程序员的心理疾病

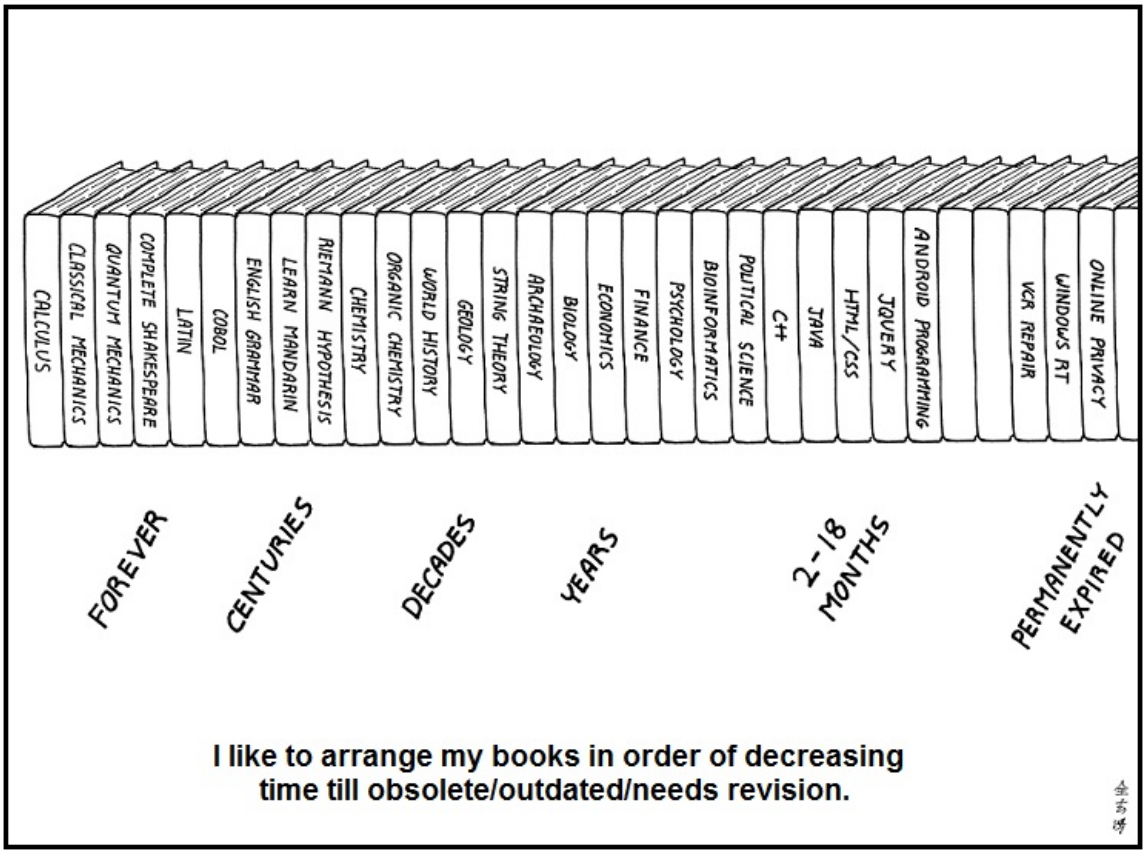
由于程序员工作的性质，他们长期以来受到的所谓“黑客”式的“熏陶”，形成了一种行业性的心理疾病。患了这种病的人对于很多新入行的人，甚至一些外行人士造成了持续的伤害。慢慢的，这些不幸的受害者也形成了“条件反射”，进而成为了这个心理变态的系统的一部分，导致越来越多的人，越来越快的变成“怪胎”。这是一件可怕的事情，所以我觉得有必要警醒一下。

这里我就简单的把我所观察到的一些症状总结一下，希望作为对于 IT 业界人士的警示，有则改之，无则加勉。也希望为遇到类似问题的新手和外行人士提供一些精神上的支持，以免他们也成为这个系统的一部分。

无自知之明

由于程序员的工作最近几年比较容易找，工资还不错，所以很多程序员往往只看到自己的肚脐眼，看不到自己在整个社会里的位置其实并不是那么的关键和重要。很多程序员除了自己会的那点东西，几乎对其它领域和事情完全不感兴趣，看不起其他人。这就是为什么我的前同事 TJ 作为一个资深的天体物理学家，在一个软件公司里面那么卑微。貌似会写点 node.js，iOS 软件的人都可以对他趾高气昂的样子，而其实这些东西的价值哪里可能跟 TJ 知道的物理知识相提并论。很多科学家其实都可以轻而易举的掌握程序员知道的那点东西，有人却认定了他们不是这个专业的，不懂我们的东西，或者故意把问题搞复杂，让他们弄不明白。

其实对于一个物理学家，他心目中知识的价值是这样排序的：



COBOL 在那么靠前的位置我觉得是用来搞笑的，不过你大致看到了很多 IT 技术在真正的科学家眼里的价值和它们的有效期。

如果力学工程师犯了错误，飞机会坠毁；如果结构工程师犯了错误，大桥会垮塌；可是如果软件工程师犯了错误，大不了网站挂掉一小时，重启一下貌似又好了。所以所谓“软件工程师”，由于门槛太低，他们的工作严谨程度，其实是没法和力学工程，结构工程等真正的工程师相提并论的。实际上“软件工程”这个名词根本就是扯淡的，软件工程师也不能被叫做“工程师”。跟其他的工程不一样，软件工程并不是建立在科学的基础上的—计算机科学其实不是科学。

垃圾当宝贝

按照 Dijkstra 的说法，“软件工程”是穷途末路的领域，因为它的目标是：如果我不会写程序的话，怎么样才能写出程序？

为了达到这个愚蠢的目的，很多人开始兜售各种像减肥药一样的东西。面向对象方法，软件“重用”，设计模式，关系

式数据库，NoSQL，大数据…… 没完没了。只要是有钱人发布的东西，神马垃圾都能被吹捧上天。Facebook 给 PHP 做了个编译器，可以编译成 C++，还做了个 VM，多么了不起啊！其实那种东西就是我们在 Indiana 第一堂课就写过的，只不过我们是把比 PHP 好很多的语言翻译成 C。我们根本不想给 PHP 那么垃圾的语言做什么编译器，让垃圾继续存活下去并不能证明我们的价值。

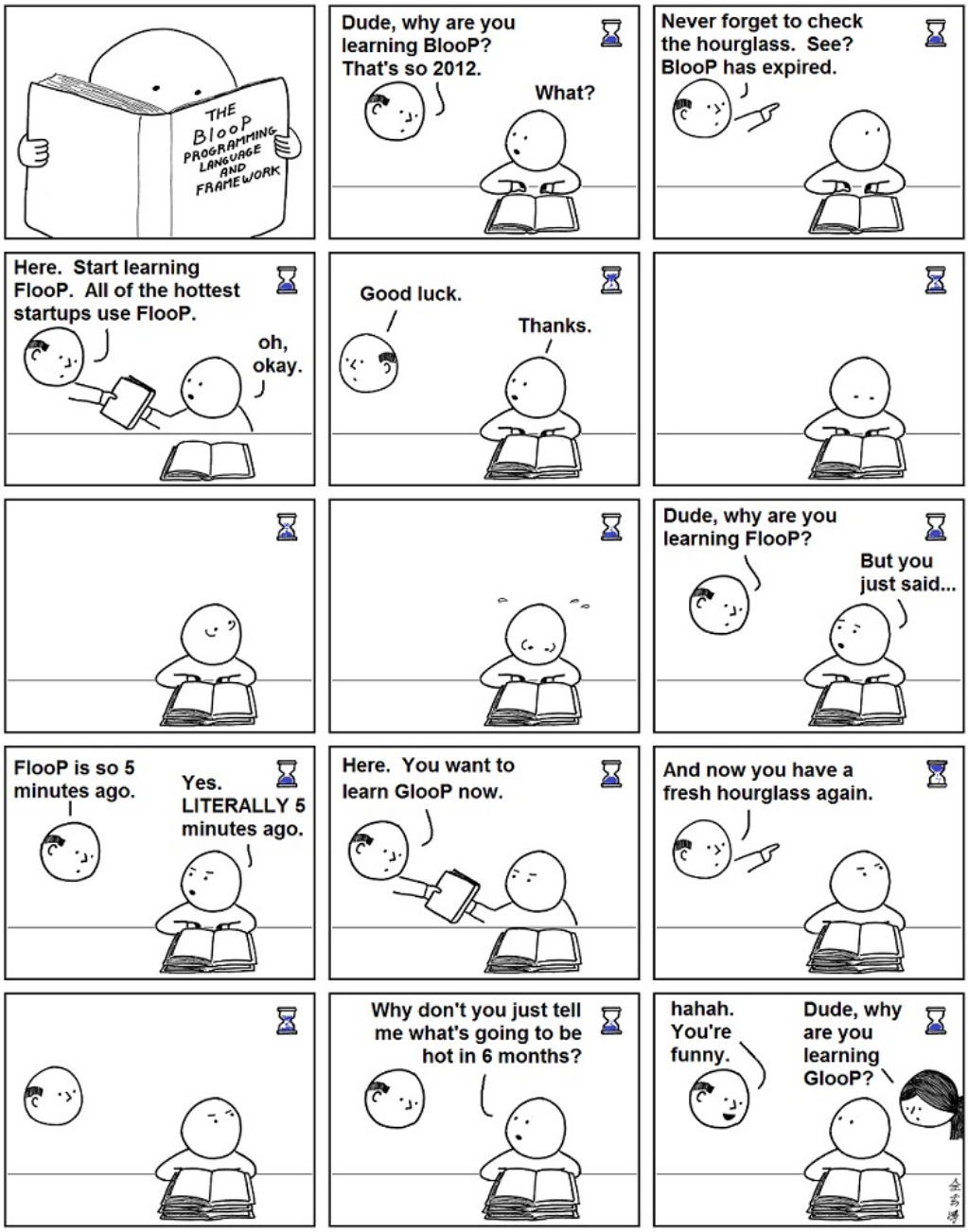
其实软件里面有少数永恒的珍宝，可惜很少有人理解和尊重它们的价值。这在其它的工程领域看来是不可思议的，然而这却是事实。由于没有科学作为理论的基础，没有实验作为检验它们的标准，软件行业的很多东西就像现代艺术一样，丑陋无比的垃圾还能摆在外表堂皇的“现代艺术博物馆”里面，被人当成传世大作一样膜拜。

为了凸显自己根本不存在的价值，又提出一些新的“理念”，就像有些现代艺术家一样，说“艺术的目的不是为了美，而是为了自由。”哦，这就是为什么你们可以自由地把那些让人反胃的东西放在博物馆里，还要买门票才能参观？

宗教斗争

当然了因为没有实质的技术，为了争夺市场和利益，各种软件的理念就开始互相倾轧。一会儿说软件危机啦，面向对象方法来拯救你们！一会儿又提出设计模式。过了一会儿又有人说这些设计模式里面有些模式是“反模式”，然后又有人把函数式编程包装起来，说是面向对象编程的克星，一会儿是关系式数据库，一会儿是 NoSQL，一会儿是 web，一会儿是 cloud，一会儿又是 mobile…… 每个东西都喜欢把自己说成是未来的希望。

这就是为什么有人说在软件行业里需要不停地“学习”，因为不断地有人为了制造新的理念而制造新的理念。在这样一个行业里，你会很难找到一个只把程序语言或者技术当成是工具的人。如果有人问你对某个语言或者技术的评价，是非常尴尬甚至危险的事情，所以最可靠的办法就是不做评论，什么都不要说。



引难为豪

在 IT 行业里批评一个技术难用，是一件非常容易伤自尊的事情，因为立马会有人噼里啪啦打出一些稀奇古怪的命令或者一大篇代码，说：就是这么简单！然后你就发现，这些人完全不明白什么叫做设计，他们以自己能用最快的速度绕过各种前人的设计失误为豪，很多程序员甚至以自己打字快为豪。

往往也就是这些自诩打字快的人喜欢使用过度复杂的方法来解决问题。我可以告诉你，我打字的速度是相当之慢的。我大量的使用鼠标，方向键，而且把 Emacs 里最常用的功能都尽量绑到 F 功能键上，这样我就可以用一个指头启动一个功能。Dan Friedman 的打字速度就更慢，而且他经常故意使用“一指禅”。为什么呢？因为我们写出来的代码非常精辟，几乎不带多余的垃圾，所以根本不需要打很快。

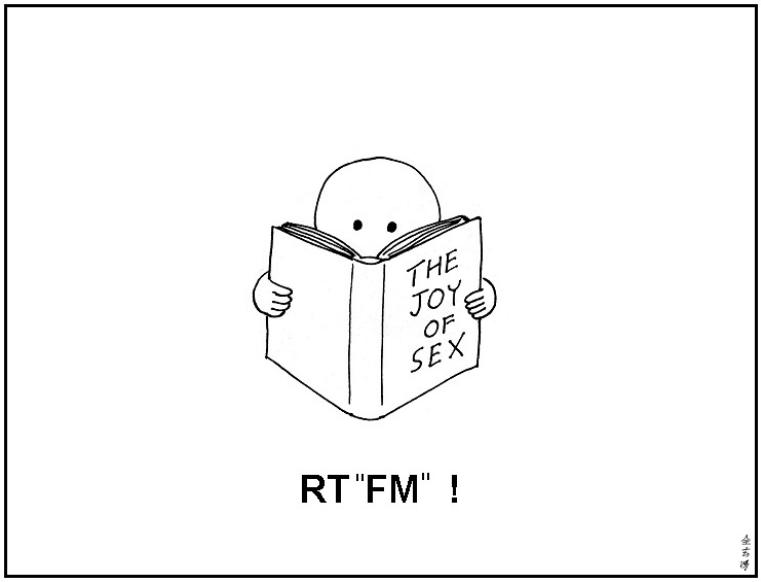
当遇到这样引难为豪的人，我的经验是，千万不要恭维他们。你必须嘲笑这些东西的设计，并且指出它们的失误之处，否则你不但助长了这些人的气焰，让这种风气继续延续下去，而且将来自己的自尊也难保了。很可惜，并不是每个人都有这种勇气把这些话说出来，这就造成了今天的局面，纷繁复杂的垃圾充斥着世界。

爱因斯坦说，你需要很多的天才和非常大的勇气，才能追求到简单。非常大的勇气…… 也许就是这个意思。

去读文档！

不知从什么时候开始，人们开始引用 Eric Raymond 的一篇叫做《提问的艺术》的文章，这篇文章后来就成为了对提问者没礼貌的借口。由于这篇文章的误导，当你希望同事能给你一个手把手的演示的时候，他们往往会丢给你一篇不知道什么时候写的文档，让你自己去读，仿佛文档就可以代替人之间的直接互动。况且不说这文档可能已经过时，里面有很多地方已经不符合最新的设计，而这意味着在潜意识里，他们觉得高你一等。

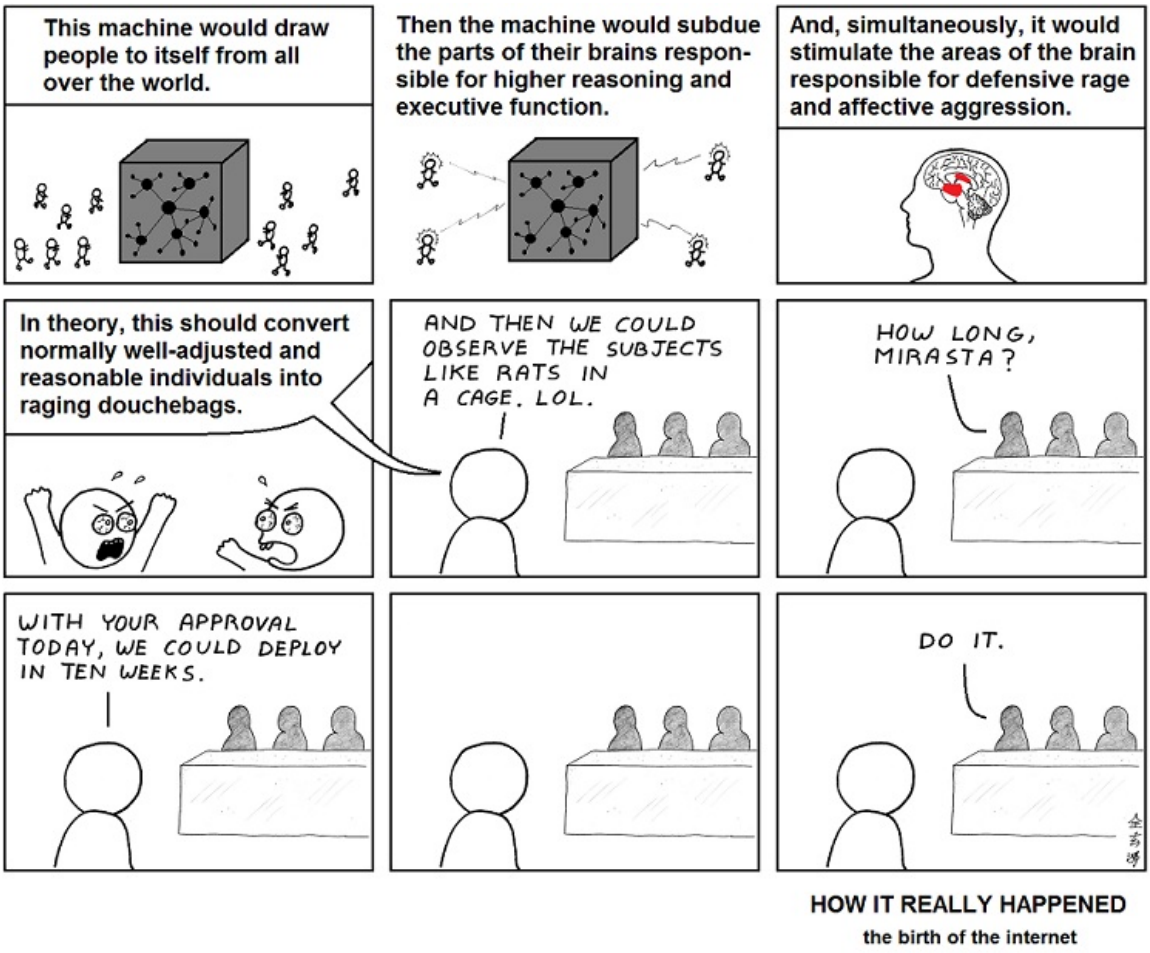
对于这种现象有一个专门的词汇，叫做 RTFM（Read The Fucking Manual）：



在 IRC 的聊天室里，由于隔着网络的屏障，这种对提问者没礼貌的现象就更加嚣张。我曾经有几次去 Java 的聊天室问一些貌似基础，而其实很深入的语言设计问题，结果没有一次不是以收到像“去读 API！”这样的回答而结束。API 谁不会读，然而我需要的是一个有血有肉的人对此的理解。所以后来我根本不去 IRC 这种地方了，因为那里面对你打字的基本上已经不是人类了。他们觉得你问问题浪费了他们的时间，好像他们一天到晚泡在 IRC 里面就是在做什么正事似的。不想回答问题，不开口还不行吗。后来你发现，原来在 IRC 里面训斥新手就是这些人唯一的乐趣，所以其实他们是非开口说话不可的。然而这次他们遇到的却不是个新手，而是一个可以把 Java 整个造出来的人。

像 Haskell 之类的聊天室貌似稍微友好一点，然而后来你发现他们显得友好是有所企图的。因为当时 Haskell 还没有很多人用，他们需要吸引新手，所以竭尽所能的诱导他们。而一旦它用户稍微多了一点，有声势了，就有人开始居高临下，成为专家一样的人物。他们就开始写书，然后就开始牛气哄哄的了。然后你就会发现当对 Haskell 的设计提出异议的时候，这些“id”们是多么的不友好，有理也说不清。所以最后你发现，其实所有语言的所谓“社区”都一个德行。如果 Haskell 有一天像 Java 一样如日中天（当然不大可能），肯定对大部分问题的答案也就是“去读API！”其实它已经在向这一步发展了。

不得不指出，《提问的艺术》等介绍“黑客文化”的文章对于这种现象的出现有着极大的责任。说穿了，写这些文章的人一般都是 Unix 的跟屁虫。这种文章试图抹去人类文明几千年来传承的文化，而重新给“礼貌”做出定义。其结果是，人类的文明因为这些文章，在程序员的世界里倒退了几十甚至几百年。很多外行人人不喜欢跟程序员说话，叫他们是 nerd，就是这个原因。



不要提问，不要谦虚，不要恭维

跟上面的症状相似，程序员世界里的一条重要的潜规则是：只有菜鸟才会问问题。所以如果你有任何机会可以自己得到答案，就不要试图向人“请教”，尤其不要显得好奇，否则你就会被认为是菜鸟。我有几次不耻下问的经历，最后导致了我被人当成菜鸟。我只是觉得那问题有趣，也许能够启发我设计自己的东西，所以吃饭时觉得是个话题可以说一下，结果呢就有人忙着鄙视你，那么小的问题都没搞清楚。正确的态度应该是诚实，直接，见惯不惊，那有什么大不了的，我什么没见过，我很怀疑。

随之而来的引论就是：不要谦虚！那些“职场经验”之类的文章告诉你的进入新的公司工作，要谦虚好问，对 IT 公司是不管用的。有的大 IT 公司有所谓的“文化”，比如叫你要“humble”，其实只是用来贬低你价值的借口。他们只是想让你安于“本分”，做一些微不足道，不能发挥你才能的工作。看看那些叫你要 humble 的人，他们 humble 吗？所以跟江湖一样，在 IT 公司里面一件很重要的事情是，亮出自己的宝剑和绝招，给人下马威。介绍自己的东西一定要自豪，这就是世界上最好的，无敌的，没有其他人能做到！不能有任何保留。不要像科学家一样介绍自己技术的局限性，否则随之而来的就是有些人对你价值的怀疑和对你自信心的打击。

另外要注意的是对于别人介绍的东西，不要轻易地表扬或者点头，否则有人就更有气势了。你要问这样的问题：这里面有什么新的东西吗？这个事情，另外一种技术早就能做了啊，没觉得有什么了不起。

以语言取人

你的软件是什么语言写的，告诉别人的时候是千万要小心的，不到万不得已最好不要说。因为十有八九，对方会立即在心里对你的软件的价值做出判断，光凭你用的是什么语言。

很多程序员都以自己会用最近流行的一些新语言为豪，以为有了它们自己就成了更好的程序员。他们看不到，用新的语言并不能让他们成为更好的程序员。其实最厉害的程序员无论用什么语言都能写出很好的代码。在他们的头脑里其实只有一种很简单的语言，他们首先用这种语言把问题建模出来，然后根据实际需要“翻译”成最后的代码。这种在头脑里的建模过程的价值，是很难用他最后用语言的优劣来衡量的。

有时候高明的程序员用一个语言并不是因为他只会用那种语言，而是其他的原因。他们的头脑里有着万变不离其宗的理念，可以让他们立即掌握几乎任何语言或者工具，所以他们对所谓的“新语言”都不以为然。可是很多人误以为他们不愿意学习“新东西”，从而从心里鄙视他们。其实计算机的世界里哪里有很多新的东西，只不过是有人给同样的东西起了很多不同的名字而已。如果连这样的程序员都不能理解你的技术，就说明你的技术设计有问题，而不是他们有问题。就像 Seymour Cray 说的，我只能理解简单的东西，如果它太复杂了，我是不能理解的。

早些年的时候，大家都认为招募某种特定语言的程序员是一种浮浅的做法，很多公司看重的都是解决问题的能力。可是近些年我发现这些浮浅的做法越来越普遍。可以说现在像 Google 这样的公司面试员工的方式和态度，其实还不如八年前我的第一份国内工作。而这种现象在使用 Python，Ruby，JavaScript 等“流行语言”的公司里就更为普遍。

跟屁虫

有些程序员对新手和同事是那么的不友好，然而对大牛们拍马屁的功夫可真是出类拔萃。我刚到旧金山的几个月有时候参加一些程序语言的“meetup”，后来我发现这种 meetup 都是宗教气氛非常浓厚的地方，跟传销大会差不多。Scala 的 meetup 里面的人几乎全都对 Scala 和 Martin Odersky 顶礼膜拜，甚至把 Rod Johnson 请来说一堆胡话。Clojure 的，当然基本上把 Rich Hickey 当成神，甚至称他为“二十一世纪最重要的思想家之一”。各种 talk 总是宣扬，哇，我们用 Scala/Clojure 做出了多么了不起的东西云云，其实只不过是在向你兜售减肥药。

很多人喜欢做这些新的语言和技术的“evangelist”，尽显各种马屁神功，然后就开始写书，写 blog，…… 目的就是成为这个“领域”的第一批专家。这就难怪了，再垃圾的语言也有一大批人来鼓吹。因为这些没真本事的人，随便把一个东西捧上天都有自己的好处。

由于受到这些“先知”的影响，有些人开始在他们自己的公司里“布道”。比如有人在 Python 的 meetup 集会时告诉我，他试图在自己的小组里推 Python，可是一些老顽固一定要用 Java，认为 Java 才是王道。很鄙夷不高兴的样子。我并不认为 Java 是很好的语言，然而 Python 也好不到哪去。它们在我眼里只不过是临时拿来用一下的工具，可是我仍然能用它们写出一流的代码。

看到这些宗教性质的聚会，我终于理解了一些地区是如何被从一个国家分裂出去，最后沦落为另外一个国家殖民地的。最早的时候，一般是派传教士过去“传经”，然后就煽动一小部分人起来造反。到后来就可以名正言顺的以“保护传教士”，“保护宗教自由”，“维持和平”等理由把军舰开到别人家门口……