**面试宝典**

**一、信息差、八股文三字经、引导面试官、前端底层的重要性**

涨薪普遍方式有两个，一个是企业内部涨薪，另一个就是跳槽。首先内部涨薪是极难的，十里挑一，极度内卷。所以大部分人就靠面试找工作这一环。工作拧螺丝大家都会，但是面试需要的是会造火箭的，对于面试准备来说准备方向是最重要的，如何搞难点亮点、优化简历，这些方法都往后排，当你坚定了方向，再来谈方法，一切就简单了。  
 程序员的职业生涯也就短短15年，竞争异常激烈，压力也很大，当然挣钱也很多，所以你要珍惜每一个提升自己的机会，接下来的三天你会听到各种新鲜的名词比如《费曼技巧》。

刻意练习，结构化思维，开气场，简历黑话，免费复活卡，业务sence，难点公式，信息差。

只要你用心学，就能应用到面试中

第一个重磅内容就是信息差，什么是信息差呢？ 再牛逼的人，也有知识盲区，不会的东西，你只要准备的好，就可以吊打他。但是你要知道，准备八股文和项目你是准备不完的，核心本质就是准备一部分高频的面试题，然后能将他牵引到我准备好的内容中。那问题是如何能让面试官问到我准备好的呢。那就是刺激他的【好奇】，面试官好奇，所以他就会被你牵引，引到你准备好的，同时面试官不会的，你就会【吊打他】，因为人都有好奇心，抓住这个点通过自己准备好的内容打死他。

那如何能够抓住他的好奇心呢？其实就是数据，难点亮点，话术。如何将这个东西细化呢？就是在简历、自我介绍 、你的回答过程中的引导。我有哪些【亮点】和【难点】，这些【亮点】都是我提前准备好的。比如说【大文件上传】，比如说【低代码】，比如说【微前端】。

甚至如果你不会那么多的项目难点的话，你也可以说你的源码的理解，只要你能拿得出手就可以。面试官内心想第一时间get你的干货，所以直接给他难点亮点就好，面试官大概率会问你说的这些难点亮点，所以我们会在这几天的课程里面无限的去引导他。无限的让他好奇，无限的吊打他。不管是从【简历】上，还是从【项目】上，还是从【源码】上，以及【八股文】上

接下来重磅知识点的就开始讲了，我们如何在八股文上让面试官觉得好奇？如果你无法引发好奇，你就会被面试官吊打。他就会问他懂的，你就会永远被牵着鼻子走，其实八股文好奇的点就是《底层》《底层》《底层》重要的说三遍！！！前端讲到底层只有三个东西【数据结构、源码、和浏览器】，所以你只要把这些研究透了，你在八股文上就能够吊打面试官。比如讲【闭包】可以讲到【浏览器的垃圾回收】讲原型链能够讲到【 Vue 的响应式】，只要你能够讲到【底层】，讲到【源码层】。当然，你光讲底层还是不够的，你还要讲【应用】。

这就是我接下来讲的，如何让面试官觉得你讲的东西不是背的？那就是讲你项目中的应用。

所以我把八股文总结出了一个话术，叫做【八股文三字经】。什么意思？其实就是任何的八股文，你都能够通过三字经的形式填装上去“【是什么】【优缺点】【讲联系】【说应用】”不要小看这几个字，字字值钱！是我研究这么久实战总结出来的。他的背后逻辑是【第一性原理与结构化思维】

讲是什么，一定要简洁直击本质 ，因为他是第一性能原理的要求，讲的太复杂，面试官无法瞬间理解，他就很懵逼，直击本质，才显得你理解深刻，我们还可以把他与上面讲的底层叠加在一起，可以说威力无穷。比如原型链本质就是一个链表（底层 数据结构），瞬间拉开和别人的差距，具体的我们就不展开了，基本上所有复杂的八股文我们都可以按照公式去套。就是三字经的公式加上底层（数据结构算法 浏览器原理 源码），这个就是讲八股文与信息差的一个应用方式，不要觉得太难，都可以提前准备的。B站上那么多视频，百度上全是答案，提前背好。面试官再牛逼，也不如你提前准备，等于是开卷考试，比如闭包 ，你按照模式去回答，【闭包是什么 优缺点 场景和源码应用】，当然你可以用chatgpt来帮你快速学习面试题，比如闭包 直接问gpt或者文心一言。

去哪里去找这些面试题？我推荐去网上，但是网上要注意一点是什么？一定要看面试题的时间，越新的，尤其是别人拍照的，越靠谱。

* 通用类型基础部分：HTML、CSS、H5C3、JS。
* 少量准备：Node.js、webpack & vite。
* 社招少量：数据结构和算法重点：场景题和提炼项目细节和难点
* 其他：兼容、性能优化、安全、项目上线、网络请求
* 框架类型vue方向：vue2/vue3 react方向：react
* 少量：uniapp、微信小程序、electron...

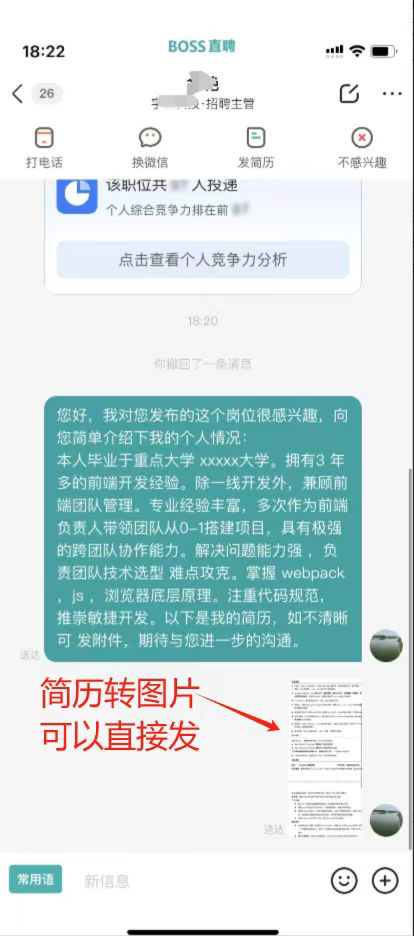
不打没有准备的战，能理解尽量去理解，实在理解不了的，可能需要背，但是背有一个不好的点就是容易忘，所以还是那句话，能理解尽量去理解。最核心的是什么？是把面试题按照刚才说的三字经的形式给他表达下来

1. **亮点公式 、业务sence 、简历、源码**

我们来实操，纯实操写简历，写高分简历。以及项目亮点，这些都是核心中的核心。首先我想引出一个话题，就是面试是从什么时候开始的？大家也可以思考一下，有些人说是从看到面试官第一眼开始，有些人说从自我介绍开始了，有人说开始介绍项目的时候开始了，其实都不对。在我看来就是在你投递简历的那一刻，面试就已经开始了。

从2023年开始，很多人拿到优质面试的机会就很少，因为他们简历很差，简历的第一个作用就是拿到一个满意的面试机会，如果你连面试机会都拿不到，更别提找工作了。简历的第二个重要的价值就在于【引导面试官】，能把第二点悟透，你才能提高自己的面试成功率以及拿【高薪】，简历相当于一个菜谱，它能够把我们所有的知识进行展示。展示完了，我们要针对每个点准备好话术。

就是我说的面试底层逻辑，就是引发好奇，吸引面试官到准备好的东西，吊打面试官。那我们先讲讲第一个点，如何拿到面试机会。先来个开胃菜，每天几点投递简历呢？最推荐是九点，打招呼词一定不要使用系统推荐，要自定义。要不然面试官看好几百份简历，他都直接略过的。然后直接把【简历变成图片】发到boss上像这样子：

投简历有两种投法：  
 一种是海投，适合20k以下想快速找工作的的简历 ，不要打满200份，150份就够了，多几个账号去投递，小心封控

另一种是精投，分析jd，适合面试机会少或者工资要求高的，精投最重要的就是匹配度  
 还有一种就是内推

如果是看 boss 的话，那它的匹配度分为这几种：

第一种就是它的业务匹配度，就是你做的业务和它做的业务是不是都是相似的  
 比如说金融，比如说 ToC，互联网，比如说 H5，比如说 PC，比如管理后台，其实这就是我们能够包装的部分。

【面试 HR 挑简历，首先他考虑的是匹配度，其次才是学历。匹配度和学历同等重要】

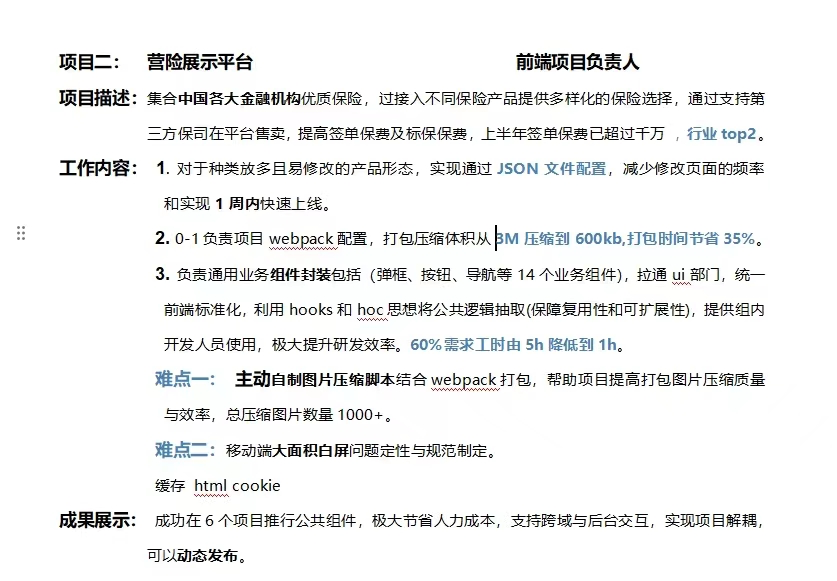
所以我们如果学历低，那我们优先找匹配度。

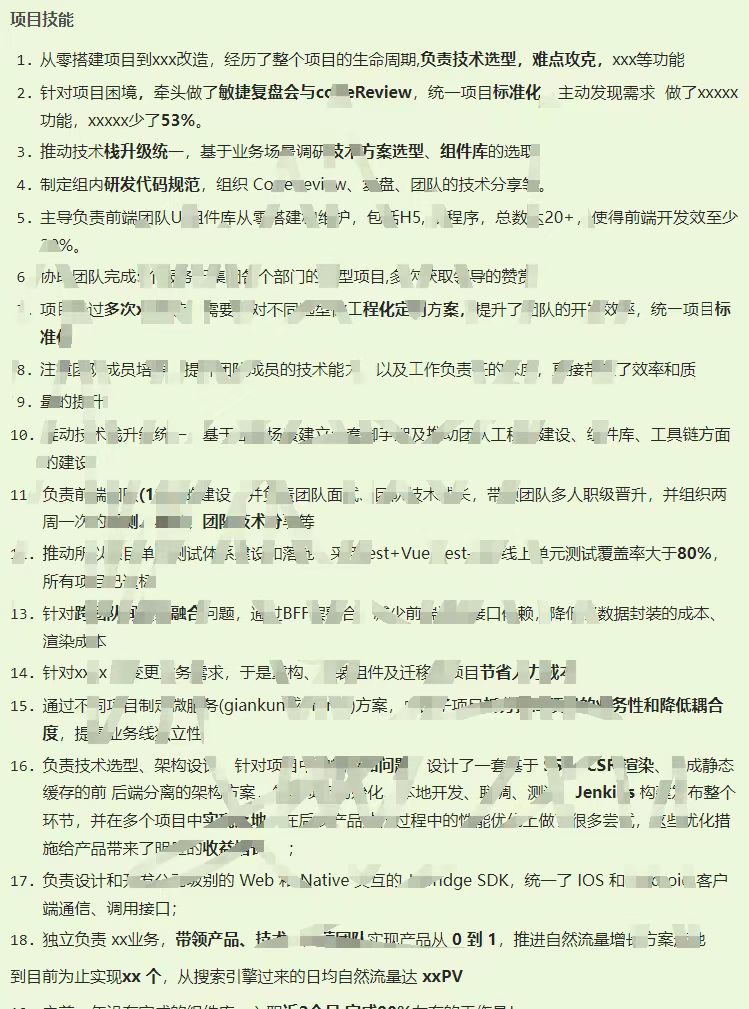
那具体如何去做这种匹配度？我就说三个维度，自己可以感受：

* 公司的匹配度
* 业务的匹配度
* 技术的匹配度

自己感受，如何快速提高这些匹配度 都是有方法的，但是背后涉及很多东西。背调真实性，技术问题的回答，业务闭环的表述。

接下来就讲简历中最重要最重要的技巧叫做引导面试官，这个其实就是之前说的，激发好奇。我们可以在简历中加一些数据，标注难点、亮点 以及加粗 甚至标蓝

太多人小看简历了，简历相当于菜谱，吸引面试官的注意。同时你在讲项目或者是八股文的时候都可以引导到你准备好的话术上。这样面试的时候更聚焦，同时简历还要有最重要的作用，就是开气场。可以看一下如何去写简历，这都是我总结的【简历黑话】。通过这些黑话在无形中就能增加我们的自信

举个很简单的例子，就是0-1。那谁都能0到1，但问题这个0-1背后是很多面试官以及业务需要的。但是我发现很大部分人他不会写这种简历黑话或者就是写不好，这就是不知道面试的标准。简历优化之前和优化之后的在面试通过率，引导面试官上差距是巨大的，所以大家一定要重视简历。

接下来我们讲面试核心中的核心，就是大家最关注的项目亮点。像大部分人来说，他们不明白什么是亮点，什么是难点。包括有一部分面试官，他们也不知道什么是难点，什么是亮点，那你连定义都不知道，那何谈准备？

【亮点 就是技术上你做的事情不是很难，但是它带来的效果非常的棒，想法非常的新颖】，满足这个点，它就是亮点，他的表达是有一个公式的就是：【我主动发现业务痛点，用技术手段解决，创造很大价值】所以你会发现有时候他的技术难度并不大，但是更加强调你对业务的理解与你创造的业务价值，这是所有程序员都忽略的，大家都觉得，我写代码关心新的技术就好了，业务，管理都是该领导做的。这是非常错误的官方观点，天天学习一堆技术栈，最终对公司价值屁用没有，学的东西毫无实践 就是demo。我们所有的一切回归终点就是如何让企业挣到钱或者省钱，产生业务价值。那如何能发现业务痛点与流程优化点呢。那就是具备业务sence ，也就是业务观。业务 sence 就像一根针，插入公司这个母体，把技术像药一样注入这个母体中，让母体循环的更好，具象点说就是帮公司省钱或者挣钱，这才是技术人最终的价值！

【业务痛点】有很多 我举一些例子

1. 部门人越来越多了，代码风格不一样，技术栈不一样，如何提高代码维护性？
2. 组件复用率很高，跨项目不能只靠复制黏贴，怎么管理？
3. 用户页面总是白屏，或者很慢，复现困难怎么办？
4. 页面每天要10个，一个人怎么做得完？

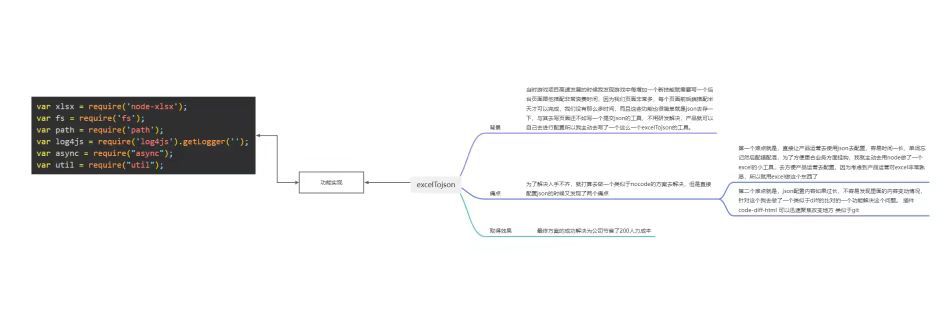
【解决方式】

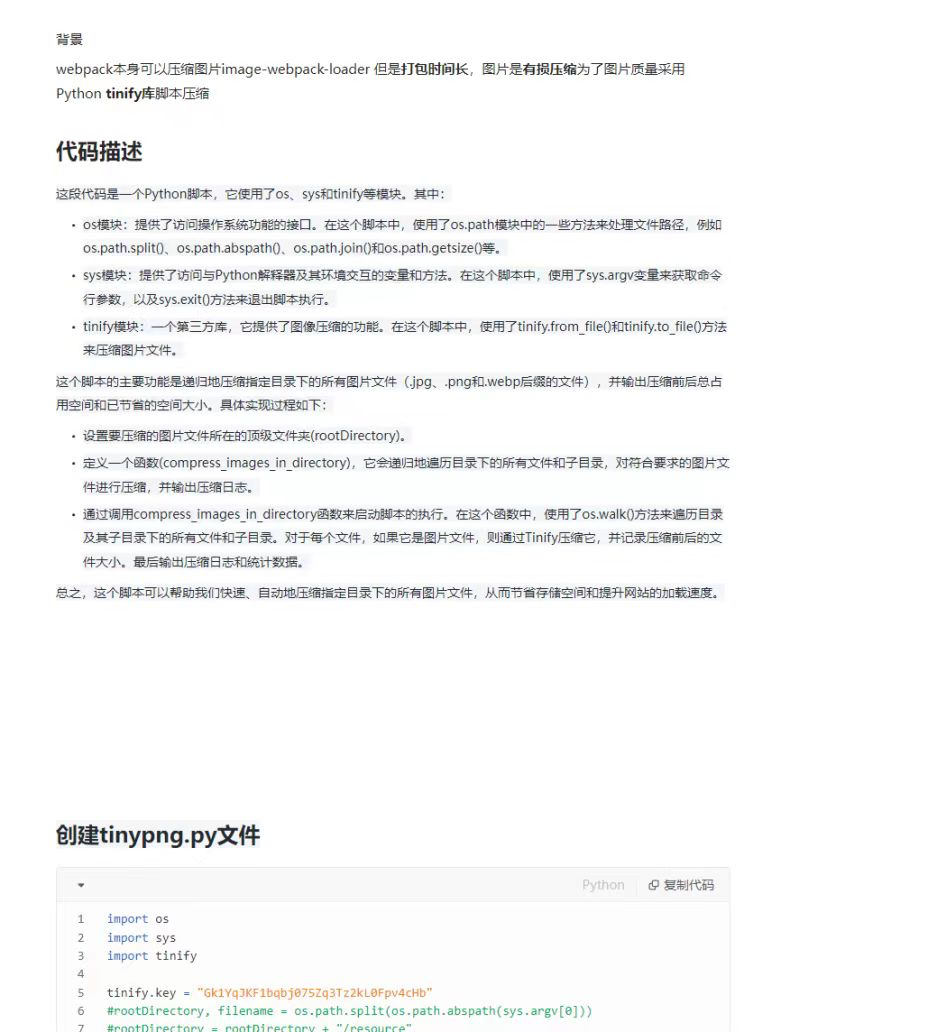
* 我们可以做 文档规范 脚手架制定 codereview机制
* npm组件库，lib库打包
* 监控埋点自研或者引入第三方平台
* 运营配置后台与投放策略系统或者低代码配置页面

很多人说，自己没做过，那你可以从现在建立自己的业务sence，在业务中实践，如果已经要准备面试了，那我告诉你都可以包装的。只要说的ok，说的明白 ，就可以。

【我举个例子，当你们项目来了很多很多人的时候，大家的代码风格不一样，于是我决定每周二拉一个 code review 会，在会上我们去统一大家的代码，同时说下大家最近做的重点事情。这样大家更有一个团队的感觉，而且代码风格，还有代码维护上都得到了统一。降低了20%的代码打回率。最后这个 codeReview会 变成了部门的例会，然后我们共建了一个文档，推广了三个部门。在会上我们还会做一些技术的分享，比如我做过前端安全与性能的分享，像这个就是非常好的一个业务的亮点，它跟技术难度并不大，但是确实带来了整个项目的好处，很多人觉得觉得这个是领导做的，但是其实面试就是把别人的说成自己的。因为你做的事情都是增删改查，你做的对业务有价值的事情都被别人占去了，那你就没什么价值，其实公司找你来并不是写增删改查的页面页代码的，而是让你创造价值，所以我们做任何事情要找到最终的价值】

【很多的小工具提效的都算亮点，比如内部的 node工具 代替了1000个后台管理页面，python图片压缩脚本 结合webpack 做到自动打包自动压所图片。自动去重重复图片，推行三个部门10个项目，总计压缩图片数量3000+。像刚才说的这些，我的面试中都有用过





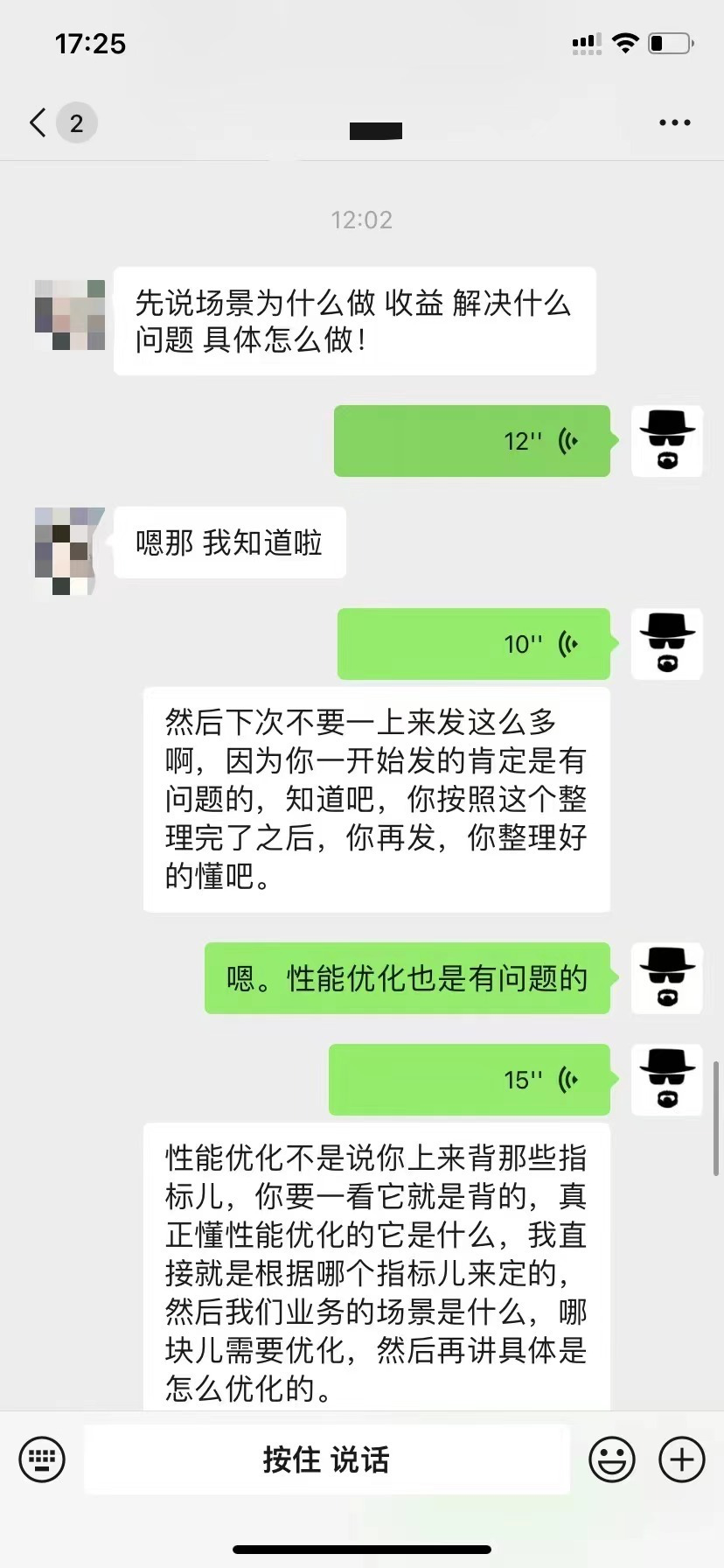
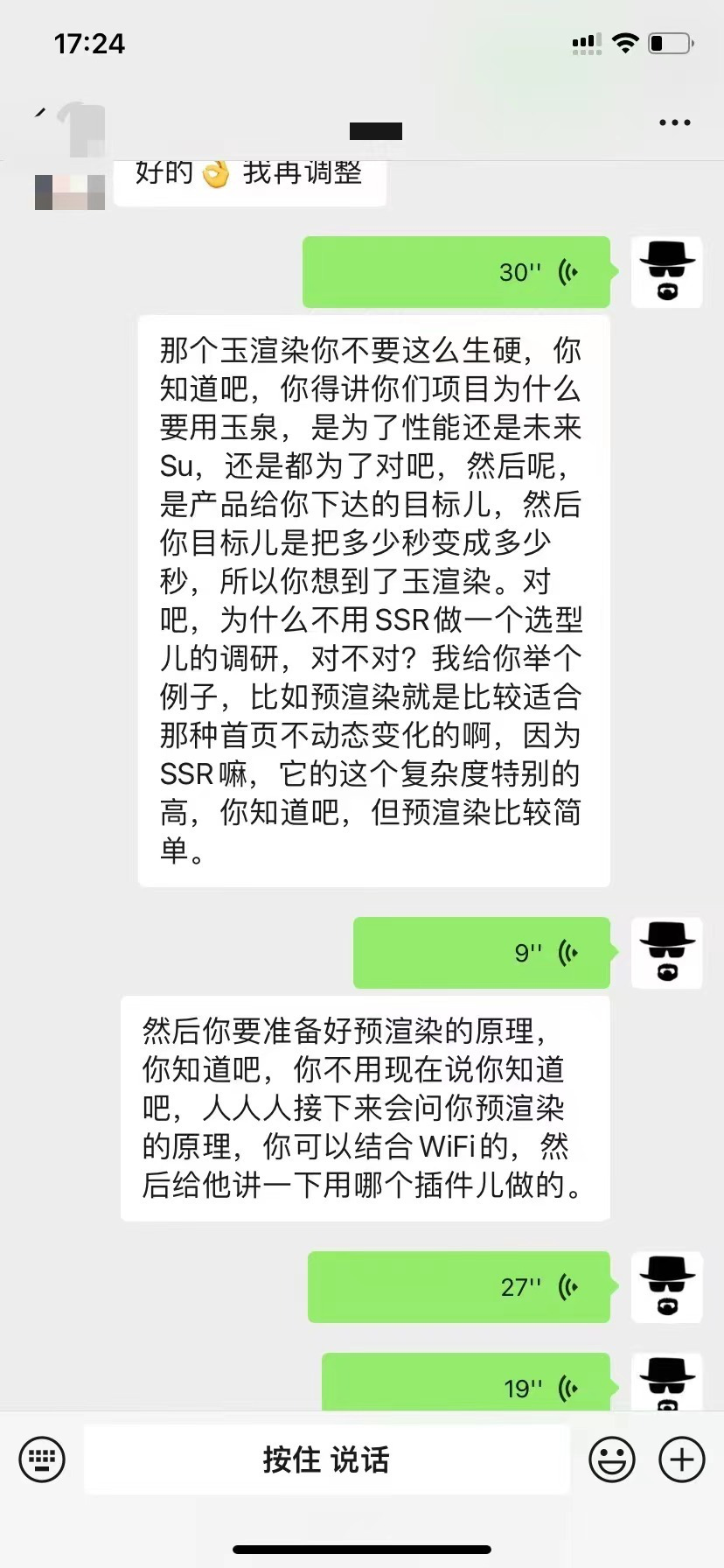
大家也可以根据我说的思路去构造，去包装自己的亮点，但是最好是能把你总结的表达出来，因为面试最重要的是说，而不是做。那如何说的真实？以及面试官会问哪些问题？你的表达自不自然？光优化也是很多 。有性能优化、体验性优化、技术赋能优化以及协作流程优化。当然除了这些优化，还有一些工具提效，这也是很重要的，但是它们都是有公式的。也就是我主动发现业务痛点，用基础的手段解决，产生了某些价值，记住这才是【业务的闭环】

那如何引导面试官？其实就是那些数据，数据是最能够打动人的，这样面试官能听出感觉你做的是有价值。那如何获取这些数据？如何去把这个公式填充的完整？面试的时候，面试官最容易问你遇到的亮点是什么？大部分做增删改查的人，他根本就没有，于是就尬在那了。面试官尴尬，你也尴尬。那你像这要怎么过？

1. **面试开气场 录音 费曼技巧 刻意练习 脑图 项目难点**

我觉得大家肯定对面试与业务是有了新的感悟，今天可以说是非常实操的一天，实操到每天如何去学习。会告诉你如何建立自己的知识体系。如何刻意练习面试，费曼技巧的应用，以及开气场与项目难点这些高阶的技能，我们先讲如何去刻意的面向面试去学习，涨薪基本都靠面试，面试考察的都是脑子到嘴的过程，也是所有人都缺乏的，大家可以思考一下自己平时是怎么学习的，打开 b 站，敲敲代码看看书，这些都是眼睛到脑子的过程，但面试不是，面试是考你脑子到嘴的过程

所以大家应该大量的去说，去表达，去录音反复听自己的回答，自己对知识的体悟，这才是正确的刻意练习，它的本质叫做费曼技巧，通过输出来倒逼自己的输入，比如第一天讲的八股文三字经，再比如第二天讲的项目亮点，还有今天要讲的项目难点，都需要你不断地去说、去结构化的表达，自己去反复的听，要站在面试官的角度去考察，但是里面会出现很多的问题，比如讲得太长、太啰嗦，没有重点，吸引不了面试官的注意



他最终去了大厂，其实它的本质就是靠了几个话术，任何人去大厂，项目难点，项目亮点没有多少的，但能保证的是他的项目难点，项目亮点是刻意练习的，是不断的录音的，不断的去说的，所以你们要做的是打造自己的几个亮点和几个难点，把面试官吸引到这里，然后吊打他。你看他就是反复的这样大量的练习，我经常在直播间展示这个案例，这个就是刻意练习的结果，刻意练习的三要素是

* 找到一个合适的难度
* 找到一个导师
* 反复大量的练习

通过输出来倒逼自己的输入是非常好锻炼自己知识的过程，因为神经元会重塑大脑的知识结构，一句话就是 你如果说不出来，你就没有深刻的理解这个知识点，知道了如何面向面试学习。接下来我讲一下咱们如何去开气场，这是非常高阶的技巧，如果你经常面试，你会发现有时候面试就像聊天，有时你一直在压制着面试官，有时你觉得一直被面试官压制，其实原理就是你的气场有没有打开，开气场底层原理就是，一个人他在10分钟之内或者7个回答之内单方面被另一个人持续吊打。后面30分钟就会被这个人压制，其实在我看来，面试就是前10分钟的事，你刚开始能不能压住他？

所以自我介绍是非常重要的，也是大家所忽略的。你完全可以在自我介绍的时候，直接引到自己准备好的难点或亮点，面试官自然会问，那你就会吊打他。但是自我介绍怎么做才好？如何能吸引面试官的注意？在面试过程中如何引到自己的难点？难点怎么结构化的表达？这些都是非常重要的点。最能吸引面试官的就是讲难点，难点其实也是有公式的，很多人不知道什么是难点，还有很多人说这个东西对于你来说不是难点，对于面试官来说它就是难点，难点是因人而异的。我只能说是有道理的，但是如果我们所有的都按这么说的话，那没有规律是无法去学习的，难点有什么规律呢？其实就涉及到了一个东西，就叫做选型，如果你这个难点都不涉及到选型的话，都不涉及到错误的方案，重试的话，它就不算什么难点。

star法则大家都听过，但使用 star 法则来介绍难点的话会很麻烦，我推荐大家用 PAR 。什么意思？

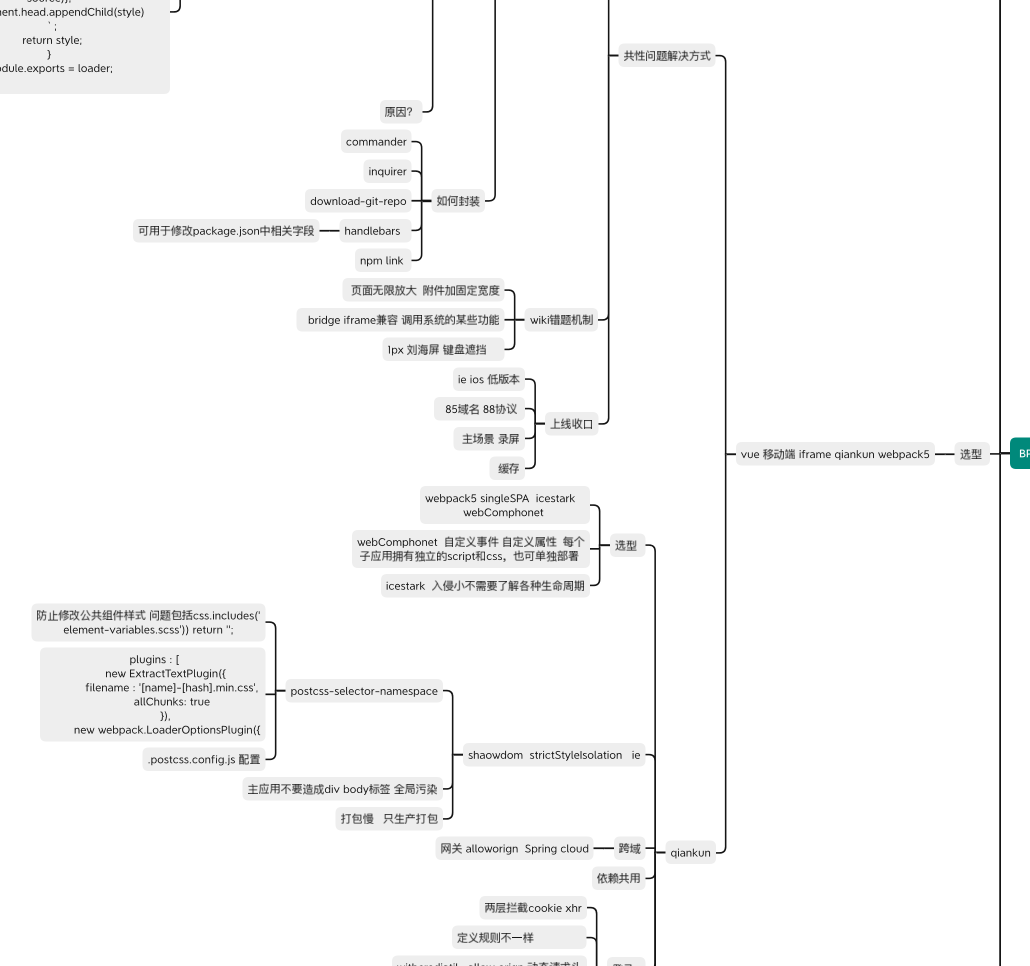
* P就是我们的背景以及问题以及痛点
* A其实就是我们要做的东西
* R就是我们的结果

其中我们如何把选型和这个 PAR 结合呢？那其实就是你要讲清楚。你碰到的难点是什么？难点是怎么选型的？难点的分类？如何分别解决 。里面又会有哪些小的难点？小的难点又是如何去解决的？如何去做的选型？业界的方案又是什么？这个就是无限嵌套

举个微前端的例子：

背景审批平台对接300个业务系统，开始用iframe 后来用前端进行微前端改造。主应用负责审批流程的节点，子应用负责展示业务表单页面。iframe 会有一堆问题比如 ios兼容 ，比如性能。所以为了产品体验需要做微前端改造

qinakun webpack5 singleSPA icestark WebComponents 无界iframe 到底用哪个？他们的优劣是什么？乾坤改造后又会出现一堆问题。跨域、 ie隔离 、登录问题、接口跨域，解决这些小问题有涉及一堆的选型与小问题。这种项目就是可以冲刺30k 40k 的复杂项目，很多人都是靠这种项目去的大厂。我们可以把它画成脑图与架构图去辅助我们表达，这是内部微前端的项目



这是帮助冲大厂学员挖掘自己的项目 画的项目架构图



当然细节又是很多很多东西，因为你面试，面试官都不是傻子。你如何能讲得通、讲得服？最佳的方式就是找懂的大佬去带你

比如上面的微前端就是内部项目，当然内部的难点有很多

比如nocode、大文件上传、性能优化、脚手架、node工具、图片压缩脚本、webpack插件等等

挖掘难点的方式有两种：

一种是挖掘自己本身的

一种是通过别人的项目或者是自己包装项目，结合我给到的方法在面试中发挥出来

其实我说的这些东西都是项目的枝和干，把枝干支起来之后再填充那些叶子，通过脑图给大家变得更丰富，每一个枝叶都建立好自己的话术，通过脑图建立它们的关系，这才能在面试的时候不断的拉扯面试官，引导面试官乃至吊打面试官，枝叶它如何能显得更加真实？显得更牛逼？就是不断的发录音，通过录音与脑图填充那些知识的。