给Java说句公道话

有些人问我，在现有的语言里面，有什么好的推荐？我说：“Java。”他们很惊讶：“什么？Java！”所以我现在来解释一下。

Java超越了所有咒骂它的“动态语言”

也许是因为年轻人的逆反心理，人们都不把自己的入门语言当回事。很早的时候，计算机系的学生用Scheme或者Pascal入门，现在大部分学校用Java。这也许就是为什么很多人恨Java，瞧不起用Java的人。提到Java，感觉就像是爷爷那辈人用的东西。大家都会用Java，怎么能显得我优秀出众呢？于是他们说：“Java老气，庞大，复杂，臃肿。我更愿意探索新的语言……”

某些Python程序员，在论坛里跟初学者讲解Python有什么好，其中一个原因竟然是：“因为Python不是Java！”他们喜欢这样宣传：“看Python多简单清晰啊，都不需要写类型……”对于Java的无缘无故的恨，盲目的否认，导致了他们看不到它很重要的优点，以至于迷失自己的方向。虽然气势上占上风，然而其实Python作为一个编程语言，是完全无法和Java抗衡的。

在性能上，Python比Java慢几十倍。由于缺乏静态类型等重要设施，Python代码有bug很不容易发现，发现了也不容易debug，所以Python无法用于构造大规模的，复杂的系统。你也许发现某些startup公司的主要代码是Python写的，然而这些公司的软件，质量其实相当的低。在成熟的公司里，Python最多只用来写工具性质的东西，或者小型的，不会影响系统可靠性的脚本。

静态类型的缺乏，也导致了Python不可能有很好的IDE支持，你不能完全可靠地“跳转到定义”，不可能完全可靠地重构（refactor）Python代码。PyCharm对于早期的Python编程环境，是一个很大的改进，然而理论决定了，它不可能完全可靠地进行“变量换名”等基本的重构操作。就算是比PyCharm强大很多的PySonar，对此也无能为力。由于Python的设计过度的“动态”，没有类型标记，使得完全准确的定义查找，成为了不可判定（undecidable）的问题。

在设计上，Python，Ruby比起Java，其实复杂很多。缺少了很多重要的特性，有毛病的“强大特性”倒是多了一堆。由于盲目的推崇所谓“正宗的面向对象”方式，所谓“late binding”，这些语言里面有太多可以“重载”语义的地方，不管什么都可以被重定义，这导致代码具有很大的不确定性和复杂性，很多bug就是被隐藏在这些被重载的语言结构里面了。因此，Python和Ruby代码很容易被滥用，不容易理解，容易写得很乱，容易出问题。

很多JavaScript程序员也盲目地鄙视Java，而其实JavaScript比Python和Ruby还要差。不但具有它们的几乎所有缺点，而且缺乏一些必要的设施。JavaScript的各种“WEB框架”，层出不穷，似乎一直在推陈出新，而其实呢，全都是在黑暗里瞎蒙乱撞。JavaScript的社区以幼稚和愚昧著称。你经常发现一些非常基本的常识，被JavaScript“专家”们当成了不起的发现似的，在大会上宣讲。我看不出来JavaScript社区开那些会议，到底有什么意义，仿佛只是为了拉关系找工作。

Python凑合可以用在不重要的地方，Ruby是垃圾，JavaScript是垃圾中的垃圾。原因很简单，因为Ruby和JavaScript的设计者，其实都是一知半解的民科。然而世界就是这么奇怪，一个彻底的垃圾语言，仍然可以宣称是“程序员最好的朋友”，从而得到某些人的爱戴……

Java的“继承人”没能超越它

最近一段时间，很多人热衷于Scala，Clojure，Go等新兴的语言，他们以为这些是比Java更现代，更先进的语言，以为它们最终会取代Java。然而这些狂热分子们逐渐发现，Scala，Clojure和Go其实并没有解决它们声称能解决的问题，反而带来了它们自己的毛病，而这些毛病很多是Java没有的。然后他们才意识到，Java离寿终正寝的时候，还远得很……

Go语言

关于Go，我已经评论过很多了，有兴趣的人可以看这里。总之，Go是民科加自大狂的产物，奇葩得不得了。这里我就不多说它了，只谈谈Scala和Clojure。

Scala

我认识一些人，开头很推崇Scala，仿佛什么救星似的。我建议他们别去折腾了，老老实实用Java。没听我的，结果到后来，成天都在骂Scala的各种毛病。但是没办法啊，项目上了贼船，不得不继续用下去。我不喜欢进行人身攻击，然而我发现一个语言的好坏，往往取决于它的设计者的背景，觉悟，人品和动机。很多时候我看人的直觉是异常的准，以至于依据对语言设计者的第一印象，我就能预测到这个语言将来会怎么发展。在这里，我想谈一下对Scala和Clojure的设计者的看法。

Scala的设计者Martin Odersky，在PL领域有所建树，发表了不少学术论文（包括著名的《The Call-by-Need Lambda Calculus》），而且还是大名鼎鼎的Niklaus Wirth的门徒，我因此以为他还比较靠谱。可是开始接触Scala没多久，我就很惊讶的发现，有些非常基本的东西，Scala都设计错了。这就是为什么我几度试图采用Scala，最后都不了了之。因为我一边看，一边发现让人跌眼镜的设计失误，而这些问题都是Java没有的。这样几次之后，我

就对Odersky失去了信心，对Scala失去了兴趣。

回头看看Odersky那些论文的本质，我发现虽然理论性貌似很强，其实很多是在故弄玄虚（包括那所谓的“call-by-need lambda calculus”）。他虽然对某些特定的问题有一定深度，然而知识面其实不是很广，眼光比较片面。对于语言的整体设计，把握不够好。感觉他是把各种语言里的特性，强行拼凑在一起，并没有考虑过它们是否能够“和谐”的共存，也很少考虑“可用性”。

由于Odersky是大学教授，名声在外，很多人想找他拿个PhD，所以东拉西扯，喜欢往Scala里面加入一些不明不白，有潜在问题的“特性”，其目的就是发paper，混毕业。这导致Scala不加选择的加入过多的特性，过度繁复。加入的特性很多后来被证明没有多大用处，反而带来了问题。学生把代码实现加入到Scala的编译器，毕业就走人不管了，所以Scala编译器里，就留下一堆堆的历史遗留垃圾和bug。这也许不是Odersky一个人的错，然而至少说明他把关不严，或者品位确实有问题。

最有名的采用Scala的公司，无非是Twitter。其实像Twitter那样的系统，用Java照样写得出来。Twitter后来怎么样了呢？CEO都跑了 :P 新CEO上台就裁员300多人，包括工程师在内。我估计Twitter裁员的一个原因是，有太多的Scala程序员，扯着各种高大上不实用的口号，比如“函数式编程”，进行过度工程，浪费公司的资源。花着公司的钱，开着各种会议，组织各种meetup和hackathon，提高自己在open source领域的威望，其实没有为公司创造很多价值……

Clojure

再来说一下Clojure。当Clojure最初“横空面世”的时候，有些人热血沸腾地向我推荐。于是我看了一下它的设计者Rich Hickey做的宣传讲座视频。当时我就对他一知半解拍胸脯的本事，印象非常的深刻。Rich Hickey真的是半路出家，连个CS学位都没有。可他那种气势，仿佛其他的语言设计者什么都不懂，只有他看到了真理似的。不过也只有这样的人，才能创造出“宗教”吧？

满口热门的名词，什么lazy啊，pure啊，STM啊，号称能解决“大规模并发”的问题，…… 这就很容易让人上钩。其实他这些词儿，都是从别的语言道听途说来，却又没能深刻理解其精髓。有些“函数式语言”的特性，本来就是有问题的，却为了主义正确，为了显得高大上，抄过来。所以最后你发现这语言是挂着羊头卖狗肉，狗皮膏药一样说得头头是道，用起来怎么就那么蹩脚。

Clojure的社区，一直忙着从Scheme和Racket的项目里抄袭思想，却又想标榜是自己的发明。比如Typed Clojure，就是原封不动抄袭Typed Racket。有些一模一样的基本概念，在Scheme里面都几十年了，恁是要改个不一样的名字，免得你们发现那是Scheme先有的。甚至有人把SICP，The Little Schemer等名著里的代码，全都用Clojure改写一遍，结果完全失去了原作的简单和清晰。最后你发现，Clojure里面好的地方，全都是Scheme已经有的，Clojure里面新的特性，几乎全都有问题。我参加过一些Clojure的meetup，可是后来发现，里面竟是各种喊着大口号的小白，各种趾高气昂的民科，愚昧之至。

如果现在要做一个系统，真的宁可用Java，也不要浪费时间去折腾什么Scala或者Clojure。错误的人设计了错误的语言，拿出来浪费大家的时间。

Java没有特别讨厌的地方

我至今不明白，很多人对Java的仇恨和鄙视，从何而来。它也许缺少一些方便的特性，然而长久以来用Java进行教学，用Java工作，用Java开发PySonar，RubySonar，Yin语言，…… 我发现Java其实并不像很多人传说的那么可恶。我发现自己想要的95%以上的功能，在Java里面都能找到比较直接的用法。剩下的5%，用稍微笨一点的办法，一样可以解决问题。

盲目推崇Scala和Clojure的人们，很多最后都发现，这些语言里面的“新特性”，几乎都有毛病，里面最重要最有用的特性，其实早就已经在Java里了。有些人跟我说：“你看，Java做不了这件事情！” 后来经我分析，发现他们在潜意识里早已死板的认定，非得用某种最新最酷的语言特性，才能达到目的。Java没有这些特性，他们就以为非得用另外的语言。其实，如果你换一个角度来看问题，不要钻牛角尖，专注于解决问题，而不是去追求最新最酷的“写法”，你就能用Java解决它，而且解决得干净利落。

很多人说Java复杂臃肿，其实是因为早期的Design Patterns，试图提出千篇一律的模板，给程序带来了不必要的复杂性。然而Java语言本身跟Design Patterns并不是等价的。Java的设计者，跟Design Pattern的设计者，完全是不同的人。你完全可以使用Java写出非常简单的代码，而不使用Design Patterns。

Java只是一个语言。语言只提供给你基本的机制，至于代码写的复杂还是简单，取决于人。把对一些滥用Design Patterns的Java程序员的恨，转移到Java语言本身，从而完全抛弃它的一切，是不明智的。

结论

我平时用着Java偷着乐，本来懒得评论其它语言的。可是实在不忍心看着有些人被Scala和Clojure忽悠，所以在这里说几句。如果没有超级高的性能和资源需求（可能要用C这样的低级语言），目前我建议就老老实实用Java吧。虽然不如一些新的语言炫酷，然而实际的系统，还真没有什么是Java写不出来的。少数地方可能需要绕过一些限制，或者放宽一些要求，然而这样的情况不是很多。

编程使用什么工具是重要的，然而工具终究不如自己的技术重要。很多人花了太多时间，折腾各种新的语言，希望它们会奇迹一般的改善代码质量，结果最后什么都没做出来。选择语言最重要的条件，应该是“够好用”就可以，因为项目的成功最终是靠人，而不是靠语言。既然Java没有特别大的问题，不会让你没法做好项目，为什么要去试一些不靠谱的新语言呢？