马申彦

前端工程师

无锡市・梁溪区
+86 18771058712
msyfls123@gmail.com
ebichu.cc/blog/about

- GitHub 资料

工作经历

高级前端开发工程师, 腾讯文档, 北京. **2020.7 - 2023.8**

- 负责腾讯文档桌面端的日常开发与维护
- 整合 native 及 web 资源，实现单机离线编辑 Office Word / Excel / PPT 文档的功能
- 运用 node-api 封装 C++ 数据库供所有主流 CPU 架构系统使用
- 整合 PWA 及数据预加载功能，大幅降低端内文档使用时的 TTI
- 通过 React 及 IPC 统一管理了 Electron 窗口的生命周期，
  将单窗口启动速度从 300ms 降低至 100ms
- 管理桌面端 CI / CD 流程，实现无人值守自动构建和半自动化发布

前端开发工程师, 豆瓣阅读, 北京. **2017.7 - 2020.7**

- 负责豆瓣阅读 web / mobile / app 内网页前端的开发测试构建上线工作
- 使用 Graphene 从零搭建 GraphQL 服务端，统一了书籍、用户信息的查询
- 在项目里推动了 TypeScript 的应用和规范建设，重构了近 30% 的历史代码
- 2019 年开始领导前端团队，2016.10 - 2017.1 为实习

社区

JSConf China 2019 Speech: 响应式编程使你看得更远 网址

第一届 QQ Tech 技术周演讲: 腾讯文档桌面端跨平台开发实践

InfoQ 采访: 掌控前端数据流，响应式编程让你看得更远 网址

教育

硕士, 中国地质大学, 武汉, 设计学. **2014.9 - 2017.6**

学士, 中国地质大学, 武汉, 工业设计. **2010.9 - 2014.6**

项目经历

桌面端本地编辑 Office 文档, 桌面端负责人, **2022.4 - 2023.4**

腾讯文档作为中国市场占有率第一的在线协作编辑应用，在弱网甚至无网环境下如何让用户仍能获得基本的编辑体验是一项长期且宏伟的目标。结合项目发展和既有技术沉淀的考量下，腾讯文档团队联合 web、桌面端和原生引擎三组，打通语言和地域的壁垒，在不到一年的时间内完成了基于 Electron 框架的伪 native 本地编辑功能，让用户无需联网即可完成 Office 文档的浏览与编辑，为办公软件国产化迈出了坚实的一步。

在项目中，我作为桌面端负责人，前期调研并跑通了通过 web 页面操作 Office 文档的 MVP 产品（PPT 部分），在各团队分工排期不清晰的情形下制定出了一套接口规范，为后续 Word / Excel / PDF 等品类接入夯实了基础。通过线上已有数据及竞品分析，确定并实施了全链路的性能监控上报机制。同时主导了多次打开同一文档的数据缓存功能开发，使得用户重复打开同一篇文档的耗时下降约 30%，并且据此缓存机制提供了文档异常退出或 Office 转换引擎出现错误时用户数据完整性保护功能 .

移植数据库, 唯一开发者, **2021.11 - 2022.1**

腾讯文档桌面端一经上线就获得了大量用户，同时随着 web 页面的不断迭代，原先采用的 JavaScript 封装数据库方案已经过时且不可维护，至此项目面临两种选择，要么继续为原方案修修补补打补丁，要么接入移动端 C++ 数据库。在综合考量维护成本和迁移风险后，我们选择了接入 C++ 数据库的方案。但该方案最大的瓶颈在于项目中懂 C++ 的工程师人数为 0。

经过简单的沟通，我发现移动端数据库实现并不复杂，只需要调用一个接口即可完成功能的桥接，但因为原本的实现仅支持 iOS 和 Android，这需要前期探索并跑通 Windows/macOS/Linux 三个系统上的构建编译流程。于是我兵分两路，白天根据现有功能代码和构建脚本在公司提供的虚拟机上实验编译，晚上阅读教程手搓一个简单的 node addon 以了解跨语言封装细节。经过两周时间的探索，最终数据库成功地运行在了三个系统多达 8 种不同 CPU 架构上。

但一个跨平台项目最重要的工作是如何让用户平滑地进行迁移，针对新旧两套数据库实现，我又设计了完整的数据发现，数据校验，数据迁移及数据恢复功能，最终在一整年时间内成功帮助两万余名老用户全自动迁移数据库 .

移动端编辑器, *web* 负责人, **2020.3 - 2020.6**

豆瓣阅读是豆瓣旗下的原创小说平台产品，随着移动阅读成为年轻人日常娱乐活动之一，文学创作者也会有移动写作的需求。豆瓣阅读本身已经具有桌面端网页富文本编辑器（基于 Draft.js 深度定制），将其适配到移动端 App 内即可快速上线作者编辑能力。

原 web 编辑器在 Draft.js 基础上使用命令模式封装了多种编辑操作，所以适配移动端主要开发工作在于将移动端实现的菜单与 web 编辑器进行桥接，我撰写了完整的双向通信协议和接口类型，同时对 web 编辑器与 UI 耦合部分进行重构。在联调过程中，帮助客户端开发同学梳理了项目早前定义好的 JSBridge 注入机制。项目已发布且运行良好.

热门小说小程序, 唯一开发者, **2018.5 - 2018.7**

豆瓣阅读网站上有很多连载小说，它们一般会以一个较为固定的周期更新最新章节，通过消息推送触达用户的机制仍不够完美且不能够满足用户主动发现并轻量阅读的需求，于是小程序作为一种新的日常阅读形式映入我们的眼帘。

小程序发布于 2016 年，2018 年时 API 已基本完善。在接到任务后，我将整个项目分为三部分，小程序主体功能的开发，小程序后台的开发以及小程序备案上线的工作。其中主体功能分为小程序 UI & 逻辑 + 后端接口改造，利用现有图书接口经过微信 Open ID 转化鉴权后输出小说的内容数据，小程序端接收到数据进行 tokenize 并解析成小程序内 UI 组件树进行渲染。第二部分是小程序内容的管理后台，复用前一步的数据对每周发布的小说列表进行展示，并提供 CRUD 功能。

小程序的上线，经历了一系列审核相关波折，以及团队间努力磋商，最终成功发布，最高获得了单日 5W PV 的访问量.

电子书筛选功能, 主要开发者, **2018.10 - 2019.3**

豆瓣阅读在线书店经历了 web 门户时代、feed 流推荐时代，最终过渡到了“聚合推荐发现搜索多种浏览方式的新书店”时代，与之配套的强力电子书搜索筛选功能也变得迫在眉睫。后端率先进行技术升级，使用 ElasticSearch 优化了老旧的 SQL 查询，前端也不甘示弱，通过 GraphQL 整合电子书 meta 信息，一次查询便可得到多级嵌套多维度的信息。

筛选与搜索一脉相承，在开发组件时就考虑到之后的搭配问题，将 UI props 和 GraphQL query 一一对应，这样在上层调用时便可自由组合。因为图书本身分类是多层嵌套的，分类筛选器以递归方式进行设计，逐级展开。为了让用户一次筛选的结果可以被方便地分享，我设计了一套基于 URL 的 form 数据序列化/反序列化机制，并针对交叉字段做了优化，尽可能提升字段间的正交性。

在完成逻辑功能的同时，我也感受到 React 在处理异步数据时可能存在 UI 与逻辑不一致的问题，于是在数据极快就绪时隐藏掉多余的转场动画，在功能上不谋而合地实现了类似 React Concurrent Mode 的效果，让用户体验更上一层楼 .

个人项目

基于 markdown-it 和 pdfmake 的富文本书写工具 仓库 *Demo*

为 Electron 应用设计的基于 Cycle.js 的开发框架 仓库

企业网站开发与配置 网址

Advent of Code 练习册（Rust） 网址

技能

框架/库: React / RxJS / Cycle.js / Svelte

跨端: Electron / NodeGui / Druid(Rust)

工具: Git / Gulp / Webpack / Rollup / Jest

语言: TypeScript / Rust / Python / GraphQL / C++（一点点） / LaTeX / 小程序

	等级	分类	年限	备注
React	■■■■□	库	7	
RxJS	■■■■□	库	5	
Cycle.js	■■■□□	库	4	非常喜欢它的世界观
Webpack	■■■□□	库	7	
Electron	■■■■■	框架	3	
Rust	■■□□□	语言	4	业余时间很喜欢
TypeScript	■■■■■	语言	5	
Python	■■■□□	语言	7	
Git	■■■□□	工具	7	

■□□□□ 初学　　■■■□□ 熟识

■■□□□ 了解　　■■■■□ 掌握

■■■■■ 精通

介绍

爱好: 看电影、烹饪美食、阅读、绘画、业余设计师

一句话介绍: 现在站在你面前的我，并不是幻影呀

更新于 2023 年 9 月 2 日