张旭

电话： 15651573371
邮箱： xuzhang3371@163.com
现居城市： 上海

求职意向： 前端工程师 / Node.js 全栈工程师 / AI Agent 应用开发工程师

个人优势 | Profile

平台能力： 近 9 年企业协作 SaaS 研发经验，熟悉开放平台、应用生态、企业权限与账号体系，具备前端与 Node.js 全栈交付能力。

架构能力： 具备大型存量应用演进经验，覆盖微前端、组件库、Monorepo、CI/CD、私有部署及性能治理。

AI Coding Harness： 长期大量使用 AI Coding，形成一套围绕项目规则、上下文、任务规划、Skills 和验证反馈的 Harness 经验。 AI Coding 个人项目：AI 育儿记录 APP – 芽芽（https://yaya.zhzxang.top）；开源项目 todo 记忆 – Sidequest（https://xuuuu51.github.io/Sidequest）

教育背景 | Education

江南大学 | 计算机科学与技术 2013.09 – 2017.06

工作经历 | Experience

阿里巴巴 – 钉钉 – Teambition 业务线 | 高级前端开发 2019.04—2026.06

负责企业 SaaS 核心业务及前端工程化建设，覆盖开放平台、企业管理后台、账号体系、业务组件库、CI/CD、私有部署和性能优化；

- **开放平台与应用生态：** 持续负责开放平台核心前端及部分 Node.js（Egg.js）后端迭代，推动应用生命周期、OAuth、开放能力及系统应用管理持续演进，支撑数千名开发者和数百家企业的应用接入。
- **企业管理后台与账号体系：** 主要负责企业管理后台前端，以及账号体系和第三方登录的前后端迭代，覆盖成员与组织、权限、安全管控、登录注册、账号绑定和 SSO 等核心链路，支撑数千家企业日常运营与配置。
- **组件库与研发效能：** 主导前端业务组件库、CI/CD 及私有部署发布体系建设，沉淀十余个组件和公共包，将日常发布操作收敛为一行命令。
- **主产品工程化与架构演进：** 主要参与 Teambition 主产品的工程化建设与架构优化，并独立完成 Web 与 H5 首屏性能专项；三周内将线上 RUM LCP P80 从 Web 2.6s 降至 2.0s、H5 3.5s 降至 2.0s。

Teambition | 前端开发 2017.08—2019.04

参与 Teambition 开放平台及应用生态的早期建设，主要负责平台核心前端与业务模块开发。

- **开放平台基础建设：** 负责开放平台早期核心前端开发，完成应用创建、配置、安装、发布、扩展接入及系统应用管理等基础能力，形成应用从开发接入到发布使用的核心链路。
- **核心业务模块建设：** 主要负责工时管理、版本管理两个模块的前端从 0 到 1 建设，上线后作为内置能力面向全量企业默认提供。

项目经历 | Project Experience

Teambition 开放平台生态建设

Teambition 开放平台面向开发者和企业，提供应用创建、接入、安装、发布、升级、OAuth、开放能力及系统应用管理。随着应用生态扩大，需要同时降低开发者接入门槛，并提升平台开放新能力的效率。

个人职责： **负责平台核心前端及部分 Node.js（Egg.js）后端开发，承担技术方案设计与关键模块落地，覆盖应用开发、接入及运行的核心链路。**

1. **低代码提升开放效率：** 独立实现了基于 Schema 驱动的动态表单方案，将组件属性、默认值、显示条件和数据源配置化；通过组件注册与动态渲染将 Schema 映射为 React 组件，统一表单状态管理和提交数据转换，并通过回调函数支持组件交互，实现扩展能力的配置化接入。
2. **开发脚手架降低工程门槛：** 独立设计并实现 Node.js CLI 脚手架，通过交互式参数、模板生成和变量替换，提供纯前端与“前端 + 轻服务”两类标准模板；内置完整的 Teambition OAuth2 授权流程，集成构建、打包及 Docker 部署方案，统一从项目初始化到部署交付的开发链路。
3. **建设 JSAPI 降低接入成本：** 独立设计并实现 TB JSAPI，基于 postMessage 构建跨窗口通信桥接，统一封装 Promise 调用、请求 ID 与回调匹配、超时及错误处理，并通过应用来源校验控制调用边界；开放任务详情跳转、authCode 获取和字段更新等宿主能力。

项目成果：
1. **开放效率：** 低代码体系累计支持 30+ 扩展能力、15+ 配置化扩展能力，将新增能力上线周期由约 2 天缩短至半天。
2. **接入标准化：** 通过开发脚手架与宿主 JSAPI，统一项目初始化、OAuth2 接入、宿主能力调用、构建打包及 Docker 部署链路。
3. **生态规模：** 支撑十数个第三方应用、数百个企业内部应用及数十个私有部署应用接入，覆盖数千名开发者和数百家企业。

Teambition 主产品架构演进

Teambition 主产品是长期演进的大型前端应用，采用 Backbone 与 React 并存的技术架构，基于 Webpack 5 和 Module Federation 构建主子应用体系。

个人职责： **主要参与主产品工程化建设与架构优化，独立负责 Web/H5 首屏性能专项，并负责一个历史 qiankun 子应用向 Module Federation 的迁移。**

1. **首屏性能优化：** 以 LCP P80 为核心指标，结合构建产物分析、Network 瀑布和自定义 Performance 埋点定位关键渲染路径，确认入口 Chunk 过大导致项目列表首屏渲染受阻；沿依赖链将水印、通知引导、重型弹窗和富文本等非首屏模块改为动态加载，并优化 Polyfill、Vendor 拆分及 Tree Shaking；在 BFF 层按页面注入关键 JS 与 API 预加载，同时减少并推迟 H5 子应用 Prefetch，降低首屏带宽竞争。
2. **子应用 MF 改造：** 将遗留 qiankun 子应用迁移为 Module Federation Remote，重构 remoteEntry 资源加载、子应用初始化与挂载链路，并适配路由基路径、publicPath 和应用生命周期；同步迁移本地开发、构建产物及发布流程，使子应用接入统一的 MF 运行与交付体系。

项目成果： 三周内将线上 LCP P80 从 Web 2.6s 降至 2.0s、H5 3.5s 降至 2.0s；遗留子应用成功纳入统一 MF 架构，统一开发与发布模式并降低后续维护成本。

Teambition 业务组件库：架构、开发与发布体系

随着 Teambition 多业务场景及开放平台生态持续发展，内部项目与生态开发者在选人、选部门、成员邀请等场景中存在大量共性需求。为减少重复开发并保持能力与交互一致，需要建设一套同时服务内部业务和生态应用的通用业务组件库，并配套统一的版本与发布体系。

个人职责： **主导业务组件库从架构设计、核心组件开发到构建发布体系的完整建设，负责技术方案和关键模块落地。**

1. **Monorepo 与构建架构：** 基于 pnpm workspace 搭建 Monorepo，按业务组件、工具和公共能力划分包边界，并通过依赖方向约束避免跨层引用；使用 Rollup 统一输出 ESM、CJS 和 TypeScript 类型声明，适配不同模块体系和 TypeScript 项目。
2. **业务组件抽象：** 统一业务状态与组件交互，围绕成员、部门和邀请场景封装异步搜索、分页加载及状态回传能力，沉淀选人、选部门和成员邀请等可跨业务复用的组件。
3. **增量构建与发布：** 使用 Changesets 管理版本变更并自动生成版本号和 Changelog，根据变更范围只构建、发布受影响的包；对接内部构建平台，将云端构建与包发布串联为一键发布流程。

项目成果： 沉淀十余个业务组件和公共包，其中选人组件覆盖全站 10+ 业务场景；日常发布操作收敛为一行命令，减少多页面切换和人工操作。