而是互相促进的。**通过这两章的学习，特别是从具体技术到通用模式这样的方式，也许你已经体会到了这一点。

那么现在，有了上面的说明，你是否已经清楚，该做出怎样的侧重了呢？

显然，我们应该明确的一点是，把几乎全部的精力都投入到这个分级的右端，换言之，**把大量的时间都花在记忆一个接一个的框架和库上面，妄图靠数量取胜，不可取。**有时更可悲的是，就连生硬的记忆本身也不讨好——都不用等到技术淘汰了，它们还都是公司内部的框架和库，离开公司就一文不值了。

因为，这只是单纯地记忆，没有比较、分析、思考这些能让这个分级均衡发展的行为。而这，我认为正是系统的技术学习中最大的误区。

我举一个真实的例子来说明这个现象，有位程序员朋友学习前端技术，最开始使用 Backbone.js，后来陆续用过 AngularJS 以及 React，现在准备转向 Vue.js。看起来似乎很勤奋，但是他自己却说他依然只是停留在上头说用什么，自己就跟着学什么的程度，自己也没有什么想法，总之越学越麻木……他自己还说“不知道何时是个头”。

接着，我想展开讲一下其它几个具体技术学习中的典型误区，而这些误区，在我看来也都和是否努力无关，本质上也都属于不恰当的“选择”。

1. 过于关注配置使用，忽略原理和场景分析。

这一点我觉得是最容易在迈入职场没多久的程序员身上见到的问题，非常普遍。老板一个命令下来，心里慌得很，就想着怎么把问题搞定，而有一些问题是知识迷局一类的，比如，怎样配置整合 SSH（Struts、Spring 和 Hibernate）。

这类问题光靠想是没法得出结论的，于是通过各种搜索、查文档、试错等等方式，花了很长时间，好不容易搞定了问题，完成了需求，又紧锣密鼓地去接下一个任务去了。**可是呢，对于刚才使用到的技术，付出了那么多，却只收获了这一个小小的迷局的解。**这投入产出比低不说，人还是很健忘的，这些具体的配置和使用，只要不用，很快就会忘掉的，下次在遇到类似的问题，可能已经忘记了这些配置，于是又要重复这一行为。

2. 过于关注编程语言的语法和语言技巧，忽略语言思考和书写时的思维模式。

比如说，JavaScript 语言，里面有很多坑，但是有一些坑是语言本身的不良设计造成的，知道当然好，但也不要因此沾沾自喜。但是对于一个写惯了 C++、Java 等后端静态语言的程序员来说，JavaScript 异步编程、函数成为一等公民等等这些颠覆以往编程思维模式的特性，才是学习这门新语言一个收获颇丰的地方（我们会在下一章介绍）。换言之，我们要写 JavaScript，就要写真正的 JavaScript——而不是写 Java，再按字面翻译成 JavaScript。

3. 过于关注具体实现逻辑，忽略了对于设计的思考和权衡。

我看到很多程序员朋友都热衷于研究源码，看完一个库，再看下一个，有些人甚至以读过源码的数量为荣，张口闭口就是“你读过多少源码”。

阅读源码当然是一件好事，但是请不要认为研究源码的目的只是“知道怎么实现”。毕竟，每个人的时间精力都是有限的，读源码尤其耗时。既然要读，就要读得有所收获，而不是凭空指望“读书百遍，其义自见”。比如说，阅读的时候，要抓住主干，去思考里面的设计思路，也就是所谓的“代码骨干”，忽略那些次要的边边角角。

4. 直接学习模式和思想，脱离了具体实践。

我已经介绍了模式和思想学习的好处，但是对于那些抽象的理论和概念，在没有实践的基础上，是很难真正“消化吸收”的。最好的方法是动手做一做，如果时间有限，至少也要阅读和了解它们都被哪些具体技术采用怎样的方式实现了。

基础知识和能力

下面我来说说第二个思路，基础知识和能力。

即便在同一级的技术上，我们的学习也应该是有轻重缓急的。我们的项目中需要的技术，当然要学，但是我们心里需要清楚那些所谓的“基础”。生活不可能完美，工作也是，工作中学当然很好，但**很遗憾的是，有很多“基础”光靠工作中学是很难有较大进步的。**比方说，算法和数据结构。

我认为，数据结构、算法、网络等等这些，都是基础知识，如果工作中的强化不够，工作以外的学习和训练还是需要的。如果你的基础不够扎实，特别是“非科班”出身的话，它就更加重要了。比较好的一点是，这些相对于“设计能力”“问题解决能力”等等来说，还是要好学很多。

除了基础知识，还有基础能力。最重要的，就是编程能力。值得庆幸的是，我相信大部分程序员都会在工作中有足够的编程时间，因此，只要专注、带着思考去写代码，认真对待代码评审，编程能力是会随着经验的增加而逐渐提升的。

最后，你正在学习的这个专栏，涉及了大量的全栈技术相关的知识，但是从内容设计上你应该可以看出，我希望能更多地介绍其中的“套路”，也就是上图中的模式和思想，有些是同类技术所共同有的（比如前几讲介绍过的 IoC），有些则是跨技术类型，但却依然共有的（比如幂等性和安全性）。

总结思考

今天的特别放送，我结合了自己的积累，介绍了我们该怎样做出选择，来应对这个“学不过来”的话题，特别是介绍了一些常见的学习误区，希望你能从我的答案中有所收获。

今天就不放特定的思考题了，因为我觉得，这一类主观性特别强的话题，很容易有不同的认识，不需要拿问题来引导思考方向了，不加限定地思考会更好。欢迎你在下面的留言区谈谈自己的看法，我们一起讨论。

扩展阅读

有一篇文章，Stop Learning Frameworks，观点比较朴实，也比较偏激，在程序员群体内掀起了轩然大波，有朋友翻译了，译文在这里。

关于传统关系数据库技术淘汰的事情，以前我写过一点小体会，可供参考。

上一篇　12 | 唯有套路得人心：谈谈Java EE的那些模式

下一篇　14 | 别有洞天：从后端到前端

精选留言 (8)

写留言

tt
2019-10-09

说到心坎里去了，坚持正确得做事。

展开

3

jxs1211

2019-10-09

1、python这样的脚本语言，是否没有对应平台可以学习，如果是，那么应该如何学习这门语言，如果用做web开发，是否只能去找对应的库来学习就可以了，还有其它需要关注的吗
2、SQL是一门半衰期很长的语言，学习sql也要是很有价值的，可以归为是语言和平台吧，是这样的吗

展开

1 1

亚东
2019-10-09

老师讲到沟通能力真的蛮重要的。我最近换了工作，同事的一些不专业的行为老是把我激怒。我感觉很难受，分分钟想离职。但是又感觉工作就是跟各种人合作，要是过不了这个坎也不行。

1

許敲敲
2019-10-09

数据结构 算法 网络 这种基础要学到什么程度呢?

展开

1

赖新宇
2019-10-12

老师，我现在作为一个前端，怎么可以找到全栈的工作，好多公司都要求相关经验，这个事情把我搞得有点头疼

展开

作者回复: 我觉得先“吃饱”，再“吃好”，找一份理想工作总不是那么容易的。如果你能拿到理想的前端的 offer，那就去做吧，在工作中再慢慢积累经验和寻找项目和团队的机会。

anginiit
2019-10-11

老师那句 光靠工作中学习是远远不够的 说得太对了，几年工作下来，体会很深，编程能力提高很快，几年基本就到顶点，但数据结构算法网络 原理等深一些的东西还是模糊的很。

没带就是没写
2019-10-11

为啥说语言平台比语言稳定呢？如果java没有热度了，没人用了，那么jvm平台就没人用了啊，存亡齿寒的关系。老师的意思是说语言的版本，比如java7 8 9 10迭代的比jvm快吗？

作者回复: 这是个泛指，其中一个原因是，你说的这个依赖关系，其实是单向的，比如说，JVM 是支持许多编程语言的，可不只有 Java 一种啊：https://en.wikipedia.org/wiki/List_of_JVM_languages

leslie
2019-10-11

其实任何都是一个过程吧：其实更多的是根据现状去补充吧；其实自己说不上自己的选择是对还是错，不过确实全栈和DevOps两门课的学习补了我的典型短板，现在又在适当的扩展去强化算法-和Nosql相关。

明确自己的主线然后不断的添砖加瓦吧：就像刘超老师在教授linux的系统的同时自己还在学徐老师的计算机原理然后补充到自己的课程中，明白自己的核心主线/主体然后强化...

展开

作者回复: 👍