# 42讲技术停滞：如何更新

我们从开始学习程序，到工作十来年，中间可能会出现几次自我感觉技术停滞了。而在这个过程中，我们也会不断地学习很多新技能，但而后其中的不少也会被淘汰在时间的旅程中。

一方面，我们在不断地打磨、提升技能，去解决工作中的问题，但久而久之，就会发现技能的提升速度越来越慢，竟渐至停滞，感觉不到进步了。另一方面，程序员所处的这个行业，技术的变化很快，潮流此起彼伏，难免产生技能焦虑。

有时，我们会不免幻想要是学会什么屠龙之技，从此高枕无忧，该多好！但这终究只是幻想，哪里又有什么屠龙之技呢？那面对技术停滞，技能过时，又该如何保持更新，与时俱进？

## 技术停滞

技术停滞是如何发生的？

程序员，最重要的就是编程技能。每天的工作可能就是编程写代码，在早期还不够熟练时，你还能感觉到进步，这种进步就是从不熟练到熟练。进入熟练期以后，你可能感觉这项技能就提升得很慢，甚至一度停滞了。

单纯的编程实战其实并不能持续地提高一个人的编程技能，想想体育运动员，又有哪一个每天的日程就只是参加比赛。运动员平时都是在进行刻意地训练，而关于习得甚至精通一门技能，最著名的理论应该是“刻意练习”，如果非要在这份练习上加上一个期限，那就是：一万小时。

关于“刻意练习”，不少书或文章中都讲了很多案例来说明它的有效性，但总结起来关键就下面三点：

1. 只在“学习区”练习，练习时注意力必须高度集中。
2. 把训练的内容分成有针对性的小块，对每一个小块进行重复练习。
3. 在整个练习过程中，随时能获得有效的反馈。

刻意练习是为习得真正的技能所设计的，它和获取知识不同，知识就是那些你知道即为知之、不知即无知的东西，这可以通过读书获得。但技能是那些你以为你知道，但如果你没做过，就永远不会真的知道的事情。

在程序员足够熟练了之后，每天的这种编程实战型工作就不会再是处于 “学习区” 的练习了，而是进入了 “舒适区” 的自动完成。真正的职业竞技体育运动员每天的日常训练都是在 “学习区” 的刻意练习，到了上场比赛则是进入 “舒适区” 的自动完成。然而很多熟练程序员的日常工作则是在 “舒适区” 的自动完成，工作之外则是另一种 “舒适区” 的娱乐休闲。

停滞，就是这样发生的。

### 技能保养

感觉停滞的技能，如果工作依然需要它，其大的技术方向发展趋势依然明朗，那么这样的技能是值得好好保养，并继续提高的。而保养和提升技能的方法，“刻意练习” 依然是不二之选。

关于 “刻意练习”，有时我们即使一直保持在 “学习区” 的重复练习，却也可能感觉不到进步，这有可能是因为重复的次数和强度还不够。我曾经就犯过这个错：英语这门技能从毕业后就停滞了（可能还倒退了些）十年，在工作十年后我重启了学习掌握英语这门技能的练习，但刚开始阶段我完全低估了需要重复练习的次数和强度。

第一年，仅仅在每日的工作之余，我会花上大约一小时来进行听说读写的练习。但即使每日都能保障一小时的时间，一年下来也不过区区 300 多小时，更别提分散在听说读写四个分支上了。最后的结果可想而知，就是那一年结束后，并没有哪一项在让我感觉到一点点的进步。

在决策科学中有一个概念叫 “基础比率（Base Rate）”：

> 所谓基础比率，就是以前的人，做同样的事，做到的平均水平。

也就是说，如果别人做这件事需要那么长时间，基本上你也需要那么长时间，因为可能你没有那么特殊，只是每个人都会“觉得”自己是特殊的、例外的罢了。所以，当我调查了下学英语人群的基数和真正算是掌握并熟练运用这门技能的人数，以及他们所花费的时间，我就知道自己大大低估了需要重复练习的强度。

重复，是有针对性的强化练习，其本身是练习过程，而非练习内容。每一次的重复过程中都需要根据反馈进行有针对性的调整，以取得练习效果的进步。

而重复的刻意练习总是辛苦的，辛苦就是我们付出的技能保养成本。

### 技能开发

技能不仅仅会停滞，还有可能会过时。

就拿我来说，我这十多年编程生涯走过来，从早年的 Basic 语言，到后来的 C，再到后来为了做 C/S 架构的项目学习了 Delphi，之后 B/S 架构开始兴起，又开始写起了 JSP，转到 Java 上来。经历了如此多艰辛的学习路线，曾经掌握过不少技能，但如今除了 Java ，其他的都过时淘汰得差不多了。

旧技术过时了，肯定是因为有另一种新的技术来取代了它，我们只需定期保持跟踪，观察现有掌握的技术是否可能被新出现的技术所取代。一般来说，新旧技术的更替都是有一定周期和一个持续的过程的，这期间给了我们足够的时间来学习和开发基于新技术的新技能。

而针对不同的学习目标，采用的学习路线也会不同。

如果需要学习新技能来解决工作上的一个具体问题，那这样的目标更适合采用深度路线学习方式，这是解决特定问题的一种捷径，属于痛点驱动式方法，能让你快速排除障碍，解决问题，而非先找到相关书籍，从头读到尾，知道一个整体大概。

一般技术书籍的组织方式都是按主题的相关性来编排的，系统体系性很强，但却不是按你解决问题需要知道的内容来组织的。所以，技术书籍更适合于在你解决问题的过程中用来参考。完整地读技术书籍能增长你的知识，但却无法快速习得技能，并解

决你的问题。

反过来，另一种情况，面临一种新兴技术，比如，近年火热的人工智能与机器学习，你不是需要解决一个具体问题，而是要对这类新兴技术方向做评估、判断和决策。那么学习的方式就又完全不同，这时采用广度路线学习就更合适。

对如何开发一门新技能，《软技能》一书的作者曾在书中分享过他的一个十步学习法：

1. 了解全局
2. 确定范围
3. 定义目标
4. 寻找资源
5. 学习计划
6. 筛选资源
7. 开始学习，浅尝辄止
8. 动手操作，边玩边学
9. 全面掌握，学以致用
10. 乐为人师，融汇贯通

这个方法和我自己在实践中养成的习惯基本一致。在深度路线学习中，对全局、范围、目标的定向更聚焦，因此寻找、筛选的资源会更窄，学习计划的迭代期更短，很快就走完了前 6 步，并进入动手实践、反复迭代的过程中，直到把问题解决。

而在广度路线的学习中，前 6 步会花去大量的时间，因为这时你面临的问题其实是对新技术领域边界的探索。这就像以前玩《魔兽争霸》游戏中，先去把地图全开了，看清楚了全貌，后面再进军时就能选择最优路径，避免了瞎摸索。

这个类比中不太恰当的是，游戏中开地图实际挺简单的，但现实的技术领域中，地图全开基本是不太现实的，一是地图实在太大，二是地图本身也在演变中。只能说尽可能在前期的探索中，所开的地图范围覆盖更广至需要去解决的问题领域。

### 沉淀能力

技术也许会停滞，技能也可能会过时，但其中的能力却可以沉淀下来，应用于下一代的技能之上。

汉语中容易把能力和技能混为一谈，在英语中，技能对应的词是 Skill，而能力对应的是 Ability。**技能是你习得的一种工具，那么能力就是你运用工具的思考和行为方式**，它是你做成一件事并取得成果的品质。

程序员爱说自己是手艺人，靠手艺总能吃口饭。五百年前，鞋匠也是手艺人，但进入工业革命后，制鞋基本就由机器取代了。手工制鞋是一门技能，它的过时用了几百年时间，但如何做一双好鞋的能力是不会过时的，五百年后人们还是要穿鞋，还要求穿更好的鞋。这时鞋匠需要应对的变化是：换一种工具（现代流水线机器生产）来制作好鞋。而现代化的制鞋机器技术实际上还进一步放大了好鞋匠的能力，提升了他们的价值。

对程序员来说，程序技能的过时周期相比制鞋技能却要短得多，每过几年可能就大幅变化了，是需要定时更新的消耗品，而能力才是伴随技能新陈代谢，更新换代的固定资产。技能用熟练了就成了工具，熟练应用工具能快速解决已碰到过的老问题。而沉淀下来的能力，是为了应对并解决新问题，甚至为了解决新问题，可以去开发新的技能或创造新的工具。

那么程序员需要去沉淀哪些能力?

作为程序员最基本的自然是代码能力。能够写程序，只能算是技能过关吧，而能写好程序，才算具备了程序员的基本代码能力。代码能力的识别，最简单的方式就是维护一份公开可跟踪的记录，比如参与开源项目贡献，在 GitHub 上留下你的代码简历。

从程序员到架构师，“架构”显然不是一种技能，而是综合应用多种技能的能力。架构师也许不像在工程师阶段需要写大量代

码，但实际没有代码能力是做不了架构的。代码能力是架构能力的底层能力要求。但仅此一项能力却也远远不足，这里就先不展开了，后面会专门有一篇文章谈架构师能力模型这个主题。

除了技术能力，如果有可能可以适当跨出技术的边界，去了解下产品、管理、运营和传播等方面的能力。为什么呢？一方面，技术能力的提升总会到达平台期，增长变得缓慢，而了解学习一下其他方面的全新能力，可能会让你找到新的成长点，重新找回快速成长的感觉。

另一方面，个人很多综合能力的差别，有时就是要靠作品来体现的。完成作品需要有一些产品思维，需要自我规划与管理能力，而推广作品需要运营和传播能力。这些相关的能力，最终都会成为你核心能力体现——作品——的放大器。

如果你是一棵树，能力是根，技能是叶；春去秋来，时过境迁，技能过时，落叶归根；沉淀下来的能力，将如春风吹又生，新的技能自会发芽。

而这一切的能力与技能之母，又叫元能力，自当是学习能力。

虽有俗语说：“技多不压身”，但实际很多技能是有保养成本的，编程技能就是一种，特别是和特定技术有关的编程技能。所以，同时保养很多技能是不太合理和现实的，更优化的选择是：**持续保养主要的生存技能，合理开发辅助技能，形成自己独有的技能组合，沉淀能力模型，发展能力矩阵**。

当时代发展，某些技能会过时，但能力矩阵不会过时，它当与时俱进；永不会有停滞的时候，它总是在进化。而对于过时的技能，除了既往不恋，我们还能做什么呢？

技能如剑，金庸老爷子笔下有一 “剑魔”，一生用剑经历 “利剑无意”“软剑无常”“重剑无锋”“木剑无俦”“无剑无招”，最终剑已埋冢，人却求败。

如今你的哪些技能已过时？又沉淀下来怎样的能力呢？


精选留言

李正阳Lee

技能和能力就像金庸作品中描写的招式和内功。小的时候看武侠剧，觉得招式比内功重要，不理解很多大师为什么需要闭关修

炼内功。毕业后，工作经验和学习场景增多，逐渐认认识到内功的重要性，改变了对招式和内功的看法。

学习武功招式时内功是基础，内功影响着学习的速度和对招式的驾驭程度。《天龙八部》有个奇怪的医生，人称薛神医，他每次为别人治病要对方把传家的武功教给他。按理说凭他多年的行医经验，功力在武林中应该是数一数二的，事实是他的功力远不如他的医术出名。导致这种情况，内功不足是一个重要的原因，他学过的招式倒是不少，但凭自己内功根本驾驭不住，可见内功的主要性。

什么是内功呢？在工作方面，我认为学习能力、专注力、执行力、认知能力都算是内功。

2018-11-07 11:42

作者回复

内功的内涵延展开来也可以很广

2018-11-07 20:19

third

核心感受：在变化的世界中，努力选择不变的东西

1，技术停滞
单纯的编程实战并不能提升编程技能，只是提升熟练度。
技能的停滞是，因为长期处在舒适区

解决方法是，刻意练习
1，只在学习区学习，（学习区，你不懂的，不知道的地方）
2，把训练内容拆解成有针对性的小块，然后进行重复练习
3，及时反馈并调整

2，技能保养
将停滞的技能进行保养，同样需要利用到刻意练习
同时注意基础比率
大家其实差不多，以前的人，做同样的事情，做到平均水平，花了多少时间，你差不多也一样。

3，技能开发
变化的世界，要不停学习新的技能
一般来说，技术的更迭是有一定周期的，给予了我们一些时间

针对不同的学习目标，做不同的学习路线
深度学习：解决一个具体问题

广度学习：对新技术方向做评估、判断和决策

练
《软技能》十步学习法
1，全方位了解
2，划定边界
3，找到你的目标
4，找你的学习资源
5，制定如何学习的计划
6，选择那些高效率的资源
7，各种尝试，慢慢学习
8，动手做，学着也玩着
9，掌控全面，开始在生活或工作中使用
10，喜欢传播这类知识，同时自己把道理悟出来

沉淀能力
技能的本质是工具
能力的本质是对工具的创造性使用（即对工具的认知和抽象）

程序员去沉淀能力
代码能力：GitHub留下你的代码简历

技术之外，拓展出别的能力。新能力会带来新的增长点，同时将你的影响圈变得复杂。

这需要有作品思维：
用产品思维创作作品
用运维能力进行运营和传播

总结
持续精进自己的保命技能
在此之外，开发一些发展技能
形成自己的技能连招

2018-12-15 12:53

helloworld
和我的看法一样，对于有长期发展前途的技术要保鲜，改扔掉的就扔掉
2018-12-13 07:47

godtrue
保持好奇之心，保持敬畏之心。
2018-11-17 22:25

作者回复

2018-11-18 14:44