Code Review Guideline

Before Code Review CR前期准备

Code Lint / Check

- [x] 【已实现】在Git hooks阶段，检查、自动修复ESLint、Style lint等CodeStyle问题，拦截问题代码、阻断Commit
- [ ] 【调研中】在Git hooks阶段或者测试环境部署CI、MR Check CI阶段，检查圈复杂度和代码重复率问题
- [x] 【试行】针对大需求的核心/复杂流程的部分，要求出技术方案（Markdown格式）给到组内或者相关人并讨论，落Git仓库/confluence wiki（可以考虑一边使用链接的形式），也相当于文档

When To Review 什么时候CR

需求开发中

- [x] 【试行】尽可能写完一部分，可以合入一部分的时候，就提MR，从而控制MR大小
    a. 比如说做完一个新的组件之类的时机
    b. 前端、BFF都有一些保证Master随时可发的手段，如隐藏路由，修改路由指向等，除了改了公共组件的情况比较麻烦，一般是可以做到的
- [ ] 【调研中】MR Check CI流程，检测本次MR影响的行数，针对不同项目制定行数标准，超过行数则阻断MR流程
    - [ ] 支持配置文件白名单
    - [ ] 支持使用Commit Message标记跳过此检查
- [ ] 【调研中】MR Check CI流程，检测是否修改到公共组件，如果修改到了，则对单测覆盖率做出一定要求，不达到要求则阻断流程
- [x] 【试行】拉一个CR专用群，在MR Check CI流程最后，根据被评审人选择的reviewer，通过群机器人通知reviewer，另一方面，CR中的疑问也可以放群里讨论留档

CR讨论会

- [x] 【试行】每周1小时左右，团队成员轮流主持，将自己认为的有价值的MR拿出来再次CR，将讨论结果记录在confluence，并发组内邮件

When Reviewing Code 应当重点关注的CR内容

CR目标：可读性高、遵从业界最佳实践、保证安全性、规避性能风险

1. 【最重要】可读性：
    a. 重点代码需要注释，且注释需要有效、易懂，不能是单纯的翻译代码这种无用注释
    b. 代码变量命名需要清晰、明确
    c. 注释原则上应该用英文写，但也可以使用双语注释
2. 代码实现问题：包括 是否遵从业界做法，组件拆分是否合理，边界条件是否考虑全，数据流是否清晰节俭 等
3. 也需要关注肉眼可见的安全和性能问题