1、写在前面

首先，我觉得我来做这个分享并不合适，既然想让我给大家说道说道，那就只能简单讲一下我的拙见，关于这些年技术之路上的一些体会，献丑了。

2、学什么

我们怎么去判定，市场需要我们掌握什么？

急功近利的方式去看，我们可以去拉勾网看看阿里这些知名部门的高 P 的岗位的 JD(招聘描述)，（我个人是极度的阿里推崇者，见谅），看看他的要求是什么，我总结下来逃不开下面几个方面：

1、 java 基础
2、 框架原理源码
3、 分布式技术栈
4、 微服务
5、 软技能

那么就是说我们自己对于这几个方面都有了底气，就成为了互联网行业亟需的高端人才 ;当然有的人会说, 我只是很纯粹地想追求技术, 活了一生就是想探求一下计算机的世界，我丝毫不反对，有这样追求的人，自己对于目标和道路应该都是很明确的，就不需要往下面听我啰嗦了 ;当然我也推荐这部分人先把我推荐的东西学好, 能极速地提高工作上的竞争力，毕竟恰饭是很重要的，做学问也不能亏待了肚子。

我并不是说建议大家学好了这些东西，就去跳槽；只是以说大实话的方式来告诉大家，我没想跳槽，但是当我面对市场的多方位挑战的时候，我也丝毫不惧怕，我有这个底气；公司层面也会希望自己的员工是有这样的素质，而不是作为公司的寄生虫，倘若离开公司，哪也生存不好，公司显然不是养老院，好的公司也不应该因害怕人员流失而阻碍员工成长。

3、怎么学

3.1、系统化

为什么要提到系统化呢，与大脑的思维模式有关，我们掌握的东西如果一直是零碎的，不成体系的，知识点之间交联的点就大概率搞不明白。举个简单的场景，如果你自己是一个长期持续学习的技术人，那么大概率会遇到这种情况，我一直在学习技术 B,学到一定程度，发现自己把技术 A 的一些疑惑点想明白了，可是之前怎么也搞不明白 A 的这些问题。这就说明了，知识的学习需要体系化，知识点之间是相互印证，相互交织，不可分割，体系化也加强了大脑对其的记忆与理解的深刻程度。

3.2、发展的眼光

很多的技术都是经过一代一代演变过来的，就是有他自己的历史。“以史为镜，可以知兴替”说明历史能折射出很多的东西，这是亘古不变的真理，人文历史值得学，技术的发展

史也是一样的。

很多人学习新的技术，就只关注最新版本或者工作使用的版本，导致对于很多的设计不明所以，就是因为没有知识点的纵向的深度。你不知道一个框架，因为什么需求而生，原始版本是什么模样，遇到了哪些问题，通过什么方案解决了这些问题，新的方案又不可避免地产生了哪些问题，整体需求变更，又需要通过什么样的方案来改进，你不明白这个演进史，你又怎么可能完全明白为什么他现在是这样设计呢？

如果我来教课，讲怎么做一个优秀的 java 技术从业者，我会不厌其烦地去讲述，从单体架构到分布式架构的过程中遇到的各种问题以及是通过什么样的方案去解决的。因为这个发展史代表了大部分框架的发展史。你理解了这个历史，对于知识的理解是有极大地帮助的。

3.3、学习的深度

系统化就是知识的横向维度，历史的眼光去看就是知识的纵向维度，加上知识的深度，我们就构建了一个完整的知识体系。我都可以想见，在我脑子里就是一个立体的，有触感的实体。

深度是要到多少，这个就很难讲清楚了。有的人觉得能调用 API，系统能跑，这个就够了；有的人觉得知道大致原理就可以了；有的人觉得看过源码就可以了；有的人觉得需要到操作系统之上，字节码以下这个界限；甚至有的人觉得要到操作系统源码及硬件设计级别。如果大家都能做到操作系统设计层面当然是最好，但是基本是不能实现的，人毕竟精力有限，而且太底层的东西也没那个必要去研究，术业有专攻，我觉得做好自己的这点东西就足够了。

我讲一下，我对自己的要求，对于 jdk 实现，nio 实现等偏底层的技术点是需要到字节码以下，机器码以上的理解。对于主流的的框架，需要到源码层面，相同类型的框架只选一个看源码，一方面是触类可以旁通，另一方面是并不是所有框架的源码都是值得看的。

推荐一下绝对值得看的源码

1、 spring
2、 jdk
3、 log4j
4、 apache commons
5、 guava
6、 velocity
7、 apache tomcat(写的很好，但框架相对庞大，不大好全篇理解)

这么讲可能还是没有那么具体的概念关于知识的深度，那么就举个简单的例子，比如 spring 的 aop，有下面几个理解的层次

1、 知道 aop 能做类似日志等的操作，能在你指定的方法前后做一些操作
2、 知道 aop 通过代理来实现，有 jdk 代理，cglib 代理
3、 知道 jdk 代理只能用于实现了接口的类的继承自接口中的方法，cglib 只能代理可以被子类获取到的方法
4、 明白代理模式的本质就是持有引用，有了引用就可以随时调用指定的方法，前后可以加任何逻辑，不管是 jdk 代理，cglib 代理，甚至是静态代理，都是为了持有引用，到了这里才有了从 0 手写 AOP 的可能。

从 1 到 4 就是知识的深度，到了一阶段，你也可以说自己会 AOP,但是对我来说四阶段才是及格线。

4、学习途径

传统的观念就是看书，要你看 thinking in java ,effective java ,tcp/ip 详解，算法导论，计算机原理，错是没错，但是要考虑适不适合自己。看书是个好途径，但是觉得不适合所有人，单纯的看书学习，我觉得对大部分人来说都不合适。

我个人对于这个方面是很开放的，你看论坛，看博客，刷题，看视频教程，线上线下培训等等的方式我都觉得可以尝试。但是要注意几点：

网上搜出来的资料有很多相互矛盾，有很多错误，可能会误导你的理解，毕竟有很多人靠着拷贝粘贴就写了大篇的博客，有的人还没你理解的到位呢，就大言不惭的教你技术了，所以要自己甄别，网络搜罗的知识，更需要秉持“尽信书不如无书”的态度，要自己多方验证，才能纳入自己的知识库；网络搜罗的资料大多不成体系，东一榔头西一棒子的，也是个大问题。

相对而言，被大众认可的书籍错误的东西较少，相对也会讲的完整，但是大家不大容易静下心来慢慢看书，吸收的效率也大概率不是很好。

对于线上的付费教程，我也是不反对的。在确定是高质量的课程的前提下，用金钱去买时间实质上是很划算的，有人帮你把知识做梳理总结，归纳整理，带你去体验别人长期实践留下来的脚印，都是对自己的知识体系的构建非常有帮助。

希望大家找到合适的适合自己的方式去学习，用轻松舒服的途径去掌握，用书籍去填补空缺，用实践去检验与丰富。

5、当下的人才论

这一部分我就是想说关于软实力的部分。当下社会就是需要大家做到毛遂自荐，锋芒毕露，才能让自己被其他人发现。在职场上，我从不提倡低调，当然我并不是说要你去显摆或者说装逼，而是毫不遮掩的散发自己的能力。毕竟现在酒香也怕巷子深，让我做一个学者我不会选择做钱钟书而会选择做余秋雨。关起门来做学问的人固然可贵，但是那些做得了学问也能上得了演讲台的人才就是更吃香。

从另外的一个角度来看，技术必然不是一切。从概率上来看，我们这些做 java 的，能一辈子走技术路线到底的人并不会有多少，大部分的人会转岗不再做技术，并且技术继续下去往下走也是两条岔路——架构师，管理岗，管理岗也不再那么关注技术细节了，而纯的技术架构的道路显然会是很孤独，艰难晦涩的。这里面就涉及到一个点，很多人谈虎色变的“程序员 35 岁中年危机”，怎么去应对呢？首先 35 岁以上的程序员在阿里多得是，没有说的那么可怕，其次应该在这之前就做好职业规划，即使以后可能不会照本宣科地走，做职业规划也是非常重要的，至少让你自己去思考自己目前的定位，自己的短期长期需求是什么，能引导自己去深入思考这些就已经值回本价了。

而如果抛掉了技术，我们还沉淀下了什么呢？如果大家觉得啥都没留下，那不是大部分人就全盘否定了自己之前的努力呢，因为他们转岗了嘛？

我觉得应该留下的是什么呢？

1、 技术影响力
2、 领导力
3、 协调能力
4、 市场的能力
5、 圈子

技术影响力就像娱乐圈的流量一样重要，极大地代表了你的实力。但是你如果没有好的口头或者书面的表达能力来传达你的底蕴，又怎么可能能影响到更多人呢；领导力来源于什么，来源于“我走过你来时的路”，我曾经在你的阶段待过，我现在走得更高了，看得更远了，我能体会你的悲伤，感受你的苦楚，并且我能用更全局统筹的方式去带你走过这份阻隔；协调能力，就是资源的调配，这里的资源包括人力、物力与时间成本，这个能力就能让你做到最大程度的“尽人事“，做到了那么成功是一件顺其自然的事情；市场的能力，代表着你埋头花了精力花了汗水在某一领域上面，如果你会思考，必然和门外的人看到的这一块世界是完全不同的，能看出行业的趋势，也就能掌握住这里面的机遇；圈子，就是关系网。在当今社会，单靠自己是成不了事的，马云尚且还需要十八罗汉呢，何况你我。

这一段讲的比较乱，总结一下。希望大家去做一下职业规划，在技术之外，请认可软技能的重要性,它的地位绝不会比纯技术能力来的低。

6、纯干货

1、 学习资料

可以关注一下“尚硅谷“这个公众号，发送”java“关键字就能拿到他的视频资料，因为之前百度网盘大批链接失效的问题，导致现在他们全部资料都可以免费拿了，算是因祸得福吧。这个的资料就是框架上的技术讲到应用层面，底层技术讲原理与少量细节，在学习的初中期还算是不错的教程，当然深度是达不到我的要求的，大家自取所需吧。

2、 学习大纲

工具箱：

- Git
- Maven
- Jenkins
- Sonarqube
- Docker
- 敏捷开发

设计：

- 设计原则
- 设计模式

分布式：

- 并发
- NIO
- Monogodb、Redis
- ActiveMQ、rabbitMQ、kafka、rocketMQ
- Mycat
- shardingSphere
- nginx

微服务：

- Spring boot
- Spring cloud Netflix
- Spring cloud Alibaba(nacos、sentinel、seata、springboot+dubbo、zookeeper)
- ELK（ES）
- ServiceMesh

源码

- Spring
- Dubbo
- Netty
- Zookeeper
- ActiveMQ

性能：

- Tomcat
- MYSQL
- JVM

数据结构与算法

软技能

- UML 建模
- 技术文档编写
- 中英文文档阅读能力

上面是我的学习大纲，大家也可以对比一下自己的掌握程度，哪些东西是欠缺的，需要弥补的，有任何问题都可以和我沟通。