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。

@Easy

二零一四年十月 北京

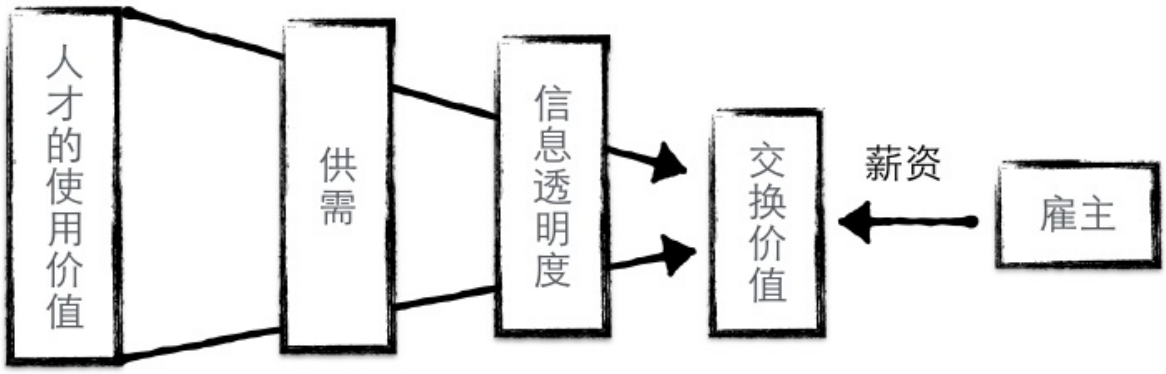
原理篇

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。

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。

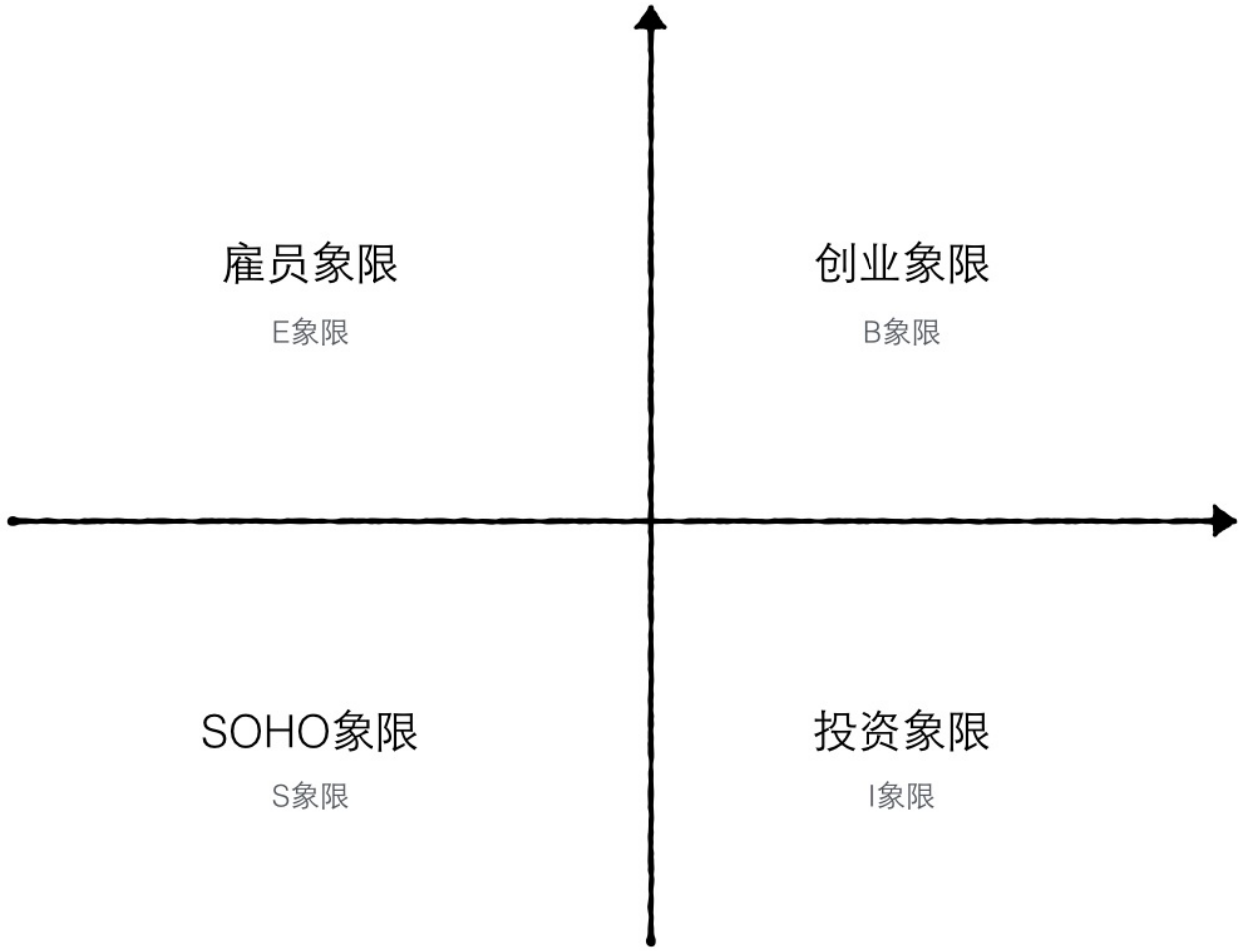
人生有时候就像一个没有终点的旅程，有人的意义是行程的边界，有人的意义是沿途的美景，有人的意义同行的伴侣。当你定下一个目标，人生就变的有了意义。

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。

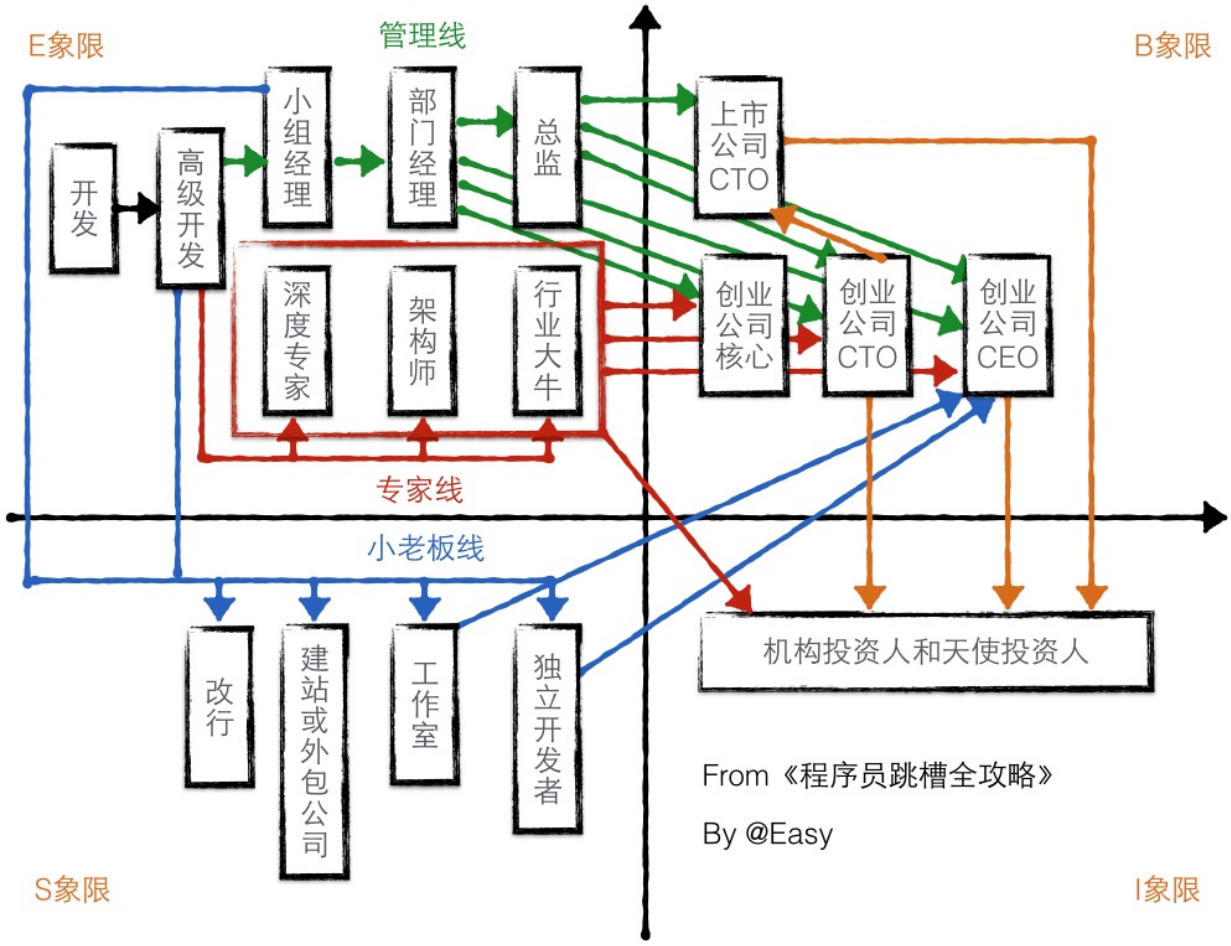
I象限 （ 让钱挣钱

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。

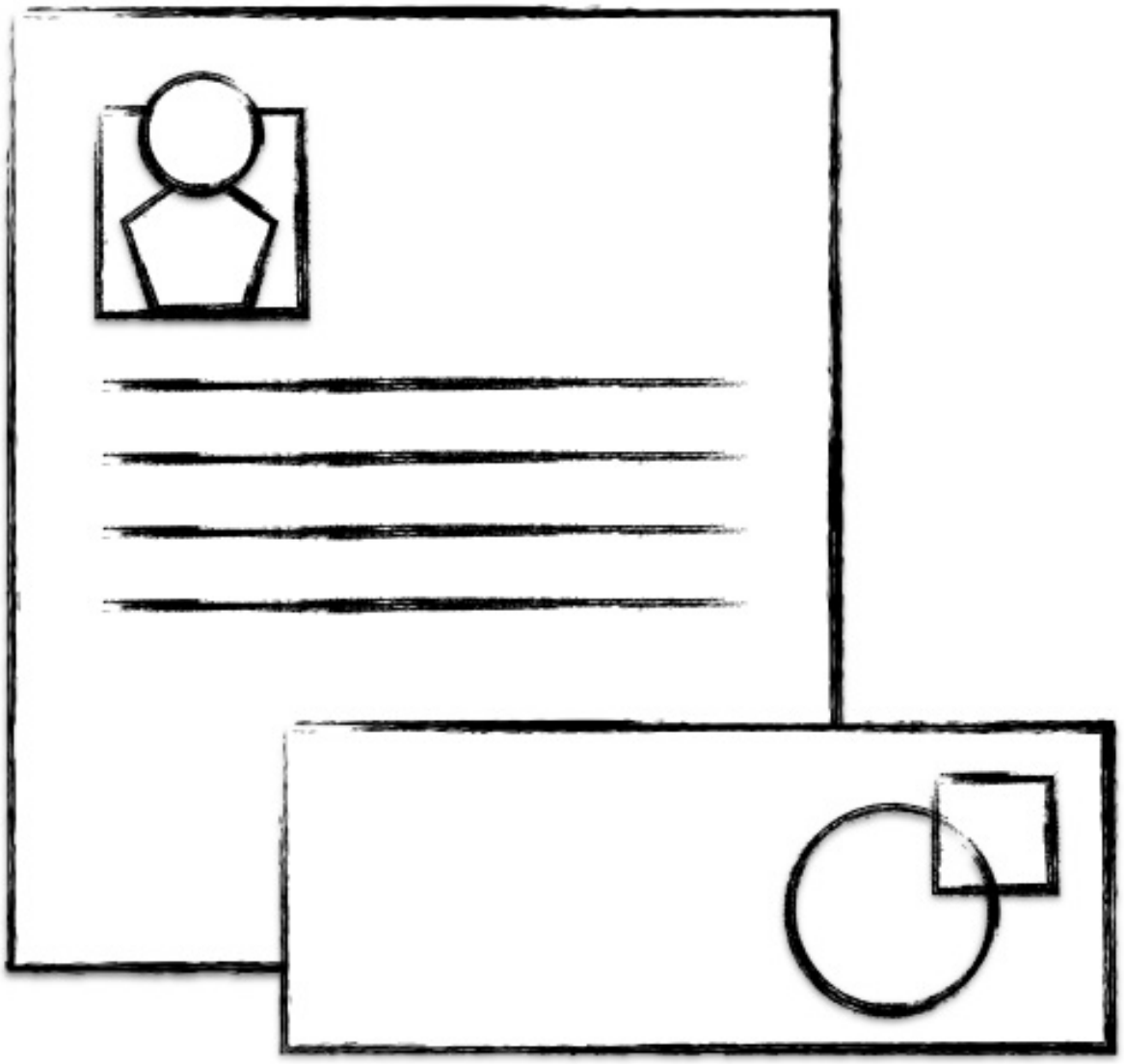


注：此图的灵感来自陈晓峰在创新工场演讲中分享的一张图。

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。

完整版目录

Table of Contents