程序员跳槽全攻略

@Easy 著

程序员跳槽全攻略

我曾经花了两个月时间，专门研究求职这件事。

那是2009年，我关掉自己的工作室后，打算重新找份工作。因为之前工作室还算挣钱，所以不是很着急。

60天时间里，我读了很多书，画了很多图，见了很多人，面了很多试。根据梳理好的节点，最后我拒了腾讯的Offer，去了新浪做云计算。

那时候SAE团队只有我一个员工（@IT人 是老大，不算，哈哈哈），国内还没有几家做云的；2013年9月我离开新浪时，SAE的日访问已经超过8个亿，云计算已经成为主流技术。

找工作是件非常重要的事情，它直接影响你1~2年，间接影响你3~5年的人生。一个潜在的机会会让你少奋斗很多年，而一次冲动的离职，会让你和千万财富错失交臂。

忘掉那些随地乱扔的小广告，还有从几十个样本做出来的所谓调查报告，换工作不是一场说走就走的旅行，而是一个深思熟虑的结果，是一项复杂的系统工程。我们建议大家每次换工作花一到三个月（的业余时间）来准备，不要嫌麻烦，只要试一次，你就会知道这是值得的。

本书分为三个部分，第一部分讲原理，第二部分讲准备，第三部分讲操作。各部分之间有较强的逻辑关系，建议依次阅读，循序渐进。

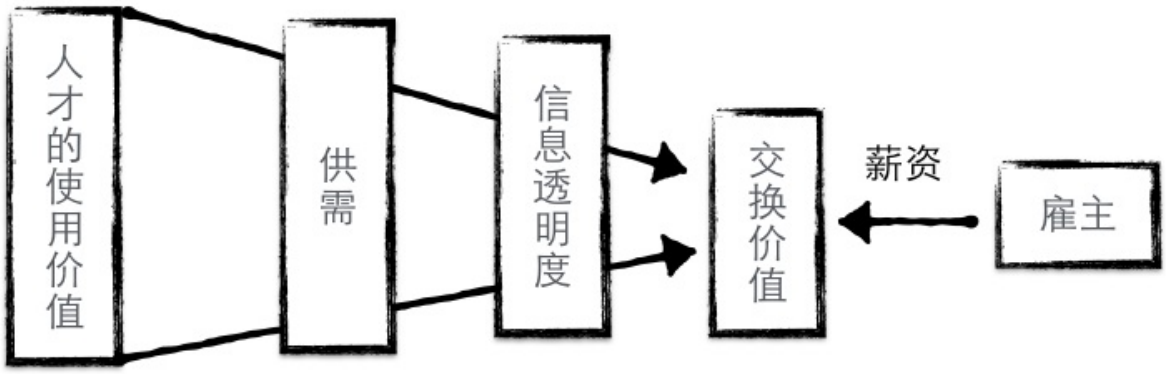
原理篇

我之所以会去研究求职，以至于最后在人才这个方向创业，很大程度上是因为以前所有的求职指导都是经验论，缺乏逻辑性。

而程序员是一种逻辑动物，只有当他们理解求职到底是一种什么行为以后，才能做出有意义的行动。

我花了很久去寻找背后的理论支持，直到我重逢了价值论。原理篇从价值理论开始，讲述我们求职行为的本质。

价值论



自从中国转向市场经济以后，市场规则就开始左右一切。虽然人才不完全等同于其他物品，但现在先让我们把人才也作为一类商品来看待。

使用价值

除了我们这些做人力资源相关行业的公司外，绝大部分公司购买人才都是为了使用，所以他们看中的是其使用价值。

这个使用价值说得更直白一点，就是人才如何直接或者间接的为公司挣钱。

使用价值不是独立存在的，而是相对于使用者存在。所以我们程序员自己的价值，也是相对于公司而言的。

有同学说，我技术很好啊，又会机器学习又会编译原理，凭什么那些写Javascript的薪水比我高一倍?

谁让你在一家建站公司上班呢。对一家做网站的公司而言，机器学习和编译原理是不能为它带来收益的，而Javascript写成的带有完美动画的交互组件却能实实在在的拉升公司产品的销售。

而同样是这个人，如果他去一家以大数据分析为核心业务的公司工作，那么他的价值就不一样了。

所以你的价值，和你牛不牛无关，只和你能为你的雇主提供多少价值有关。（当然，大多数情况下，你能力越牛越能提供更多价值。）

这是最根本的规则。

增加自己的使用价值很简单，提升自己的业务能力就好。

供需

有个80后自嘲的段子是这么说的：

> 读小学时，大学不要钱；读大学时，小学不要钱；还没工作时，工作是分配的；可以工作时，得自谋职业；没挣钱时，房子是分配的；能挣钱时，发现一辈子的薪水也买不起房子。

我不知道国内人才市场是什么时候市场化的，就算成悲催的80后开始工作时吧。人才市场化意味着你有了选择公司的权利，同时也意味着公司有了选择你的权利。

这个时候，交换价值就出来了。虽然交换价值以使用价值为基础，但它更容易受供需的影响。

简单的说，当企业的职位空缺远少于找工作的人数时，人才的价格就会下降；而当企业的职位空缺比找工作的人数更多时，人才的价格就会上升。

在过去相当长时间里，我们都处于前一种情况。这意味着求职者要彼此竞争，而招聘方可以选择要价更低的候选人。这在大家大学毕业找工作时应该深有体会。

幸运的是，供需也是可以调整的，技巧就在于选择更好的细分市场。因为需求的多样性是存在的，所以如果你能在一个大的需求中切入一个需大大大于求的细分市场，那么你就能得到远高于其他人的回报。

举个例子，同样是管服务器，普通运维工程师和云计算运维工程师的薪资差异是非常大的。一个普通运维要变成云计算运维，需要补充的知识并不是特别多。所以你只要合理安排好自己的职业规划，比如以相对较低的薪资到类似新浪云这样的地方工作一到两年，你的能力和交换价值都会大幅度提升。

信息透明度

当人才市场很小的时候，信息是很透明的。因为很容易了解到各自的情况。

但当信息量变大后，你就会发现虽然整个市场很大，但只有你接触到的才对你有意义。

比如北京现在有100家公司都在招聘PHP，但你只知道其中3家，这个时候，其他97家公司的存在对你而言是没有意义的，即使这3家给你的薪资比其他公司低，你也只能被迫接受。这就是信息透明度对我们求职的影响。

没有网络招聘的时候，我们很难对这些公司进行比较，折腾过几家公司后，就屈服了。

有了网络招聘，求职者活得稍微好一些了，可以不出门看到全国的招聘情况；但JobBoard形式的招聘站是为招聘方设计的，它们通过构造信息不对称，向求职者优先显示那些付费却未必最好的公司，迫使求职者以更低的薪资为这些可见的公司工作。（这无可厚非，所有中介体都是通过信息不对称来收费的）

所以要想拿到足够好的薪资和获得足够多的机会，我们要学会和信息不对称进行抗争。一定要在短时间内获取到大量的机会，这样才能「 做选择题 」而不是「 做判断题 」。

关于如何改变信息不对称，操作篇中的「 渠道 」部分我们会详细说明。

跳槽不是什么

跳槽不是为了追求价值最大化

虽然前边我们从纯商业的角度讨论了跳槽这件事，但实际上我们并不推荐求职者单纯地通过薪资的高低来选择雇主。原因有两个。

工作是生活的一部分

你每天有24小时，上班8小时，它是你三分之一的生命。 如果你只是因为高薪选择了一家上班不开心的公司，那么接下来三分之一的日子你都会在痛苦中渡过。而你得到的，只是多出来的那么每月几千块钱而已。

职业的不可逆性

别的事情错了我们可以重来，而职业选择上，每一步都是好几年的时光。一旦走下去，就很难回头，因为一旦换职业方向、哪怕换个开发语言，都意味着你之前的积累很可能前功尽弃。所以眼光放长远点，不要只看眼前的高薪，更要看好未来的发展，方向错了，跑得越猛输得越狠。

跳槽不是找工作

跳槽不是找工作，而是换工作。所以它和你刚毕业时找工作不一样，它是有成本的。

到一家新的公司，需要放弃原来公司的期权，需要重新熟悉业务，重新和同事搞好人际关系，上班路上需要花更多的时间，甚至需要离开自己熟悉的城市。

你并不是一无所有，虽然往往只有在你失去后才发现。所以跳槽之前要考虑好自己的付出和收益，衡量好得失，没事别裸辞。

跳槽不是加薪的砝码

不要以跳槽为砝码去谈加薪，这种迫胁性的谈薪资方式会使主管对你丧失信任。不管涨薪成不成功，他都不得不为你的离职做好准备，而当准备完成后，你就可以走人了。

类似的，当你提跳槽时，如果你的主管通过加薪来挽留你，最好也不要同意。

程序员如何谈薪资

如果你对现在公司很满意，只是觉得薪资太低，那么可以先和你的主管聊聊。

我一般这么谈加薪的：

- 首先，讲一讲自己最近在工作上的成长，看主管是否认同；
- 然后，从能力提升角度，向主管要一个更大的发展空间和更大的业务挑战；
- 最后，问问当你的能力达到这个新的水平时，薪资是否可以同步提升起来。

这样谈有几个好处：

- 它建立了一个谈话基础，那就是薪资应该和能力相匹配；
- 它不谈现在的薪资，而是谈未来，一方面表明你对现在的薪资还算满意（以减少主管顾虑），另一方面对主管来讲，他没有立刻给你涨薪的压力；
- 它体现了你对自己能力提升的渴求，而如果在谈话中出现任何不顺利，你可以把中途把对话结束，根本不提加薪的事情。

当然，坏处就是你得隔段时间才能加薪了。

跳槽到底为什么

说了那么多跳槽不是什么，那跳槽到底为什么呢？

求职的本质

在跳槽之前，你想过我们为什么要工作么？

其实并不是真的如我们父辈们认为的那样，人活着就是要有一份工作的。

工作最重要的目的是累积金钱。这是我们在这个现实世界用以维持生计的必需资源。

挣钱这件事，至少有四种形式：帮别人挣钱（打工）；为自己挣钱（SOHO）；雇别人挣钱（企业主）；让钱自己挣钱（投资）。

为什么我们总是选择第一种呢？因为第一种门槛和风险都最低。 而求职只是我们在自己资源和能力不足以进入后几种挣钱形式时，通过出卖自己的劳动能力换取资源的阶段性行为。

当然，如果能在第一种形式下获得自己满意的物质回报和精神满足，你也可以永远不考虑其他形式，但其他形式在挣钱的效率和数量上是有优势的。

这个我们在稍后的章节会细聊。

跳槽的意义

跳槽其实是打工这种挣钱形态下，我们进行自我调节的方式，通过合理地跳槽，我们可以寻找到市场需求和自我实现的最佳匹配，从而在金钱和成长上双丰收。

「市场需求」描述了企业渴求的员工，「自我实现」描述了你想要的生活；当它们重合到一起，梦想就照进了现实。

这会带给你几年非常愉快的工作生活，但随着我们不断成长，职位和能力可能出现不匹配，这时候我们又需要通过跳槽来再次调节。很多时候，这也是很无奈的事情，不进则退，人在江湖身不由己。

跳槽的原则

我有一个很好用的跳槽原则，之前发到微博上被赞了上百次：

永远不要因为「现在很差」而跳槽，要因为「未来更好」而跳槽。只有这样才能保证你一直往上走。

跳槽的节奏和路线图

时间并不是用来衡量是否应该跳槽的指标，能力才是。从长远角度讲，我们是有一个大理想存在的，比如升职加薪迎娶白富美当上CTO。

而当我们还是一个应届毕业生的时候，我们是不可能直接变成CTO的。所以我们从CTO往下一层层画出节点来，大概这个样子：

应届生 → 初级程序员 → 小组主管 → 部门经理 → 总监 → CTO

然后我们会给每一个节点设置一个达标能力和一个参考时间。当我们的能力已经开始可以胜任下一个节点的工作时，我们就会开始寻找相关机会，最理想的是在当前公司进入下一个节点。但从小组主管开始，能否成功进入下一个节点就不光由能力决定了，有时候你遇到一个永不跳槽也永不被提拔的上级，你就会长时间得不到成长，当这个时间到达我们设定的参考值时，我们就不得不通过换公司来前进。

有些时候，我们也会根据具体情况对职业线做出调整，比如当你小组主管做的很出色，但部门经理这个节点却很难达到时，我们可以这么走：

应届生 → 初级程序员 → 小组主管 → 初创公司CTO → B轮公司CTO → 上市公司CTO

但不管怎么调整，自己头脑里边要有清晰的目标。我们总是现在风险最小的路径（比如当前公司）寻求成长，不成功时再考虑其他的路径。

准备篇

本篇是一些需要花较多精力进行准备的事项。

JobDeer职业画布

在做商业模式设计时，有一种叫做商业模式画布的工具，它可以在一页纸上清晰的描述各方面的影响；由于人才市场有很强的商业属性，以商业模式画布为基础，我们创造了JobDeer职业画布。

它看起来像这样：

下面我来解释下各个部分。首先，JobDeer职业画布是以价值论为基础的，所以最中间就是这个价值主张：「 我能给雇主带来哪些价值 」。

在它左边，是「 如何构造价值 」；在它右边，是「 如何传递价值 」。

如何构架价值

我是谁，我有什么资源

这部分是对自己能力和资历的一个梳理。

我的竞争优势

这部分是基于自己的能力和资历，我们认为自己比别的求职者更有优势的地方。

注意除了写上你已经有的优势外，还可以写上你可以有的优势。然后我们可以在准备期把这些暂时还没有的优势变成现实。这就是为什么我们建议大家提前1~3个月来准备下一次的跳槽。

谁可以帮助我

这部分是指可以帮助你构建价值的人。我们把内部推荐放到这个地方的原因是，推荐你的人会为你做背书，从而证明你的高价值。推荐你的人是否认识和了解你，是否愿意赌上自己名声为你做背书，这很重要。某些网站提供的内部员工推荐并不能为你的价值加分，因为推荐人根本不认识你，这时内部推荐就降格成一种渠道了。

如何传递价值

雇主需要什么样的人

这部分其实属于价值主张部分的，它详细描述了雇主的需求。

怎样让雇主知道你

这部分我们会在「求职渠道」章节中详细介绍。

怎样宣传和证明自己

这部分我们会在「个人品牌」章节中详细介绍。

预估收益

完成了上边的规划以后，我们就要开始来计算收益了：

按照上边的规划，我需要为这次求职付出哪些成本，比如放弃原来公司的期权；学习哪些东西，比如在一个月内学会Swift。

如果我成功入职这家公司，我会有哪些收益，比如能在国内最好的云计算团队研究动态扩容；比如每个月的薪水增加5k。

如果我求职未成功，哪些投入可以在对其他公司的求职上重用，哪些不能，我是否承受得起。

在思考完这些以后，我们就可以得出一个详细的求职规划。在求职过程中，你还可以随时对画布进行更新，来判断要不要接受某家公司的offer。

本书接下来的内容，将针对上边九个方面做详细的讲解，大家可以随时回到本节，对照JobDeer职业画布来体会。

自我认识和自我实现

你该去什么样的公司、做什么样的事情、拿多少钱，都取决于一个问题：你想成为一个什么样的人。工作只是人生的一部分，是用来支撑你人生价值的核心框架之一。在你自己没有想明白的时候，没有人能帮你。

正如前文所说，跳槽是为了寻找「自我实现」和「市场需求」的最佳匹配，但我经常发现我们的候选人对自己的人生并没有目标。

对于没有人生目标的同学，我有两个建议：

第一，给自己定义一年期的目标。我曾花了很长的时间去思考人生的意义，但最后却发现意义都是我们赋予它的。

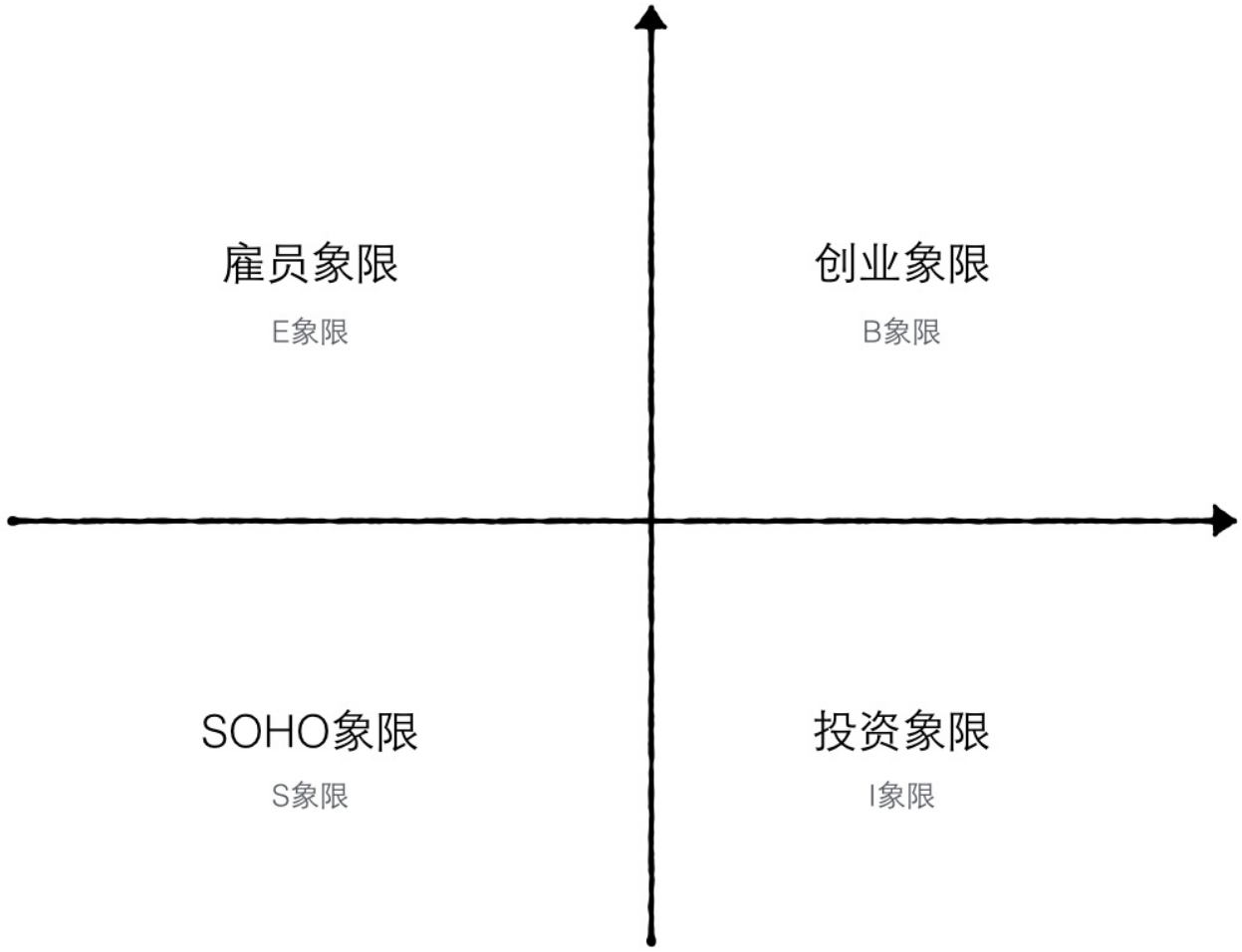
人生有时候就像一个没有终点的旅程，有人的意义是行程的边界，有人的意义是沿途的美景，有人的意义同行的伴侣。当你定下一个目标，人生就变的有了意义。

第二，如果你暂时没有发现人生的意义，那么就多挣点钱，因为等你有一天发现它的时候，一定用得上。

你想要什么样的生活，你想成为什么样的人，这些东西弄清楚后，你得先有一个清晰地人生规划，才能有一个清晰地职业规划。

程序员职业的四大象限

写《穷爸爸富爸爸》的那个胖子——罗伯特清崎，从现金流向将人类分到了四个象限，从而总结出来了这么一张图。



这四个象限分别描述了不同的挣钱方式，这里我们从程序员的角度来进行解读。

E象限（ 帮别人挣钱

E象限是雇员象限，一般来讲，我们刚工作时都在这个象限里边。这里是风险最小的地方，只要你准时上下班别一个SQL把公司的数据库Drop掉，基本都能每月领到应得的银子。

程序员的世界是靠实力说话的（真好），所以如果你刚开始工作，那么你刚刚进入成长期，不顾一切的学好用好各种技术就行，不要想太多。当你工作了两到三年，成长成为资深程序员后，你才开始有资本选择路线。

E象限有两种典型的发展路线，专家线和管理线。它们之间最大的区别是专家线主要和机器打交道，而管理线主要和人打交道。专家线分析机器和程序，优化性能和数据；而管理

线控制资源和进度，随时要和下属谈心、向上级汇报。如果在你眼里，人类特别是女人，是复杂而难以理解的存在，而你也不打算改变你的想法，那么你不太适合管理线。

对于安分守纪的程序员来讲，风险最小的E象限本应是天堂，因为比起改变世界，他们更多的只是很单纯的喜欢写代码。但现实很残酷，北上广的房价高不可攀。你可以不在乎房子，你女朋友能不在乎吗？你女朋友不在乎，她妈能不在乎吗？再考虑到将来小孩上学之类，只要你还打算留下来，房子其实还是必需品。

在E象限，有一批幸运儿通过公司的期权和股票获得了足够多的财富，比如阿里的核心员工们。但公司上市这种情况并不太多见，所以更多的人主要还是用过月薪在获取收入。

S象限（ 为自己挣钱

E象限的薪资通常是有天花板的，很多公司总监的月薪也就3万到5万，扣掉税和每月花销，其实攒不了太多钱。如果公司一直不上市，那么回报就不会太高。 有时候我们为公司创造了很高的价值，我们却无法直接从里边获得收入，但如果是自己的公司，我们就可以把挣的钱可以全部放自己腰包里。于是有一部分人就选择了S象限，为自己打工，这个路线我叫它小老板线。

小老板线是有风险的，如果你长时间没有生意，就要饿肚子了。所以你要有卖得出去的东西。比方说，我们可以开一家微博应用外包公司，给微博的粉丝服务平台做应用。这种面向企业的业务利润不错，一年一个单子就够本、两个单子就挣钱。但这种生意的难度在于你能得到单子。

所以在S象限要活得舒服还是有技巧的：如果做外包，一定要有一个不错的客户渠道；依赖于大平台的项目最好能花点钱成为平台的合作伙伴；建站也是Web程序员们做得多的方向，现在可以顺便把移动APP也给做了，很多简单需求用HTML5打个包就能卖几万块钱。

如果你不懂做关系（尤其是小城市），好吧，我猜你不懂，那么就只能用免费+收费模式了。首先把你要做的业务中标准化的部分开发成产品（如CMS）在网络上免费传播，而其中需要定制的部分就可以收费了。开源和免费的Web产品很多，但同质化严重，很少有细分市场的产品，用心定位的话，养活一个小公司绰绰有余的。

S象限因为是自己开公司，通常员工也不多，所以可以有一种很悠闲的活法，那就是逃离北上广、回归大自然。去一个风景优美空气清新的二三线城市，在湖边山脚弄一个小工作室，写点小众的iOS和Android应用，卖给美国人，既没有房价的压力，还能花着人民币挣美金，也是不错之选。

B象限（ 让雇员挣钱

B象限是创业象限，玩法和S象限很不同，它是以规模化为前提的。投资、上市和出售是这个象限的关键词。

如果你从来没在创业公司待过，那么我不建议你独自创业。如果你没有独立做过能挣钱的软件、上万用户的免费APP、粉丝数5万以上的大号、每天PV10万的网站，那么先别离职创业，先选一个你喜欢的用业余时间感受下。不光是能力问题，也是喜好问题。我见过不少很厉害的程序员CEO，他们过得并不开心。如果你不喜欢伺候一群爷（也就是你的用户），那么别做CEO，还是做一个静静敲键盘的美男子吧。

对于程序员来说，B象限是有一条低风险的捷径的，你可以选择到在创业公司做CTO，如果公司能快速成长，那么你就成为了快要上市公司的CTO；如果公司不幸挂了，那么换一家创业公司接着当CTO就好了。CEO需要为创业公司的失败负担很大的责任，而CTO不需要，他只要用心做好技术就行了。

在这里要和大家强调一点，同样是CTO职位，初创团队的CTO和相对成熟公司的CTO差别是非常大的。

A轮（不一定精确，大致如此）以前的公司，主要在寻找商业模式，会频繁的变更需求，对开发速度要求更高，这时候CTO只要能敏捷的开发产品就OK；A轮以后的公司，着力于规模化，会有大量的推广，可能在某些时间点遭遇高并发，同时技术人员、设备会迅速增加，这时候CTO需要考虑业务的高可用、还要能处理好团队、资源的管理工作。这时候CTO需要迅速的跟上公司的发展速度，否则投资方会建议从大公司挖一个，平心而论，这也是没有办法的事情。

这事有好有坏，坏处是作为初创团队CTO你的压力大了，好处是如果你是被挖过来的那个人，那么你就实现了一次跨级的提升。

E象限中，技术大牛和总监经常会因为拿到投资进入这个象限；S象限中也同理，好的产品也经常被投资人看上。

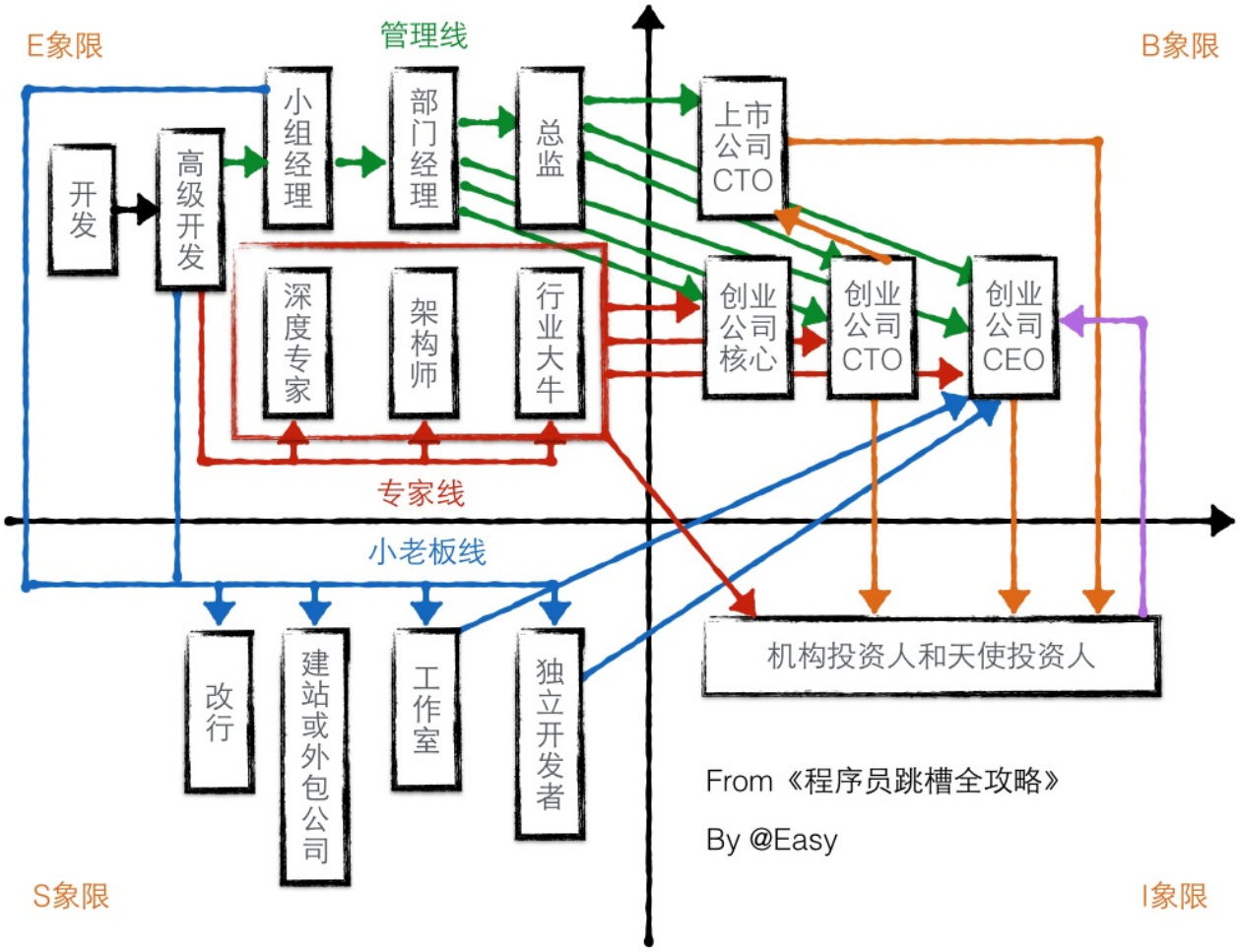
I象限 （ 让钱挣钱

如果你创办或所在的公司成功上市或者被收购，那么恭喜你，你很快就有了少则几百万，多则过亿的资产。这个时候，工作对你来说就是完全可选的了。

但钱多了，让钱保值增值却是你的新课题。于是很多人开始做天使投资，其实技术人做投资存在一定优势，因为可以很好的规避掉产品的技术风险。正因为如此，很多投资机构也很喜欢有技术创业背景的同学，所以投资行业的程序员也开始多了起来。

职业路线图

当我们把四个象限中常见的节点和流向都画出来，我们就有了一张清晰的程序员职业路线图，相信从这张图里边，你可以看到很多熟悉的身影。



注：此图的灵感来自陈晓峰在创新工场演讲中分享的一张图。

市场需求的分析

在认识了自我，想清楚了自己未来要走的大致路线后，就可以根据你自己已经掌握的技术，定出你下一步想要到达的节点。

比如大学毕业后，你已经在一家建站公司工作了两年，已经熟悉了门户、论坛、商城等常见网站的开发，接下来你希望能使自己编写的代码更规范、更优雅、能承载更多压力，所以你希望下一家公司有较大的业务量、成熟的技术体系和NB的技术团队。

定量分析

接下来我们就要开始做市场调研了，把符合你标准的公司一个个列出来，没在你所在的城市也没有关系。去他们的官方网站，从你想要应聘的招聘启事复制下来，放到数据库里边（写个脚本就可以了，我相信对你来说不是什么问题）。样本量稍微大点，至少50份以上吧，然后分词，按词频降序列出来。很快你就可以了解到这些公司对这个职位的要求是什么。

来自JobDeer.com的分析数据

为了给JobDeer投放关键词广告，我们提取了JobDeer和快简历上一周时间的全部几百份招聘启事，进行了上述的分词和排序。因为当初的目的只是为了投放，所以没有专门分离出技术关键词，但意外的是，结合关键词对应的词频，我们可以清晰的看到招聘方对程序员这个职业的要求。

word	count
开发	191715
熟悉	98495
经验	80356
工程师	77135
能力	67110
技术	65459
以上	61837
产品	57489
工作	56993
优先	52054
设计	50191
负责	44781
相关	42809
php	37571
前端	34621
web	33370
熟练	32522
要求	32483
良好	32475
android	30737
系统	29799
精通	28934
项目	28518
优化	27575
具备	26129
使用	26083
linux	25758
团队	25497
框架	25377
java	25211

word	count
掌握	24841
应用	24630
问题	24193
本数据由 @Easy 分析	
架构	22570
沟通	22406
ios	21650
平台	21397
岗位	21026
测试	20844
移动	20700
编程	20622
互联网	19768
具有	19640
数据库	19505
代码	19380
职责	19317
了解	19208
经验者	19096
学历	19082
javascript	18778
职位	18770
css	17623
html	17572
研发	17243
用户	17080
解决	16790
计算机	16406
数据	16393
进行	16208

word	count
性能	16141
学习	16135
语言	15970
mysql	15749
参与	15700
实现	15256
本科	15156
管理	14965
理解	14904
网站	14814
专业	14812
精神	14781
分析	14507
运维	13916
合作	13870
2年	13824
软件	13492
能够	13365
维护	13197
公司	13168
算法	13051
网络	12643
基础	12369
一定	11838
至少	11799
常用	11435
研究	11250
需求	11226
完成	11224
编写	11107

word	count
工具	10871
大型	10730
文档	10476
善于	10459
独立	9915
扎实	9828
python	9610
我们	9402
各种	9224
流程	9121
c++	8811
服务器	8677
描述	8502
高级	8481
方案	8335
体验	8322
服务	8308
业务	8250
以及	8213
强烈	8186
开源	8114
习惯	8092
深入	8054
一种	7945
js	7713
功能	7514
安全	7404
交互	7316
经理	7224
规范	6891

word	count
页面	6877
任职	6871
客户端	6861
脚本	6807
shell	6682
考虑	6580
环境	6451
app	6402
积极	6366
两年	6289
技能	6254
模式	6228
主流	6221
丰富	6093
3年	6034
阅读	5792
核心	5774
原理	5755
jquery	5630
手机	5624
面向对象	5597
提供	5490
质量	5474
数据结构	5425
英文	5397
常见	5341
配置	5315
深刻	5313
协议	5296
实际	5254

word	count
com	5245
http	5235
知识	5157
编码	5120
1年	5105
基本	5044
标准	5030
操作系统	5014
压力	4958
协作	4931
兴趣	4805
方向	4739
制定	4672
资格	4662
搜索	4649
意识	4646
推荐	4632
模块	4584
unix	4575
优秀	4525
表达	4524
自动化	4474
tcp	4455
ajax	4448
逻辑思维	4429
浏览器	4397
承受	4343
可以	4341
部署	4335
提升	4335

word	count
方面	4289
分布式	4281
实践	4281
根据	4254
自己	4230
基于	4214
广告	4212
redis	4201
主动	4199
后台	4189
机制	4186
系统设计	4171
简历	4170
拥有	4127
英语	4099
责任心	4094
mvc	4088
协调	4077
处理	4069
配合	3993
lamp	3990
objective	3970
ip	3937
api	3926
iphone	3917
行业	3890
需求分析	3884
或者	3844
其他	3830
快速	3791

按技术职位的招聘关键词分析

在发现词频对求职的指导意义后，我们对自己数据库中最近发布的2000份招聘启事进行了分析，它覆盖了最热门的8个技术职位，平均每个职位400个JD，其中NodeJS和CTO样本不足400，约200左右，但职位对技术点的要求还是一目了然的。

我们还在这些数据的基础上，开发了各个职位的简历模板，放到了Github上，受到了大家的热烈欢迎，很快上了排行榜第一，现在star已经超过2000了。

下边是这些职位的技术关键字和对应的简历模板地址：

PHP

word	count
php	959
mysql	525
web	280
linux	269
css	246
javascript	241
html	196
ajax	151
jquery	120
sql	105
mvc	104
lamp	91
js	76
apache	76
xml	68
unix	68
div	67
nginx	62
yii	56
thinkphp	56
redis	52
xhtml	51
http	50
shell	49
oop	41
json	41
memcache	38
zend	36
java	31
api	31

word	count
ci	31
svn	27
python	27
codeigniter	27
html5	26
nosql	25
discuz	24
smarty	23
mongodb	21
cms	21
oracle	21
w3c	19
framework	18
lbs	17
git	17
memcached	17
tcp	15
lnmp	15
ip	14
cakephp	13
server	13
app	13
rest	12
crm	9
android	9
uml	9
css3	8
middot	8
webservice	8
php5	8

完整的PHP程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/php.md

iOS

word	count	word	count
ios	1383	im	18
iphone	312	ue	17
objective	286	lbs	17
app	276	gui	16
sdk	205	sqlite	16
xcode	196	cocos2d	16
ipad	155	uikit	16
ui	153	service	16
http	138	ipod	15
mac	83	macos	15
tcp	83	udp	15
socket	81	ood	13
cocoa	58	linux	12
xml	58	apple	11
object	51	macosx	11
json	51	gcd	11
appstore	49	middot	10
os	47	coredata	10
android	36	shark	10
api	36	review	10
interface	31	runtime	9
web	30	itunes	9
builder	29	opengl	8
java	27	storyboard	8
touch	25	ndk	8
instruments	22	and	8
github	22	swift	7
mvc	20	html	7
oop	18	js	7
html5	18	obj	7

完整的iOS程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/ios.md

Android

word	count	word	count
android	1830	j2me	10
java	386	mysql	10
ui	180	oracle	9
app	178	html	9
http	149	sql	8
sdk	135	tv	8
tcp	95	mvc	8
socket	93	lbs	8
api	60	service	7
xml	48	review	7
framework	48	im	7
eclipse	41	code	6
linux	38	mobile	6
json	28	view	6
os	28	studio	6
ndk	27	stackoverflow	6
ios	27	xmpp	6
sqlite	26	o2o	5
andriod	25	ue	5
html5	25	objective	5
web	23	js	5
github	21	blog	5
jni	20	andorid	5
bug	17	rom	5
svn	15	launcher	5
activity	14	restful	5
gui	14	emsp	4
git	13	webservice	4
wifi	10	apk	4
3g	10	androidsdk	4

完整的Android程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/android.md

Web前端

word	count	word	count
web	889	pc	28
javascript	596	nodejs	28
css	555	firefox	26
html	430	angularjs	25
jquery	323	ps	25
js	311	fireworks	24
html5	307	extjs	23
ajax	196	mobile	22
css3	176	safari	22
w3c	168	mvc	22
div	156	jsp	22
php	134	node	21
xhtml	106	backbone	21
java	92	ruby	20
ui	78	github	19
photoshop	75	asp	18
dom	63	android	18
xml	56	ios	18
json	54	ie6	18
yui	51	sass	17
flash	45	wap	16
bootstrap	43	ie	16
python	43	mootools	16
dreamweaver	38	mysql	15
linux	33	flex	14
ext	33	firebug	13
seo	32	bom	13
app	31	webapp	12
prototype	29	less	12
pc	28	mac	10

完整的Web前端程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/web.md

Java

word	count	word	count
java	730	shell	39
spring	305	python	38
web	260	hadoop	37
mysql	250	ssh	35
oracle	207	nosql	35
linux	198	sqlserver	33
j2ee	182	mongodb	33
javascript	177	uml	32
sql	176	svn	32
hibernate	169	json	27
html	139	maven	27
tomcat	132	unix	27
struts	128	nginx	26
jquery	116	webservice	25
jsp	106	jdbc	24
ajax	96	memcached	23
css	94	resin	22
ibatis	84	tcp	22
mvc	77	socket	21
servlet	71	jvm	21
xml	70	db2	19
js	62	springmvc	19
eclipse	51	io	17
mybatis	51	service	17
jboss	47	soa	16
struts2	47	websphere	16
weblogic	46	mina	14
redis	46	android	14
apache	45	extjs	13
server	45	erp	12

完整的Java程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/java.md

C/C++

word	count	word	count
linux	369	nosql	16
windows	178	nginx	15
mysql	110	ui	13
tcp	96	cocos2d	13
unix	93	gdi	12
python	72	app	12
socket	71	boost	12
sql	68	smarty	12
android	67	svn	12
oracle	61	atl	12
java	59	wtl	11
http	51	3d	11
ios	51	win32	10
shell	47	studio	9
web	44	hadoop	9
stl	41	visual	9
mfc	39	gis	9
vc	31	sip	9
server	31	postgresql	8
lua	31	redis	8
php	24	2d	8
perl	24	hive	8
html	23	sqlserver	8
javascript	23	sqlite	7
gdb	21	dsp	7
api	21	uml	7
pc	20	ffmpeg	7
gcc	18	sybase	7
sdk	18	p2p	7
udp	17	xml	7

完整的C/C++程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/c.md

NodeJS

word	count	word	count
js	129	svn	8
nodejs	105	ajax	8
node	97	osx	7
javascript	60	websocket	6
linux	53	xml	5
web	53	socket	5
mysql	48	server	5
mongodb	43	ubuntu	5
redis	26	json	5
php	26	experience	4
python	26	sql	4
html5	25	angular	4
css	21	ping	4
java	18	memcached	4
api	17	stackoverflow	4
express	17	prototype	4
git	16	os	4
nosql	15	async	4
tcp	15	apache	3
jquery	14	meteor	3
html	13	promise	3
ruby	13	backbone	3
shell	12	koa	3
unix	11	cgi	3
app	11	scope	3
github	10	udp	2
css3	10	xhtml	2
nginx	9	callback	2
restful	9	seajs	2
mvc	9	angularjs	2

完整的NodeJS程序员简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/node.md

架构师

word	count	word	count
java	364	unix	36
web	219	oop	35
mysql	186	html5	34
linux	159	jboss	29
php	133	hbase	28
oracle	109	js	28
j2ee	109	rose	28
spring	101	websphere	25
sql	95	sqlserver	25
uml	83	memcached	24
android	74	mongodb	24
javascript	68	json	23
html	62	shell	22
css	59	nginx	22
soa	57	crm	22
ajax	57	weblogic	22
hadoop	54	apache	19
hibernate	54	webservice	19
ios	50	jsp	18
nosql	48	erp	18
mvc	47	ooa	17
jquery	47	memcache	16
struts	46	javaee	15
tomcat	44	lucene	14
python	42	windows	14
server	40	ssh	13
xml	39	saas	13
ibatis	38	powerdesigner	13
redis	38	bi	13
tcp	36	aop	12

完整的架构师简历模板 见这里
https://github.com/geekcompany/ResumeSample/blob/master/architect.md

最后再附送一张CTO技能表。从词频可以看到技术关键字明显变少，软技能的要求上升。（因为这次只提取了技术关键词，所以软技能相关的关键词没有出现在列表中）

word	count
ios	34
java	34
android	32
php	25
app	20
mysql	19
web	18
linux	12
javascript	10
sql	7
oracle	6
cto	6
tcp	6
cg	5
sdk	5
xml	5
css	5
html	5
3d	4
redis	4
python	4
socket	4
rose	3
ui	3
jee	3
o2o	3
bigtable	3
mongodb	3
object	3
xmpp	3

定制你自己的市场分析

以上是我们对通用职位的分析，而你可以针对自己的情况做得更精准一些，比如你可以在数据采集时加入最低薪资限制，这样你可以看到同样是做PHP，会哪些技能的人薪资更高一些。

定性分析

定性分析比较简单，找一下你想要去的公司的工程师，吃个饭聊聊天就能知道基本情况了。如果你有朋友在那个公司，让你的朋友介绍下；如果没有，微博上搜一个公司的员工也很容易。

在做以上分析的时候要记住一件事情，你不是要做一份行业调查报告，你只需要着重了解你关心的情况就可以了。

根据需求调整自己的定位

当你了解了市场的需求后，就要开始和自己的能力进行匹配，看看哪些地方是你的强项，这些在写简历的时候要重点突出；哪些是自己的弱项，争取在准备期强化它，把自己提升到一个新的层次。

这里给大家两个建议：

- 学会观察技术趋势
- 投资新兴市场和细分市场

站在风口不一定能飞起来，但站在冰山上必然会沉下去

和大家讲讲我自己的经历。我是2002年开始学习PHP的，当时ASP非常流行，而我只是一个计算机系的学生，并不具备对行业趋势的判断能力，只是因为ASP太受欢迎，以至于图书馆的书都被借完了，我只好借了一本放在旁边的PHP。

于是在大学时我用PHP写了很多开源项目：留言板、相册、CMS；靠这些东西得到了一家建站公司的兼职工作，除了把学费挣回来以外，还轻松进入新浪实习。到新浪刚过了几个月，新浪也全面转向了PHP——而我正好是我们组最懂PHP的。你可以想象接下来我是多么如鱼得水，我和我的小伙伴们用PHP写完一个人才项目后，又用PHP重写了PV近亿的手机新浪网发布系统，一切都风调雨顺有惊无险，以至于后来去另一家公司面试时，面试官问我经历过最失败的项目时，我完全答不出来。

十年时间过去了，PHP依然如日中天，用我十年前学习的PHP技能，依然可以在一流互联网公司找到一份不错的工作，而ASP——现在谁还知道ASP是什么？

好了，我无意拉仇恨，其实我只是想说同样是一门语言，它的投入产出比是非常不同的。

学会观察技术潮流

有一本老外写的书里边讲，如果你要知道什么语言流行，就去看看技术Guru们都在用什么。这话固然不错，但知其然不知其所以然并不是件好事。

从根源上讲，一个技术是否流行，和人们使用技术的方式密切相关。PHP能大规模流行，并不是因为它的语法好看，而是因为人们使用技术的方式从单机转向了网络，而PHP正好是所有语言里边最专注于Web的。ASP之所以衰落，和网站大量使用Linux服务器有关。

苹果手机大规模流行后，Objective-C的使用量开始极速上升。在很多程序员眼里OC的语

法完全是异端，但这丝毫没有影响他们用异端语言大把挣钱。后来OC的语法苹果自己都看不下去了，于是他们推出了Swift。

仔细观察下周围，现在人类（是的，普通人类而不是其他程序员）是怎么使用技术的？很多人晚上回家已经不开电脑了，他们用电视盒看电视，用平板玩游戏，用手机吐槽。而这些设备很多都是Android系统，那么你觉得Andriod程序员会便宜吗？

由于有了多个设备，个人的数据需要在多个设备之间进行同步、分散到各处的数据也需要统一管理，所以云计算尤其是云存储的需求会凸现出来。

知道了多屏合一以及APP流行的背景，那么PHP的未来会如何？

PHP最大的优势在于它的胶水特性——简单快速的把HTML和业务数据粘在一起然后显示出来。如果只有浏览器，那么PHP还可以通过响应式设计的方式来兼容移动设备。

但是当我们有了手机APP的时候，服务器就必然需要API了。换句话说，多屏合一其实是要求数据和显示分离的——这不是原来的PHP最擅长的（想想单进程这件事），单纯写API接口的话，无论是NodeJS还是Go都颇具优势。所以如果PHP长期在API方面没有改进的话，它的增长会放缓。

好消息是，PHP最近在性能方向提升明显，由鸟哥主导的PHPNG，也就是PHP7，性能表现直追HHVM；Yar项目在实现Rest API时表现卓越；而由韩天峰同学主导的Swoole项目，让PHP在消息通信上的性能不逊色于NodeJS。

这些项目正是技术大牛们实际工作场景中遇到问题，而针对原来的PHP做出的扩展和改进。这些改进会影响未来几年PHP的流行程度。

投资新兴市场和细分市场

新兴市场

新兴市场对程序员来说，就是一种新的语言、一个新的平台、一套新的框架。新兴市场因为刚刚兴起，所以几乎所有人都在同一个起跑线，特别适合后进者。我认识从一个2011年开始学习iOS开发的同学，他能力中等，但现在已经算是很资深的iOS工程师了，月薪超过2万；而如果他那时选择去学习PHP，那他现在只能算个中级程序员。

并非只有一个平台的兴起才有机会，很多框架技术也会带来机会。比如说Cocos2d Javascript版。以前开发游戏需要学习OC或者Java，而Cocos2d Javascript版本的出现，让你可以用纯JS同时开发一套运行在浏览器、iOS和Android平台上的游戏——不用改一行代码，由于是把JS解析后直接扔给OpenGL运行，在手机上性能也很不错。这对JS工程师来讲，就是一个全新的、可以进入游戏行业的机会。

细分市场

如果你只能进入一个成熟市场，那么记得给自己确立一个细分的定位。前几天国内知名的漏洞报告平台乌云招聘PHP开发工程师，职位薪资不是很高，但是我觉得是一个很不错的机会。乌云平台每天被攻击一百多次， 在乌云工作几个月以后，你就能写出来可能是国内最安全的PHP代码，如果再适时的分享下你在乌云工作的经验教训，一个专注于安全的PHP专家就跃然纸上了。这就是细分市场，比你懂安全的没你懂PHP、比你懂PHP的没你懂安全。

合理的调整自己的市场定位，可以让你在职业路线图上走得顺风顺水。

构建个人品牌

所谓个人品牌，就是关于你这个人的品牌。

公司品牌不是个人品牌

在大公司工作的同学经常有一种错觉，就是把公司的能量当做自己的能力，很多时候，你能把事情做好，别人愿意和你合作，不是因为你个人，而是因为你供职的这家公司。

我就有切身体会，之前和技术社区谈合作都是以新浪云的名义去谈，可以很轻松的拿到很好的结果；当我自己创业后，再和他们去谈合作时，他们已经没有时间见我了。所以这种影响力并不是个人品牌，它不过是公司品牌在你身上的折射而已。

一身相随的个人品牌

个人品牌是你可以带走的东西，可以从上一家公司带到下一家公司，和手机、工作、女朋友不一样，可以陪你一辈子。

想想老罗，他其实就是一个典型的个人品牌。很多学生去听他的英语课，不光是因为他的课好，更多的是因为喜欢这个人；所以当他不说相声改行做手机了，这些人就也不听课了，改成买他的锤子了。

就像我们同事吐槽的，当你微博有了4000万粉丝的时候，你卖个啥都能挣钱，这就是个人品牌的能量。

以前只有明星才能成为个人品牌，但随着自媒体和社交网络的爆发，普通人也可以拥有自己的品牌了。它不需要你去演电影、上春晚，只需要你开一个微博或者做一个公众号，然后持续的吸引喜欢你风格的人。

如果你是一个医生，你可以和大家分享医疗常识和急症室的各种趣事；如果你是一个PHP程序员，你可以教大家使用PS切图和介绍你喜欢的格子衬衣品牌。

这并不是很难的事情，只要你贡献出高价值的内容，粉丝是会慢慢有的。比如看过这本书的同学觉得很不错，那么他们就会关注我的微博（ @Easy ），因为这样以后我发布新的好东西的时候，他们就能在第一时间知道，所以随着这本书的传播，我的粉丝也会增加（笑）。

程序员如何做个人品牌

对于技术人员，下边这个列表我是建议你要有的：

- Github账号，不解释。
- 技术博客，可以直接放到Github上，Big更高。
- 微博，最好能加V，用于业内交流。
- 技术社区账号，比如stackoverflow。

有了这些账号还不行，还得有持续更新的优质内容。这里我要严肃批评一些做技术的同学，太低调了。

以前我有个同事非常牛，经常搞定各种高难度的东西，但就是特别低调，不愿意整理和分享出去。我因为记性不太好，遇到什么问题不管大的小的，我都往微博上记，那时候为新浪移动云做可行性调研时写了很多Tips，几个月下来发现粉丝涨了一两千。现在这位技术大牛经常找我帮转发招聘启事。

所以勿以善小而不为，勿以技小而不分享。平时遇到的大小问题可以零星记录到微博上；相对大块的东西，可以写成文章放到博客，通过微信推送给订阅的朋友；成系统的东西，可以在文章基础上整理成迷你书。

做粉丝和虚荣无关

粉丝并不是虚荣，它是你个人品牌的量化表现，中国最不缺的就是人，品牌就是雇主在芸芸众生中不选择别人，唯独选择你的理由，所以好好去经营它吧。

学会沟通和写作

我知道对于很多程序员来说，沟通和写作是非常难的事情，这里不打算开写作培训班了，只是和大家分享一点立马用得上的提升写作水准的技巧。

条理性

其实我们技术人员工作中大部分的写作，都是应用文写作，并不需要你有丰富的想象力、风趣幽默的文风、小清新的玻璃心或者历经沧桑的过去，只需要你把事情讲清楚就好了，所以只要内容条理性够好，听众就会很OK。

我们可以用列表来写一个提纲，比如今天你要汇报最近一周的工作，那么先写一个一级列表，像这样：

- 优化招聘方请求
- 提升通知到达率
- 订阅系统升级

然后，再为每一项写小的列表，比如：

- 优化招聘方请求
    - 根据关键字对请求加减分
    - 增加被拒绝请求的优化提示

- 提升通知到达率
    - 添加短信通知
    - 微信下行通知换成模板通知

- 订阅系统升级
    - 添加被动订阅模式
    - 修改界面，改为关键字订阅和条件订阅二选一模式

不多写了，再写这周周报都写完了。那当我们汇报的时候，要先点后面，先给概要再给细节。具体起来就是这样：

> 上周我们主要做了三件事情，分别是优化招聘方请求、提升通知到达率以及订阅系统的升级。为了优化招聘方请求，我们分别采取了根据关键字对请求加减分和增加被拒绝请求的优化提示的措施，这两项都已经在周三完成，已经测试并上线，用户反馈数据正在收集中，预计下周一可完成，当天下班前我会发邮件给大家……

这样就很非常清楚了。

逻辑性

虽然缺乏逻辑性是产品经理的而不是程序员的通病，这里也提一下吧。我们写文章或者做汇报时，前后文之间是应该有明确关系的，后一句对于前一句，或者是后续、或者是论据、或者是总结、或者是并列。不要搞跳跃性思维，不能前边一句说你去了洗手间，后一句说你吃得很饱，这会出人命的。

只要你牢牢记住上边的技巧，写一篇条理清晰，逻辑顺畅的文章还是很容易的。

走完分享的最后一公里

毫无疑问，程序员是非常愿意分享的群体，正是这样才有了数不尽的开源软件，我现在正在使用的GitBook就是其中之一。

但是很多程序员在分享这件事上虎头蛇尾。我们分享的目的，是让别人能够理解、重用我们的劳动成果。如果我们只是将代码直接push到GitHub上，其实是达不到分享的目的的。

我们走过了程序开发这千里长征，我们一定要坚持走完分享这最后一公里。为自己的项目写概要说明文档，为新手用户写Quick start，将项目提交到各个技术资讯站，为感兴趣的同学提供讨论和交流的场所。

充分的交流不但会让你的影响力扩大，更会聚集各种有意思的想法，往往让你喜出望外，获得新的启示。

渠道

下边给一些常见的渠道，如果是技术文章：

- 首先可以发布到你自己的技术博客
- 然后同步到微博（可以用ifttt）
- 如果反响不错，可以再通过微信公众号推送给订阅读者
- 提交到 startup news 和 cdsn的极客头条

针对文章的受欢迎程度，我们还可以进行二次加工

- 根据文章内容制作PPT，通过slideshare和微盘分享
- 定期精选系列文章，更新到最新后整理成PDF，通过微盘分享
- 对于特别受欢迎的教程类文章，可以做screencast，通过在线教育网站（比如优才网、慕课网等）进行传播

如果是开源项目，当然就是GitHub了。

开始你的开源项目

开源项目在技术求职中是大规模杀伤性武器，如果要面试的公司正在使用你写的开源代码，你会有非常高的加分；即使不是那么有名的开源项目也可以让面试官很清晰的了解你的编码风格、架构能力，从而节省很多不必要的面试笔试时间。所以现在就开始你的开源项目吧。

通过开源项目转型

经常有候选人和我说，我很喜欢XXX语言，但是在公司没有机会做，所以我想跳槽到一家使用XXX的知名公司进行学习。

这种想法的愿景不错，但往往很难实现。因为从招聘方来讲，它不是做免费教育的，它是一家商业公司，所以它总是去招性价比最好的人选。

除非你之前的工作经验能很好的移植到新的领域，否则为什么不直接找一个应届生来培养呢？他们处于职业的成长期，对于薪资不敏感，又有更充沛的学习精力。

所以如果你想转型，做一个开源项目是非常有帮助的。它让你在新领域的经历不是一片空白，也向招聘方证明了你对这个领域的真实兴趣。反过来，如果你对招聘方说你对一个语言「非常感兴趣」了好几年，却从来没有用它写过一个项目，很可能被贴上光说不练的标签。

开源项目不是遥不可及的

并不是一定做出WordPress这样的项目才行，其实很多有名的开源项目不过是一些细节上的改进，比如iScroll这个项目，它其实只是处理滚动条的小Tip而已，技术上没特别的难度，代码量也不大，但由于大家都不想在这种细节上花太多时间，反而让iScroll大规模流行，最后苹果和微软甚至雇佣过它的作者做兼职。

所以开始一个开源项目其实很简单，找一些自己在做项目时遇到的费事费时的小细节做好，然后开源就可以了。

举个例子。比如我们在做图片列表的时候，如果图片高度不同，我们就要截图，很容易把脸给截没了。但其实JS版的人脸识别库已经在github上开源了，那我们就可以做一个可以识别人脸的智能截取一定高度的图片的jQuery插件，先给自己用，再开源给其他人。

随着用户的增加，我们会添加对不同版本浏览器的支持，添加对手机的支持。这样用的人就越来越多，他们会帮我们口口相传，最后我们就有了一个很不错的开源项目了。

比起技术能力，更多的是「来自于真实的需求」以及「持续更新的毅力」，这就是做好开源项目的秘诀。

提升架构能力

架构能力和写作一样，不是能一蹴而就的东西。这里只是在理念上和大家分享下。

在我看来，软件本质上是一种能力，是封装好的、可高速、廉价、重复执行的能力。

我读过很多关于架构的书，也写过数百个大大小小的项目，一路实践下来，个人觉得最重要有两个原则，DRY和正交性。

DRY

DRY是Don't Repeat Yourself的缩写，翻译过来就是「不做重复事」。

这正是一个逼近软件本质的原则，它指导我们把经常使用的功能抽象成库，把重复出现的代码重构为可重用的框架模块。如果你用DRY来要求自己，很快你就会发现自己抽象和架构能力的飙升。

半自动化

但是我们活在现实世界，所以我们不可能把所有的事情都给自动化了，有人类尤其是女人参与的活动，往往会毫无规律可循。

但我们不能放弃，不要二元思维，除了手动和自动，我们还可以半自动化——让机器做完所有繁杂的常规操作，人类来处理需要智慧的那一点点工作就好，这也能极大的提升工作效率。

正交性

正交性的意思是，功能和功能之间应该尽可能不互相干扰。只有这样，我们才能有效的控制每个部分的行为。所以功能之间的依赖尽可能少，如果有，规则一定要明确，不要试图去做一些自作聪明的事情。

比如JobDeer之前的推送通知是在候选人发布时自动发的。一直用着不错，但有一天有一个候选人需要重新发布，但我们不想推送通知，这时候我们就傻眼了。这是因为发布功能和推送功能不是正交的。

后来我们把发布和推送功能分开，在发布成功后，询问是否需要跳转到推送页面。这样发布人才不会影响推送；推送信息也不会依赖发布了。Keep it simple stupid 就是这个意思。

API其实也是强化正交性的利器，它通过接口规范确定了互不影响的功能，又通过接口协议隐藏了后端实现，去除了对实现技术的依赖性。SAE在这点上就受益匪浅。

操作篇

本篇开始讲求职过程中的实际操作。

求职材料

简历的本质

在写简历之前，我们必须清楚的了解一件事情，那就是简历是什么?

它不是人生履历，不是项目清单，也不是技能大放送。

简历的存在只有一个目的 —— 帮你约到面试。只要能达到这个目的，简历可以是一段视频，一个开源项目，一张照片，甚至是一行字，比如:

> I wrote python

当然，绝大部分简历的形式，就是我们所熟知的，是一篇文章。即使你通过其他方式获得了面试，当你入职的时候，还是要有这么一份纸质简历的，所以不要想着偷懒。

简历要说什么

FAB

介绍自己？错。越是好的职位竞争越激烈，光介绍你自己是远不够的，要推销你的自己才行。

一份好的简历，要低调的告诉招聘方，爷很NB。 那么，如何才能低调的NB着呢?

不光要说明事实，更要通过FAB法则来增强其说服力。

- Feature：是什么
- Advantage：比别人好在哪些地方
- Benefit：如果雇佣你，招聘方会得到什么好处

给论据别给论题

写简历和写议论文不同，过分的论证会显得自夸，反而容易引起反感，所以要点到为止。这里的技巧是，提供论据，把论点留给阅读简历的人自己去得出。

论据要具体，最基本的是要数字化，再好点的论据要让人印象深刻。每天PV8个亿，这是数字化；访问量超越Google App Engine，这是让人印象深刻。

下边写一段实例，其中内容是虚构的:

- 2006年，参与了手机XX网发布系统WAPCMS的开发（这部分是大家都会写的）。
- 作为核心程序员，不但完成了网站界面、调度队列的开发工作，更提出了高效的组件级缓存系统，通过碎片化缓冲有效的提升了系统的渲染效率（这部分是很多同学忘掉的，要写出你在这个项目中具体负责的部分，以及你贡献出来的价值）。
- 在该系统上线后，Web前端性能从10QPS提升到200QPS，服务器由10台减少到3台（通过量化的数字来增强可信度）。
- 2008年升任WAPCMS项目负责人，带领一个3人小组支持着每天超过2亿的PV（这就是Benefit。你能带给前雇主的价值，也就是你能带给新雇主的价值）。

这是一个比较基本的FAB的应用，还有很多细节可以优化。

对比体现成长

有同学问，如果我在项目里边没有那么显赫的成绩可以说怎么办?

讲不出成绩时，就讲你的成长。因为学习能力也是每家公司都看中的东西。你可以写你在这个项目里边遇到了一个什么样的问题，之前怎么解决的，之后解决的，新方案好在什么地方，你是寻找到这个新方案的，最终这个方案的效果如何。

具体、量化、有说服力，是技术简历特别需要注重的地方。

PS：不要在简历中造假，技术圈很反感这个，一旦被发现后果会很严重。

工具和模板

书写工具

技术简历最理想的书写格式是Markdown，纯文本可以保证任何书写工具都可以打开它，而你也可以很好的在简历嵌入代码。

但看我们简历的不一定是技术人员，所以直接发送一个Markdown文档过去是很冒险的，我们一般将其转为PDF。

为了方便大家写简历，JobDeer开发了在线简历工具DeerResume，使用它，你可以在浏览器里边用Markdown书写简历并实时预览，最后一键生成为PDF。

这个是DeerResume的简历书写界面：

这个是DeerResume的简历浏览界面：

除了可以生产投递用的PDF，你还可以用它做自己的在线简历。添加上阅读密码后，就只有知道密码的人才能查看了。

DeerResume是开源的，你可以自己搭建（ Github地址：https://github.com/geekcompany/DeerResume ），也可以直接使用我们搭建的多用户版本：http://digitcv.com 。更详细的使用，可以观看视频教程。

不管你是否使用DeerResume，我们都建议你一直维护一份Markdown简历。不要等到找工作的时候才去更新简历，每到一个里程碑，都应该更新简历，这样可以帮你回顾你最近的经历对你职业规划的影响，提醒你多去做值得写入自己履历的事情。

简历模板

为了帮助大家更好的写做简历，我们准备了一份程序员简历模板，你只需要根据每部分的引导，换成自己的内容，就可以得到整体不错的简历了。这个模板我们已经发布到GitHub了，还针对8种热门职位做了关键字优化（用来Hack不懂技术的HR），地址是 https://github.com/geekcompany/ResumeSample 你可以star下来，这样我们有更新的时候就能第一时间知道。

下边是一份通用程序员模板，括号里的是我们JobDeer顾问编写的说明，建议在简历书写完成后统一删除。

联系方式

（HR会打印你的简历，用于在面试的时候联系，所以联系方式放到最上边会比较方便）

- 手机：135** （`如果是外地手机，可注明。如经常关机，要写上最优联系时间`）
- Email：goodman@gmail.com （`虽然我觉得QQ邮箱无所谓，不过有些技术人员比较反感，建议用G`）
- QQ/微信号：6*（`提供一个通过网络可以联系到你的方式`）

个人信息

- 胶布帝/男/1990
- 本科/萌鹿大学计算机系
- 工作年限：3年
- 微博：@JobDeer （`如果没有技术相关内容，也可以不放`）
- 技术博客：http://blog.github.io (`使用GitHub Host的Big较高`)
- Github: http://github.com/geekcompany (`有原创repo的Github帐号会极大的提升你的个人品牌`)

- 期望职位：PHP高级程序员，应用架构师

- 期望薪资：税前月薪15k~20k，特别喜欢的公司可例外
- 期望城市：北京

工作经历

（工作经历按逆序排列，最新的在最前边，按公司做一级分组，公司内按二级分组）

ABC公司 （2012年9月 ~ 2014年9月）

DEF项目

我在此项目负责了哪些工作，分别在哪些地方做得出色/和别人不一样/成长快，这个项目中，我最困难的问题是什么，我采取了什么措施，最后结果如何。这个项目中，我最自豪的技术细节是什么，为什么，实施前和实施后的数据对比如何，同事和领导对此的反应如何。

GHI项目

我在此项目负责了哪些工作，分别在哪些地方做得出色/和别人不一样/成长快，这个项目中，我最困难的问题是什么，我采取了什么措施，最后结果如何。这个项目中，我最自豪的技术细节是什么，为什么，实施前和实施后的数据对比如何，同事和领导对此的反应如何。

其他项目

（每个公司写2~3个核心项目就好了，如果你有非常大量的项目，那么按分类进行合并，每一类选一个典型写出来。其他的一笔带过即可。）

JKL公司 （ 2010年3月 ~ 2012年8月 ）

MNO项目

我在此项目负责了哪些工作，分别在哪些地方做得出色/和别人不一样/成长快，这个项目中，我最困难的问题是什么，我采取了什么措施，最后结果如何。这个项目中，我最自豪的技术细节是什么，为什么，实施前和实施后的数据对比如何，同事和领导对此的反应如何。

PQR项目

我在此项目负责了哪些工作，分别在哪些地方做得出色/和别人不一样/成长快，这个项目中，我最困难的问题是什么，我采取了什么措施，最后结果如何。这个项目中，我最自豪的技术细节是什么，为什么，实施前和实施后的数据对比如何，同事和领导对此的反应如何。

其他项目

（每个公司写2~3个核心项目就好了，如果你有非常大量的项目，那么按分类进行合并，每一类选一个典型写出来。其他的一笔带过即可。）

开源项目和作品

（这一段用于放置工作以外的、可证明你的能力的材料）

开源项目

（对于程序员来讲，没有什么比Show me the code能有说服力了）

- STU : 项目的简要说明，Star和Fork数多的可以注明
- WXYZ : 项目的简要说明，Star和Fork数多的可以注明

技术文章

（挑选你写作或翻译的技术文章，好的文章可以从侧面证实你的表达和沟通能力，也帮助招聘方更了解你）

- 一个产品经理眼中的云计算：前生今世和未来
- 来自HeroKu的HTTP API 设计指南(翻译文章) （`好的翻译文章可以侧证你对英文技术文档的阅读能力`）

演讲和讲义

（放置你代表公司在一些技术会议上做过的演讲，以及你在公司分享时制作的讲义）

- 2014架构师大会演讲：如何通过Docker优化内部开发
    - 9月公司内部分享：云计算的前生今世

技能清单

（我一般主张将技能清单写入到工作经历里边去。不过很难完整，所以有这么一段也不错）

以下均为我熟练使用的技能

- Web开发：PHP/Hack/Node
- Web框架：ThinkPHP/Yaf/Yii/Lavarel/LazyPHP
- 前端框架：Bootstrap/AngularJS/EmberJS/HTML5/Cocos2dJS/ionic

- 前端工具：Bower/Gulp/SaSS/LeSS/PhoneGap
- 数据库相关：MySQL/PgSQL/PDO/SQLite
- 版本管理、文档和自动化部署工具：Svn/Git/PHPDoc/Phing/Composer
- 单元测试：PHPUnit/SimpleTest/Qunit
- 云和开放平台：SAE/BAE/AWS/微博开放平台/微信应用开发

致谢

感谢您花时间阅读我的简历，期待能有机会和您共事。

求职邮件

求职邮件只需要简单的写上在什么地方看到招聘启事，应聘什么职位即可。

最好在正文中写上完整的文本简历，然后在附件里附上PDF版本。这样招聘方更容易查找到你。

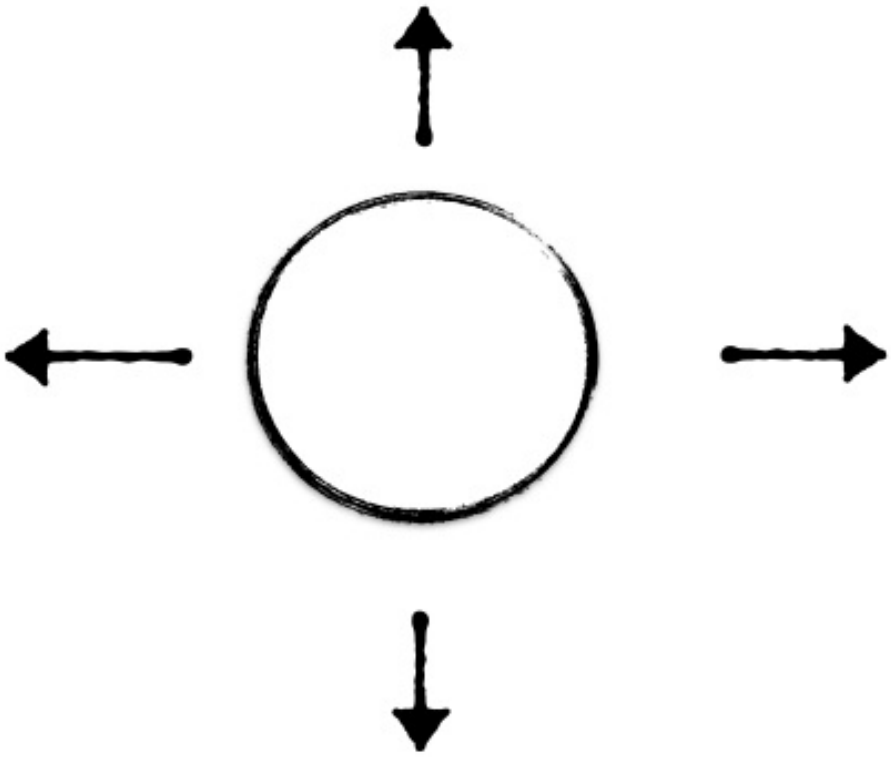
邮箱

如果你要使用QQ邮箱，请确保你的用户名看起来像个程序员。建议使用Gmail或者自己的域名邮箱（QQ邮箱就支持域名绑定）。

文件名

作为附件的PDF不要起名为「个人简历.PDF」，而应该是「 XXX简历.PHP开发工程师.PDF 」，这样当HR分拣简历文件时能第一时间看到你。

求职渠道



人脉：最优途径

在求职过程中，信任是非常重要的，原本冗长复杂的人才筛选和鉴定流程，因为信任，可以简单。所以人脉推荐是最好的求职渠道，没有之一。

如果你要换工作，第一时间是找了解你的朋友，问问他们公司有没有好的机会。可以发短信打电话，也可以发邮件。如果你在大公司待过，离职员工群是一个机会非常多的地方。尽量不要找不认识你的人做推荐，这样没有背书，和后边要讲的直投效果相当。

竞拍：遍历潜在机会

竞拍是仅次于人脉的优秀求职渠道。大部分程序员都比较内向，更喜欢和机器打交道，所以在找工作的时候，能帮忙的朋友并不多。这个时候，竞拍就变成了最靠谱的渠道了。

什么是人才竞拍

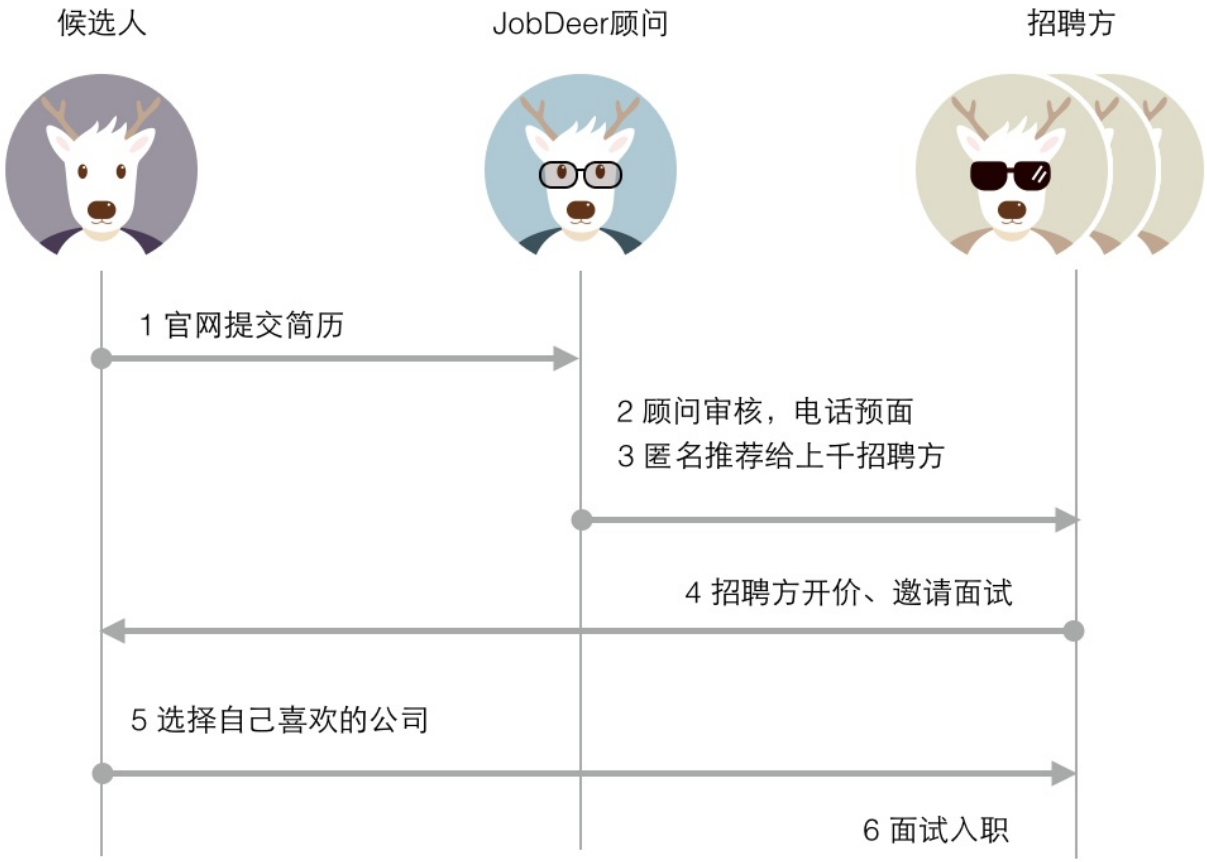
首先和大家解释下人才竞拍这个新兴的模式。最近几年，中高级人才的供需发生了很大的变化，优秀人才开始供不应求，成为日益明显的卖方市场。

传统的招聘网站是为买方市场设计的：企业把职位贴出来，求职者投简历去抢这个职位。竞拍模式是反过来的，把人才放出来，招聘方发送面试邀请来抢人。

模式的反转最大的意义在于，求职者的机会变多了，地位变高了，体验变好了。你看，如果去招聘网站，你需要从N家公司的招聘启事里边挑出自己感兴趣的，然后投递简历等结果。一般投十份简历已经算很多了，你潜在的机会也就在这十家公司里边。

竞拍模式现在刚刚兴起，各家细节不同，以我们JobDeer为例吧。

如果你到JobDeer来，我们会为你准备一封匿名推荐信（我们有很强的隐私保护机制，类似TCP的三次握手），把你推荐给上千家招聘方。



如果他们对你感兴趣，就会给你发面试邀请。里边包括了公司和职位的说明、以及能提供的薪资范围。

除了省下了阅读招聘启事、一家家投递简历的麻烦，最最重要的是，我们帮你遍历了潜在的职业机会。

在招聘网站，你不知道的公司机会会全部Miss掉；而在JobDeer，这些潜在机会会在几个小时内整整齐齐的出现在你的面试邀请页面。

这些机会里边很可能包含了你之前没想过的，但其实特别感兴趣的公司。

还有很多大公司的秘密产品团队以及偷偷拿完投资在做改变世界的产品的创业公司，为了不被竞争对手发现，它们都在JobDeer秘密招人。

不管是朋友还是猎头推荐，都很难帮你在几个小时触达上千家招聘方，而JobDeer可以，因为我们的产品就是为此设计的。

上个月，JobDeer收到面试邀请最多的是一位前端，共计87个；入职最快的同学是一位PHP，早上10点半推荐，中午已经约好4个面试，第二天面到第二家公司时第一家公司的offer已经发过来，当即确定，共计耗时28个小时。

哪些情况不适合竞拍渠道

竞拍渠道都是有门槛的，如果你是应届毕业生，或者经验很少，就不太适合竞拍渠道。另外几乎目前所有竞拍服务都只针对热门职位，所以如果你的职位不是那么热门，找行业猎头效果反而会更好一些。

猎头：求职中的隐私保护

其实相对于招聘网站来讲，猎头的服务要好很多：有专人和你联系，帮你约面试，提供的职位也比较好。

但是这两年由于大量小型猎头公司涌入，猎头这个渠道已经变得很乱了。

比如卖简历，当你把简历给了一个猎头，就等于把简历给了全天下的猎头。我自从2013年把简历给了一个猎头，平均每周两个猎头电话，持续至今。

再比如骗简历，就是其实这个猎头并没有为某大公司招聘，但它假装是，给你开出各种条件，当你把简历发给它以后，就没有然后了。

很多猎头公司都靠上边两种方式来扩充自己的人才库。在这个行业里边，违规行为被当做日常，候选人的隐私完全得不到保障。这也是为什么一般猎头都是有着外国名字的中国人，他们不希望或者不敢让你知道它是谁。

所以如果你万不得已要用猎头，给你提供两个建议。

使用小号

不要在简历中留自己的常用电话号码，可以买一个号专门求职用。

特别推荐阿里通信最近推出的「 亲心小号 」APP，直接安装好就可以申请小号了，免费的当天有效，每晚可续期三天；懒得续期的可以买专属小号，每月5元。这样你在给猎头的简历中可以留这个小号，当找完工作后，释放掉就好，从此再也不用受猎头的骚扰了。

使用来电拦截工具

如果你不幸和我一样，已经把手机号码给过了猎头，那么可以使用360手机卫士或者搜狗号码通之类的来电拦截工具。幸好有了它们，我才可以在不良猎头从不间断的骚扰中正常的工作和生活。

常规渠道

如果前边的渠道对于你来说不好用，那么就使用常规渠道吧。

第一首选是企业本身的官网，然后是技术社区的招聘版，最后是招聘网站。下边是用得比较多的一些网站：

- 技术社区的招聘版
  - V2ex http://v2ex.com/go/jobs
  - ChinaUnix http://bbs.chinaunix.net/forum-32-1.html
  - OSchina http://www.oschina.net/job
  - ThinkPHP http://www.thinkphp.cn/topic/job.html
  - 前端乱炖 http://www.html-js.com/job

- 行业招聘网站
  - 内推网 http://neitui.com
  - 拉勾网 http://lagou.com

直投：绕过HR

HR是一个很苦X的职位，不但要负责招聘，还要管理员工保险、薪资一大堆杂事，所以状态不好时漏掉几份简历实在很寻常。

加上我们都是技术岗位，HR本身是不懂技术的，所以他们也很难准确的进行简历筛选（关于如何添加关键字来Hack非技术HR，前文已经说明过了）。

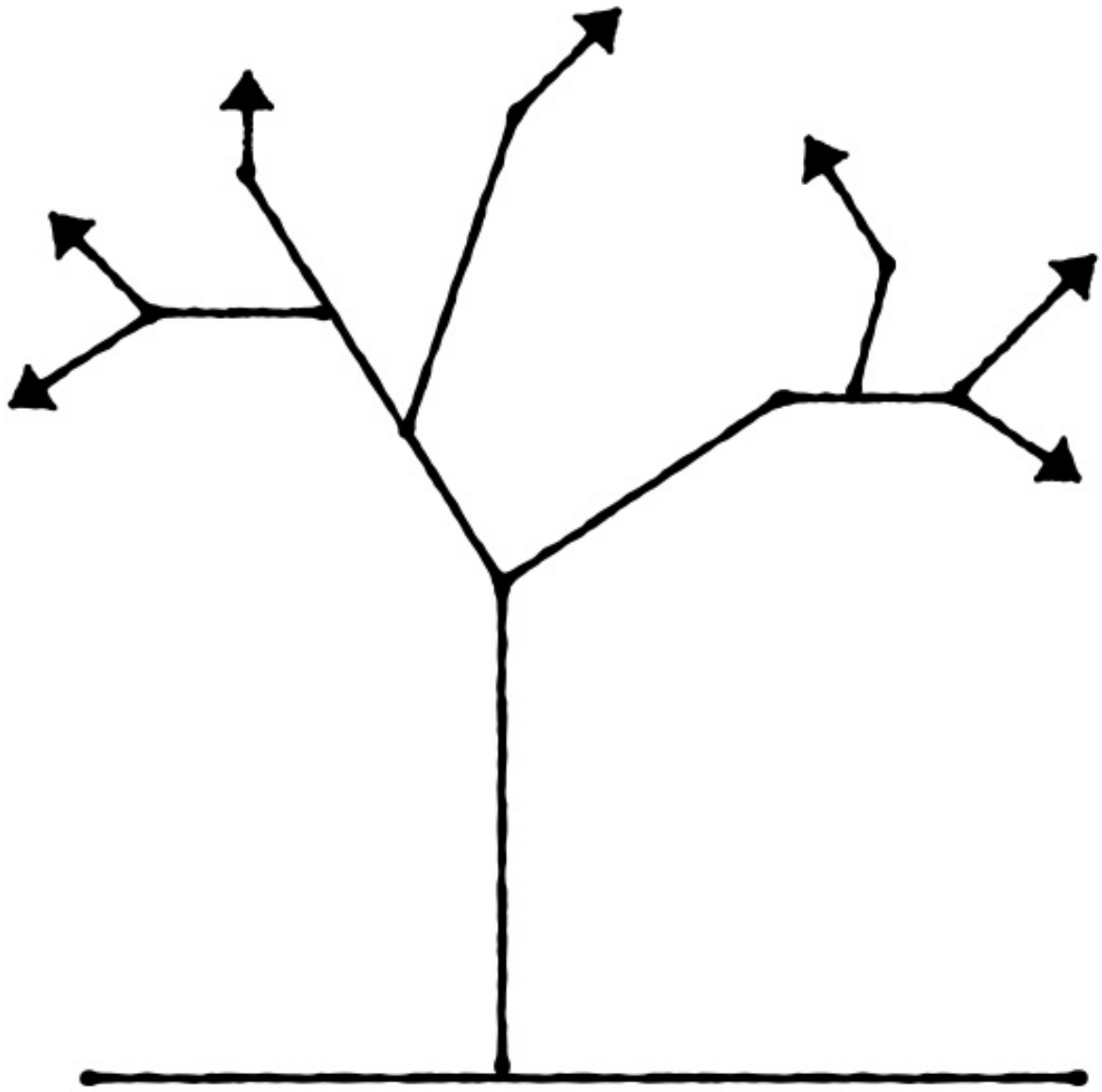
这种情况下，对技术主管进行直投是很好地补充手段。有了微博，要找这些技术主管其实并不难，只需要在微博搜索页面，按公司搜人就可以得到该公司的员工列表，然后发私信或评论给他们，问问他们公司是否招人，是否可以帮忙推荐简历，大部分技术人员还是很nice的。

直投类招聘网站

如果你在微博上找不到合适的人，那么可以去直投类的招聘网站，比如和Jobdeer属于同一家公司的快简历，最近还有一家叫Boss直投的，没详细了解。

求职渠道就为大家介绍到这里，请根据自己的具体需求来选取。

面试准备



知识补全计划

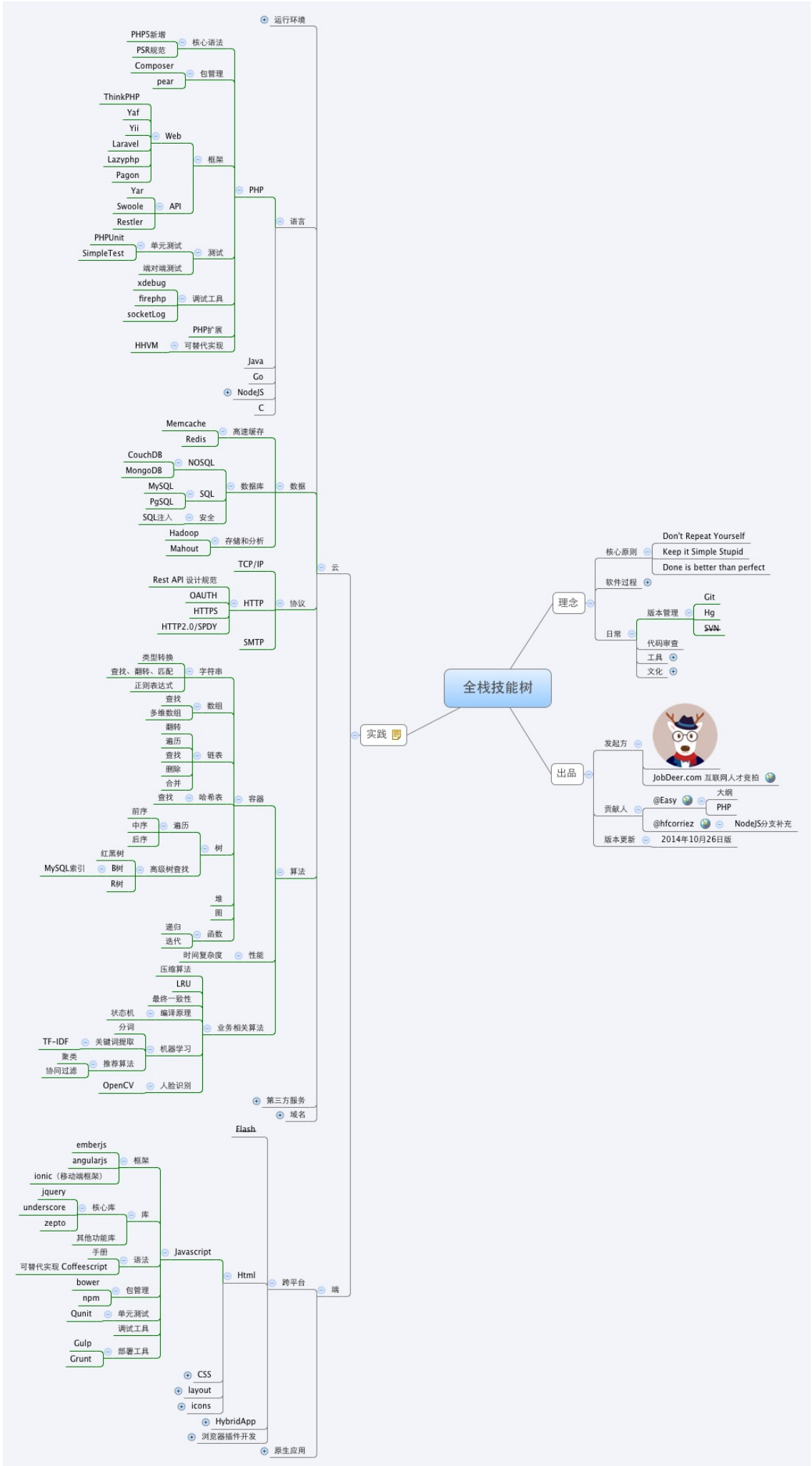
遍历简历知识点

面试时，很多问题都由简历引申出来，所以首先将你自己简历中提到的所有知识点进行复习，思考可能遇到的问题。

面试是对知识的一种测试，而我们日常工作是对知识的消费和积累，它们的侧重点不同。所以不要以为你天天在用的东西就不需要去复习了，要记得在概念和常识层面去复习它。

技能树

为了帮助大家从全局层次进行复习，我梳理了一张「全栈技能树」的思维导图供大家参考。


全栈技能树
理念
核心原则
Don't Repeat Yourself
Keep it Simple Stupid
Done is better than perfect
软件过程
日常
版本管理
Git
Hg
SVN
代码审查
工具
文化
出品
发起方
JobDeer.com 互联网人才竞拍
贡献人
@Easy
大纲
PHP
@hfcorriez
NodeJS分支补充
版本更新
2014年10月26日版
实践
云
运行环境
语言
PHP
核心语法
PHP5新增
PSR规范
包管理
Composer
pear
框架
Web
ThinkPHP
Yaf
Yii
Laravel
Lazyphp
Pagon
API
Yar
Swoole
Restler
测试
单元测试
PHPUnit
SimpleTest
端对端测试
调试工具
xdebug
firephp
socketLog
PHP扩展
可替代实现
HHVM
Java
Go
NodeJS
C
数据
高速缓存
Memcache
Redis
数据库
NOSQL
CouchDB
MongoDB
SQL
MySQL
PgSQL
安全
SQL注入
存储和分析
Hadoop
Mahout
协议
HTTP
TCP/IP
Rest API 设计规范
OAUTH
HTTPS
HTTP2.0/SPDY
SMTP
算法
容器
字符串
类型转换
查找、翻转、匹配
正则表达式
数组
查找
多维数组
翻转
链表
遍历
查找
删除
合并
哈希表
查找
树
遍历
前序
中序
后序
高级树查找
红黑树
MySQL索引
B树
R树
堆
图
函数
递归
迭代
性能
时间复杂度
业务相关算法
压缩算法
LRU
最终一致性
编译原理
状态机
机器学习
分词
关键词提取
TF-IDF
推荐算法
聚类
协同过滤
人脸识别
OpenCV
第三方服务
域名
端
跨平台
Flash
Html
Javascript
框架
emberjs
angularjs
ionic（移动端框架）
库
核心库
jquery
underscore
zepto
其他功能库
语法
手册
可替代实现 Coffeescript
包管理
bower
npm
单元测试
Qunit
调试工具
部署工具
Gulp
Grunt
CSS
layout
icons
HybridApp
浏览器插件开发
原生应用

其中绿线部分是可能会深入考察的，应该多留意。当前版本比较偏重于Web开发，还有很多待细化的地方，期望大家来补充。我们已经将图和源文件放到GitHub上，会持续更新，欢迎Star。

GitHub地址 https://github.com/geekcompany/full-stack-tree

常见面试问题

先说一个小技巧，在和面试官沟通时，如果提到对方公司，尽量用咱们XXX，少用你们XXX。这样可以提升亲切感，在讨论比较尖锐的问题时，更像是内部的探讨，而不是外部的指责。

当然，你要注意下语境，你可以说我想知道下咱们新浪是如何处理Feed流的，但不要说咱老婆今天几点回来……

为什么要离职?

不要说上一家公司的坏话，而要说现在这家公司的好话。两个原因，一是你现在如何吐槽上一家公司，将来也可能如何吐槽未来这家公司；二是要谨记我们提过的因为「未来更好」而跳槽的原则，这会让面试官感受到你是积极向上的。

遇到过的最大的挑战以及解决的办法?

这是试图了解候选人在压力情况下的表现，你可以挑一个最具有代表性的情景来回答，突出自己面临压力时，如何冷静的分析问题和解决问题的。不一定非要讲难度最大的，你可以说「以前的挑战太多，记不太清了，讲一个最近发生的事情吧」，从而将问题降级。

你希望三年后，成为一个什么样的人?

这是对你职业规划的考察，看是否和招聘方能给你的空间匹配。在回答中应该强调成长性和稳定性。

你还有什么要问我们的?

反向提问一般会在面试结束时出现，可以相对随意的问一些你关心的问题，但最好问一个带有「兴趣指标」的问题。

这个问题其实就是为表明你对这家公司的兴趣的，没有人会反感候选人对自己的产品好奇，尤其是创业公司（PS：不要对产品细节挑刺，等入职以后再挑）。

比如你到新浪云去面试，你可以问问新浪云的代码存储是否有排重，按什么规则排重的；沙箱是如何做安全防护的之类。

比如你到JobDeer来面试，，你可以问问我们是如何做周边，不，是如何挑选人才的，如何保证人选那么受招聘方欢迎。

不管对方是否为你解答，你的好奇心对方已经感受到了。

知己知彼

如果你知道面试人是谁（比如是通过人脉推荐或者直投方式得到的面试），那么在面试之前记得仔细读一读他的博客和微博。一方面是为了面试，另外一方面也是看你自己是否喜欢这个主管的风格，万一不适应，可以早做调整。

准时和礼节

和招聘方约定好的时间应该尽量准时，晚到的话，应该发短信告知。如果你已经确定了offer，不想再面试了，应该给之前承诺参加面试的招聘方群发短信或邮件告知。

离职

按新劳动法，员工离职只需要提前一个月书面通知雇主即可，不需要通过雇主同意。但我们还是应该尽可能的做好平滑交接，保证原公司的业务顺利进行，圈子不大，冤家路窄，说不定哪天还要和这家公司打交道呢，和原公司保持好关系是一件有好处没坏处的事。

离职访谈

不少企业对离职员工有例行的离职访谈，很多同学觉得要离职了，就把几年屯下来的槽全吐了。

这里提醒大家一下，可以对具体的规定吐吐苦水，但尽量避免对同事，尤其是上级，尤其是上级的人品发表评论。某些HR很八卦的，最后添油加醋一传，信息来源人还是你。

说什么？老规矩，要讲因为新的机遇比较好，所以才离职；原来公司同事都很好，有些舍不得，所以现在才走；给公司的建议挑两个无关痛痒的提就好了，没必要当真。

另外提醒一下，记得开离职证明，新公司报到时要用的。

后记

为什么我们要自己做职业生涯规划?

记得电影《社交网络》里边，CFO同学在知道自己股权被稀释时说了一句话，「我以为那些律师是我的律师。」

其实我们大多数人对HR几乎都存在类似的误解——你以为她是你的HR，其实她只是公司的HR。她们care的是如何编个理由用老板给的那点小钱留住一个高性价比的人才，而不是真正有助于你发展的职业路线图。昨天还含情脉脉和你讨论人生的知心姐姐明天就可能变成拿着劳动合同逼你主动离职的凶婆娘。和人性无关——这就是她们的工作，越专业的HR越擅长。

所以，你要自己来做这件事情。

每每看见优秀的程序员往往因为太专注于工作，在一个位置长期得不到发展（很讽刺），我都想冲上去，告诉他应该如何求职、如何跳槽、如何规划自己的职业和人身。

但是我也是一个内向的前程序员，不擅长和陌生人聊天，于是我决定写一本书。这本书从2013年底开始规划，陆陆续续花了一年时间。并不是因为内容太多，而是因为期间进行了大量的重写。到今年十月，整个体系算是清晰了，于是我开始补充文章。

说实话，我不能保证这本书里的那些观点结论都是对的，但我把自己思考的方式、使用的工具都详细的描述了出来，买鱼送渔，期望能对你有所帮助。

感谢你的阅读，如果你喜欢本书，请推荐给你的朋友们。

本书的勘误和交流，请访问我们在GET社区的百科页面。

你也可以关注我的微博 @Easy 进行交流。

Easy

二零一四年十月 北京

Table of Contents