hyperink

知名博主和程序员Jeff Atwood力作
Coding Horror精华文章首度结集出版
程序员必读

高效能程序员的修炼

Effective Programming: More Than Writing Code

软件开发远不只是写代码那么简单……

[美] Jeff Atwood 著　　陆其明 张健 译

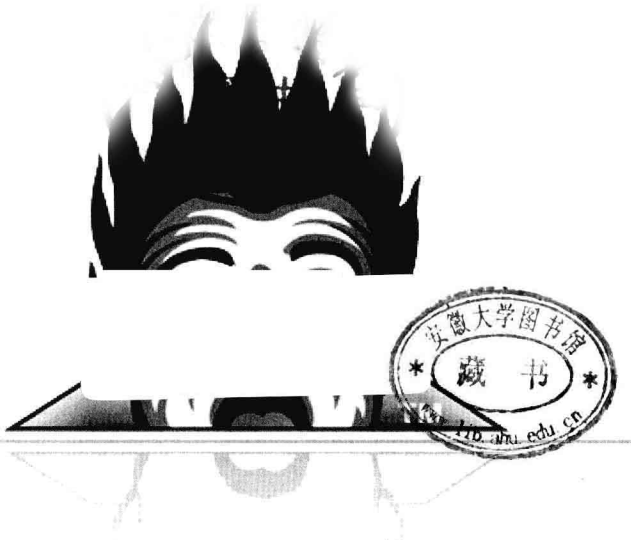
人民邮电出版社
POSTS & TELECOM PRESS

hyperink

高效能程序员的修炼

Effective Programming: More Than Writing Code

[美] Jeff Atwood 著　　陆其明 张健 译



人民邮电出版社
北京

图书在版编目（CIP）数据

高效能程序员的修炼 / (美) 阿特伍德 (Atwood, J.) 著 ; 陆其明, 张健译. -- 北京 : 人民邮电出版社, 2013.7
ISBN 978-7-115-31898-5

Ⅰ. ①高… Ⅱ. ①阿… ②陆… ③张… Ⅲ. ①程序设计 Ⅳ. ①TP311.1

中国版本图书馆CIP数据核字(2013)第099942号

版权声明

◆ 著 [美] Jeff Atwood
译 陆其明 张 健
责任编辑 陈冀康
责任印制 程彦红 杨林杰

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
大厂聚鑫印刷有限责任公司印刷

◆ 开本：700×1000 1/16
印张：17.5
字数：293 千字 2013 年 7 月第 1 版
印数：1 – 3 500 册 2013 年 7 月河北第 1 次印刷

著作权合同登记号 图字：01-2012-6320 号

定价：49.00 元

读者服务热线：(010)67132692 印装质量热线：(010)67129223
反盗版热线：(010)67171154

内 容 提 要

Jeff Atwood 于 2004 年创办 Coding Horror 博客（http://www.codinghorror.com），记录其在软件开发经历中的所思所想、点点滴滴。时至今日，该博客每天都有近 10 万人次的访问量，读者纷纷参与评论，各种观点与智慧在那里不断激情碰撞。

本书是 Coding Horror 博客中精华文章的集合。全书分为 12 章，涉及迈入职业门槛、高效能编程、应聘和招聘、团队协作、高效工作环境、用户体验、安全问题、测试、社区管理、营销广告、人生思考等话题。作者选取的话题，无一不是程序员职业生涯中的痛点。其中，“程序员的八种境界”、“程序员的《权利法案》”、“结交混世魔猴”等文章早已脍炙人口，在程序员圈子里广为流传。

本书的写作风格风趣幽默，且充满理解和关怀，适合从新手到老手的各个阶段的程序员阅读，也适合即将成为程序员的计算机和相关专业的学生阅读。本书能够帮助读者更多地关注程序员职业生涯中的人性和人文因素，成长为真正的高效能的程序员。

作 者 简 介

Jeff Atwood

我叫 Jeff Atwood。我和妻子住在加州的伯克利。我养了两只猫、3 个孩子，家里还有一大堆电脑。20 世纪 80 年代，我拥有了第一台微电脑（德州仪器的 TI-99/4a），并开始了我的软件开发之路，参与了微软 BASIC 产品的大量实施工作。20 世纪 90 年代早期，伴随着 Visual Basic 3.0 和 Windows 3.1，我的工作转移到了个人电脑上，尽管我同时也使用 Delphi 早期的几个版本，花了大量的时间写了很多 Pascal 代码。现在我对 VB.NET 或 C#用得都比较顺手，尽管 C#区分大小写的特性着实让我感到有点别扭。目前我还在学习 Ruby。

我自认为是一个相当有经验的 Web 软件开发者，对软件开发过程中的人文因素特别感兴趣。这从我给其他开发者推荐的读物清单上能够体现出来[①]。电脑真是让人神魂颠倒的东西，但对于使用它们的人来说，它们很大程度上是一面镜子。如果要研究软件开发艺术，单纯研究代码是不够的，必须同时研究写下那些代码的人。

2004 年，我创办了 Coding Horror 博客。我未曾想过多地引人注目，但它确实改变了我的生活。后来发生的很多事情都或多或少跟这个博客有关。

2005 年，我在 Vertigo 软件公司找到了一份理想的工作，并因此搬迁到了加利福尼亚。如果你感兴趣的话，可以去网上看一看我以前的办公室。

2008 年，我决定自己创业。我和 Joel Spolsky 联合创办了 stackoverflow.com 网站，并把它最终做成了 Stack Exchange 问答网络引擎。在互联网网站规模排名中，Stack Exchange 现在位于前 150 之列。

2012 年初，我做了另外一个决定：离开 Stack Exchange。孩子们在渐渐长大，他们需要我；同时我也在思考我下一步的方向……

---

① 参见本书附录《程序员必读之书》。

译 者 简 介

陆其明，曾是一名 C++程序员，多年从事流媒体应用与软件开发。自 2004 年起，连任 4 届微软 MVP（最有价值专家）。现任乐威软件（上海）有限公司研发部高级经理，主要负责 OTT 电影分发系统的移动客户端软件开发。在软件行业辛勤耕耘十余载，在团队建设、流程管理、项目管理等方面积累了丰富的经验。已经出版的著作有《DirectShow 开发指南》、《DirectShow 实务精选》、《Windows Media 编程导向》、《脚本驱动的应用软件开发方法与实践》，译作有《代码之道》。新浪微博：豆巴陆其明。

张健，2000 年毕业于南京大学。多年来专注于嵌入式系统设计工作，涉及 SOC 架构、内核及设备驱动、应用系统构建等多个领域。在杭州士兰微电子公司任职期间，主管嵌入式软件研发工作，有丰富的团队及项目管理经验，并于 2008 年获得 PMP 认证。现居悉尼，在 Open Access 公司任职高级软件工程师。

译 者 序

出版社的冀康一开始来找我谈翻译这本书的时候，我的第一反应是：这兄弟真是不知道我现在有多忙！我每天要处理 200 多封邮件；在资源有限的情况下经常要同时带 6～7 个项目，而且每个项目的交付计划都很紧，压力很大；每天起码工作 12 个小时，有时候还要熬夜跟美国同事开会；周六基本上也是工作状态……我哪里还有空来翻译书？！

后来，当我了解到这本书的作者是 Stack Overflow 网站的创始人 Jeff Atwood，还有书的内容实际上就是从作者的博客网站 Coding Horror 精选而来的文章时，我开始有些心动了。Jeff 的成就是值得尊敬的！这本书的主题和风格也是我喜欢的：一篇一个话题，针对性很强，讲的都是我们程序员自己的事。据说篇篇都很受人关注，引来读者评论无数。我的好奇心愈加强烈了：Jeff 都说了些什么呢？好惭愧，我以前居然都没读过他的文章！

做吧！我一定能从 Jeff 的这本书里学到不少东西。而且，既然这些东西这么好，我一定要把它们介绍给更多的中国读者。时间哪里来？挤呗！时间是挤出来的！于是，我把所有的零碎时间都利用上了：陪儿子上早教课时在教室外守候的时间、午后休息的时间、晚上睡觉之前、坐地铁时、外出办事等候的时间甚至去美国出差和去泰国度假的途中……这本书就是这么“挤”出来的！

再说说这本书的内容吧。其实得先说说 Jeff 其人。他无疑是一位杰出的程序员。那是因为他写代码很厉害吗？其实，他在第 1 章就指出了，“成为一名杰出的程序员其实跟写代码没有太大的关系。做程序员确实需要一些技术能力，当然，还要有坚韧不拔的精神。但除此之外，更重要的还是要有良好的沟通技巧”。所以，这本书不是讲某种特定的编程语言的，也没有涉及太多具体的代码问题。因为 Jeff 认为，“如果要研究软件开发艺术，单纯研究代码是不够的”；他更多关注的是软件开发过程中的人文因素。因此，本书涵盖的主题非常广泛，包括对程序员的素养、做事方法、价值观的探讨，也谈到了编程风格、软件测试、团队合作、用户体验、社区管理、网络安全、市场营销等方面的问题。书中的很多观点并不是 Jeff 独创的，但他

旁征博引、博采众长，把许多很棒的东西汇聚在一起，给读者奉上了一顿饕餮大餐。

Jeff 给我们指明了“程序员的八种境界”。他还在第 2 章中指出：作为程序员，大家不能只顾着埋头写代码。任何能让你成为一名更好的程序员的事情都值得去关注，别担心你会因此而分心、少写了 N 行代码，因为磨刀不误砍柴工。

Jeff 是我们学习的榜样。这本书特别适合初入行的程序员阅读。对于已经工作过几年但碰到了发展瓶颈而倍感迷茫的老程序员来说，这也是一本极好的读物。我们不能总是低着头走路，而要时时抬起头看看路。希望 Jeff 的这本书能为你指点迷津。

本书的前半部分由张健翻译，后半部分由我翻译。张健是我大学时候隔壁寝室的兄弟。谁也不曾料到，十几年后的今天我们会有这样的一次合作。张同学平时工作也很忙，久居海外，对中文表达已有些许生疏。他翻译的部分我帮忙做了些润色，以保证全书有较为一致的风格。我们还在翻译的过程中添加了大量的“译者注”，这些背景知识往往都很生动，有助于读者更全面地理解原书的内容，也是本书的看点之一。

翻译工作历时半年，译稿几经修改——翻译一遍，再修改至少三遍——交稿时我虽已尽心竭力而问心无愧，但我相信书中还有很多地方可以改进。努力永无止境！感谢冀康给我这次机会，让我翻译这么棒的一本书。感谢 Jeff 在翻译过程中给予的支持，他对我的问题回复得很及时，还提供了很多额外的信息。同时也要感谢我的爱人谭洁红，她帮忙做了些美工，还审阅了部分译稿，在文字表达方面给我提出了很多建议。这么多感激之余，其实我还有些愧疚，因为半年来我很少有时间去陪我的儿子豆豆。令我感到欣慰的是，当我向他解释爸爸在忙什么时，他能够理解，然后一个人乖乖地去睡觉。而他以前是要在床上听我给他讲完一个故事后才入睡的……

写书或翻译书都很辛苦！我不得不牺牲很多业余时间，也少了跟家人交流的机会。每次我都对自己说，“这是最后一次了！”是吗？也许吧……关于本书的内容或者翻译上的问题，需要跟我交流的话，请爱特我吧。我的新浪微博：豆巴陆其明。

2013.4.1

陆其明

目　录

第 1 章
Chapter 1

入门须知

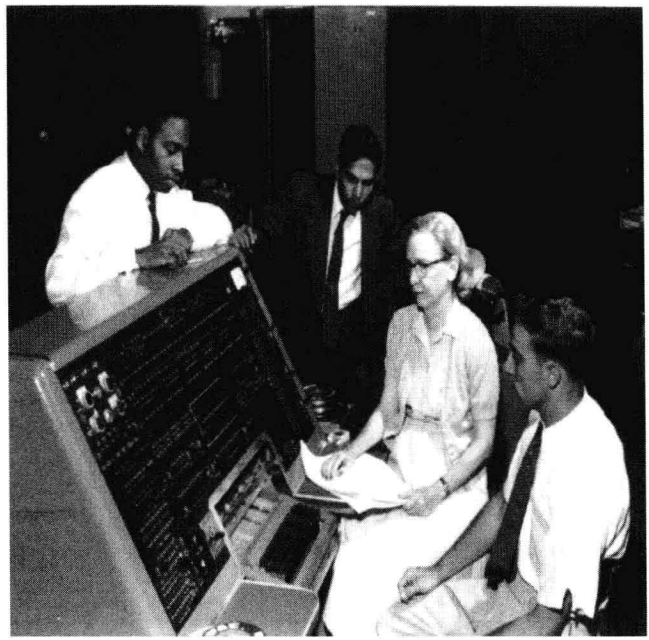
你想成为一个程序员

“不是所有程序员在他们的职业生涯中都渴望相同的东西。但是，思考一个程序员在 10 年、20 年、30 年甚至他一生的时间里所能取得的成就，还是挺有意义的！”

平心而论，我觉得大多数想要学习编程的人都是被一种盲目的信仰刺激所致，他们认为代码都是善良的、可以信手拈来。其实，你需要了解它的另一面，那就是在代码世界里还有很多悬而未决的问题，并且没有银弹这样的神秘武器去解决。如果在全面了解了利弊之后，你还是想学习编程，那就用尽一切办法开始学吧！如果在了解了那些编程的阴暗面之后，你轻易就动摇了，那么你不应该学习编程——这世上还有很多其他的事情值得你花时间去学习，它们无疑比编程更加有益并且易于实践。你可以遵循 Michael Lopp 的教诲，学着做个更好的沟通者。你也可以追随 Gina Trapani，学习怎样才能提出更好的解决方案。依我的经验，编程只是整个解决方案中极其微小的一个部分。为何非得在这棵树上吊死呢？

在最早期的计算机上，每一个用户都必须成为程序员，因为那个时代没有任何应用软件。如果你想让计算机做点什么，你就得自己写程序。年代稍近一些的计算机，在启动以后就会直接进入 BASIC 解释器，它还闪动着一个略显友好的光标。

在我看来，软件开发的整个历程，就是程序员们耗尽毕生精力去编写代码，以使其他人能从代码编写工作中解脱出来（他们就不必成为程序员了），从而可以很方便地使用计算机来做他们真正需要做的事情的一个过程。因此，宣扬“每个人都需要知道如何去编程”，我认为是一种倒退！



我完全支持互联网基础知识的普及。然而，为了能成为一名好司机，是不是每个人都需要透彻地掌握汽车的工作原理呢？我们必须普及机械自动化的基础知识，并且把手工艺课提升到跟英语、数学一样重要的地位吗？对普通人来说，知道该如何换轮胎以及何时把汽车送去保养不就足够了吗？如果你家的马桶堵住了，为了把它修好，想必你不会花上两个星期的时间去 toiletcademy.com 学习高级水管工的课程。事实上，浏览一个简单的网页，即学即用，应该就足够了。

那么，从最抽象的意义来看，到底什么是代码呢？

代码是：

1. 一组信号，用以在消息传递过程中表示字母或者数字；

2. 一组有特定含义的符号、字母或者词条，使得消息传递更加保密或简洁；

3. 一组符号或者规则，用以向计算机传递指令。

——摘自《美国传统英语词典》

代码是打孔卡？远程终端？Emacs[①]？Textmate[②]？Eclipse[③]？Visual Studio？C？Ruby？JavaScript？在20世纪20年代，人们普遍认为学会如何使用滑尺是非常重要的。时至20世纪60年代，机械制图又登上了神坛。而如今，这些知识都已经不再重要了。除了那些在“Code: The Hidden Language of Computer Hardware and Software”（代码：计算机硬件和软件的隐语）一文中提到的，我不太愿意向别人推荐任何特定的编程方法。因为我不确定再过20年或者30年，编程是否还会是现在这个形态。对于现在的孩子们，将来他们的编程可能就像玩《我的世界》[④]或者《传送门2》[⑤]之类的游戏那样……

但是，每个人都应该试着写上一小段程序，因为这在一定程度上会使你的思维变得更为犀利，是不是呢？

或者大胆试想一下，也许把《大英百科全书》从头到尾完整地读一遍也能达到同样的效果。说实话，我更建议人们先花些时间去想想，什么样的问题才是他们真正热爱和感兴趣的，然后再好好研究研究这些问题。生命中最困难的，是想清楚你真正想要做的事情，而不是学上一堆假设将来会有用的东西。如果说你的研究和探索最终引领你走上了编程之路，那就用尽一切方法去学习吧，我的祝福与你同在……当然，我的祝福你听听也就算了，它帮不了你。

所以，我绝对不会提倡为了学编程而学编程。我鼓励的是毫无保留地追求你的快乐。举个例子，我曾经收到过这样一封电子邮件：

---

① Emacs是一款强大的文本编辑器，在程序员和其他以技术工作为主的计算机用户中广受欢迎。——译者注

② Textmate是Mac平台上著名的文本编辑器软件。——译者注

③ Eclipse是一个开源的、基于Java的可扩展的开发平台。——译者注

④ 《我的世界》是世界上第一款沙盒建造游戏，玩家可以在一个三维世界里用各种方块搭造建筑物。——译者注

⑤ 《传送门2》是一款益智游戏，玩家需要丰富的想象力和空间思维能力，通过各种方式寻找隐蔽通道或是解开各种机关和谜题。——译者注

我是一名律师（兼注册会计师），现年45岁，正试图尽快放弃目前的律师职业，并全力以赴寻找我的下一个事业。事实上，我雇用了专业人士来帮我做这件事。作为“寻找自我”环节的第一步，他们让我回顾我那漫长而又曲折的职业生涯，找到那些在我的职业生活中我真正享受工作的时刻。

作为一个工作在个人电脑革命时代的会计师（当我在安达信会计师事务所开始我的第一份工作的时候，我们甚至会为手动更新资产折旧计划表而向客户收取费用），我花了很多时间学习如何使用电脑、打印机和应用软件（有人用过 VisiCalc 吗）。而当我受聘为一家大型医疗系统的健康保障计划财务分析师的时候，这些准信息技术相关的事情在我工作中的比重达到了顶峰。在我开始那份工作的第一天，我发现我的前任仅仅留给我一页静态的 Excel 电子数据表，传说它可以用来为一个有7家医院的医疗机构“分析”一份涉及数百万美元的托管保障合同。没办法，我只能开始构建我自己的电子数据表，但是它很快就超出了 Excel 数据库功能的容限。因此我只能自学 Access。接下来很快又把 Access 电子数据表方面的功能利用到了极限——我需要调出成千上万条病患数据记录，套用特定的计算公式，最终得出结论：在假定完全相同的使用情况下，我们从签署的合同中所能得到的付款会更多还是更少。

不得不承认，从任何专业的角度来看，我当时的工作都不是编程。我想办法让 Access 做到了一些微软的技术支持人员告诉我 Access 做不到的事情。但我也仅仅是使用最基本的控制命令，来让一个现有的应用程序服从于我的意志。我依然清晰地记得，我当时很快乐。我每天花 12～14 个小时往公式单元格里输入无限嵌套的命令，我是那样的兴奋，以至于当我不得不停下来休息的时候，我会感到非常失落。

我创造了那个“怪物”并且让它为我效力的经历，至今仍是最能让我感到满足的职业成就，即便我后来担任了另外一家健康保障机构的首席财务官（一个当时可以满足我所有职业抱负的成就），都无法与之相比。不仅仅是因为那些工作，还有在我尝试、失败、尝试、调试、孜孜不倦地构建那个基于数据库的大家伙的过程中，所认识的那群志同道合的分析师和系统管理员。我知道了复活节彩蛋和那些编程的学问，并且破解进入了医院的主服务器。以我当时的收入水平，那台主机的价格已

经远远超出了我所能承受的范围。遗憾的是，我后来还是选择了继续追求我所谓的“职业目标”，并且最终在我憎恨的职业中做着我厌恶的工作。”

上面例子中的这个人，第一，他碰到了自己感兴趣的问题；第二，他尝试着解决问题；（最终很自然地）第三，问题引领着他去学习编程。而且，他热爱编程。事情也理应如此。我并不是因为有人告诉我学习编程很重要才成为程序员的；我成为程序员是因为我想改变我所玩电脑游戏的规则，而学习编程是唯一的途径。一路走过来，我也爱上了编程。

我只是想说，如今我再一次站到了人生的十字路口，在干着准编程工作的那些日子里，虽然平平淡淡，但我却非常享受我的工作。我心里仍然能听到来自它们的“塞壬[①]之歌”。我想问你的是，你觉得像我这把年纪的人，还能学习编程并且达到一个让别人雇用我的水平吗？我并不想一边在纽约城里忙碌奔波，一边又只把编程当作一个副业。恰恰相反，我完完全全真诚地希望变成一个善良的程序员，并用我的时间来创造一些有价值的东西。

遗憾的是，以程序员自居会使你的职业生涯受到诸多限制，特别是对一个曾经担任过首席财务官的成功人士来说，这个转变可不算小。跟钱打交道的人往往会挣很多钱——看看华尔街就知道了。

然而，这不是钱的问题，对不对？我们要说的是“热爱”。如果你想成为一个程序员，你只需追随你快乐的感觉，并且爱上代码。真正的程序员很快会结交到其他一些真诚的善男信女——一些像他们一样毫无保留、疯狂地热爱代码的人。欢迎来到程序员的部落！

如果你在读这篇文章，并且在想，“让这个 Jeff Atwood 见鬼去吧，他到底是谁呀，凭什么对我是否应该学编程说三道四？”而我想对你说的是：“好样的，你应该具备这种独立思考的精神！”

---

① 塞壬（Siren）来源于古希腊神话传说，在神话中她被塑造成一个人面鸟身的海妖，飞翔在大海上，拥有天籁般的歌喉，常用歌声诱惑过路的航海者而使航船触礁沉没，船员则成为塞壬的腹中餐。——译者注

程序员的八种境界

在求职的时候，相信很多人都被问过这样的问题：“你对自己未来 5 年的职业是怎样规划的？”每当我被问及这个问题的时候，我的脑海里总是会浮现出 Twisted Sister[①]乐队在 1984 年拍摄的一段视频里的这个场景：一位老师对着他的学生大喊，“我想要你告诉我，不，是告诉全班同学，你究竟想过怎样的生活？”

你自然会想：我要变得很牛！至少也得成为一个很牛的程序员。尽管这个问题看起来并不像其他一些同样老掉牙的问题那么严肃（如“你觉得你最大的弱点是什么？”），但很可能你还是觉得难以回答。也许有时候你表现得太牛了，不屑于回答这样的问题。但要小心了，你可能会冒犯别人。

在我看来，这个问题同样是一个相当严肃的问题，值得好好思考一番。不是为了应付面试官，而是为了你自己。

对于这个问题，大部分人都会选择一个不痛不痒的回答来敷衍面试官。但它也确实引出了一个更为深层次的问题：一个软件开发人员究竟该有怎样的职业生涯？当然，我们选择这一行是因为我们喜欢这一行，并且幸运地得到了老天的眷顾，让我们干上了这

① Twisted Sister 是一支源自美国纽约的摇滚乐队，成立于 20 世纪 70 年代早期。后来经过将近 10 年的奋斗，他们在 20 世纪 80 年代初有了较大的突破。然而，由于他们的叛逆形象和古怪行为，在他们最风光的时候，被美国政府看作是重金属毒害青少年的一个典型例子，于 1988 年被迫解散。——译者注