

此简历在线 web 版本: <https://i-want-tobelong.github.io/resume/>

# 个人信息

- 喻智勇/男/1998
- 学历: 大专 (西安交通大学计算机科学与技术)
- 博客: <http://i-want-tobelong.github.io/>
- github: <https://github.com/i-want-tobelong>
- 阅读书签: <https://www.gettoby.com/p/0qhnlxhp4lfh>
- 工作经验: 四年
- 期望职位: 前端组长/前端高级工程师
- 期望城市: 深圳/广州
- 期望薪资: 面谈

# 技术栈

以下为我熟悉的技术栈(使用 react hooks 较多):

- 基础: [html5](#) & [css3](#) & [javascript\(esnext\)](#) & [node-lts](#)
- 预编译语言: [typescript](#) & [scss](#)
- 前端库: [react hooks](#) & [vue-next](#)
- 脚手架: [create-react-app](#) & [vue-cli](#)
- 跨平台桌面应用框架: [electron](#)
- 打包工具: [webpack](#) & [rollup](#) & [vite](#)
- 编译器: [babel](#) & [esbuild](#) & [swc](#)
- 单元测试: [jest](#) & [react-hooks-testing-library](#)
- 编辑器: [vscode](#) & [neovim](#) & [android studio](#)
- 代码规范: [eslint](#) & [stylelint](#) & [prettier](#) & [husky](#) & [commitlint](#) & [lint-staged](#)
- 文档: [markdown](#) & [swagger](#)
- 版本管理: [git](#)
- 包管理工具: [npm](#) & [yarn](#) & [pnpm](#) & [bun](#) & [nix](#) & [scoop](#) & [cargo](#) & [pacman](#) & [\[paru\]](#)
- 操作系统: [nixos](#) & [archlinux](#) & [macos](#) & [windows](#) & [wsl](#)
- 终端: [windows-terminal](#) & [kitty](#) & [wezterm](#)
- shell: [zsh](#) & [fish](#)

以下为我了解的技术栈

- 系统级开发: [rust](#)
- 微前端: [qiankun](#) & [emp](#)
- 小程序开发框架: [taro](#) & [tauri](#)
- 跨平台桌面应用框架: [react.node-gui](#) & [tauri](#)
- webassembly: [wasm-pack](#)
- 原生移动端: [java](#)

更多可能(我对我的技术栈没有限制, 因为看完文档都能写的很好)

# 个人优势

自学前端并积累了整套前端自学资源, 会基础的后端开发。五年编码经验, 四年开发经验, 涉猎广泛, 熟悉前端框架选型, 脚手架搭建, 自动化测试, 持续集成持续部署, [gitlab](#) 私服搭建, 可以配置各种脚手架提高组员开发效率和开发体验, 日常使用 [linux](#) 发行版声明式系统 [nixos](#) 进行开发, 并业余从事 [nixos](#) 开源社区的建设, 熟悉函数式编程理念, 写得一手同事都夸赞的干净代码。

# 联系方式

- 手机: 13606640872
- 邮箱: [floatshuyin@gmail.com](mailto:floatshuyin@gmail.com)

# 工作经历

- 公司名称: 深圳十方融海科技有限公司
- 职位类型: 前端开发
- 工作内容: 负责 erp 中台, 用户数据可视化中台的开发与维护
- 在职时间: 2022.03 - 2022.08

- 
- 公司名称: 深圳观麦科技有限公司
  - 职位类型: 前端开发
  - 工作内容: 负责面试、erp 项目的进销存与分拣模块、以及 PC 端 electron 和安卓端分拣软件的开发与维护,
  - 在职时间: 2021.03 - 2022.02

- 
- 公司名称: 西安猫兜灵智能科技有限公司
  - 职位类型: 前端组长
  - 工作内容: 框架选型, 脚手架搭建, 组件开发, 前端自动化测试
  - 在职时间: 2019.12 - 2021.02

- 
- 公司名称: topcoder
  - 职位类型: 前端开发
  - 工作内容: 前端 web 页面开发
  - 在职时间: 2018.06 - 2019.11

# 项目与开源库

## 基础库

### @sfe/video-player

@sfe/video-player 是从中台和 C 端抽离出来的基于 h5 video 标签的自研播放器。

我主要处理了离职员工遗留的大量的 `typescript` 类型错误和 bug。

- 修复了 package.json 未声明 type 字段导致引用改库的项目类型报错问题。
- 修复了自动播放功能
- 修复了弹窗点击事件冒牌导致的进度条响应了该事件回退进度的问题
- 修复了播放器控制栏 DOM 层级错误导致的错误的层叠上下文问题。
- 添加了全屏和倍速功能
- 使用 `rollup` 和 `rollup-plugin-typescript2` 对该库打包，并使用 `tsc-alias` 修复了类型导入路径的别名问题
- 配置 `eslint` & `stylelint` & `prettier` & `husky` & `commitlint` & `lint-staged` 以保障代码质量与规范 git commit message. 配置了 `vscode` 编辑器 `"source.fixAll": true` 配合 `prettier` 和 `eslint` 实现保存代码自动格式化, 在提高代码质量的同时提高编码体验.

### @sfe-core

@sfe-core 是中台和 C 端的前端基础建设库，包括编译构建和 api 生成等基础功能。

- 修复了根据 `swagger` 生成 `typescript` 接口函数时无法处理多维数组的 bug
- 升级 `eslint` 和 `prettier` 以及 `typescript` 和 `swc` 以支持更新的 esnext 语法
- 添加了接口路由根据请求接口对新老 axios 实例进行派发，实现了渐进式的接口迁移

## 移动端

---

### 分拣软件

春节前一周临时收到一位移动端分拣软件客户的智崎电子秤无法适配的问题，我根据官方文档快速上手 `java` 并安装了 `android studio` 更换串口通信库为 `SerialPortHelper` 后重新打包 apk 完成了分拣软件安卓端对智崎电子秤的适配，获得了客户的好评!

## PC

---

### 分拣软件

分拣软件 是一个桌面软件，可通过串口和蓝牙连接电子秤与电子秤通信，还可以连接打印机打印分拣标签。

分拣是整个 ERP 系统的重要环节。

技术栈方面使用 `electron` & `react` & `typescript` & `mobx`.

我使用 [html-react-parser](#) 重构了打印页面模块，接入了 [react](#) 生态，解决了在单个 html 文件里开发导致的许多问题，并使用 [promise](#) 队列解决打印过快时打印机漏打的问题。

## Web

---

### 观麦 ERP

观麦 ERP 是一个将物流、财流、信息流集成化管理的应用系统，包含采购、销售、库存、客户、财务等模块。

技术栈方面使用 [gm-react-app](#) & [react hooks](#) & [typescript](#) & [tailwindcss](#) & [react-router](#) & [mobx](#).

主要负责 erp 项目的进销存模块需求评审，开发以及维护。

### 学习观

学习观是一个应用多模态学习的在线学习平台。

技术栈方面使用 [create-react-app](#) & [react hooks](#) & [typescript](#) & [emotion](#) & [react-router](#) & [easy-peasy](#)，并大量使用了 [react-use](#) 库中的 hooks。

我主要负责框架选型，工程化配置，代码规范，单元测试，以及视频模态区的编写。

具体工作内容如下：

- 使用 [create-react-app](#) 脚手架创建项目，并使用 [craco](#) 自定义 [create-react-app](#) [webpack](#) 等配置，避免了 eject。
- 配置 [eslint](#) & [stylelint](#) & [prettier](#) & [husky](#) & [commitlint](#) & [lint-staged](#) 以保障代码质量与规范 git commit message。配置了 [vscode](#) 编辑器 `"source.fixAll": true` 配合 [prettier](#) 和 [eslint](#) 实现保存代码自动格式化，在提高代码质量的同时提高编码体验。
- 编写 README 文档，主要是写明了一些开发环境的需求(Requirements)，以及快速开始运行项目的步骤(Getting started)，还有就是推荐一些提升编程体验的编辑器插件(Enhance your development experience)，最后是 QA 部分会写明一些常见问题的解决办法，比如包管理工具 [yarn](#) 的淘宝镜像设置，帮助新人快速融入到项目开发中。
- 使用 [react-testing-library](#) 和 [jest](#) 编写前端路由的快照测试。
- 配置 [webpack](#) [postcss](#) 插件，使用 [purgecss](#) 剔除未被使用的 css class，以减少编译后的 css 文件大小。
- https 请求库选用支持 [node](#) 与 [promise](#) 的 [axios](#)，[axios](#) 还有拦截器插件机制，通过编写拦截器很方便的实现了 [jsonwebtoken](#) 的获取与自动续期。
- 编写 [useAxios](#) hook，并使用 [swr](#) 增强 [useAxios](#)，还使用 [react-hooks-testing-library](#) 编写了 [useAxios](#) hook 测试。
- 编写 loading 和 404 页面，页面路由时 loading 使用的 [react suspense fallback](#)，配合 [react.lazy](#) 异步加载页面，优化页面请求大小。
- 搜集资料做 [react hooks](#) 全局状态管理库的选型，并输出了一篇[比对文章](#)：最终选用了 [easy-peasy](#)。
- [css-in-js](#) 库的选型时参考 [awesome-css-in-js](#) 后选用了 [emotion](#)。

- 视频模态区的播放器组件使用 [videojs](#), 进度条使用的 [react-compound-slider](#), 并编写了 `usePlayOrStop` / `usePlaybackRates` / `useVideoQuality` / `useVideoTime` / `useVolume` / `useFFFR` 等 hook.
- form 组件和 form 验证选用 [formik](#).
- i18n 选用 [react-i18next](#).
- 动画库选用 [react-spring](#), 对于简单的动画还使用了 [animate.css](#) 库, 对于 [animate.css](#) 我封装了一个 hook: [use-animate-css](#).
- 编写 `useSocialShare` hook, 这个 hook 封装了各个社交网站的分享 api.
- 更新项目依赖项并跟进 `Breaking changes`.
- ...

## 学习观后台管理系统

技术栈方面使用 [create-react-app](#) & [react hooks](#) & [typescript](#) & [emotion](#) & [react-router](#) & [easy-peasy](#) & [ant design pro](#).

我主要负责框架选型, 工程化配置, 代码规范, 单元测试.

具体工作内容如下:

- 编写 `useRouter` hook 实现配置式动态路由, 路由守卫等功能.
- 配置 [webpack](#) 插件 [mock-api](#).
- ...

## TC3.0 Marketing Site Prototype

[topcoder](#) 的 marketing site 页面

- 为 marketing site 添加 svg 动效
- 为 topcoder 主页导航栏原型添加交互动画

线上地址: <https://www.topcoder.com/>

项目地址: <https://gitlab.com/FloatingShuYin/tc3-marketing-site-prototype>

## connect-app

参与 topcoder 的 connect-app 项目开发

- 使用 React 编写 [draft](#) 编辑器的插件实现类似 [google docs](#) 链接编辑模块的功能

线上地址: <https://accounts.topcoder.com/connect/registration>

项目地址: <https://github.com/backtolife2021/connect-app>

## 开源项目

## devos

devos 是构建我的 nixos 系统的 nix 代码的所在地. nixos 是一个 linux 发行版, 特点是使用图灵完备的函数式语言 nix 来编写和构建属于你自己的声明式操作系统.

而 nix 是一款适用于 linux 和其他 unix 系统的强大包管理器, 使包管理变得可靠且可重复.

想象一下, 一个通过 uefi 和 grub 引导的运行在 linux 内核上的现代操作系统的构成部分, 从黑色背景的 tty, 到登录管理器然后是桌面环境, 桌面环境的 shells, trays, docks, themes, widgets, ime 以及日常开发使用的所有软件开发工具以及它们的配置项都以声明式的文本代码形式编写并组织在一个 git 仓库中, 它们大多是 foss 的, 并且拥有与你的 cpu arch 兼容的二进制源, 其余的则从各种 git 源拉取源代码, 你只需要输入一行命令, 从代码中声明的一切, 下到磁盘分区, 上到桌面主题, 它们由各式各样的编程语言编写, 使用各式各样的编译器, 在此刻都编译并构建为你自己声明的操作系统.

## localsend

localsend 是使用 flutter 框架编写 airdrop 的开源跨平台替代品, 可以运行在 android ios mac windows linux 多个平台上, 以在局域网中互传文件.

flutter 继承了 gtk 长期存在的问题, gtk 拒绝提供合理的默认值, 并且拒绝最终用户调整, 并将责任交给软件开发者. 目前使用 gtk 的软件将默认使用客户端装饰.

这对一些运行在 wayland 合成器上拥有服务器窗口装饰的桌面环境来说会带来一些不一致性, 我对此进行了修复.

## autohide-tdrop

### autohide-tdrop

终端是我们开发过程中必不可少的工具, 而一个通过快捷键打开的可自动隐藏的下拉式终端则可以提高我们在开发过程中频繁的从其他软件中切换的效率

autohide-tdrop 是一个使用 rust 实现的命令程序, 通过监听 x11 服务器发出的焦点切换事件来自动隐藏 tdrop 管理的窗口, 可以在失去焦点时自动隐藏终端或由 tdrop 管理的其他应用程序.

支持 x11 和 wayland 下的 xwayland.

## zaza

zaza 是一个使用 typescript 来编写 neovim 配置的项目。

neovim 现已支持使用 lua 来编写配置, 而 typescripitolua 则可以将 typescript 代码转换为 lua 代码, 因此使用 typescript 来编写 neovim 配置是可能的。

如果要使用 `typescript` 来编写 `neovim` 配置, 首先要做的就是提供 `neovim lua` api 的类型文件。  
此处参考 `lua-dev.nvim` 的做法, 从 `neovim` 获取到 `message pack` 文件, 并从 `vim-lsp` 获取到 `builtin-docs.json` 文件, 这些文件描述了 `neovim` 的 `lua` api, 我们可以将其作为输入, 通过 `typescript` 的编译器 api 生成并输出 `d.ts` 文件。

`message pack` 的获取通过编写一个 `archlinux` 的 `dockerfile`, `git clone neovim` 然后执行生成 `message pack` 的 `python` 脚本, 最后将生成的 `message pack` 导出 `docker` 到宿主机即可。  
由于 `message pack` 与 `builtin-docs.json` 描述的信息不全面, 因此生成的 `neovim lua` api 的函数参数类型多为 `any`, 需要人工的在此基础上对常用 api 做一层封装, 提供类型安全的便捷的上层 api。

## use-animate-css

使用 `typescript` 编写的 `react hooks`, 封装了 `animate.css`

🌟 Live Demo: <https://codesandbox.io/s/useanimatecss-tyklw?file=/src/A.tsx>

项目链接: <https://github.com/backtolife2021/use-animate-css>

## 文章

---

## development-environment-manual

这是一篇使用 `windows` 系统包管理工具 `scoop` 搭建开发环境的教程, 特点是几乎所有操作都是通过纯命令行的方式去做的, 这意味着在 `windows` 系统上也可以通过自动化脚本一键配置开发环境。

包括以下编程语言:

- `c/c++`
- `golang`
- `java`
- `javascript`
- `python`
- `rust`

开发环境的搭建涉及四个方面, 软件安装, 环境变量, 编辑器, 等宽字体

软件安装方面, 使用 `windows` 包管理工具 `scoop`

通过包管理工具 `scoop` 下载安装软件将会自动处理软件依赖, 添加环境变量等琐碎事情

编辑器则推荐的开源编辑器 `vscode`

除此之外还推荐了支持 `ligatures` 的 `monospaced` 字体: `firacode`, 对中文与 `powerline` 的支持都很好的:

项目链接: <https://github.com/backtolife2021/development-environment-manual>

# 致谢

感谢您花时间阅读我的简历，期待能有机会和您共事。