我眼中的各种编程语言

Java

喜欢Java的人肯定喜欢打字。我指的就是敲打键盘上的键。你得不断地重复又重复。

设计Java系统的人是个疯子，他解决问题的方式就是，设计模式。如果你把设计模式看作是这个语言中解决问题的一种方式，那么你会发现Java里有许多这样的设计模式。

另一方面，Sun的这些家伙的确是费了点心思在Java规范上的，这使得它能运行在嵌入式系统上，所以这块我们还是坚持在使用它。我很难相信Python或者C在我的手机桌面系统上运行。

还有，那些个目录又是怎么回事？我必须得使用Eclipse，因为只有它知道怎么跳过那1000个字长的路径名。如果我在应用的同一个目录下放10个类，会不会伤害到某些人？

C

C是精确的。当我用C写程序的时候，如果搞定了，我知道它是靠谱的。它就像是用一把小刷子在画一幅巨作。在这么详细的层面上写代码需要一种不同的心态。当你坐下来写C的时候，在动手之前你就得规划好到底怎么写。否则后面肯定得费很多工夫去改。

如果你的经验足够丰富，内存泄露这种事就不太会找上门。它的第二特性——malloc/free总是形影不离。你不能忘了任何一个。否则就像是忘了冲水或者关灯。你就这么做就是了。

有句话说得好，如果你打算给房子上漆，一把好刷子可远远不够。我猜你肯定想要个大滚轴。如果让我写一整个应用或者系统，能不用C的话我肯定不用。

C程序想要进行改动可得费老劲了。当我写算法的时候，我知道第一遍肯定是不会对的，所以我通常都先用Python写，搞定了之后再翻译成C的。

C++

它就是个有string类的C。同时还有数组，列表，队列等东西，你可以用它们来实现你想要的。一言以蔽之：别想着自创新模板。这太困难了。除了这 个，C++还改良了一下C，用C++你可以写出非常不错的软件。它这个额外的特性使得它可以用于一些大型系统上，只要大家都还遵循同样的约束的话，难度还 不算太大。

JavaScript

这是个没人喜欢的语言。不过它喜欢你。当你刚开始学习它的时候，你可能会写出一些非常糟糕的代码，把对象用作字典，别的对象作KEY，不过这样也是OK的，因为这些代码运行起来也没有什么问题，只要浏览器还支持JavaScript就好。

JavaScript没有连接器，因此所有的代码都共享一个命名空间，不过还好大家都知道这一点，所以还能一起和谐相处。

CoffeeScirpt

CoffeeScirpt是一个解释器，它将那些长得像Ruby的奇怪的语言逐行地翻译成JavaScript。它是一个拥有所有外来语法的JavaScript——括号，方括号，额外关键字移除。只有代码的基本含义还保留着。

CoffeeScirpt挺不错的。如果你要写很多代码的时候，它能让你提高至少25%的效率。你可以一次在屏幕上看到更多行的代码。

当你用CoffeeScript写代码的时候，你得时刻记住这是要生成JavaScript的。问题就在这。你得先去学习JavaScript。项目来的新人都得先学JavaScript，然后才能学CoffeeScript，最后才能去学习项目代码。

node.js

我也希望能爱上它。我觉得我给过它机会了。它的回调让我无法忍受。我知道会有这么一天，因为某个原因，其中一个回调并没有出现，然后我的应用就会堵在那一直等待。真是要了命了。

还有一点就是，它几乎没有内建任何东西。如果你要做某件事情，总是会有一大堆模块来实现这个功能的。该选哪个呢？如果出现问题了，哪个模块会有人来支持？

Scala

Scala是一门函数式，强类型的语言，它会编译成JVM代码。

我是在工作中学的Scala。有一家初创公司的生产系统用的是它，我是在后期才加入他们的。

这让我看到了Scala丑陋的一面：类型推导。类型推荐被它用到了极致。每个对象都有类型，不过想确定它是什么类型的，你得检查不同分层上的好几个文件才行。Scala也继承了Java的文件夹的坏毛病，因此你要查找某个类型的话得进入好几层目录才能找到对应的那个文件。

简而言之，Scala是极好的——对于那些最初的开发人员而言。新加入的成员为了熟悉现有的代码，得有一个很长的学习曲线。

Erlang

Erlang也是我曾经想爱上的一位。我真的努力了。它是一门美丽的函数式语言，它可以写出很精致的小模块，它们以一种精确的方式进行通信，你的系统可以运行10年以上，因为它能处理未知问题，如果必要的话还会重启，然后继续运行。

不过它的结构太复杂了。开发似乎要停留在伯克利发明socket的那个年代。当前时代所需的东西几乎一样都没有。为什么开发一个简单的WEB服务需要费这么大的工夫？

Go

Go很容易学习，对于新人而言也是如此。它使用40年前的语言概念来构建一个健壮的异步系统，但它让你能像写同步代码一样编程。你可以不费吹灰之力写出1000个可以安全工作的线程。

在库支持方面它仍需要改进。当我想做某事的时候，该用哪个库——github上2011年的那个，还是2013年开始的那个半成品？一个是官方主页链接的，不过它的官方主页看起来并不是最新的。好吧，我觉得我还是自己写一个吧。。。

还有，为什么追加元素到数组里也这么费劲？

Python

在Python里，不管你想做什么都会有一个对应的库，如果你用的是Linux，它绝对是不二选择，因为它可以一键安装。

如果你想做些数字处理或者科学运算，选择Python吧，你值得拥有。

Python中的字符串即可能是文本的也可能是二进制的，因此你得上来就学习下文本编码的东东。

Python 3

Python 3和Python有许多共同的特性，不过它却是门不同的语言。由于它比较新，因此支持的并不是很好。我也想使用它，不过总会有那么一个库，它是只支持Python 2的。