2024 前端面经

本文主要给大家带来一些上岸面试者的面试经历和经验，希望这篇文章能够给正在找工作的同学们一些帮助。

字节跳动面经

一面：聊项目

作为前端负责人在前端基础设施做了哪些事情

1. 开发阶段，包含了技术选型、项目创建、模板化、脚手架工具等方面的工作

2. 部署阶段，如何去做自动化的 CI/CD、如何将项目部署到服务器

3. 质量保证，尝试使用一些自动化测试框架进行项目测试，同时对项目配置进行收敛，减少配置的修改，以保证每个项目的基础设施统一

4.如何提效，通过脚手架工具来实现零配置启动部署项目，通过模板、组件以及对应的 schema 调用方式来降低开发门槛和效率提升

Vite 的底层原理是什么

1.这部分确实是还没有去看，只知道是基于 esbuild 和 rollup 实现的，就直接说不知道了。

你觉得 Vite 相比于 Webpack 哪些方面更好

1. 本地开发的构建速度快，由于 Vite 是基于 ESM 和 esbuild 的，所以在本地开发时整体的构建速度会比 webpack 更快

2.使用简单，Vite 内置了很多 loader 和配置，让开发者可以零配置跑起来一个项目，而 webpack 则是需要写很多复杂的配置

Vite 的热更新原理是什么

1. Vite 本地启动时会创建一个 WebSocket 连接，同时去监听本地的文件变化

2. 当用户修改了本地的文件时,WebSocket 的服务端会拿到变化的文件的 ID 或者其他标识，并推送给客户端

3.客户端获取到变化的文件信息之后，便去请求最新的文件并刷新页面

介绍一下之前的监控告警是怎么做的

1.分为两部分，第一部分为日志的埋点和上传，包含了代码日志和业务日志。主要是自己实现了一个基于 WebSocket 的日志服务，在客户端项目加载的时候启动 WebSocket，然后通过提供的 log 方法在代码中去进行日志打点。服务端收到上传的日志之后传入到公司内的数仓，之后通过数仓的 API 实现日志的查询。
对于业务数据的埋点则是在对应的用户操作时进行埋点上报。

2.另一部分则是告警的实现，这部分利用公司的统一基础设施去做。在拿到前面的埋点信息之后，在公司的告警平台可以看到对应的埋点数据的趋势图，根据趋势可以设置告警阈值。告警阈值主要是通过人工去指定告警策略并根据实际情况进行调整和优化，以实现更准确的告警。

怎么去监控业务指标情况

1.根据业务流程中对应环节的埋点数据可以计算出业务指标，如提单成功率 = 支付成功的埋点数 / 购物车页点击下单的埋点数。获取到各个业务指标的计算公式之后，可以在告警平台去配置对应的趋势图，之后按照前面的方式设置监控和告警即可。

小程序性能优化做了哪些事

1.小程序本身是双线程架构，渲染层和逻辑层相互独立，通过微信原生中的 JSBridge 进行通信，因此性能的损耗主要花费在通信过程中。而在通信过程中影响性能的点最主要的则是 setData 的频率和数据量。

2.所以主要做的优化手段就是去减少 setData 的次数，同时当 setData 数据量过大时对数据量进行拆分，分为多个 setData 去执行，从这两者中找到一个平衡。

3.除此之外，还利用了 Wxml 节点压缩，CSS 样式合并，以及请求预加载等方式进行性能优化。

【代码题】实现一个 React 的 useState hook

```
js
复制代码
const useState = defaultValue => {
const value = useRef(defaultValue);
const setValue = newValue => {
if (typeof newValue === 'function') {
value.current = newValue(value.current);
 } else {
value.current = value;
 }
  }

//  触发组件的重新渲染
 dispatchAction();


return [value, setValue];

    }
```

二面

D2C 做了哪些事情

1.主要是做了一个 sketch 插件，通过插件的开发语言 cocoascript 和其中的 webview 的方式来实现的。为了降低开发成本，插件面板内的所有内容都是在 webview 中去开发前端页面。当从面板中拖拽某个图标或组件出来时，通过 message 将组件的信息传递到插件原生种，之后通过 cocoascript 开发 sketch 文件的查找和放置画板的逻辑。

2.为了简单实现，我们在插件安装的时候内置了符合业务规范的 sketch 组件包，因此面板中的每个组件都有唯一标识与 sketch 组件一一对应。最后导出页面时同样是对画板中模板和组件进行遍历，找到对应的组件代码并进行组装。

小程序的日志和监控服务做了哪些事情

1.分为两部分，第一部分为日志的埋点和上传，包含了代码日志和业务日志。主要是自己实现了一个基于 WebSocket 的日志服务，在客户端项目加载的时候启动 WebSocket，然后通过提供的 log 方法在代码中去进行日志打点。服务端收到上传的日志之后传入到公司内的数仓，

之后通过数仓的 API 实现日志的查询。
对于业务数据的埋点则是在对应的用户操作时进行埋点上报。

2.另一部分则是告警的实现，这部分利用公司的统一基础设施去做。在拿到前面的埋点信息之后，在公司的告警平台可以看到对应的埋点数据的趋势图，根据趋势可以设置告警阈值。告警阈值主要是通过人工去指定告警策略并根据实际情况进行调整和优化，以实现更准确的告警。

作为前端负责人在技术上做了哪些基建

脚手架

1. 提供前端脚手架工具，支持一行代码自动完成项目创建，同时调用 gitlab 的 API 完成远程仓库的创建，最后自动生成相应的 CI/CD 脚本实现自动化部署

2. 将项目的配置项进行收敛，包含 ESLint、Prettier、TSConfig、Vite 的等，将标准的配置文件全部内置在脚手架当中，只提供部分配置项以单独的脚手架配置项透出

3.提供自动化命令，包含代码格式化、质量检测、本地开发、生产打包等

框架

1. 前端框架部分主要是对一些公共模块和服务进行了单独的封装，包含请求模块、状态管理、路由等，所有的功能都由框架导出给开发者直接调用

2.提供了业务通用能力的封装，如 PDF 预览、统一图表展示、富文本编辑器等

组件

1. 组件部分主要是基于 Antd 去做一些样式和改造以及更上层的组件封装

2.对常见的 CRUD 页面封装成模板，并提供 JSON2Page 的使用方式，以实现通过 JSON 配置直接生成页面

富文本编辑器是如何实现的

1.一开始是通过 slate.js 去实现的，slate.js 提供了富文本编辑器的核心能力，基于这些核心能力我们可以在上层实现各种复杂的富文本编辑器。但是随着业务对富文本编辑器的功能需求越来越多，我们就需要在 slate.js 上投入更多的人力去开发这些新的功能需求，这显然在我们这样的小团队中是不太现实的。于是就考虑找一款功能完善的富文本编辑器，最后选择了 jodit，一款基于 TS 实现的富文本编辑器，整体代码比较易读，便于后续对其进行二次开发。

有没有在质量和稳定性上做一些事情

包含两部分，一部分是开发过程中的质量保障，主要有两点一个是单元测试，通过 Jest 去实现，另一个则是 UI 的自动化测试，通过 cypress 实现。第二部分是对线上稳定性的监控，这个是基于开源项目加二次开发实现的。具体可以分为以下几个步骤：

• 异常捕获：对异常进行分类，不同分类通过不同的方式进行捕获，如 addEventListener('error')监听 JS 代码异常,window.onerror 监听资源加载异常，xhr.addEventListener('error')监听请求异常等

• 异常上报：可以通过 WebSocket 进行上报，或者直接通过 HTTP 请求上报，在上报的时候需要考虑到弱网的缓存，以及高并发的数据合并情况

• 数据接收：启动一个 Node 服务，在接收到异常数据之后直接存在数据库中即可

• 数据使用：对上传的异常和监控数据进行分类查询，并提供对应的监控告警机制，当超出阈值范围时发送告警邮件

【代码题】描述下列代码的输出结果和原因

// 最害怕这种题目了，总是想不明白，一开始说了结果是 3，后来面试官让我再想想//
仔细缕了一下后发现，fn() 里的函数声明虽然会提前，但不会提前到最外层，只会到 fn() 的顶层//
里面的 foo = 3 修改的其实是里面函数的那个声明，不会影响到外部，所以最终的结果应该是 1var foo = 1;

```
function fn() {
foo = 3;
return;
funciton foo() {
// todo
}
}
fn();
console.log(foo);
```

【代码题】按照版本号由小到大排序

样例输入：versions = ['0.1.1', '2.3.3', '0.302.1', '4.2', '4.3.5', '4.3.4.5']
输出：['0.1.1', '0.302.1', '2.3.3', '4.3.4.5', '4.3.5']

```
function compareVersions(versions) {
 return versions.sort((a, b) => {
 const tempA = a.split('.');
 const tempB = b.split('.');
 const maxLen = Math.max(tempA.length, tempB.length);
 for (let i = 0; i < maxLen; i++ ) {
 const valueA = +tempA[i] || 0;
            const valueB = +tempB[i] || 0;
            if (valueA  === valueB) {
 continue;            }
 return valueA - valueB;
  }
 return 0;
 });
}
```

三面

三面的时间不短，唯一担心的点就是我的项目经历技术复杂度一般，可能会跟面试官的期望不符。

介绍一下业务做了哪些系统

先从现在公司的业务讲起，然后根据业务内容介绍了对应的系统，我觉得这个问题主要是表现出自己对业务的理解即可。

在以往的工作经历中有哪些最有成就感的项目

详细讲了一下之前做的监控和告警项目，并且如何通过技术手段来推导业务指标，并根据业务指标的监控数据来反推业务的发展。具体技术上的关键点做了详细介绍，但是整体复杂度并不高，主要还是从收益上看属于低成本高收益的项目。

当时为什么考虑现在的公司

从三个方面进行考虑：

1.在互联网行业待了一段时间，想要看看其他行业是什么样的，同时自己对金融还是比较感兴趣

2.大厂“打螺丝”不能充分发挥出自己的价值，想要到小公司去做更多的事情，让自己变得更全面

3.有一定的创业精神，选择了一个创业团队，可以收获一些创业的经验

现在为什么又要从现在的公司离开

有两个原因：

1.团队发展不看好：新换了一个老板，对团队的定位发生了变化，由创新的探索性团队变成了支持性的 IT 团队，跟自己的期望不符

2.个人发展受限：从技术上来看，产品用户量较少，且复杂度较低，技术上持续在做输出，但是输入很少；从管理上来看，团队规模趋于稳定，后续不会再有更多人让自己来带

对自己未来的发展规划是什么

也是可以从两个方面来讲，一个走技术路线，一个走管理路线。

技术路线：

1.对自身已经掌握的技术持续精进，并通过技术手段去回馈团队和业务（如前端架构、异常监控、性能优化、指标体系等）

2.在某一个技术方向上做到突出，能够沉淀出相应方法论，并建设出系统性的平台，在部门及公司内部普及应用

3.保持对新技术的热情，持续扩宽技术广度，对团队的技术栈持续迭代，保持团队整体技术的竞争力

管理路线：

1.定目标 —— 包含业务支撑、技术成长和团队培养三部分

2.看执行 —— 主要是对现有流程的问题进行梳理，让过程更加规范化、流程化，同时需要发掘高潜，给其空间快速成长

3.拿结果 —— 需要有可量化的具体数据来作为结果的衡量依据

【代码题】实现一个拼手气抢红包算法

提供了一个 RedPackage 的类，初始化时传入红包金额和个数，需要实现一个 openRedPackage 方法，每调一次都进行一次“抢红包”，并以 console.log 的形式输出抢到的红包金额。

```
class RedPackage {
    money = 0;
    count = 0;
    _remain = 0;

    constructor(money, count) {
        this.money = money;
        this.count = count;
        this._remain = money;
    }

    openRedPackge() {
        // 已经抢完了
        if (this.count <= 0) {
            console.log('红包已经被抢完啦~');
            return;
        }

        // 只剩一个红包
        if (this.count === 1) {
            this.count--;
            console.log(this._remain);
            return;
        }

        const ratio = Math.random() * (this._remain / this.money);
        // 这里会涉及到一个JS计算精度的问题
        // 正常应该用第三方库或者手写单算法实现一个精准的加减乘除
        // 这里为了简单就这么直接做了
        let youGet = (this.money * ratio).toFixed(2);
        const tempRemain = +(this._remain - youGet).toFixed(2);
        const allLeast = this.count * 0.01;

        // 如果剩余的金额不够每人一分钱，那么需要减少本次获得的金额
        if (tempRemain < allLeast) {
            youGet = +(this._remain - allLeast).toFixed(2);
            this._remain = allLeast;
        } else {
            this._remain = tempRemain;
        }
        console.log(youGet);
        this.count--;
    }
}
```

阿里

一面

自我介绍

小提示：说实话，自我介绍说多了，都可以流利的直接背下来了，一大段话在两分钟内；

前端参数配置有没有做成动态取的（从后端）

小提示：面试官的意思就是，如果配置成 HTTP 方式配置，就可以变成动态配置方式；

直接回答了没有，我给的原因是如果 SDK 初始化时通过 HTTP 去拉配置表，可能会造成主应用的性能损耗；

数据清洗的实现逻辑？

在一开始的**整体架构设计**中已经说明：

1.数据清洗是为了白名单、黑名单过滤等的业务需要，还有避免已关闭的应用数据继续入库；

2.数据聚合是为了将相同信息的数据进行抽象聚合成 issue，以便查询和追踪；

3.这样子假设后续我们需要在数据库查询：某一条错误，产生了几次，影响了几个人，错误率是多少，这样子可以不用再去 ES 中捞日志，而是在 MySQL 中直接查询即可；

4.并且，我们还可以将抽象聚合出来的 issue ，关联于公司的 缺陷平台（类 bug 管理平台），实现 issue 追踪 、 直接自动贴 bug 到负责人头上 等业务功能；

削峰限流的实现逻辑

假设说，有某一个时间点，突然间流量爆炸，无数的数据向服务器访问过来，这时如果没有一个削峰限流的策略，很可能会导致机器 Down 掉，

所以说我们有必要去做一个削峰限流，从概率学的角度上讲，在大数据量的基础上我们对于整体数据做一个百分比的截断，并不会影响整体的一个数据比例。

前端暴力削峰有没有考虑以用户为比例；

小提示：这个问题确实我有一点意外；原先都没有考虑过前端的削峰限流直接以用户为维度进行处理；

人话：在初始化 SDK 的时候，就 Math.random 来决定这个用户是否需要上报；这样避免了有一些用户的数据上报一部分后，数据链路追踪不完整的问题；

原先确实没有考虑到，聊后有所收获；

监控哪些性能指标

小提示：我回答这个问题，都会先回答：我将性能指标分为以用户为中心的性能指标和以技术为中心的性能指标，然后再分开聊；

LCP 的值，如果被骨架屏干扰怎么办；

这个问题一开始确实懵了几秒钟，之前都没有考虑过这个问题；然后就跟面试官讨论可行的方案，讨论了好几分钟；

我提出的方案：类似于用 mutationobserver 采集首屏时间的逻辑，采集 DOM 元素变化最大的点，并且让骨架屏的 DOM 在权重里面为 0；就可以忽略骨架屏对于 LCP 的影响；

最后面试完后研究了一下，发现官方对于 LCP 的取值，本身就不取骨架屏的 DOM 元素，它只取文本、img 等有效元素所在的 DOM（笑哭）；

上报 PV 还附带哪些数据

小提示：聊到上报 PV 时，会附带将各种元数据带上去，包括当前页面信息、用户来路、用户跳转方式、用户 UID、会话 ID、登录 ID；

在线协作项目相关

这个项目更多的是架构设计方面的东西，因为跟公司业务强相关，问题答案就不细写了；

包括了下面这些方面的问题：

1.在线协作表格的整体架构设计

2.每个分层之间的细节设计实现、职责

笔试：

上面的聊完后，面试官说发个笔试链接给我，我们做几个题，然后就把链接发到我的邮箱，打开三道笔试题；

1.监听页面的所有 a 标签 `click` 事件，判断 `href` 是否以 `https` 开头；

2.手写 `Event` 事件，包括 `on`、`off`、`once`、`trigger`

3.爬楼梯，LeetCode 原题，但是我忘了是 70. 爬楼梯 还是 746. 使用最小花费爬楼梯

二面技术面

自我介绍

小提示：自我介绍尽量保持在 2~3 分钟之间；语速均匀，不要结巴；可以多说一下自己非常熟悉的项目，试着将面试官的注意力往自己熟悉的方向拽；

为什么要自研监控平台

小提示：因为目前市面上成熟的监控产品很多，如果你是自研的监控平台，势必会被问道这个问题，可以提前先总结归纳一下；

我的大概回答：

1.可以实现前后端全链路 API 监控

2.接入公司缺陷管理平台（bug 平台，实现一系列的监控告警闭环流程）

3.可以自定义业务的监控维度和数据分析维度（一些具有独特监控角度的业务，需要自研）

4.这里面试官比较在意的点在于：自研相较于使用 Sentry 是否更有意义价值，自研的契机是什么，讨论了很久；

全链路 API 监控是做了什么？SDK 在设计时考虑了哪些问题？

我的大概回答：

1.如何设计成支持多平台、可拓展、可插拔的 SDK

2.如何设计 SDK 的上报策略

为什么不用 gif 图片，跨域怎么办

小提示：上面回答了上报策略后，面试官追问为什么不用 gif 图片、跨域怎么处理；

Image 是以 GET 方式请求图片资源的方式，将上报数据附在 URL 上携带到服务端，而 URL 地址的长度是有一定限制的。规范对 URL 长度并没有要求，但是浏览器、服务器、代理服务器都对 URL 长度有要求。有的浏览器要求 URL 中 path 部分不超过 2048，**这就导致有些请求会发送不完全**。

至于跨域问题，作为接受数据上报的服务端，允许跨域是理所应当的；

有没有考虑用 gif 分片上报

小提示：因为前面提到 xmlhttprequest 进行合并上报，作为 sendbeacon 的兜底方案；就问有没有考虑不是使用 xmlhttprequest 而是采用分片的方式进行上报；

上报失败怎么处理

小提示：这种题目，就将自己的实现、思考都说出来即可；

我的大概回答：

如果请求失败，会将数据隔几秒重传一次，如果再次失败；就中止所有的上报；将数据存入 localstorage；

下次进入时，判断 localstorage 要不要进行重传；

采集了哪些性能指标

小提示：我回答这个问题，都会先回答：我将性能指标分为以用户为中心的性能指标和以技术为中心的性能指标，然后再分开聊；

[以用户为中心的性能指标]

什么叫以用户为中心的性能指标呢？其实就是可以直接的体现出用户的使用体验的指标；目前 Google 定义了 FCP、LCP、CLS 等体验指标，已经成为了目前业界的标准；对于用户体验来说，指标可以简单归纳为 加载速度、视觉稳定、交互延迟等几个方面；

1.加载速度 决定了 **用户是否可以尽早感受到页面已经加载完成**

2.视觉稳定 衡量了 **页面上的视觉变化对用户造成的负面影响大小**

3.交互延迟 决定了 **用户是否可以尽早感受到页面已经可以操作**

而针对于上面三个方面，我们可以采集以下 **多种指标** 进行衡量

[性能指标评分是怎么做的]

小提示：这种问题，先说明不能对所有的应用设置统一的基准线

然后再说说不同指标的权重不同，分数占比不同等；

我的大致回答：

首先有一个结论，那就是对于所有的应用设置统一的性能指标基准线是没有意义的；因为不同项目的体量、设计都是不同的，标准也就不同；

所以说我们在平台上有一个设计，那就是可以在界面上设置具体指标的评分基准线；我们会有一个默认的性能指标基准线以及指标比重；指标包括：FP、LCP、CLS、FID 这几个以用户为中心的性能指标，然后用户可以自己在默认基准线和比重的基础上进行调整；

然后最后会根据设置的基准线进行计算得出评分；

错误聚合怎么实现

小提示：前端通过生成错误 hash 值，判断是否同一个错误

SDK 如何生成错误唯一 ID

上面我们有提到一个 错误 ID，它的作用分两种：

1.在客户端用以实现会话级别的上报去重；

2.在服务端用以实现相同错误的数据聚合；

但在实际应用中，我们如果仅仅根据 错误信息、错误行号、错误列号、错误文件 来进行判断，可能还不够准确，所以，我们需要将堆栈信息纳入聚合算法中

我们根据错误堆栈，解析出了 错误文件名、错误函数名、错误行号、错误列号等信息；

我们再利用上述的所有信息，最终生成一个 hash 值，这个值就是能够完全的描述这个错误的唯一性的 ID；

告警逻辑 & 策略

小提示：提到了告警逻辑有宏观告警和微观告警，还追问微观告警是否会带来很多误报情况（网络情况等）

首先，监控告警不是一个易事，在什么情况下，我们需要进行告警的推送？

我们先来了解两个概念：宏观告警 和 微观告警；

key	宏观告警	微观告警
告警依据	是否超出了阈值？	是否有产生新的异常？
关键指标	数量、比率	单个异常
比对方法	时间区间内的 异常数量、异常比率	新增的异常且异常 uid 未解决

- 宏观告警 更加关注的是：一段时间区间内，新增异常的数量、比率是否超过了阈值；如果超过了那就进行告警；
- 微观告警 更加关注的是：是否有新增的、且未解决的异常；

我们团队这边目前做的都是微观告警；只要出现的新异常，它的 uid 是当前已激活的异常中全新的一个；那么就进行告警，通知大群、通知负责人、在缺陷平台上新建 bug 指派给负责人；

在线协作项目细节 & 架构设计 & 难度深挖

这个项目更多的是架构设计方面的东西，因为跟公司业务强相关，问题答案就不细写了；

问题包括：

1.整体项目的架构设计分层

2.聊聊技术选型

3.自己的角色 & 负责的架构分层

4.网络层的职责 & 设计

5.网络层的难点 & 挑战 & 解决

6.如何保证 OP 消费有序且正确

7.OT 实现的算法逻辑 & 架构方案逻辑

8.为什么要做在线协作项目，能给公司带来什么产出；

有什么想要问我的

小提示：在面试的最后，抓住这个机会问一些内容，包括说团队、技术、等等；

我的大致回答：

介绍一下团队和团队业务

想要招一个什么样的人；

对我的面试印象是怎么样的

HR 面

HR 面我就列举一下具体的问题，没有答案了哈，总计 1h

聊天

简单自我介绍

毕业时对读研和直接工作的考虑

追问：跟父母沟通读研和工作时是否有意见不统一的时候，怎么解决；

本科的时候有没有什么实习经历

回答：没有实习，但是有在学校研究实验室有过经历

追问：导师是怎么评价你的

追问：导师让你做前端你是什么想法

追问：导师让你做研究的时候有没有不开心、困难、挫折的事情

追问：遇到了挫折的事情后有没有做什么总结反省呢

你现在所在的公司的收获是什么

为什么想要离开呢

心中对阿里、对岗位的期待

业余爱好

使用淘宝的时候，有什么可以觉得做得更好

回答：我举例了淘宝购物车的例子，购物车扩容；

追问：还有什么觉得淘宝没改，或者改没有改好的地方吗

追问：觉得拼多多、京东有什么做得比淘宝更好的地方吗

觉得这一段时间，对于阿里巴巴来说过的不顺，比如股价跌、垄断，你怎么看待这个事情

你觉得自己未来还有什么可以提升的，在非技术层面

网易

一面：视频面，70 分钟

1.mobx、redux 区别

2.react diff 算法

关键词：节点查找，同级比较

3.react 事件机制

关键词：事件代理，冒泡

4.原生事件哪些不冒泡，react 如何处理

5.react-redux 原理

关键词：context，provider，带上 dispatch

6.父组件 C 有两个子组件 A、B，B 有 C 传来的 props。问如果 C 传递给 B 的 props 改变了，A 会怎样的处理，执行哪些钩子

7.讲下 router

关键词：链接和视图同步

8.react 15 16 有哪些钩子不同

9.http 缓存

10.前端安全有了解吗

关键词：xss，csrf

个人感受：网易一面难度整体略微比字节跳动二面低点（就我遇到的而言），感觉网易挺狠的呀，一面都这么难，以为要凉

二面：现场面，50 分钟

两天后 hr 打电话说一面过了，邀我去北京现场面，小激动

1.讲讲你用 three.js 做的这个项目

2.3D 的立体图如何实现

关键词：正方体形，球形

3.react 生命周期

4.componentWillMount 和 componentDidMount 的区别

关键词：真实 dom

5.react 学习中遇到的难点

6.H5 项目如何适配

关键词：vw，rem，fastclick...

7.node 中间件机制

关键词：请求截获，挂上属性

8.generator, yield。附加题：co 模块如何实现

关键词：线程让权，状态机

9.xss 及防御

关键词：储存型、反射型、dom 型

个人感受：现场面等了好久，以为自己走错房间了，我想如果面试官不稍微解释下为啥迟到我就直接不面了。面试官也没有提前准备，拿着简历看了一分多钟才开始

三面：现场面，30 分钟

由于二面直接过了，二面的面试内容还没有上传上去（因为是现场），三面的面试官同样也没有准备，像是 hr 临时拉上去的，看了简历一分钟才开始。面试过程中还不停用手机催促赶紧发来一、二面内容。所以也没问些啥

1.mobx、redux 区别

2.H5 项目都干了些啥

3.做的项目流程是怎样的

4.如何规划一个项目功能

5.印象深刻的 eslint 规则

6.react 15 16 的区别

关键词：钩子，移除模块，createProtal...

个人感受：现场面有点失望，我一直是网易的死忠粉，或许是期望越大失望越大。不过我看网易的现场面确实准备的不充分，等了很久的人不止我一个，甚至还有人来到现场都给安排的视频面，可能是面试官太忙吧，或者出差啥的

整体感受

网易整体面试难度还是有的，就是现场面处理的不是很好。作为死忠粉的我还是要说一句，可能确实是因为面试官太忙了，三面的面试官说面了我马上就急着有事去做。另外在现场等待时，网易的小哥哥小姐姐们也很养眼，颜值在线。嗯，就扳回到这吧