前端架构技术体系

一、整体技术体系

二、前端技术规范

- 前端团队全部的开发规范： 开发规范
  - npm私有包发布和版本管理流程**（废弃）→ NPM私服**
  - VUE业务组件数据流规范(草案)**（废弃）**
  - 前端代码规范Code Guide**（重点关注）**

三、前端工程

1. **2018年3月**
   - 从 JQuery 切换到 Vue.js 技术栈。
   - vue-cli 2设计，webpack等配置暴露在工程中。
   - 多工程。
   - 组件库基于elementUI做的扩展发布megvii-ui 1.x，开发了自己的复杂UI组件，完全拷贝了v-charts源码来做charts。
   - insight-utils封装了axios。
   - 洞鉴5.0。
2. **2018年12月**
   - 加入npm私服，和后端共用。
   - 重新设计了module，引入pages。
   - 引入了公共业务组件工程common-web。
   - 洞鉴5.1，慧寻1.0。
3. **2019年12月**
   - 切换到单工程。
   - Module和菜单的完善。
   - 拷贝elementUI（megvii-ui 2.x），分离megvii-charts、megvii-icons，megvii-http等，原来的复杂UI组件临时放在megvii-ui-pro中。
   - 灵探2.0，洞鉴6.1，城管1.0等
4. **2020年4月**
   - 满足业务需求的布局修改。
   - 工程本地开发和构建效率提升等。
   - 昆仑1.0，慧寻2.0

核心原理

权限控制逻辑

前端工程更新（城管1.0后的更新）

常见问题：

为什么没用使用vue-cli 3?

- 切换到Vue.js技术栈的时候，vue-cli 3还没有发布。
- 不想随着vue-cli的不断升级而去升级架构和项目的模板。
- 我们也希望能构建自己的脚手架和工具链。

为什么一开始是多工程?

- 在做洞鉴5.0的时候，考虑到前端模块过多的开发体验。
- 产品当时有其他形态（菜单完全不同的自由组合形式），按照模块的归类分为多工程和负责模块集成的portal工程。

为什么后来又汇总成单工程?

- 多工程带来了一系列维护上的问题，定制化在项目定制方面页存在问题。
- 没有单独的时间去对多工程的方方面面做进一步完善。

四、现状和规划