如何管理前端项目

@被删

1

前端项目的搭建

一、选择合适的框架

项目的技术选型—前端框架

基于团队成员偏好和能力，选择适合的前端框架

框架	优势	不足
Angular	• 大而全 • 多人协作友好	• 相对笨重 • 设计和使用的概念很多（如依赖注入/注入器/令牌、指令、模块化、AOT 等），入门成本较大
React	• 对前端编码侵入较少 • 轻量，可灵活搭配各种状态管理工具、脚手架等	对于大型复杂项目，需要自行搭配其它配套工具来解决
Vue	• 对新人友好、文档和社区较完善 • 轻量，可灵活搭配各种工具进行开发，官方也提供完整的全家桶解决方案	• 如指令和语法糖有一定的概念门槛 • 对于大型复杂项目，需要自行搭配其它配套工具来解决
自研	• 可满足业务特殊场景和需求	• 开发和维护的成本较高 • 容易成为历史债务（屎山）

Angular = React/Vue + 路由库（react-router/vue-router） + 状态管理（Redux/Flux/Mobx/Vuex） + 脚手架/构建（create-react-app/Vue CLI/Webpack） + ...

二、选择合适的工具

项目的技术选型—工具库选择

基于项目规模，选择是否需要路由管理、状态管理等工具库

1. 路由库：跟随框架（ngRoute、vue-router、react router）

2. 状态管理工具：

    - 事件通信（Event/Emitter）
    - 单向数据流（Redux、Vuex）
    - 响应式数据流（RxJS、Mobx）

3. 其他：

    - 构建工具（跟随框架、Webpack、Gulp、Rollup）
    - 组件库（Antd、ElementUI）
    - 请求库（Axios）

如何进行技术选型？？

技术选型四字箴言：合适就行！！！

1. 项目规模
2. 业务场景
3. 团队技术栈
4. 实用性 > 高级感

2

前端项目的维护

一、解放我们的双手

项目的管理维护—自动化工具

基于成员规模和项目状态（快速上线、稳定维护等），选择是否需要代码构建、自动化测试等自动化工具，以及搭建持续集成、持续部署等自动化流程

1. 人越多，事情越多
2. 技术债务在积累

持续集成 CI + 持续部署 CD

- 代码规范校验自动化
- Git 版本管理（分支合并/打 Tag/创建发布分支等）
- 自动化测试

代码合入、部署、发布、灰度、监控自动化

二、约束我们的步伐

项目的管理维护—流程规范

使用一致的项目规范，包括项目代码结构、代码规范、开发流程规范、
多人协作规范等内容

1. 代码规范（命名、目录规范、编码规范）
2. 开发流程规范（Git 分支、代码合入、Code Review、代码发布）

} 使用工具保证！！！

项目的维护永远是程序员的大头！！！

前人种树，后人乘凉

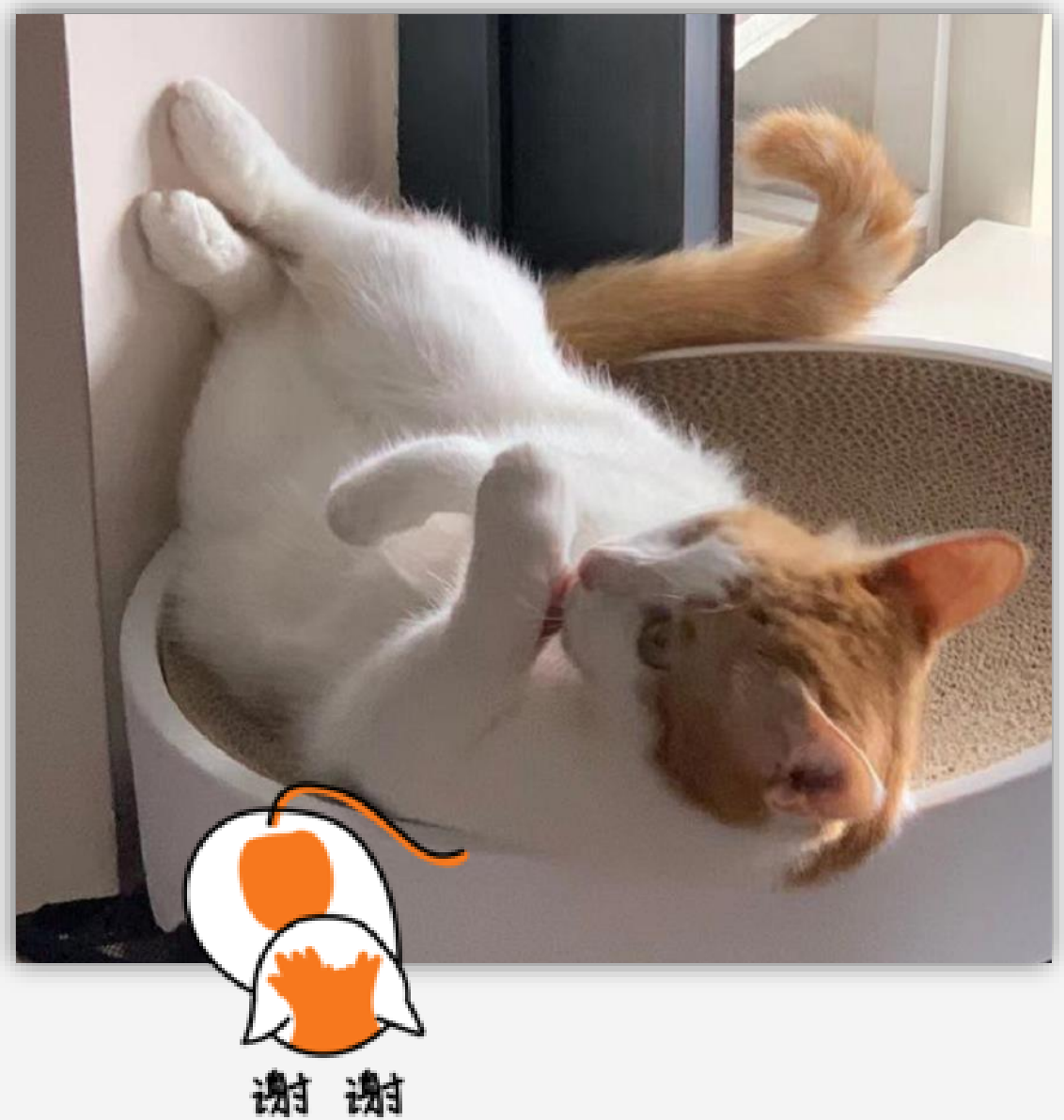
前人拉屎，后人铲屎

三、控制前进的节奏

项目的管理维护—进度把控

1. 工作量评估
2. 分工排期
3. 风险评估（时间和质量）
4. 复盘和总结

具备 Owner 意识



Github: godbasin
@被删