李嘉奇

在职, 2周内到岗 | 五年工作经验 | | |

15201582721 | hi@jackyli.me | **前往在线版本 ↗**

掌握的技能

- 架构设计: 具备大型应用架构设计能力，熟悉微前端、Monorepo等架构模式，有丰富的性能优化经验；
- 前端框架: 熟练掌握 Vue、React、Nuxt、Next 等主流前端框架；
- 跨端开发: 熟悉 RN、Taro、uni-app 等跨端开发技术，具备小程序开发经验；
- 混合开发与交互: 熟悉 Hybrid 混合开发，熟悉 JS-Bridge 技术及 H5 与原生交互流程；
- 数据可视化: 熟练使用 ECharts、G6、F2 等可视化工具，并具备封装通用图表库的实践经验；
- 数据结构与算法: 熟悉常见数据结构，了解常用算法及设计模式；
- 网络协议与 API 开发: 掌握 HTTP 与 RPC 协议，拥有 GraphQL 开发经验；
- 服务端开发与优化: 具备 Node.js 开发经验，熟悉 Nest.js、Koa 等框架；
- 前沿技术: 关注大模型及 AI 领域，使用 GPT 提高工作效率（例如 Cursor 等），并探索将 Dify 知识库与大模型相结合构建智能问答系统。

工作经历

小红书 - 前端开发工程师

2022-08-11 ~ 至今　北京

- 主导 IDEA 大数据平台架构重构，采用 Monorepo 架构提升系统可维护性，性能提升 27%；
- 负责公司通用图表库 Delight-Charts 的架构设计与核心开发，提升团队开发效率；
- 主导性能优化专项，通过代码分割、懒加载等技术手段显著提升应用性能；
- 推动前端技术栈升级，引入新技术提升开发效率和用户体验。
- 开发过程中，与每个板块负责人员保持交流沟通，确保需求按期交付

ByteDance Ltd. - 前端开发工程师

2021-08-06 ~ 2022-06-10　北京

- 负责飞轮投放平台、巨量引擎官网、千川官网以及 dou+ 的开发与维护
- 参加 Okee 组件库、橙子建站平台物料的开发与维护
- 参与公司项目的讨论、需求评审、技术评审，并细化排期，确保按期交付需求
- 以前端视角对产品进行体验优化

教育经历

湖北・荆门 | **本科・软件工程**　荆楚理工学院

2015-09 - 2019-07

项目经验

IDEA 大数据平台 - 小红书

技术栈: Vue3 + axios + monorepo + Nest.js + Thrift

主要职责

- 负责系统架构优化、权限控制和性能提升等方面的工作，确保项目的高效稳定运行；

- 主导模块重构与配置化升级，优化数据结构，封装通用组件，提升开发效率和代码复用性。

主要做的事情

- 架构优化: 采用 Monorepo 架构对应用模块进行拆分，降低模块间的耦合度，提升代码的可维护性和扩展性，便于团队协作和版本管理；
- 权限控制: 引入 RBAC-1 权限模型，精细化管理权限点，实现基于角色的权限控制；
- 性能提升: 通过拆分子包、非核心 JS 异步加载及 CDN 缓存优化，显著提高页面加载速度，提升用户体验，降低网络带宽负担；
- 模块重构: 对洞察模块进行全面重构，重新设计数据结构并封装通用 Hook ，提升代码复用性和模块可维护性 ，为系统的后续扩展打下坚实基础；
- 配置化升级: 抽离常量 JSON 配置，使得部分筛选项和设置可以动态配置，提升系统的灵活性，并降低硬编码风险
- 组件封装: 封装 Notify 下载组件和业务中常用的组件，增强了组件的复用性与灵活性，提升了开发效率
- 微前端嵌入: 嵌入其他业务线模块，实现跨业务线的功能模块复用，进一步增强系统的灵活性和可维护性。

工作成果

- 整体页面加载速度提升27%，重新设计代码结构降低后续维护难度；
- 通过 RBAC-1 权限模型 实现了精细化权限管理，增强了系统的安全性；
- 微前端的方案实现了跨业务线的功能复用，为团队带来了更高的灵活性和扩展性；
- 重构功能模块并利用 AI 进行 Code Review ，提高了团队开发效率和产品迭代速度。

DelightCharts 图表库 - 小红书

背景

- 业务上: 公司内部亟需构建一套可复用、高扩展性的通用图表库，以展示出主题相同的图表样式。
- 需求上: 图表库的可扩展性、可维护性、可复用性、性能优化、用户体验优化等。

主要职责

- 项目搭建、架构设计、数据格式设计、插件体系设计、文档优化等
- 提升图表库的扩展性、可维护性与用户体验

主要做的事情

- 项目架构: 采用 Monorepo 架构统一管理代码，结合 Tree shaking、拆分子包技术优化包体积
- 数据格式设计: 根据 echarts 的数据结构，抽象出 dimensions 与 metrics ，支持多种图表适配
- Hook 封装: 借鉴 Hook 思想，封装通用 hooks 供团队调用，提升代码复用率
- 插件体系: 提供一套插件体系，提高图表库的可扩展性；比如 数据转换插件、tooltip 渲染插件等
- 核心封装: 封装 chart.core 文件，管理 ECharts 实例与事件，逻辑清晰，易于维护
- 用户体验: 利用 Dify 知识库 + Deepseek 大模型，优化文档搜索体验，精准识别用户意图，提高搜索准确度
- 文档优化: 封装 Sandpack 功能，实现代码在线编辑、预览、分享功能
- 前沿探索: 探索使用 ChatGPT 辅助进行单元测试与 Code Review ，优化测试覆盖率和代码质量

工作成果

- 成功构建了一套高效且具有强大扩展性的图表库，大幅提升了开发效率和代码维护性，显著降低了项目迭代成本；
- 通过精心设计的数据格式与插件体系，增强了图表库的灵活性和可扩展性，使其能够更好地适应不同业务需求；
- 优化了代码结构，提高了维护性和复用性，为团队节省了大量开发时间。

抖音 Dou+ 模块 - 字节跳动

技术栈: Next.js + JSBridge + axios + Nest.js + SequelizeORM + Thrift

主要职责

- 通过前后端协同及架构调整，优化页面静态化、预渲染效果和多平台适配，确保功能模块高效稳定运行。
- 参与系统性能监控，及时发现并解决性能问题，确保系统稳定性和安全性。

主要做的事情

- 跨端兼容: 封装统一 JSBridge 方法，兼容 Web、Native 和 Iframe 内嵌场景
- 页面优化: 使用 SSR、SSG、ISR 技术实现页面静态化和预渲染
- 长列表优化: 利用 Suspense 与 IntersectionObserver 优化任务中心页长列表，封装自定义 hook 降低维护成本
- 推送与适配: 实现 WebHook 站内信 push ，基于屏幕 DPI 进行低端手机分辨率适配
- 代码稳定性: 使用 Jest 进行单元测试，提高代码稳定性和维护效率
- BFF层设计: 使用 Nest 框架聚合 RPC 接口，封装中间件统一身份校验，并设计接口枚举 code 降低开发成本
- 自动化与监控: 利用 Slardar 进行性能监控及异常上报，结合飞书 SDK 实现异常提醒

工作成果

- 显著提高页面加载和响应速度，优化了用户体验，降低了开发和维护成本，同时实现系统稳定性和安全性的双重提升。
- 通过前后端紧密协作，不仅提升了系统的性能，还加速了产品的迭代和上线速度，进一步提升了团队的工作效率和协作水平。

橙子建站 - 字节跳动

技术栈: React + Nest.js + axios + Koa + MongoDB

主要职责

- 参与系统的组件化开发，设计并实现拖拽式页面编辑器，提升平台的易用性和灵活性；
- 参与服务端 API 接口的开发，确保用户设计的页面能够与实时数据进行有效绑定。

主要做的事情

- 组件开发: 创建可复用的基础组件，并将项目中重复的组件封装成业务组件；
- 组件渲染器开发: 通过将业务组件打包成 UMD 格式，实现组件的远程加载；
- 用户数据操作与反馈: 实现了基于后端事务机制的实时反馈，确保在数据提交过程中，如果操作失败，用户能够及时得到正确的错误提示，避免数据不一致或误操作；
- 文档编写: 编写技术文档，梳理组件的设计、功能的实现、组件的使用，方便团队成员快速上手并进行协作；
- 数据存储: 使用 MongoDB 数据库存储页面的配置和组件的状态；
- 后端性能优化: 在请求处理时采用 Redis 进行缓存，来减少对数据库的高频访问；
- Redis 缓存预热: 提前将页面的模版等需要高频访问的数据加载到 Redis ，避免数据库启动时的性能瓶颈。

开源项目

FightingDesign ↗

一个 Vue3 组件库，支持 TS、Tree shaking。