万能的林萧说：我来告诉你，一个草根程序员如何进入**BAT**

引言

首先声明，不要再问LZ谁是林萧，林萧就是某著名程序员小说的主角名字。

写这篇文章的目的其实很简单，算是对之前LZ一篇文章的补充和完善。

之前LZ写过一篇《回答阿里社招面试如何准备，顺便谈谈对于Java程序猿学习当中各个阶段的建议》，那篇文章LZ主要介绍了如何应对社招面试，以及如何进行Java学习。

文章的反响还不错，不少猿友都感叹，“如果早看到这篇文章，说不定我已经成大神了。”

但是LZ只能很遗憾地告诉你，LZ那篇文章并不能让你成为大神，只能让你成为一个比大部分人优秀一点的程序猿。而且LZ自己现在都还没成为大神，如何能让你成为大神？

但是，LZ可以手把手的告诉你，LZ作为一个非科班毕业，出身于三流大学的草根，是如何凭借自己的努力，进入到国内一流互联网公司的。

心态

看到这个标题，有的同学可能会说，“LZ，你不是要告诉我们如何进入BAT吗？怎么一上来就写心态？”

没错，LZ第一个要说的，就是心态！

原因很简单，文章下面即将提到的学习内容，如果你没有一个好的心态，是不可能进行下去的。所以，先过了心态这一关，再来谈别的吧，否则你肯定会死在半路上的。

说起来，很多群里的新人都爱问LZ，“你是怎么学习的？为什么我学不进去呢？为什么我一看书就困呢？”

以上这种现象，基本上就是两个原因，第一个原因是心态浮躁，总觉得看书好像没啥用啊，看了好像还是觉得没学到什么，过不了多久就忘了。第二个原因则是目标感不强，不知道自己要达到什么程度才算可以，所以也就干脆“一瓶子不满，半瓶子晃荡”了。

针对第一个原因，说到底就是个心态问题。总想着看书能够立竿见影，从菜鸟一下子蜕变为大神，如果你老是抱着这样的心态去看书，你特么不瞌睡才是邪门了。

而第二个原因，LZ觉得有时候人要适当的激发内心的欲望，无欲无求这种高逼格的事，等你七老八十了再说，现在趁着年轻，心中有点欲望其实并不是什么坏处，只要你没被欲望懵逼了双眼就行。

说起欲望这事儿，LZ觉得有必要给大家看一下LZ当初的欲望，相信从这封邮件里，你就能感受出LZ当时的欲望。这种欲望，会激励着你前进，但是你要切记，在前进的途中，调整好自己的心态，戒骄戒躁。

学习

说完心态，咱们来谈谈学习这事儿，还是老规矩，由于LZ是Java后端出身，所以接下来的内容，会与Java后端息息相关，非Java后端的同学可以适当参考，但切勿照搬。

本篇文章和《回答阿里社招面试如何准备，顺便谈谈对于Java程序猿学习当中各个阶段的建议》不同，那篇文章更多的是从学习的角度去谈的如何学习，而本篇既然咱们是要谈如何进入BAT，那么咱们就从面试的角度来谈学习这件事，LZ会谈谈一流互联网公司对于Java后端程序员的要求，相应的，也会谈谈如何达到这样的要求。

为了简单起见，LZ将这些要求分为三个层次，分别为基本要求、可选要求以及加分要求，接下来，咱们就一个一个的来谈一谈。

一、基本要求

基本要求就是指，你必须要学会的知识，而且这里面大部分内容，在面试里出现的概率都是极高的。因此，这部分内容你没有选择，只能选择啃下它，你可以花一年，也可以花十年，或者带到棺材里学习也可以。

1）语言的基础部分：

基本要求的第一个，当然是语言的基础部分。基础部分其实就是语法以及一些关键字的作用，像一些if/else、for循环这类基础的语法，以及一些new、class、public这类的基础关键字，大部分情况下面试问的是比较少的，因为这部分内容，只要你写过几年Java，基本上都没有什么问题。

那么基础部分的重点，其实主要就是**static**、**final**、**transient**、**volatile**这一类的关键字，以及内部类、泛型这一类的高阶语法。

说到static，首先要记住的最重要的一点就是，类属性中被static所引用的变量，会被作为GC的root根节点。作为根节点就意味着，这一类变量是基本上不会被回收的。因此，static很容易引入内存泄漏的风险。

如果一个面试官让你解释static关键字，你告诉他static可以修饰属性、方法和内部类，以及修饰之后又有什么效果的话，那么面试官基本上不会记住你这个回答，整个印象就是平庸。

但是如果你说完以后，补充一下说道，你曾经遇到过一个内存泄漏的问题，就是因为static修饰的一个Map类型的变量导致的，最后排查了堆栈信息找到了问题的所在，并且解决了这个问题。那么，面试官这个时候内心中对你的印象，就会不自然的提升几分。

而且，对于static，更深入的理解是，static会将所引用的属性、方法以及内部类，与类直接产生引用关系，而非与类的实例。这就是为什么，你可以使用类名.属性、类名.方法以及类名.内部类名，来直接引用一个被static所修饰的属性、方法或者内部类。

如果你没有用static修饰，那么你就必须使用实例才能引用这些方法、属性或者是内部类，最典型的就是内部类。相信很多同学都好奇过，为什么一个没有被static修饰的内部类，必须要这么声明。

```
1  OutterClass.InnerClass innerClass = new OutterClass().new InnerClass();
```

因为你没有使用static修饰InnerClass，所以你必须new出来一个OutterClass的实例，才能在此基础上new出内部类的实例，因为内部类只能通过外部类的实例才能引用。如果你使用了static修饰，那么你就可以这样使用内部类。

```
1  OutterClass.StaticInnerClass staticInnerClass = new OutterClass.StaticInnerClass();
```

这两种方式最大的区别就是，第一种方式，如果你想要获得InnerClass的实例，你必须有一个OutterClass的实例，所有其实这种方式你创建了两个实例，所以有两个new关键字。而第二种方式就好理解一些，静态内部类不依赖于外部类的实例存在，因此只需要直接创建内部类的实例就可以了，所以只有一个new关键字。

static说的有点多了，不过LZ其实不光说了static关键字，也一起连同内部类的语法也大致都说了下。那么接下来，基础部分还有一个比较考验人的东西，就是volatile关键字。

这个关键字的重点就三个字，就是可见性。但是面试的时候，你说出可见性三个字，基本上满分100的话，最多只能得到20分。剩下的那80分，就要靠你用硬功夫去获得了。

所谓的硬功夫，其实就是要整明白，在并发当中，可见性到底是什么意思。那么，为了弄明白可见性什么意思，就需要你了解什么叫主存和工作内存。

只有把这些概念都搞明白了，你才会知道volatile的真正作用到底是什么。不过有一点要提醒你的是，volatile并不保证同步，这一点一定要记住。不光是应付面试官，在真正使用volatile的时候，也要注意这一点，否则很容易出现问题。

好了，基础部分就说这么多吧，LZ挑了一些有代表性的说了下，归根结底，这一部分就是要你非常清晰的了解Java当中的关键字和语法，这里所谓的了解，是清晰的了解其实现原理，而非简单的会用而已。

2）Java运行时环境

Java运行时环境就是**JRE**的中文翻译，本质上其实就是指**JVM**。

首先对于JVM必须要知道的是，JVM与Hotspot的关系。JVM更多的是指JVM规范，而Hotspot是JVM的一种实现，也是我们最常用的JVM实现。你可以把JVM规范当做接口，Hotspot当做实现类，这样去理解会比较简单一些。

此外，**JVM**最重要的三个部分必须要非常清楚，内存划分、**class**加载机制以及**GC**策略。搞清楚这三部分不仅仅是为了面试，也是为了让你对于**Java**有更深刻的理解，这对于你的**Java**生涯非常有帮助。

而且，关于内存划分，还有一点要注意，咱们常说的划分方式，其实是指的Hotspot的划分方式，而非JVM规范所规定的。

Hotspot的内存划分简单说分为三个部分，Young Generation（年轻代）、Old Generation（年老代）以及Perm Generation（永久代）。其中的Young Generation（年轻代），又分为Eden、From和To，其中From和To又统称为Survivor Spaces（幸存区）。

正常情况下，一个对象从创建到销毁，应该是从Eden，然后到Survivor Spaces（幸存区），再到Old Generation（年老代），最后在某次GC下消失。

当然，一个对象也可能直接在Eden里死掉，也可能一直在Old Generation（年老代）存活，这些都是有可能的。

关于内存划分，可以自己没事用内存分析工具看看，比如jmap、jvisualvm等等，观察一下各个区域的内存变化，结合实际去了解一下。

关于classloader机制的学习，可以结合tomcat去学习，了解清楚tomcat的classloader机制，看tomcat是如何保证各个APP之间的类隔离的。如果可能的话，看一下tomcat中classloader的源码，或者看一下LZ的一个开源项目niubi-job，当中也包含了与tomcat类加载机制相似的部分。

至于GC，需要清楚GC Roots都有哪些，以及如何判断一个对象可以被回收。此外，GC的算法和策略也要有大概的了解，具体的可以参见LZ关于这一系列的文章，地址为http://www.cnblogs.com/zuoxiaolong/category/508918.html。

3）并发知识与**concurrent**包

要想进入一线互联网公司，这部分内容必须要会，否则的话，你始终都只能停留在比较low的段位。

关于并发知识，最重要的两个概念一定要搞清楚，那就是可见性和原子性。其中可见性与前面提到的volatile关键字是息息相关的，可见性只是并发领域里的一个概念，而volatile则是Java语言中，实实在在保证变量可见性的关键字。

前面说了，要弄清楚可见性，就需要搞清楚主存和工作内存。关于主存和工作内存，其实又属于JVM的知识范畴。所以从这里就可以看出来，知识都是有关联性的。

原子性其实相对于可见性来说，反倒更好理解一些，相信那个万年不变的银行汇款的关于事务的例子，就足以大部分人理解原子性这个概念了，它其实就是一个或多个操作，被视作一个整体的意思。

有了并发的基础知识以后，你就需要研究一下**concurrent**包了。这里面的东西其实是一个宝藏，一旦你需要写并发相关的功能，你会发现这里面的东西非常实用。

其中ConcurrentHashMap是面试最容易被问到的一个类，几乎所有的面试都会问你，ConcurrentHashMap和普通的同步HashMap有什么区别。

这个问题其实需要你知道两个知识就可以了，一个是HashMap的数据结构，一个是锁分段的技术，具体的LZ这里就不解释了，大家自己下去找相关资料看吧。

此外，**concurrent**包里有一个非常重要的类，叫做**AbstractQueuedSynchronizer**，几乎所有的**concurrent**包内的并发工具类，都是基于这个抽象类扩展出来的。因此，把AbstractQueuedSynchronizer这个类研究透彻，非常有助于你理解concurrent包。

最后一点，面试的时候还经常会被问到的一个问题，就是ReentrantLock和synchronized关键字有什么区别。

记得LZ之前组织过的YY面试活动里，LZ问过很多次这个问题，但几乎所有人都答不出来。这只能说明一个问题，那就是大部分人在用synchronized和ReentrantLock的时候，并不会考虑这两者到底用哪个好一些。

其实它们的区别很简单，简单的说，就是synchronized由于是底层JVM实现的互斥，因此效率会高一些。而ReentrantLock的功能则比synchronized更多，比如定时获取某个锁，多个等待条件等。

并发这一部分是一个程序员进阶的重要部分，希望所有Java程序员都可以重视这一部分。

4）设计模式和反射

设计模式和反射这部分内容，LZ个人觉得是一个高阶程序员必须精通的部分。

用好了这部分知识，可以让你在实际开发中少写N多代码，而且还可以使得程序的结构更加良好。

关于设计模式LZ这里就不多做介绍了，具体的可以看LZ的设计模式系列文章，地址是http://www.cnblogs.com/zuoxiaolong/category/509144.html。

关于反射，其实就是reflect包里的内容，这个包里的类其实并不难，主要是得多用，多看。比如Java领域里最常用的spring框架，里面其实大量充斥着设计模式和反射的真实使用场景，没事多研究一下，绝对让你受益匪浅。

5）文件**IO**、**NIO**、网络**IO**以及网络协议

文件IO、NIO以及网络IO这一部分也是工作当中要经常用到的部分，因此也必须要掌握。

其中NIO更多的是了解其原理，此外，tomcat中有多种协议的实现，其中包括了BIO、NIO和APR，这三者一定非常清楚它们的区别，这个可以在connector的protocol属性配置。

至于网络IO部分，其实就是net包里的内容。这里面的内容是非常常用的东西，比如你调用HTTP-API，那么就需要使用这里面的类。在这个restful-API泛滥的时代，你少不了要使用HTTP协议调用API。

此外，在了解这部分的时候，网络协议也要适当的了解一下，最典型的TCP和HTTP协议是一定要了解的。

在LZ参加的面试中，基本上TCP协议是一定会问的，虽然这可能和LZ的简历写了TCP协议有关，但比如TCP协议的重试机制，三次握手的过程，TCP与UDP的区别这一类的知识，还是要了解一下的。

至于HTTP协议，相对来说就简单很多了，应用层的协议主要是知道其协议格式即可，比如都支持哪些header、每个header都是什么含义等等。

6）小结

好了，到此为止，基本要求就差不多介绍完了。细心的猿友可能会注意到，这些内容其实和LZ上一篇文章《万能的林萧说：一篇文章教会你，如何做到招聘要求中的“要有扎实的Java基础”》，有不少的相通之处。

没错，其实基本要求这部分，差不多就是要求你有扎实的Java基础。这也是所有一线互联网公司，基本都会写在招聘要求地前几条的要求。

因此，要想进入BAT，那么这一部分的内容一定要了解，而且这部分的内容对你实际开发也是非常有帮助的，并不仅仅是为了应付面试。

二、可选要求

看到可选要求四个字，或许不少人会认为这部分不太重要。但是LZ可以很负责的告诉你，这部分往往才是决定公司要不要你的重要指标。

因为基本要求达标以后，公司主要挑选人才的标准其实就是可选要求这一部分，在之前《回答阿里社招面试如何准备，顺便谈谈对于Java程序猿学习当中各个阶段的建议》这篇文章中，LZ曾经提过差异性这个词，其实这一部分就是差异性的体现。

接下来，LZ就带大家看看，到底都有哪些可选的要求。此外，LZ要提前说明的是，这些可选要求，没有必须会和必须不会的内容，尽可能多的了解，总是不会错的。

1）Spring、Mybatis框架

框架这部分其实不用多说了，spring和mybatis框架的原理和源码，如果你可以非常精通的话，那么这一定能成为你巨大的优势。

如果你是专门做WEB开发的Java后端猿，那么spring和mybatis框架基本上你是肯定要用的。精通Spring和mybatis框架不仅为了面试，对于你日常开发也有巨大的帮助，你可以做很多架构上的优化，为你的战友省去很多重复性的工作。

关于Spring框架，最核心的当然是IOC，其次便是AOP、MVC这两部分了。好好研究这三部分的源码，会让你从大部分程序员当中，脱颖而出。至于mybatis框架，主要还是关注它如何实现动态SQL。

而且，待你研究透彻以后，你完全可以自己尝试去造轮子，说不定能得到意想不到的收获。

2）Linux服务器

这一部分其实原本是运维应该精通的部分，但是作为一个Java后端猿，如果你可以精通linux服务器，那么对你排查线上问题，是有很大的帮助的。

大部分程序员都只知道一些常用的Linux命令，对于Linux系统本身的文件系统、网络以及IO等等，是完全不了解的，这其实也包括LZ自己。但是，LZ见过身边有一些程序员，对于Linux玩的非常熟练，这不光光体现在多会几个命令，而是对整个Linux系统的了解。

可以预见的是，这些人在排查问题的时候，往往会更容易找到问题的根本。因为程序问题往往并不是最难解决的，异常这东西见多了就都知道怎么回事了，大不了看看源码也总能找到原因。最难解决的是环境问题，而环境问题无非就是操作系统层面的问题。

而显然大部分情况下，Java运行的操作系统都是Linux。

3）数据库优化

说完Linux，紧接着LZ要说的就是数据库了，这原本应该是DBA应该精通的部分，但作为一个Java后端猿，数据库基本上也是最经常打交道的了。

而且大家都知道，一个应用的性能瓶颈，往往都出现在数据库这一端，因此，一个Java后端猿如果可以精通数据库的话，那么对于你工作的实际帮助，也是非常大的。

相信不少人都碰到过SQL过慢的情况，这个时候，如何通过加索引、SQL分析和优化的手段，将SQL的执行时间优化到一个可接受的范围内，其实还是比较考验人的。

反正，这玩意儿LZ是半斤八两的水平，基本的优化是没有问题的，但稍微复杂一些的就不行了。

所以，这一部分足够成为你的优势，体现出你的差异性。

4）消息服务

除了Linux和数据库以外，消息服务也是当今互联网公司里，必不可少的一个组件。

常见的消息组件比如rabbitMQ、activeMq，包括一些其它的开源消息组件，比如rocketMq。这里面任何一个，如果你可以精通其原理的话，也会成为你有力的竞争条件。

其实消息服务的重点，无非就是如何保证最终一致性、消息的顺序，包括消息事务等等这一类的问题。

虽然LZ本人对此不是很了解，但LZ很确定，这一部分如果你可以有自己独到的见解的话，一定会大大增加你的成功率。

5）缓存服务

说了消息服务以后，相信缓存服务大家也一定不陌生了。

常见的缓存比如memcached、redis这两个，如果你能搞清楚其中一个的话，也会给你加分许多。毕竟现在的互联网应用，缓存也是必不可少的了，因此如果你能完全hold住缓存这一部分，那么你的差异性也就有了。

在缓存服务当中，有几个问题也是比较常见的，比如缓存满了怎么办，缓存的实时性如何处理，内存结构如何规划，分布式的情况下如何处理增删节点时缓存的命中问题等等。

6）负载均衡器

负载均衡器，这是最后一个可选要求了。

常见的负载均衡器就两种，一种是软负载均衡，比如nginx、Apache、lvs这一类的。另外一种则是硬件负载均衡，常见的主要就是F5。

这两种方式各有优劣，其中硬件负载均衡如要用于简单应用、大访问量的场景，而软件复杂均衡则主要用于复杂应用，较小访问量的场景。当然了，两者还有一个不得不考虑的区别是，硬件复杂均衡一般都是非常贵的，而软负载均衡则基本上没有任何成本。

在负载均衡器方面，也有一些问题是比较常见的。比如如何保持会话，如何做流量控制，负载均衡策略都有哪几种，如何检查后端服务器的健康状态等等。

7）小结

好了，到这里，可选要求就说的差不多了。

细心的猿友会发现，这6个要求其实对应的就是Java后端开发中，最常接触到的一些东西。比如spring框架和数据库，这两者基本上是个Java程序员都接触过吧。

其余四个包括Linux服务器、消息服务、缓存服务以及负载均衡器，也是一样的，大家在实际工作当中，应该或多或少都接触过这几个东西。

但是真正能对其中一个非常了解精通的人，相信并不会太多。也正因为如此，如果你做到了，才可以体现出你的差异性，这或许会是你拿下offer的重要筹码之一。

但是，LZ这里必须要再强调一下的是，这几样东西大多数人或多或少都会有一些了解，包括上面LZ提到的问题，不少人也都知道答案。

然而，光知道答案是远远不够的，这并不足以成为你的优势，你需要对这些问题有着深刻的了解，以及有着自己独特的见解，才足以让它成为你的优势。

三、加分要求

最后一个便是加分要求了，加分要求虽然不如基本要求和可选要求重要，但它也与可选要求类似，往往拿下offer的最终原因，正是这些看似不是必要要求的部分。

接下来，LZ就带大家一起来看一下，都有哪些可以加分的部分，这一部分其实在之前那篇文章《回答阿里社招面试如何准备，顺便谈谈对于Java程序猿学习当中各个阶段的建议》中已经提到过，这里就再详细说一下。

此外，LZ要强调的是，这些加分要求中，在某些特殊情况下，可能会成为基本要求。

1）数据结构与算法

这一部分内容不用多说了，大家都懂的。精通数据结构与算法，绝对会成为你的一大亮点。

因为大部分程序员的这一部分基础都不太好，包括LZ本人，面试的时候如果问到算法一类的问题，LZ基本上就两个字：不会。

以前LZ还看过Java集合框架的一些源码，对于一些常用的数据结构还有一定的了解。但是现在，LZ已经基本上忘光了，就连最基本的冒泡排序，可能LZ都不一定能写的对。

因此可以预见的是，数据结构与算法绝对是非常加分的一项。而且，在你面试一些与算法相关的职位时，这个加分要求还可能会成为基本要求。

2）计算机操作系统

计算机操作系统原理，是非常底层的内容。

这部分内容比较难，里面讲的都是一些最基本的底层原理，例如内存、指令、系统IO等等。LZ之前也研读过一本关于操作系统的书，也写了一系列文章，地址是http://www.cnblogs.com/zuoxiaolong/category/518480.html。

不过LZ看的还是不够全面和深入，如果你可以将操作系统研究透彻的话，那么在面试的时候，你完全可以以此作为突破点，展示你的亮点。

3）计算机网络

其实网络这一部分，对于程序员来说还是比较重要的。

LZ最近正在做的事情，就经常会碰到一些网络上的问题，虽然很多时候，这些问题其实可以找专门的网络人员去解决，但如果你自己对此不够了解的话，对于你的工作还是会造成很大的障碍。

而且，要想精通TCP/IP协议，如果对计算机网络不了解的话，还是很难真正理解的。

因此，计算机网络部分如果你可以精通的话，这也绝对会成为你的一个加分项。

4）熟练使用一种脚本语言

脚本语言在很多时候是很方便的，而且也非常实用。

LZ最近就被迫正在使用Python做很多事情，其实用了以后你会发现，虽然Java也可以实现同样的目的，但确实选择合适的语言，会帮你节省大量的精力。

因此，如果你可以熟练使用一种脚本语言，比如Python、shell等等，这也必定会成为你的加分项。

5）你的**github**和博客

这点相信大部分人也都知道，如果你拥有自己的github和博客，并且里面有不少有价值的内容的话，那么一定会为你加不少分。而且，说起github和博客这件事，LZ还有一个关于自己真实的故事，在文章的最后给大家分享，这里就暂时不提了。

此外，就不说面试这回事，平时在github写写代码，在博客里写写文章，总结总结自己的技术和职场，也是非常有好处的。相信不少猿友都已经体会到了这其中的益处，LZ也就不再多说了。

毕竟说多了也无益，最主要的还是自己要真真实实的去做，如果你希望可以在这方面加分，那就从当前做起，并且坚持下去。

6）小结

到这里，加分的要求就说的差不多了。

其实能够加分的内容还有很多，LZ只不过列出了比较常见的几种而已，比如你有其它一线互联网公司的背景，这也是可以加分的。只不过这种加分项比较难达到，而且，这里更多说的是草根程序员，因此LZ这里就没多说。

总的来说，加分要求和可选要求一样，都是你致胜的关键部分，因此如果可能，还是要在加分要求上下一些功夫的。

学习小结

关于学习这部分，到这里就说的差不多了。

就像上一篇《万能的林萧说：一篇文章教会你，如何做到招聘要求中的“要有扎实的Java基础”。》文章里说道的一样，其实大部分一线互联网公司，对于招人的技术要求就两个，扎实的Java基础和一个一技之长。

扎实的Java基础，其实就是本文中基本要求的部分，而一技之长，其实就是可选要求和加分要求中任意挑选一个就可以了。

当然了，不可否认的是，可选要求和加分要求中，你会的越多，成功率就越高，这点是毋庸置疑的。但是如果你一点优势都没有的话，就算你Java基础再扎实，其实也很难进去，因为你这样的人太多了，无法在众多面试者中脱颖而出。

虽然不排除你运气特别好，当时公司正好急缺人，而且没有其他更好的面试者，导致你很幸运的拿到了offer，但毕竟这种概率实在是太小了一些。

说来说去，知识是摆在那里的，不会跑也不会动，就看你学或不学，以及什么时候学。

有的人毕业后一两年就达到了，有的人用了三五年才达到，而有的人，则是一辈子都没有达到。要做什么样的程序员，就全看你自己了。

素养

说完了心态和学习，咱们来谈谈一个程序员应该有的素养。这部分虽然看似对面试没什么帮助，但其实LZ有时候觉得，这比技术更重要，因为它们可能会影响到你程序员生涯的发展。

一、代码风格

说到程序员的素养，第一个就是代码风格。

虽然代码风格并没有绝对正确的风格，但是在满足基本的Java代码风格的前提下，你应该逐渐形成自己的代码习惯，而且必须是一个好习惯。

说个最简单的例子，不管你多么厉害，如果你的变量命名是用拼音来命名的话，那么别人对你的印象一定是，这是个非常low的程序员。

其实程序员有时候和艺术家很像，一个专注于绘画的艺术家，一般都会有自己的风格。说得夸张一点，可能他的画只要拿出去，就有人能认出来，这其实就是一种风格。

作为一个程序员，你也应该有你自己的代码风格，虽然在工作中，为了大家更好的通过代码沟通，你或多或少的需要做一些妥协，和大家保持一致的风格。

但是你自己的开源项目，它应该是你的艺术品，你在雕琢它的同时，其实也是在形成你独特的代码风格。

而且，有的时候，你的开源项目，可能可以直接或间接的帮助你，获得一份不错的offer。

二、写作能力

看到这个或许有的人会很意外，但是LZ个人觉得，写作能力是一个程序员应该有的素养。

代码风格只是你写代码的素养，你还需要有写文字的能力和素养。一个程序员是否专业，文档和注释也是一个很重要的衡量标准。

因为不管你的技术多么厉害，别人看到的，除了你的代码以外，就是你的文档和注释。这部分能否写好，在很多时候，直接决定了别人对你的印象。

因此，写作能力其实也是一个程序员应该有的素养，至少LZ一直是这么认为的。因为不管这个人的技术多么厉害，如果他的文档和注释写的一团糟，丝毫体现不出专业二字，那么他的形象一定会在LZ心中大打折扣。

结尾

本文从心态、学习和素养三个方面简单谈了谈如何进入BAT，但其实这也是一个程序员学习和提高自己的过程。

在提到github和博客时，LZ说了要在文章的最后，给大家讲一个关于LZ自己的真实故事，其实这个故事就是LZ来杭州的真实经历。

LZ之所以能进入现在的公司，其实很大一部分就是因为LZ有一个经常更新的博客，和一个造了几个轮子的github。说起来，LZ真的是运气非常好，或者说是平时的努力，给LZ带来了好运气。

两年多以前，LZ特别想进入中间件团队，但是发简历老是没有回音，于是LZ就给中间件团队的leader，写了一封求职邮件，表明自己非常想进入中间件团队，做服务于技术人员的工作，希望能够得到面试机会。

可惜的是，当时这个中间件的前辈并没有回复LZ的邮件，原本LZ以为是自己的简历没有打动对方。不过LZ后来才知道，其实是因为当时这个前辈已经离开了中间件。

不过，两年以后，也就是2016年过完年以后，LZ的博客里无意间收到了一封短信，正是中间件团队的成员发给LZ的邀请。

可以看到，这封短信正是今年过完年LZ收到的。刚开始的时候，LZ还没注意到有这么一封短信。直到过完年上了十来天班以后，差不多快三月份了，LZ才注意到短信箱里的这个短信。

LZ清晰记得，当时看到的时候是周四，LZ看到的时候很意外。

虽然当时LZ并没有跳槽的打算，那段时间也从未投过一封简历，不过出于不聊白不聊的原则，LZ还是当天就把简历发给了这位前辈。

出乎意料的是，周四和周五仅仅两天，LZ接连不断的经过了四轮电话面试，就从前辈这里得知，LZ非常顺利的通过了面试，就等着拿offer就可以了。

当时听到这个消息的时候，说实话，LZ真的觉得和做梦差不多。

原本一两天前，LZ还处于刚过完年假的不适应期，同时还在纠结手里的需求怎么实现。突然之间就要奔赴杭州，进入自己期待已久的公司，期待已久的部门，展开另外一番职业生涯了。

LZ当时真心觉得，这世间的事情，实在是太特么神奇了。当时LZ脑子里不自觉地冒出来的一句话就是，功夫不负有心人啊！

LZ日积月累的写了这么多文章，终于没有白写，给LZ换来了一个机会。

以前LZ觉得，像什么“功夫不负有心人”这一类的话，都是给别人灌鸡汤打鸡血的话而已。但真正自己遇到的时候，才会真切的体会到，有的时候道理就是这么简单，区别就在于你做还是没做。

就像这篇文章里LZ写的一样，今天你看到了，你做了，那或许未来的某一天，你会感谢今天努力的你。但如果你看到了，没有做，或许你依旧还是那个，整天抱怨工作没劲，加班太累的人，然后在碌碌无为和怨天尤人中度过。

LZ也不想给大家灌什么鸡汤，打什么鸡血，但是LZ想告诉大家，程序员这个职业很公平，相信不少人当初踏上这条路的时候，也是觉得程序员这个职业充满着奋斗的气息，可以依靠自己的努力改变自己的命运，而不像某些职业一样需要拼爹。

在程序员这条路上，努力，你就有很大希望成就自己的目标和梦想。不努力，那你就基本上原地踏步下去，直到被后浪拍死在沙滩上。

如何抉择，其实完全看你自己，只要你不后悔你的决定。所以，少点抱怨，多点行动。

如果你能接受平庸下去，那就平庸下去，幸福的过完你的一生，也没什么不好。否则，你就坚持努力下去，直到达到你一个又一个目标。

最怕的情况就是，心中充满了欲望，不甘于平庸，却又不愿意努力的人，这样的人，如果做了程序员，真的是一种悲哀。

原网页地址：

http://blog.jobbole.com/107224/