跨界、困惑与架构师跨界、困惑与架构师

程序员成长之：跨界、困惑与架构师

1. 如何看待“大跨度入行编程”——跨界程序员？

从我的角度来说，角色的变换，是一个从兴趣到职业的过程。兴趣可能持续一段时间就不再感觉兴趣了，就放弃了，但职业不一样，是担起的担子，是对一群人包括自己的一种责任，有些事不喜欢不仅得做，还要做的漂亮。

从法学院出来当程序员其实并没有想象中那么难 。在学校那段时间，晚上经常一写代码就忘记时间，看到外面天亮了才睡觉。从不知道如何压缩 PNG 到解决编码上的问题。那时候似乎不需要技巧，一切都是新的，每个点都是新的，凭着一股单纯的热爱去学习，通宵所感受到的不是疲惫反倒是一种享受。但工作了，要做好却并不像入门那么简单。一方面是兴趣消退，一方面是工作并不能像兴趣一样只做自己喜欢的事，也不能随意创作，需要按照画好的 UI 和预定的 UX 来做。所以开始需要一些技巧，或者说一些工作的信条。

工作信条方面。到现在还印象很深刻的是，刚工作那会对于任务的第一感受是「这个网站究竟要做到几时才会停下来?」，一切都似乎没有终点，人也在工作中变行焦虑起来。这是很难受的一件事。万幸的是，多方的影响，从完成一件任务的心态慢慢变成如何打磨一个拿的出手作品。这也是现在一直在实践的一个技巧：一方面要求自己不只是简单地去完成一个任务；另一方面是不要只跟同事做比较，写文章、开源代码、出作品接受更多人的挑战。

年轻的时候一直有一个疑问没有解开：人与人之间能力是否真的相差十倍甚至百倍千倍？因为武侠小说或者电影里的英雄通常能以一敌十，而现世生活中人与人之间积累的财富更是有巨大的差别。后来慢慢解开，正常情况也是，社会资源的流向总是倾向那些准备多一点点，优秀多一点点的人。比如学习成绩好会加分，加分会考上更好的学校，好学校会得到更好的教学环境和更优秀的同伴，诸如此类。

工作也一样，做的多承担更多责任，承担更多责任获得更多回报。除了工作三年积累二十万拿着爸妈给的六百万全款买了上海中环内的房子这类非正常的方式外，能者多劳、多劳多得这种套路一直没有变过。

从学习方法来说。入门一个新领域都会挑选一本比较系统的书来看，读懂后开始动手做一个相关的项目。比如 CSS 看《精通 CSS》，JS 我会推荐《Pro JavaScript》，设计有一本叫《写给大家看的设计书》是一本不错的入门书，后来学 SQL、PHP、Go 一样，刚转到管理岗那会也是看了好几本管理书边学边用。似乎总能得到不错的结果。

关于如何从非科班生变成技术总监。大家的方法都不同，但有一点一定是相同的 —— 不设限。如果大家只看法学方面的书，就不会写代码，自然不会成为程序员更别说技术总监。前端可以学点设计，自然也可以学后端，甚至操作机器。去年得了公司的管理奖，算是对自己管理上的阶段性肯定。因此上半年给自己定了个目标是看懂财报，学习从会计的角度是如何看公司的。

2. 如何看待“编程终极目标”——架构师？

我曾问过很多自称热爱代码的程序员的发展规划，大多都回答说期望成为一名架构师。而在招聘一方，有的团队会过滤掉多次提起架构一词而一点不提具体内容的简历。可见，虽然在大多数程序员眼里，架构师是神圣的，但又不得不承认事实是：“架构”和“架构师”是最常被滥用的。那些写能 PPT 而不能写代码的人，只做和事佬而不考虑软件快、稳、便捷的人，都称不上做“架构”更别提“架构师”。

那么什么样的人可以称为“架构师”？

据称架构一词源于建筑行业，架构师这个职位，不管是前端还是后端，职责是相同的。而用规划一次房屋的装修来描述架构师这个职位的职责是非常合适的。

建立一套 Web API 就像在定装修风格。要选择注重重 CRUD 的 RESTFul 式，还是请求自定义性更强的 GraphQL 式，又或者是简单的 JSON-RPC 式，这就像装修风格是选要简洁的日式、粗犷的美式还是奢华的欧式。定方向和选型这件事无处不在，架构师必须根据实际需求，做各种决策，为后面各部分整体结合

打好基础。

灯光、墙面、家具等各个部分都需要根据风格精心设计、执行和不断修正，才可能达到原定目标，架构也一样。拿光线控制来说，施工人员可能会忽略你注重的一些细节：暖色的书房氛围；明亮且能切到影院模式的客厅；装在合适位置才不会刺眼的背景灯。在每个环节的执行上，架构师既要设计，又要保证对每个角色充分理解，必要时不排除动手编写重要环节的功能，而在经验或考虑不足的点上一旦出现问题就必须迅速调整。空有一个好的设计而没有好的执行，是非常让人惋惜的。

值得一提的是，选用最好的卫浴用品、最贵的过滤器并不是获得最佳洗浴室体验的关键点。同样，软件架构并不是说把每个部分做到最好再拼凑起来就能达到佳效果。最好洗浴室体验的关键点在于折中和妥协。例如，在水压不是特别高的情况下，把过滤器安装在总闸虽然能让用水达到最健康的状态，但会导致淋浴的水压不够，进而使体验大打折扣。把过滤器安装在厨房出水口可能是最佳的平衡，既保证水压又保证了用水的健康。分成多个部分是解耦，而协作的平衡是内聚。低耦合、高内聚是架构师处理软件各部分协作的终极目标。

装修有很多细节，例如，若不喜欢晾衣服且生活在有“黄梅天”的上海，可选洗烘一体机；房子面积不大，可选扩展型家具；对通风质量要求比较高，可安装新风系统。软件架构也需要考虑很多细节，例如客户需求、实际环境、技术可用黑科技之类、安全、重用、扩展等。而这些细节方面的考虑，并不是一个刚入门的新人能做到的。

总的来说，称得上架构师的人，必须是具备丰富系统设计经验且能保证设计执行的设计师和决策者；必须参与设计、开发执行和测试但又不局限于一个角色。也许架构师并不一定全是这样，这仅代表个人看法和期望。

3. 如何解答程序员的成长困惑？

跳槽如何选择？

跳槽这件事我并没有什么话权，裸辞和看缘分似乎是我一直的方法，当然谈薪资也一样，所以不止一次被老婆说根本不懂得如何去赚钱。不过从选择团队上，我有一个不错的建议，也是我自己选择的底线，两种团队：一是选择喜欢的一个产品；一是选择一个有趣的团队。

选择一个喜欢的产品。如果你喜欢饿了么让全世界都足不出户就可以享受各种美食，那么有什么理由不加入？工作就是人生追求这件事，简直求之不得。喜欢一个产品总是很直接的，所以选择起来非常简单。

为什么要选择一个有趣的团队？一个被业务缠身的团队是无法有趣的，没时间；一个没有水平的团队是无法有趣的，烂事一堆都焦头烂额了，哪来有趣？有趣的团队一定是好玩的，且热爱生活的，这样的团队通常不仅可以学到新的酷的，还会得到一群有趣的朋友。但如何判断一个团队是否有趣呢？看他们的作品，是不是总是为了生活更简单，比如喜欢自动化，喜欢抽象，交互简单且有效等等；或者看他们是否有很特别的方式在玩，比如找个海岛、包个酒吧之类，或者去深山里玩狼人杀。

无论是否有这样的团队，我觉得有一点比选择更重要的是，成为一个某个产品或团队因为有你而变得更好的人；如果没有这样的产品或者团队，那么我的方式是 —— 创造一个。

技术和管理如何互转？

关于技术和管理如何互转上，很多人觉得因人而异，照自己的喜欢的来选就可以了。做了这么久的技术，转到管理岗于我，就像从法学院毕竟转身成为程序员一样，是一个新的挑战，当时义无反顾接受。如果你也一样，碰到一些两个单选都可以的情况，照我的看法是，既然生命只有一次为何一直重复做同样的事。事实上，每次选择的结果似乎都还不错。

顺路分享一个有趣的心路历程。上上周人在美国 LA，当时第一次在那边开车，陌生的 JEEP 大切诺基、山路、晚上，并且中间被警察拦下来过一次，不过这些都不重要。重要的是在公路上大家都开的超快，当时我也开的特别快，朋友说刚想睡就被我一个弯就惺醒了。事后大家都觉得我开车真的有点太吓人。而当时心里却只有一句话 —— 如果一切都在掌控之中，说明一切跑的并不足够快。如果生活总是在掌握之中，说明我们跑的并不足够快，甚至只是原地踏步。

其他就不说了，分享一个初做管理的心得和一个成就优秀团队的重要的策略。

心得。大家都可能摸清了套路，写代码你只要做的比预期多一点点，代码写的抽象一点点，大家都会开始叫你大神，说你就是那样棒棒的。所以做管理的时候，碰到别人不会的问题，你通常过会想 —— 「走开，让我来」。千万别这样做！不管你懂或者不懂这句话的真正涵义，千万别这样做。然后，你就会开始得到很多很好的助手。这个就不多解释了，有些事可以边学边做，越受打击越强大。

策略。永远不要静止地看团队，特别是在招人上。什么叫静态地看待团队？举个例子，有多少业务申请多少资源，然后业务永远都在原地踏步，团队永远在只是一个建立时一样的团队。我的策略是这样的，一方面半年前与半年后招聘同一级的人一定要比原来的要求更高，且重点挑选团队缺少的人才；一方面看半年到一年后希望变成一个什么样的团队，而去做相应的 5% 左右的人才储备。看似浪费资源，但长远来看，相比因为招不到人支持不了业务的时候才是真正的浪费。

统观大前端之概念及团队落地

4. 如何看待大前端的概念？

大前端通常是指所有客户端，因为会有 Native App 和 Web 两个前端；另外还泛指不仅仅是负责静态内容，而是向后端扩展负责更多的内容，比如除 Model/DB 层以外都称为前端。简单来说就是前端及接受前端的层。方面其实不是很重要，如果喜欢跟 UI 打交道，那么选择大前端就没错了。

在参加大咖说的直播过程中，有观众问到一个问题 —— 前端会消亡吗？从社会分工的角度来说，似乎目前还是比较稳定的职位。但现实世界变化特别快。以前我们看父母辈都可以在一家公司工作上 2、30 年，而今一家公司是否能存活这么久还是一个问题，几乎可以说，这个世界上唯一不变的就是变化，而且越来越快。所以无论选择什么方向，一方面尽力做到最好；另一方面不要给自己设限，去接受更多挑战以提升自己；这样可能是比选择一个方向要更靠谱的方法。

5. 饿了么大前端团队的定位是什么？

（1）为什么叫“大前端”团队

大前端这个词最早是因为在阿里内部有很多前端开发人员既写前端又写 Java 的 Velocity 模板而得，而我们部门成为之初所承担的内容不仅仅是前端，还包含 CDN 和 Nginx 层，所以取名“大前端”。时至今日，大家所说的大前端已经包含了前端、Node、Native-Like (Hybrid / Weex / RN)，甚至包括 Native App 开发。

（2）我所理解的“大前端”

在我看来，**“大前端”是一种变化形态多过于固定的职责范围，是“前端”职责范围的延伸，是对这个社会分工因为能力范围和因此所达到效率提升的一种进化形态**。给大家分享个小道消息：CTO 曾多次开玩笑说 —— 你们负责的已经不仅仅是前端，要不就改名“大全栈”吧。这部门的名字很霸气但同时也太高调，所有并没有接受 BOSS 的提议，而是继续沿用“大前端”这个部门名。

（3）饿了么大前端团队的职责

如上面所说的，除了纯 iOS / Android App 的开发，其他都是我们团队职责所在，同时我们还负责公司 HTTP API 层，有一些自己运维的系统。

从分工来说来，目前我们有架构与机动组负责框架、规范、工具的生产；Node 研发团队负责公司 Node 业务的基础设施和业务支撑；多个业务前端团队支持不同的 BU 前端。这里值得一提的是，**架构与机动组会对每个业务团队至少派出一名架构师长期、深入地了解业务团队所遇到的问题并反馈到架构团队，并在解决方案提出后协助推动**。

在具体职能分工之外，**各团队也有以项目而组织起来的虚拟团队**，比如由我们部门负责的“游戏中心系统”就由 Node 研发团队和架构与机场组中的成员组成；又如小程序团队；亦如发起一次由“93 后”独立组织的招聘；专栏编辑团队、官微小分队、对内对外分享会小分队，等等。除了大家看到的开源产品，内部的所有项目都是“内部开源”，特别鼓励大家提 Pull Request 和相互 Code Review。这些与部门所创建的文化息息相关，且如你可能猜到的，**大多合作都是一旦有人提出，即自发组队**。

6. 饿了么大前端如何看待和落地新技术？

我们是如何看待新技术的？在面试前端 Leader 候选人的时候，我通常会问一个问题：你如何看待技术债，有没有办法可以避免？几乎任何程序员都讨厌还技术债，所以才会有那句“挖坑一时爽，填坑火葬场”。因为

痛苦才会非常值得我们去思考和解决。技术债从某种程序上代表着过时的技术，而新技术也将在未来某个时刻变成新的“技术债”。饿了么大前端是如何回答这个问题的？就是我们对新技术的看法。

我 2014 年加入饿了么，那会 PC 和 Mobile 都还是后端渲染的模式，使用 Bootstrap 和 jQuery。我进去的**第一件事是用 Angular.js 重写移动网站，并且前 / 后端分离**，提倡后端即服务。在高速发展期利用移动网站支撑了当时 10 倍增长的业务；**第二件事是重构 PC 站，把 web2 升级到 web3（Code Name），同样是前后端分离**，到 14 年底 15 年初，基本实现完全分离。重构一方面是提高前端协作的效率，一方面是提升两部各自的掌握力 —— 只要 API 约定一致，内部封装可以自己随时变更（提升）。**在此之后，我们的方向一直是比较激进的技术模式 —— 新业务可以用任何框架，大家自由选择**；旧业务只要负责团队（人）有能力升级，那就鼓励用最新的。由于后端已经完全分离出去，所以从掌握力大大提升，加上这种受鼓励的激进技术模式，Angular.js 1.x 这种当年的新技术在日渐变成技术债的今天，也已经几乎全被重写成 Vue.js 和 React.js。

当然，也像今天大家能看到的，**当大家都还在转发关于 PWA 文章的时候，我们已经和 Google 合作并把 PWA 上线**；开源的项目大多是内部成熟应用的项目，而开源的产品亦让我们成为 Vue.js World Ranking 最高的团队；Weex 方面，我们是除阿里内部团队外最早上线的大型用户。**这些看起来快速和无止追求新技术的背后，其实并没有大家想的那么了不起，仅仅是因为团队文化本身就鼓励利用新技术解决问题**。

如果一定要拿 Vue.js 来举例，可能和你想象不一样的是，不用“落地”，**仅仅是因为有人说了一句“WOW，Vue.js 写起来好简洁啊，大家要不要一起来试试？”。然后，一个团队，两个团队... 几乎所有团队都开始应用，几乎所有新项目都在用**。一位 IBM 的朋友告诉我，他申请在项目试用 Vue.js，上级说在半年后试用，结果半年后又被推翻了。所以你看，在合适的文化土壤里新事物就是一种常态，如果做一个项目用什么技术还要上级主管确定“能不能做”，那本身就不是一件简单的事。

我们 **对于技术选型通常来说要求是 —— 是否提升饿了么运行的效率或者团队开发的效率？是否能 hold 住？有没有人负责到底？符合这样的条件，就会被推动**。当大家都在说 HTTPS 是好东西的时候，我们已经推动全网上线，诸如此类 —— Webp、Https、Vue.js / React.js / Angular.js、Weex、PWA 都是如此。比如大家可能去年就关注到我们在上线 HTTP/2，而今天饿了么大前端内部已经做过 HTTP/2 Server Push 的实验，可以想象在不久的将来，线上应用将会大面应用。这就是我们的选型和落地模式。

7. 饿了么大前端团队的特色是什么？

特色？如果只用两个字来回答就是 —— 散养。但仅仅这样描述大家会一脸问号。外部对我们的评价是：“新技术跟进好快啊”、“怎么又又又出去玩了”、“下班很早”、“好多大牛啊”、“开源的东西获得好多星星啊”，诸如此类，但这不是我们要特地我呈现的，只是一种表像，或者说是一种副产品。

一个团队的风格与创始人有很大的关系，比如喜欢愉懒的我会更多考虑自动化；又如有洁癖所以会有代码规划和 Code Review；还有大家看到的爱玩，所以会经常有团建、下午茶这样的文化；但另一方面，**我并不想让团队仅仅是和我一样有大家喜欢的，同时充满缺点，而希望是不断尝试的，兼容并包的**，让每个人的闪光点都成为集体中有特色标志的。所以我有自己的一套，就是前端所说的“散养”式管理，说起来可能会很大，重点说几点：

1. **招聘最好的人**。最好的人不是业界里的所有明星，更重要的是 **能从某方面给带队带来提升的人**。这些人通常自驱能力强，只要有一个方向就能推动事件的发生和发展。加入的人会被要求不要以学习姿态加入这个团队，而是以加入会让这个团队会让其变得更好为姿态，成长就会成为副产品。
2. **鼓励创造结果而不是追求上班时间**。如果我们的目标是页面加载时间不要超过 800ms，那么目标就是 800ms 而不是上 12 个小时的班。
3. **营造环境**。我们有最好的人，我们追求结果而不是追求上班时间，我们鼓励主动和主人公意识，我们创新以打破规则 ，我们声明所有人为自己而生为用户工作而非老板，我们会包个酒包或找个海岛玩到天亮。有很多东西是要刻意去营造的，创新土壤，主动的意识，热爱生活的文化，鼓励什么就会

聚集 / 培养一群什么样的人。

因为这样的管理方式，通常大部分事情都会被内部很好地解决，而我也得到更多的时间去思考如何做和决定做什么；团队也因为成员不断成长闪耀不同的光芒而变得更好。如果以官话来说，就是我们要发现一种“可持续发展”的模式。这种模式目前运行的不错，无论是业务上的，还是团队文化本身，抑或是加入成员的成长，都是让人高兴的。但，更好的方式仍在探索，如果说只分享一个点的话，那就是千万别用“管”的方式，而是“理”顺，就会顺理成章。

至于是不是盲目追求新技术。上面我们已经谈过技术选型的要求，最最重要的也是最最根本的问题“是否进升饿了么运行的效率或者团队开发的效率？”，我相信如果大家能回答好这个问题，就解决了“盲目”追求的问题。

最后说一句，**无论是管理上、技术上、生活上，预留一定的空间和自由度，一定会带来惊喜**。具体就不解释了，大家自行理解，或许某天我们遇到可以用这个话题开场，就开聊起来了。

8. 大前端模式的利弊有哪些？

“大前端”模式的特点前面已有提及，即是 **对前端本身的一种能力范围延伸**。移动开发团队，我这里指包含移动网页、Native-Like、Native App 这样的团队，如携程；纯大前端的团队如美团和饿了么北京研发中心，只要是客户端的无论是网页还是 App 都在单一个团队；饿了么不仅有大前端，还有各 BU 的 Native App 团队，甚至还有专注于移动基础框架的公共的移动技术团队。

不同的分工模式，其利弊通常与公司状态、团队本身所创造的价格有很大的关联；虽然大家都是“离用户最近的工程师”，但没有公式可抄。

就拿饿了么大前端来说，**最开始是因为业务的快速发展，除了 C 端部门，其他新成立的部门前端工程师极度紧缺，为了资源的高效配置，才成立了大前端这个部门**。这是当时公司业务的状态。

创始人和 CTO 曾问我：“你觉得如果今天合了，几时会分？”当时，我的想法是，如果 **一个业务前端团队发展到 10 人左右，在负责好本身的业务外，能建立且不断升级自己的技术基础设施又具备有自己的文化，是一个分的时机**。时至两年后的 **今日，我已不再是这样的想法**，并且新成立的 BU 更愿意把人放到大前端。

这是为什么呢？我们从以下几点分析：

1. 如果一个大团队，并不能提便利的基础设施，不能创建自由且充满可能的文化，不能持续提升自己并帮助成员提升其自身水平。也就是大家经常谈到的技术追求、归属感、成就感。那么，打散到业务组可能更灵活。所以 **前端业务团队的分与合归根到底在于 —— 大团队是否创造比小团队更高的价值**。
2. 大多数人高估了“前端 + 后端 + 产品”坐在一起的效果，认为这样就能完美解决问题。很多时候，**对于程序员来说更少的打扰，更多的思考**（比如坐内部电梯去找某个人太慢了，就会开始思考是不是有必要去找某个人）**过后的沟通，是更高效的沟通**。
3. 划分框架、机动与业务团队，一方面基础设施共享（框架 / 工具）有更多重用，人员调度可以省不少资源（小团队 _ 需求少的团队可以合并支持）且又有专注于业务的团队，似乎是最前非常不错的划分方式，而在 10 个人的团队中是很难做到的。

由此，我们可以看出，**一方面是业务影响，另一方面也依赖团队本身产生的价值**，今天我们要分还是合，其利弊计算出现在决定做出之后，带领团队的人能否利用“合”的优势去产生出更大的价值。这个问题的答案就是利与弊的答案。

9. 如何看业界大前端团队的现状？

我们团队每年都会以个人或者团队名义邀请多位前端业界大牛来内部交流，也会组织内、外部交流会，这通常是几个电话和微信就搞定的事，总体交流还是比较多的。即使没有专门的会议，也会偶尔相互通通气。

我没有具体统计过业界现状，但从我这边交流过的团队来说，**现在很多团队都是大前端方向，或向这个方向发展的比较多**。有少部分像携程、腾讯这种体量本身就很大职能划分也比较明确的公司，做的事可能是“大前端”但分开仍是比较偏向于 JS+HTML+CSS 这样的纯前端模式。
这里也期望读者所在的团队，如有新的实践和想法我们可以偶尔探讨。

10. 要不要组建大前端团队？

前文也有提到，**要不要组建大前端团队，一方面是看业务是否有需求，另一方面是看合能否比分开带来更大的价值**。
除非人极少，通常来说业务大多数情况都会随公司的发展变得越分越开，而价值则主要是关于人和文化。
目标不是为了合而合，或者追随业界模式，而应该着眼于是否优化了公司的人才架构从而优化业务。
如果一定要从具体实施点上来说，这里说两点：

- 框架团队和业务团队应该同时设立，并且框架与业务不能相互脱离，具体可以参考饿了么大前端的模式 —— 相互渗透；
- 在大职能团队下，尽可能以业务划分团队，不要以职能划分团队；反之亦然。