你好！

这是一份前端工程师的简历。 95 年男生，18 年应届本科毕业生。2 年前端相关实习经历，期间使用过多种现代前端框架；使用 react-native 开发过 iOS 应用程序；负责过 vue 服务端渲染项目；对 Web 前端性能优化有一定的见解。具有良好的自我驱动能力以及自学能力。喜欢面向用户的开发，希望能做出用户真正喜欢的产品。如果你对我有兴趣，就请继续读下去吧。

基本信息

- 姓名：曹俊铨
- 出生年份：1995
- 学历：大学本科
- 英语水平：CET-4 基本英文文献阅读能力
- 联系邮箱：hire@xdxd.work
- 联系电话：<请向我发邮件，我会在回复中注明>
- 个人网站：https://cdog.me

教育经历

上海工程技术大学 / 电子电气工程学院（2014.9 - 2018.6）

- 全日制大学本科，广播电视工程专业（网络通信工程）。已于 2018 年 6 月 1 日毕业，并获得工学学士学位。

工作及项目经历

上海燎野网络科技有限公司（2016.4 - 2017.6）

「燎原」创业社交 Web 前端项目（2016.4 - 2016.7）

- 使用 angular 1.x 编写的社交类门户网站。项目中使用了 SASS ，Flex 布局，RxJS 等技术构建了一套能够适应手机、平板电脑和一般电脑屏幕的响应式单页 Web 程序。我负责了其中一部分业务模块的编

写。

「燎原」创业社交 react-native ios App项目（2016.7 - 2017.6）

- 应用连接：App Store
- 基于 react-native 构建的 iOS 客户端应用程序。使用 mobx 管理应用数据状态。我从 2016 年 7 月份开始接手这个项目，与另外两个同事一起，完成了该应用的若干个迭代，期间我参与重构以及新编写了数个重要功能。这个项目使我对 iOS App 的开发有了一定的了解，打破了我以往对 js 编写的客户端应用响应慢，体验差的印象。通过我们的优化，最终达到了与原生相近的用户体验。并且借助于 code push 等热更新技术，可以很方便的进行版本迭代。

上海哔哩哔哩科技有限公司（2017.7 - 2018.6）

直播房间页 Web 前端项目

- 线上地址：哔哩哔哩音悦台
- 该项目是对老的房间页的一次全新重构，重写了全部的代码。
- 该项目使用了 Typescript，我参与的第一个用于生产的 vue 项目。使用了 vuex 管理应用的数据状态。由于之前的项目中使用过 mobx，向 vuex 的过度并不是太困难。房间页与普通的前端页面的不同之处在于，房间页的主体是播放器。我负责了一部分与播放器交互的工作，克服了文档少、工期紧等问题，通过加强和同事沟通、自行尝试 api 等方式较好地完成了工作。

直播首页 Web 前端项目

- 线上地址：哔哩哔哩直播，二次元弹幕直播平台
- 该项目是对老的首页的一次重构，重写了大部分代码。
- 同样是 typescript + vue 编写地项目，配合 SSR 化改造以及一系列运营产品侧的需求对老的首页进行的重构。由于首页是一个网站的门户，所以首屏性能是至关重要的。我在这个项目中负责了首屏性能的专项优化，实践了一系列性能优化策略，包括延迟加载非重点内容、嵌入关键样式表、js 脚本初始化时机的优化等工作。首页同样存在播放器，并且用户对于首屏时间的感官很大程度上取决于播放器流的载入速度，所以我们同样与播放器的同事进行了密切的合作。最终，在用户感官上，直播首页基本能够做到秒开；数据上，首屏时间基本控制在 1s 左右。

直播 vue 服务端渲染（SSR）项目

- 线上地址：直播首页 / 直播分区列表页-游戏分区

- 该项目是我主要负责的项目，分为两期，一期为直播首页接入主站提供的 jinkela SSR 框架；二期为直播对 jinkela 进行读写分离改造，并接入分区列表页。

- 一期主要以直播首页的 SSR 化改造为主，摸索并熟悉主站提供的 jinkela SSR 框架。在临上线前的压测时，发现直播首页在服务端渲染的时候，存在一定程度的内存泄漏问题，会导致在并发量高时频繁的 504 错误。通过上网查阅文献资料，使用 chrome 调试工具进行反复排查后发现，是由于业务代码打包时的一处配置错误引起的。我们在进行了紧急修复后，确认问题解决，保证了项目的正常上线。

- 二期主要是对于 jinkela 的改造和重构。本质上，jinkela 使用的是 vue 社区提供的 SSR 方案，其在性能上存在瓶颈。一期使用了内存缓存解决了一部分的性能问题，但是在后续测试中发现，内存缓存在高并发情况下存在被穿透的风险。一旦内存缓存被穿透，请求量直接打到 SSR 核心逻辑上，容器的 CPU 很容易被打满，导致应用失去响应而雪崩。并且一期的方案无法保证多实例情况下缓存的一致性问题。于是二期，我们引入了 Memcache 缓存；针对缓存穿透的雪崩问题，设计了读写分离的架构。所谓读写分离即将原有的 SSR 服务拆分为两个独立服务，分别是读取和写入服务，二者都只与 Memcache 交互，避免了请求量直接打到 SSR 核心逻辑的风险。同时对主站的代码进行了 TypeScript 化改造，提升了可维护性。最终，由直播改造过后的 SSR 框架顺利上线，经历了若干次考验后证明了具有良好的稳定性。

诉求

- 希望能够做 *用户喜欢的* 产品
- 是个二次元爱好者，希望能做和爱好有关的工作
- 希望获得和 *能力* 相称的薪资

感谢您看到结束，目前我正在积极寻找更好的工作机会，欢迎与我联络：hire@xdxd.work

愿双眼所及的视线 带梦想穿越境界线