走心分享：前端 Offer 提速之如何写出有亮点的简历

技术漫谈 2021-12-10 08:32

以下文章来源于Tecvan ，作者范文杰

Tecvan

All or nothing, now or never 👉

声明：本文所有观点均代表我个人意见，不代表与公司、部门、团队立场/意见，关于内容的对错、价值观由读者自行判断，欢迎友好沟通探讨，希望能和而不同。

技术漫谈

有知识，有温度，有态度。技术漫谈，新一代技术人的聚集地！

91篇原创内容

公众号

先来个灵魂拷问：**「你与他人相比，有什么能形成明显区分度的优势条件？」**

这里有两个层面的问题，一是**「如何识别出你的优势条件」**，毕竟大多数人大多数时候可能都是在做业务，临到写简历的时候要求总结日常工作中跟别人不一样的点，确实挺难的，怎么办？第二个问题是你可能已经挖掘到自己的优势，但是**「在简历里面怎么组织内容，怎么表达才能突出，让面试官迅速 get 到点呢」**？

这事并不容易，我见过的不少简历，特别是5年以下的同学很多都写的不达预期，有一些是真的平平无奇，有一些是明明有不错的经历，但就是没有表达出来，几分钟内很难 get 到亮点。最近刚好有不少人找我内推，我都会尽力帮着看看有没有什么明显的问题，在沟通过程中慢慢总结出了一些共性问题，于是有了这篇文章，希望能帮助正在或即将找工作的同学。

本文不会讲太多基础问题，例如格式、字号、字体等问题，这些网上已经有很多文章，没必要重复讨论。本文会更聚焦于内容，聚焦于**「如何在有限篇幅内突出你的个人优势」**，包括如何在日常工作中挖掘亮点，如何组织语言让面试官能够迅速理解你的亮点，以及需要避开那些可能会造成负面效果的坑。有任何想法意见欢迎留言讨论，如果对你确实有帮助希望不要吝啬您的赞，这对我很重要，能激励我持续写更多文章。

如何挖掘亮点

重点是持之以恒的记录，积累足够的素材。在此基础上学会识别对于求职来说什么是加分项，什么不是。

坚持记录

写作需要素材，写简历当然也需要素材，简历的素材就来自于我们日积月累的工作，可以养成习惯，有意识地将一些经历以文本的形式记录下来。

记录的方式有很多，比如技术博客、项目日志、年度总结甚至是周报，这种书面形式的留存总结能够随时 review，所谓好记性不如烂笔头，这些信息最终可能就变成你简历的重要素材。

当然，也没必要事无巨细记流水账，可以把有限的精力放在一些重要节点上：

- 项目启动时，技术选型的过程、思考、论据、结论
- 项目结束时，执行过程的复盘、反思、重点难点、数据指标
- 使用开源框架遇到问题时，调试过程、逻辑推导、解决方案
- 学习新技术时，
- 解决性能问题时，优化前后有多大的提升、具体有哪些优化措施，用了哪些工具，如何实行

这些节点都是个人成长的良好契机，把它们记下来，记录下你在这个过程中都遇到了哪些问题，分别是怎么解决的，写简历的时候翻一番总比凭感觉回想靠谱得多。

识别亮点

在积累足够多的素材之后，就可以根据面试的公司、业务、目标岗位从素材中选取更可能被面试官相中，也就是所谓“亮点”来组织简历了。

亮点应该是那些能让你显得与众不同的经历，比如说：

- 做过一些深度的性能优化，并且有比较大的性能收益，能量化提升空间的
- 做过一些业务逻辑特别复杂、业务影响力特别大的项目
- 推进过一些制度、工具，深刻影响团队乃至整个公司的工作流程、工作方式，且整体有提效作用
- 用一些不太常见的技术，解决过对前端来说比较偏门的问题，例如视频直播

- 做过有一定名气，能真正解决技术问题的开源项目，demo、awesome-xxx 类的不在此列
- 深入学习一些工具的用法，以此解决了一些工程化、开发效率、性能方面的问题
- 给知名开源项目，提交过真正复杂有意义的MR，typo 类修复不在此列
- 钻研过一些框架的原理，并能持续输出足够多的有技术深度的文章，或者明确解决过项目中出现的复杂问题
- ...

这个列表还可以继续罗列下去，不同人，甚至同一人随着经验、认知的增长在不同时期都会有不同的判断标准，所以这里没有标准答案，尽力就好。

什么不是亮点

梳理过程要注意避开哪些不能给你加分的信息，要理智地反思一遍，这段经历是否足够复杂？是否足够表现出你的最高水平？对于这里面用到的技术，你真的掌握的很好，能应对面试吗？

这里也列举几种反模式：

- 单点技术突破不算亮点，例如解决了某个 UI 框架的单个样式bug，体量太小
- 做了很多项目，不能称之为亮点，只能证明你可能已经工作了很久
- 技术框架、工具一直停留在用的阶段，对内部实现原理完全不清楚
- 仅仅解决一些很寻常，很普遍，网上有大量现成方案的问题不能算亮点

如何表达亮点

积累足够多素材之后，接下来需要探究一下如何通过简历高效传达给预期读者。面试官通常都很忙，特别是很多大厂面试官可能每天要浏览几百上千份简历，如何组织内容才能高效传达你的信息？如何在短时间内抓住面试官的注意力？更进一步的，如何引导后续面试的内容？

首先是基本格式，这方面比较简单，上网找个你觉得最简洁清爽的模板就行了，我个人比较喜欢这个。基本格式之外更重要的其实是内容，如何在短短一两页内呈现你的能力、专业度、人设等，下面展开聊聊。

树立技术人设

所谓人设，可以简化理解为我们做过什么，以及我们将要做什么。

做过什么

落到简历上，通常需要以项目经历、掌握技能这两个角度呈现，项目经历是简历的核心组成，大多数面试官都非常看重这一part，千万不要盲目写，要有条理，有次序，有重点，我个人总结出几条规则：

- 有意识地挑选几段能突出某项、某系列技能的项目经历，例如你要突出 vue ，那么就应该尽量围绕这个主题展开，避免一会是vue，一会是 Lua 这种牛头不对马嘴的情况，要让面试官能立即 get 到你的技术专长就是 vue
- 组织好语言，项目经历在时间轴上从远到近，围绕你所设定的主题逐步细化、深化，例如最开始的项目经历里面你只是用了这项技术，后续逐渐开始更好地应用生态；更理解实现原理并能够解决复杂的性能、工程化问题；甚至更进一步开发了一些有价值的开源工具，或者输出了一些高质量的文档反哺社区。要让面试官能够通过项目经历感受到你从小白逐步成长的过程
- 前面两点都是在表现深度，对于工作3年以上的同学，通常既要求有深度，又要求有一定的广度、视野，说实话这并不容易做到，有一个方法就是围绕上面树立下来的深度，向外扩展补充与前端基础强相关的工作经历，比如说 http、TLS、http2、TCP 等网络栈相关的性能优化、版本升级经历；或者，内存泄露的排查修复经验、FPS、FCP、FMP 之类指标过低的优化经验等等；又或者一些更复杂的开发场景，例如编辑器、编译器、可视化、复杂动画、多媒体等等

总结下来，尽量做到一专多能，既有深度又有广度，深度能够帮助面试官迅速判断你的技术栈，降低心智负担，看起来不累；广度能够帮助面试官识别你的学习能力、潜力、对复杂开发场景的承受度等。在基本技术人设之外，最好还能顺带传达出你对所在行业的认知深度，后面会聊到。

近期准备做什么

很多面试官喜欢问：你未来3-5年的职业规划是怎样的？很多人会觉得“职业规划”这玩意儿挺虚的，不愿意花时间去认真梳理。我的观点，职业规划首先是给自己看的，是给你自己设定了一条路径，日常工作中需要不断做出选择，心里的这条路径越明确，做决定的成本会越低，会看到自己不断在接近目标。

对于面试官来说，这个问题大部分情况下首先考察你对自己的职业生涯有没有足够清晰的认知和目标感，三天打鱼两天晒网就跟频繁跳槽一样，没办法让你在垂直领域积累足够的深度；其次，考察你规划的天花板，如果没有表现出技术、职业野心，那容易让人 judge 你的发展潜力；最后，考察你的规划与团队的 match 程度，这就见仁见智了，没法一概而论。

所以对人对己都很有必要先花点时间，想清楚自己未来3-5年要做什么，做到什么程度，树立一个明确的职业目标。这个话题有点脱离本文的主题，因为你很难在简历中表达出你的职业规划，不过可以换个角度，在简历中以附加材料的形式呈现，比如个人博客、github。

博客的话可以围绕你设定的职业规划，连续一段时间围绕这个主题写多篇博客，让面试官感受到你既有想法，又确实有在这个方向上努力。

Github 的话也是一样的，连续一段时间在这个主题上输出一些代码质量较优的仓库，通常面试官进来第一眼是看star，其次是看代码风格，如果不能攒到 100 star 以上，就尽量把代码写好看一点，这也是加分项。

量化

数字是个大杀器！正确使用各种量化指标能让你的简历更有重心，更有可信度，更容易获得认可。很多东西可以量化，比如说：

- **「性能提升」**：性能优化通常是一种很综合很复杂的场景，需要足够的知识深度，需要灵活运用各种调试工具，所以面试官通常看到这种经历都会多加关注，如果能推断出优化前后的指标变化就更好了
- **「业务提效」**：这方面通常是引入或者创造了某类工具，改变或优化原有工作流程达到局部或全局更优解，从而提升整体效率，优化方向不局限于开发团队内部。这类优化与业务紧密结合，换个业务方向的面试官可能很难从你做的事情 get 到点，如果能提供一些具体的优化数值是有助于读者做判断的，可以是流程提效了 xx %、工单完成率提升了 xx，达到xx、响应及时度提升了 xx 之类的
- **「业务推进」**：假如有幸参与到一个发展比较猛的项目，而且你在这个项目中是比较核心的成员，那么可以考虑总结一下从开始到你准备离开的时候，项目的业务指标有多大的增长
- **「影响力」**：影响力这个概念就比较主观难以量化了，但是也可以用别的指标从旁佐证，比如工作期间做了 xx 次部门内分享、xx 次公司范围分享、xx 次行业大型分享；或者是，输出了 xx 份博客之类的

注意，前提是正确，不要为了量化而刻意捏造或者拍一些不存在的数字，拍出来的数字通常很容易识穿，面试时容易露馅，没必要。建议在日常工作中养成用数据思维，包括业务上的，技术上的，特别做一些优化的时候，记录优化前后的数值情况，写简历时自然有素材。

业务深度

先分享一下我个人经历，我曾经在一家特别小而美的人工智能创业公司工作了三年，虽然职能是前端，但是过程中并没有把自己的工作边界圈的泾渭分明，经常很发散地去支援服务端、数

据甚至是运营团队的工作，比如：

- 用 Python + celery 开发定时结算系统，降低对账成本，压缩结算周期
- 用 node + ffmpeg + docker 做了一套视频流式处理工具，能够根据配置对一批视频做抽帧、截取片段、压缩、转格式等操作
- 某个 POC 性质的项目中，用 Python + caffe 调用深度学习模型实现图像识别服务，配合浏览器上调用 media 接口将摄像头画面传回服务器识别出画面中的物品

短期来看确实没有明显收益，但是在我离职写简历时，发现这些经历串联起来，让我对深度学习的工作过程、原理、局限性、工具、指标等概念的理解已经足够支撑我在面试过程应对各种问题，后面找工作的时候聊到这一部分都会特别顺利。

这里不是鼓励大家去做很多前端领域之外的事情，只是想表达对业务、合作团队的了解与洞察程度可以映射出你对工作的投入度，而市场通常都会比较青睐投入度高的人。所以在日常工作中可以有意识地用各种方法跨出职能边界，去了解其他团队在做什么，怎么做的，平常会用哪些工具技术，有没有存在什么问题，这些问题有没有办法用前端的技术解决，等等。

当你对行业形成足够立体的认知之后，写简历、面试的时候可能就可以展现出在这个领域的横向认知，反过来说如果你过去对工作的认知一直停留在前端领域内，隔壁在做什么，怎么做，用什么做；业务线接下来有什么计划，可能需要用到什么新技术，这些问题都回答不上来的话，面试官容易怀疑你对工作的投入度。

项目经历怎么写

不要只写你做了什么，更重要的是突出你用什么方法，解决了什么问题，收益是什么，要能够形成一条完整的逻辑闭环，面试官才有足够信息来判断你项目经历的价值。

比如说，对于这样一段经历：在 XXX 项目中引入性能及异常上报工具，后续团队内基于回收的数据有针对性的做了一些优化，我曾经收到一份简历是这么写的：

> 集成监控SDK，包含页面测速，错误异常，API质量，白屏异常，URL异常，收集项目中的各类错误信息并上报，通过 performance API进行测速分析，封装基础库ajax上报API错误信息；在资源监控系统通过对各个端上报的指标进行清洗，聚合以及数据分析，错误模块聚合sourcemap还原源代码，便于修复线上问题；重构项目代码与调用链，加载时间缩短20%；

这里面有一些明显问题：

- 语言组织太过平铺直书，感知不到重点
- 监控SDK是啥？集成方式又是怎么样的？这里面有多少工作量？有那些难点？
- 形式与描述不好，阅读成本高
- 清洗、聚合、数据分析分别又是啥？前端在这里面做了什么？
- 错误模块聚合sourcemap？只有错误模块吗？聚合又是什么鬼？聚合还原完为什么就能“便于修复”线上问题？sourcemap 的原理又是什么？
- “重构项目代码”与前面说的“集成监控SDK”是什么关系？为什么要写在一起？
- 加载时间具体是指哪个指标？具体做了什么缩短的？

总结下来，我个人觉得问题主要是描述不清晰，很难理解这到底是一件什么事情，怎么做的，最后收益又怎么样。比较好的方式应该是：

> - 引入(基于) xxx 搭建性能与异常监控体系，覆盖 FCP/FP/FMP/TTI/LCP 等性能指标；覆盖白屏、页面奔溃、JS 异常、http异常等错误场景。
> - 在上述监控体系基础上，逐步推演出核心性能指标模型，以此为决策依据逐步执行图像合并、代码分包、缓存策略优化、首屏渲染优化、SSR 等措施，前端性能平均指标提升 xx%，QPS 提升 xx%
> - 在上述监控体系基础上，优化项目 CI 工序，接入基于 webpack 的 sourcemap 映射能力，线上问题能够直接映射回源码堆栈，线上问题平均修复时间降低 xx%

这不是最好的表述，但是这已经充分说明了：用什么方式方法、具体解决了什么问题、最终收益是什么，相比于前面的写法，叙述上更严谨也更容易理解一些。

如何做减法

简历内容在“历”，但是预设条件是“简”，不要写成流水账，不要事无巨细写成了自传。简历内容绝非多多益善的，写的越多阅读成本越高，越难以抓到重点，所以应当适度精简。

注意项目路径

如果你已经有比较丰富的项目经历，千万不要不做选择全部往简历怼，不是项目经历越多越好，应该根据结合个人情况，精心挑选几段有代表性的，例如：

- 能表现出技术深度，或者业务复杂度、业务体量的
- 能表现学习能力的，例如曾经为了使用一个文档缺失的框架，花了一段时间看完源码总结出用法，最终能够
- 能够构建起“成长路径”，这一part在“树立技术人设”一节已经讲的比较细了

尽量避开这些类型的项目：

- 单纯练手，复杂度、业务价值都特别低的
- 太过简单的，例如简单的活动页
- 仿 xxx 型的，培训班很喜欢搞这种练习题，不少候选人就拿这个当实际项目写到简历上了

慎用技术名词

前端简历中常常会有一 part 总结自己的技术栈，这里一定一定要慎重，我经常遇到很多技术栈特别广泛，但凡用过的技术都往上面写的简历，一个是看起来、分析起来累；一个是面试过程一问三不知。我建议使用任何技术名词前可以先问问自己：

- 这种技术会给你的简历加分吗？
- 你的使用频率、了解程度足够高吗？足够应对可能出现的各种技术问题吗？
- 这项技术足以让你与其他候选人拉开距离吗？

比如说，“我曾经用 grunt 搭建过一套完整的工程体系”这样的经历不能说完全没有价值，但放在 webpack、vite、snowpack、parcel、rollup 大行其道的当代，这份经历能给你加分吗？反而可能会让面试官质疑你技术更新迭代的速度会不会太慢了？

又比如说，“我曾经写个一个带视频的网页”这样的经历还不足以支撑你在简历里面写“具备视频编解码能力”，如果更进一步“我曾经基于 HLS + FFMPEG 实现动态视频流服务，配合 video.js 实现按需播放”(如我的另一篇博客 HLS + ffmpeg 实现动态码流视频服务 所说)这种程度，那大可以说你“理解常用视频封装、编码格式，能根据应用场景搭建流畅的视频播放体系”。

站在一个面试官的角度，单纯的堆叠名词反而容易让人质疑你的知识深度和对自己的认知的准确性，适当的裁剪往往更能突出优势。

慎用形容词

在我第一次写简历的时候，有一篇文章印象很深，细节忘了但是里面有一个很重要的观点：不要写精通！我觉得特别对，因为大多数人对大多数技术的掌握程度并没有达到这个深度，如果你真的自认有精通的点，那有可能是事实也有可能是不知者无畏。

举例来说，你觉得你精通 HTML 吗？那么：

- input 标签的 type 属性有哪些可选值？分别对应什么功能？浏览器兼容程度如何？
- 什么是标签的语义？为什么要有语义化？有谁会消费这些语义？怎么评估语义是否恰当？
- aria 属性是什么？怎么编写合理的 aria 结构？又是谁，以何种方式会消费这些属性？

又或者，你觉得你精通 vue 吗？那么：

- Vue2 的双向数据绑定是什么？如何实现的？这个过程如何影响 props、computed 属性？
- 如果上面的问题你理解了，那么 Vue3 呢？
- Vue 如何将 template 转换为 render 函数？又是如何识别出标签对应的组件？组件层级之间的创建顺序是怎样的？渲染顺序又是怎样的？

这个列表还可以无限列下去，所谓学海无涯，谦虚一点总没坏处的。我个人的做法是绝不写“精通”，因为我自知对任何一个技术点都远远没有达到精通的程度；但是会写1-2个熟悉的技术项，并且书写顺序上会尽量靠前；此外会再补充一些理解，对于那些把握不够的点会忽略不写。面可以广一点，例如网络协议、构建工具、开发框架都写一些，但总量尽量保持到3-5个。

这里可能会有同学，特别是实习生、应届生担心技术栈不够广会不会反而拿不到面试机会呢？这其实也是一种学习策略的问题，如果你站在面试官的角度，放在你面前的两份简历，一份内容看起来是少但明显能感觉有足够深度；另一份堆砌了很技术名词，但是名词之间看起来没有明显关联，你会倾向那一份？

总结

简历不容易写，技术人员的简历更不容易，为了心仪的工作花多点时间沉淀一份优秀的简历是非常有必要的。

聊一聊装饰者模式

知了一笑

ERGM的几种稳健性检验思路

蜗牛飞

数据指标体系搭建（理论篇）

Lin王发林