如何确定自己是否适合做程序员？

成为一名好的程序员，势必要经历一段时间的学习和探索，下面的内容能够帮助你尽快找准未来方向——

一、程序员成长的必备条件

程序员的工作，说到底就是要编出满足需求的程序，如果这项工作你做得好，你就发展得好，所以，程序员的成长，都是围绕着「编出满足需求的程序」来展开的。

第一，程序员要了解工作的终极目标是满足需求。

很多程序员在这一条上就走错方向了，他们误以为，程序员的价值就在于把技术玩得神乎其技，于是，他们脑子里想的，是如何应用各种时髦、炫酷、看起来很牛的技术，而完全忽略了工作的最根本目的是要满足需求。

很不幸，这种程序员还不在少数，而且因为这种想法表面上看起来还挺迷惑人，所以很有市场，我甚至听过一个初创公司的 CTO 这样说：「如果程序员不用最新最牛的技术，那还做这行干吗？」他们在工作中就是把各种最新的时髦技术都用上，不出意外，这家初创公司已经倒闭了。

如果你能够意识到，程序员的工作是要满足需求，你就已经强过了大部分二吊子的程序员，因为你看问题的出发点就会不一样。在做技术选择的时候，你就会更可能做出正确的选择。

有的程序员会说，需求满足不满足，那是公司的事情，管他呢，我只要学习到最新最酷的技术，公司倒闭了好歹我还能找到下一份工作。且不说这样的想法很自私，从「学习」的角度说，也是错的。我接下来要说下一个程序员成长的必备条件——学习能力。

第二，程序员成长必须要有学习能力。

每个程序员都知道，这个行业有浩如烟海的知识量，且不说层出不穷的编程语言和框架，光是各种概念和风潮就已经让人应接不暇了。这个行业的特点就是技术发展快，没几年就会有一次革命性变化，几年前微服务还只是一个最佳实践的候选，现在就是开发大型后端应用的标准配置；几年前整个行业都在说大数据，这几年整个行业都在说人工智能……快速发展的行业，带来了快速增长的知识和技术。

那么，我们是应该展现我们的学习能力，把这些都学了吗？

当然不是！

学习能力的一个很重要组成部分，是知道「哪些需要学，哪些不需要学」，不做区分，什么时髦学什么，别人鼓吹什么学什么，那就会贪多嚼不烂，最后样样都稀松。

为什么我在前面反复强调，程序员应该首先明确自己的领域方向，还有确定自己会在什么类型的公司工作呢？因为这些选择将决定什么样的知识对你是最有价值的。

假设你确定自己的发展方向是移动端开发，服务的公司定位是初创型公司，那么，一个新的手机操作系统出现，你就应该更加关注，相对地，一个新的机器学习算法，你就没必要花太多时间了解；一个快速开发模型出现，你值得关注，相对地，一个超重量级的软件管理流程，你知道有这么回事就足够了。

把学习精力放在和你的方向相关的领域上，你才能获得最佳的投入产出比，当然，并不是说对和自己领域无关的东西完全不管不问，从扩大自己知识面的角度来说，你需要了解这些知识点，但是你不应该被这些东西分散有限的时间。

面对新技术变革的挑战，抓住自己专属的领域，伤其十指不如断其一指，先做到一个领域的专家水平。

第三，程序员发展要和团队发展联系起来。

这个行业，没有一个人是可以单打独斗的，你可能觉得自己可以单枪匹马完成一个项目，但是，你可以闭上眼睛思考一下，如果你的工作中少了一些同事的支持，你是否真的可以完成？

如果没有产品经理给你组织明确的需求，你是不是无法开始编程？

如果没有项目经理给你协调各个部门之间的进度，你是不是无法获得对应部门的支持？

如果没有测试人员给你做测试，你是不是也没有信心让程序上线?

如果没有其他程序员给你做代码审核，你是不是也没发现潜藏的代码缺陷?

如果你对上面的问题全都不以为然，觉得你一个人能够独立搞定所有的事情，你真的没必要来读这本书，恭喜你，你已经可以成为一个非常独立的自由职业者了。

不过，对于绝大部分程序员，都应该正视这样一个现实——你不是一个人在战斗，你是在一个团队中工作。

所以，不光要关注自己的成长，也要关注团队的成长，当团队出现问题的时候，要想办法解决，当队友遇到困难的时候，要帮助他们解决。

如果你的团队发展得不好，你一个人就是有孙悟空的神通，也无法按期把满足需求的产品推出；如果团队发展得好，你哪怕只是像沙和尚一样跟着取经团队走，最后也能修成正果。

小结一下，程序员成长的必备条件，根本上就是要明确目的是提供满足需求的程序，为此，要明白满足需求最重要，要正确发挥自己的学习能力，把自己的发展和团队发展关联起来。

前面我们已经介绍了程序员的成长，不过，这是基于 3 年以内工作时间的前提，接下来，我们要把眼光放长远一点，来介绍一下在一个更长时间里程序员如何规划自己的职业生涯。

二、如何确定自己的成长目标?

为了达到成长的目的，必须有针对性地进行训练，首先要确定自己针对的目标。

请先让自己回答下面的这些问题：

5 年内，我希望自己对这个领域的了解达到多少？10%、30%、50% 还是 100%?

5 年内，我希望自己在这个领域能够影响到多少人？3 人、10 人、100 人还是 1000 人?

这时候可能就有人看不下去了，他们要喊：不要这些虚的，你就直接告诉我，5 年之后怎么让我的年薪达到 XX 万元，其他的都不要说。

抱歉，作者只懂提高能力和影响力，不能给你开工资，到底怎么达到这些年薪，在提高能力和影响力之后都有可能，如果没有能力和影响力的提升，单纯的空想是没有意义的。

上面说到，关注于对领域知识的了解，代表深度，自己能够影响多少人，代表广度。只有在深度和广度上都有投入，才能在职业生涯上发展得顺利。

这个世界我们无法把握，但是至少能够把握自己，通过提高自己的能力来实现自己的目标，人的能力可以分为「硬技能」和「软技能」两个方面。

「硬技能」指的是那些外在的、可以量化衡量的技能，我们在学校里学到的大部分课程教育的都是硬技能，因为这些技能可以通过考试分数来衡量。对于程序员，编程能力就是最重要的硬技能，一个程序员一天能够开发一个程序模块，就比需要一周才开发同样模块的程序员硬技能要强，可见，硬技能是很容易量化衡量的。

「软技能」指那些内在的、感性的、难以量化衡量的技能，比如专业精神、人际交往能力、领导和管理能力，我们没法给一个人的软技能打分，因为没有这样一个量化衡量的方法。

接下来，我们分别介绍如何提高「硬技能」和「软技能」。

三、如何培养自己的硬技能?

硬技能，对于程序员就是创造软件相关的专业知识和技能，培养这方面知识，方法其实很简单，无论提高任何硬技能，请按照这三个步骤来做：

第一步，确定目标，定义清楚测量自己能力的方法；

第二步，学习对应知识和技能；

第三步，给自己一个测试，看是否达到了目标，如果达到，你就完成了，如果没有，回到第一步继续。

因为硬技能是可以衡量的，所以都可以用这种套路来提高自己。举个例子，你想要学 习 Python 语言，你先确定这个月的目标是要能够独立编写一个游戏，然后你开始学习，每过一

周，你就尝试用 Python 编写一个小游戏，如果能够完成，代表你的目标达到了，否则，你需要继续学习。

提高硬技能的方法看似很简单，但是要做到并不容易，我在知乎上也被无数次问这样的问题：「我看了 XXX 这本书，看到第 X 章的时候就看不懂了啊！」每次我都试着去了解这些程序员的困难，最后发现问题根源都是——基础差。

IT 这个行业每一个知识点都是建立在很多其他知识点上的，例如，如果你不理解时间复杂度，你就不会理解为什么数据库需要索引，而要理解时间复杂度，你又必须理解算法和数据结构基础……依此类推，实践中一个小的功能点，要求你懂很多基础性知识。

要获得扎实的基础，最直接的就是接受过计算机专业的本科教育。有很多网上的朋友问我，如果没有上过计算机本科怎么办？很抱歉，我真没有什么好方法，如果我能够有一个捷径可以代替计算机本科教育，那我也太牛了，计算机本科教育如果能够这么轻松被替换，也就没有存在价值了，对吧？对于想要从事这个行业的小朋友，不要被一时的快感或者利益蒙蔽眼睛，不要在高中毕业甚至初中毕业的情况下就去参加工作，磨刀不误砍柴工，请一定至少考一个计算机专业的大学本科，这样你才能有一个扎实的基础。

那只有初中或者高中学历的人就不能当程序员吗？当然不是，实际上，我见过一些高中毕业当程序员的，他们有的很努力，也工作得不错，但是，他们经历的辛酸是别人难以想象的，他们的发展天花板却是谁都看得到的。

对于已经失去接受计算机专业本科教育的机会的朋友，你也不要灰心，补救的方法也很简单，你找到任何一家你心仪的计算机专业院校，打听清楚他们的专业课程，然后按照他们的课程来学习，这当然不能和接受全日制教育那么好，但是这是最好的补救办法。我说过，没有类似《九阴真经》《九阳真经》这样的秘籍，你看完之后就能打通任督二脉，如果你非要说有，那就是计算机本科的教材，没有其他捷径。

有了扎实的计算机基础之后，接下来就要多实践，这个行业是一个建立在实践上的行业，优秀的程序员只有在实践中才能培养起来。简单说，就是少做些无病呻吟的鬼扯，多一些实际创造价值的贡献。

提出问题，谁都会；解决问题，就不是谁都能够做到的了。

根据我的观察，职业发展良好的程序员，都有一个特点，那就是他们是擅长解决问题的人，而不是抛出问题让别人解决的人。

我举一个非常具体的例子，团队里代码质量很差，三天两头线上出 bug，搞得开发团队一方面有新功能的开发压力，一方面还要抽时间去修线上 bug，这时候团队肯定会有怨言的。

这时候，只会提出问题的人，只会抱怨：「项目一团糟，都不知道怎么办了。」「大家都没有干劲，这样下去肯定要散。」「bug 越改越多，都不知道为什么还要改。」

好的问题当然是解决问题的一半，但是，如果单纯地提出问题，甚至是以消极态度提出问题，那就只是添乱。

能够提出解决办法的人，会说：「我们把团队分为两组，一组集中精力开发新功能，另一组专心修复线上的 bug。」「我们的代码分支管理可以踩进 gitflow 的流程，避免代码提价混乱。」「我们需要研究分析一下重复出现的 bug 的类别，针对最多出现的类别进行改进。」

上面的「解决问题的方法」只是一些例子，重点就是，这些方法是针对实际问题的对策，能够帮助团队渡过难关。能够提出解决办法并且实施，不光是对自己能力的锻炼，更重要的是，真正解决问题的人能够获得组织的青睐，获得更多的晋升机会。

在确定了学习方法、学习态度之后，「硬技能」的培养就是要花时间去反复训练，没有任何捷径，下面是一些直接的提高硬技能的方法：

1. 当你加入一个新项目时，从 fix 一个 bug 入手开始工作，因为你的同事未必有时间帮你过一遍代码，而你找得到的文档往往都是过时或者错误的，而 fix 一个 bug 是最快速的能让你熟悉一个项目代码的方法。

2. 当你学习一门新的语言时，用这种语言编一个你每天都会用的程序，比如一个「待办任务列表」（Todo List），或者做一个弹珠小游戏，写这样的程序的好处是需求非常清晰，因为你已经见过类似功能无数次了，你不用纠结需求（虽然实际工作中你依然要纠结），你可以心无旁骛地应用你的编程语言。

3. 如果有时间，你写过的程序，都重新写一遍，你会发现第二次写的会和第一次大不一样，越是不一样，越说明你在这期间

进步大。

4. 每天花一两个小时来学习工作之外的技术，如果工作紧张腾不出这个时间，那就每周末腾出几个小时来学习。

5. 不要浪费时间和别人争论语言或者框架的优劣，把争论的时间用在实际上手编程上，你不会后悔。

6. 无论如何，给自己定一个目标，每个月不能生产少于 1000 行代码，这里说的是新写的代码，不是修改别人的代码。

7. 当你积累了多年的工作经验，对某一个领域非常熟悉的时候，可以考虑扩展自己的广度，成为其他领域的专家，这是让你扩展影响力的必经之路。

四、如何培养自己的软技能

相对于硬技能，软技能在程序员职业发展中的作用更大，为什么这么说呢？因为，硬技能的学习材料实在太饱和了，各种语言、工具、框架的介绍可以说是长篇累牍地重复，只要你有足够的时间和耐心，学好一个硬技能不成问题；与硬技能相反，软技能的培养是一个很冷门的领域，更多的是靠程序员在工作和生活中自己去领悟，没有人给予教导，偶尔能够开悟掌握软技能的程序员，就会如虎添翼。更重要的一点，这个行业里需要的是人与人之间的协作，一个人的硬技能只能做好一个人的工作，处理人与人之间关系的软技能却能让一个团队产生合力，达到 1+1>2 的效果。

我见到过很多程序员，他们觉得技术是一切，有技术就是牛，忽视了软技能的培养，这就很影响他们的成长，表现在这些方面：

- 只能自己做工作，不能教别人做工作，最后累死自己；
- 费好大劲做出的成果，但是表达不出来，团队和领导看不出这个成果有多重要；
- 唯技术至上，看不起技术刚入门的同事，甚至出口不逊，造成矛盾。

这样的例子数不胜数，相反，软技能过硬的程序员，即使技术上并不是最强的，职业发展却会顺风顺水，真的，最后人和人之间差距，更多的是软技能之间的差距。

软技能是一个比较模糊的范围，可以包含但是不受限于下面这些能力：

1. 表达能力；

2. 自我营销能力；

3. 与他人沟通能力；

4. 说服他人的能力；

5. 组织能力；

6. 培训指导他人的能力；

……

如果要总结上面这些「软技能」的共同之处，那就是，这些技能都是和「人」打交道，而不是和电脑、逻辑、程序打交道，所以，「软技能」的核心，就是和人打交道的能力。

一说到和人打交道，就不得不提「情商」这个概念。情商高的人软技能就一定高吗？其实未必，一个人可以情商很高，但是一到演讲的时候就情绪紧张，情商更多的是一个人的为人处世态度，并不完全代表技能，有一些和人打交道的技能，比如演讲能力，依然需要训练才能获得。

所以，情商高只是基础，软技能依然需要训练。

程序员这个职业，因为长时间和电脑打交道，而且和电脑打交道就能够获得工资，所以往往会让人产生错觉，以为不需要培养和人打交道的能力。

其实啊，软件开发还是和人打交道的艺术，且不说产品经理、项目经理和程序员之间的沟通是和人打交道，程序员之间的交流也是和人打交道。前面我说过，要成为更高阶的程序员，你的「影响力」就要扩大，而你所影响的，就是人类啊，所以和人打交道的能力是必修课。

在这里，我们不需要把「软技能」神秘化，我们可以明确程序员需要提高的软技能方面，请努力做到下面这些方面。

1. 每一个知识点，能够解释得不懂计算机的人也能够理解，你不一定需要让对方和你一样精通，但是你可以用比喻的方法让对方理解。

2. 你能够通过演讲，把自己的理念传播出去，为什么要这么做、怎么做，你能够通过半小时到一小时的演讲说清楚。

3. 你能够让团队成员和你交流如沐春风，可以有观点不一样，但是不要红脸吵架。在和人交流的过程中，请记住卡耐基教诲我们的一句话：「每一个人都希望自己是重要的。」记住这一点，让每一个人都感觉自己重要，你就可以做到每一个人和你的交流都很愉快。

如果你做到了上面三点，你的软技能已经强过绝大部分程序员，足够让你成为资深级别以上的程序员。

我在微软工作的时候，有一次一个工号在 100 之内的资深员工来给我们做培训，培训内容覆盖了硬技能和软技能。关于硬技能部分，参加培训的员工都没有任何问题，但是当他介绍软技能培养的时候，大家就觉得难以把握了，于是，我就问了这位资深前辈一个很具体的问题：「如果我只能做一件事来提高自己的软技能，那这一件事应该是什么？」

这位前辈说：「你觉得你日常所能接触到的人里，谁的软技能最高？」

我想了想，说：「我们部门的架构师吧。」

这位前辈又问我：「为什么你觉得他的软技能最高？」

我说：「因为他情商高，说话风趣，再复杂痛苦的事情他也能理清楚，所有人都说和他说话总是很愉快。」

然后这位前辈就给出建议：「那你就和这个架构师约一个两个月以后的一对一会议，一个人是肯定会接受两个月之后的会议的，然后你就通过和他的接触，学习他的做事方式，向他看齐。」

后来我按照这位的方式做了，发现这一招真的非常好使，当有了一个明确的榜样去学习之后，看似虚无缥缈的软技能也变得非常具体易学了。

可见，虽然「软技能」比较软，难以量化衡量，但是，也有一个对个人来说可以「量化」的培养方法，这个方法也非常简单：找到你接触范围内的「软技能」的榜样，然后努力让自己变成他那个样子。这个榜样，可能是一个口才很棒的领导，也能是一个情商超高的项目经理，重要的是你认可他就是你几年后想要成为的层次，你就可以和他多交流接触，看他平时都看些什么书，看他平时有什么编程习惯，看他怎么和别人交流，看他平时怎么处理问题，即使你只学会他的十之四五的本事，你很快会发现自己已改头换面，成为大不一样的人，这时候，你的眼界更高，就可以设立一个新的榜样，迈向一个新的台阶。