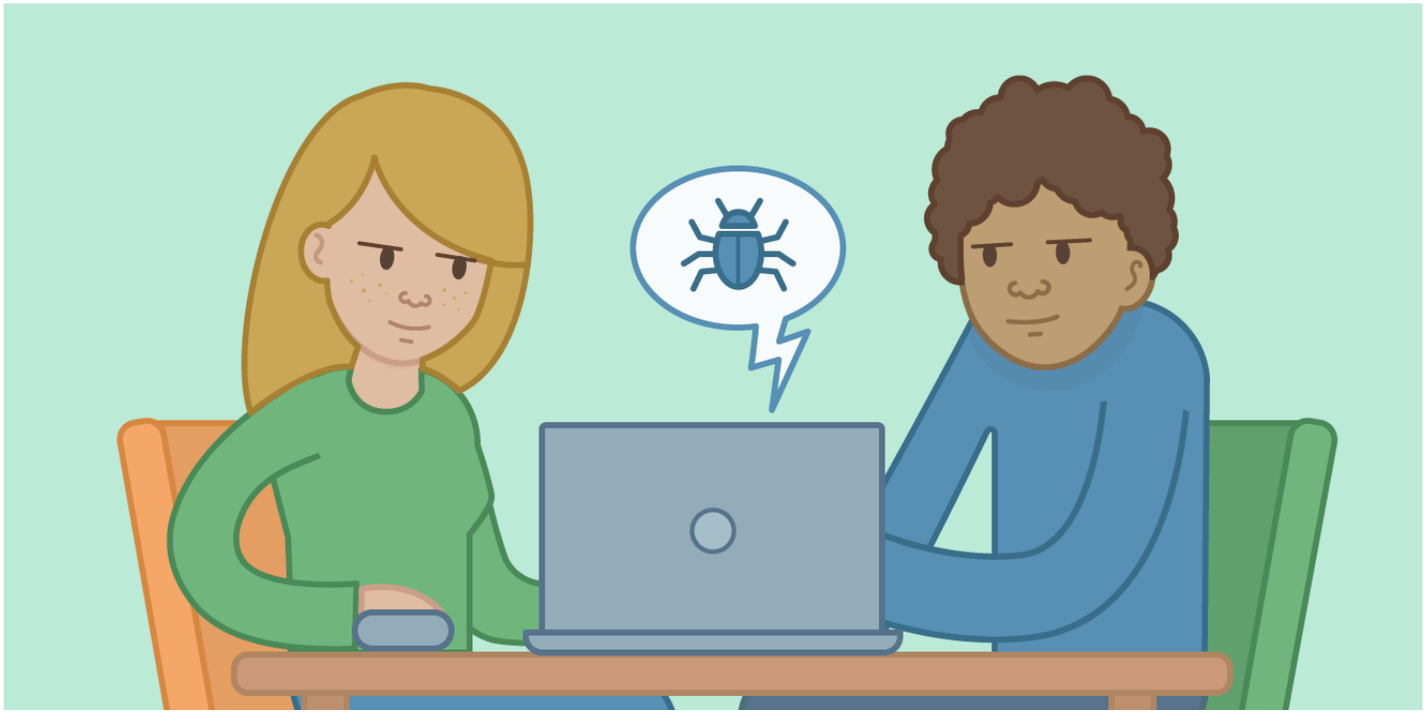
Code Review

1

什么是 Code Review

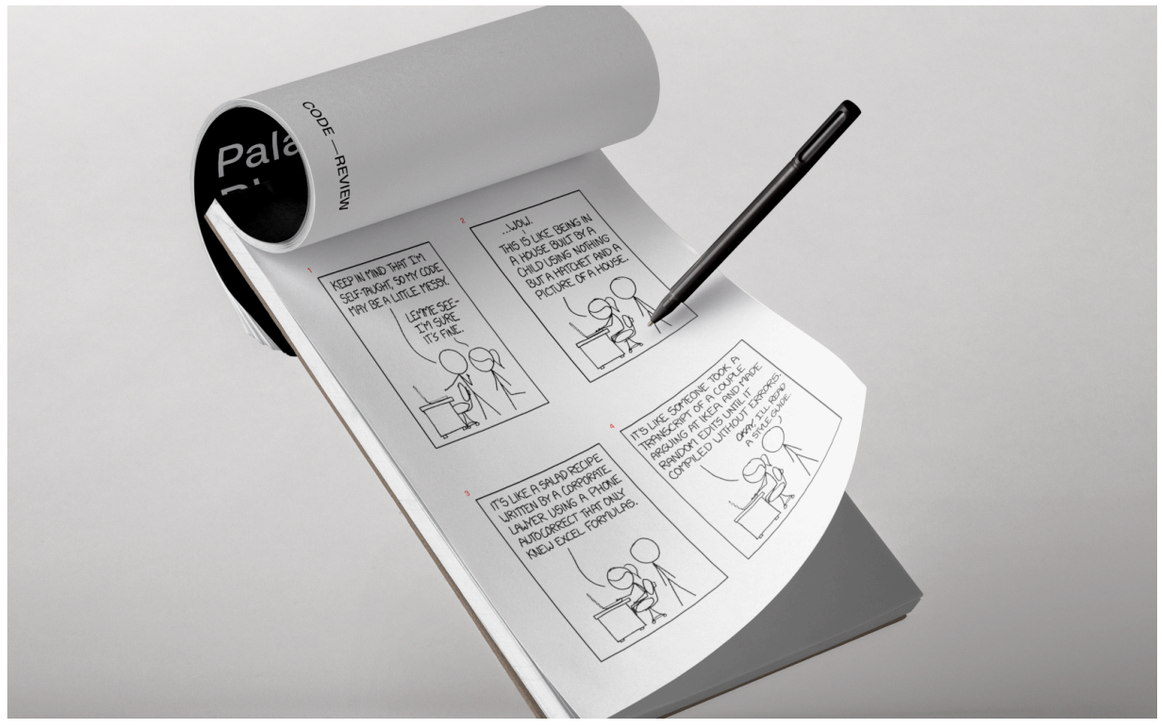
形式

- 在软件开发过程中，对计算机源代码系统化地审查
- 常常是以同行评审（Peer Review）的方式进行



为什么 Review

- 提高代码质量
  - 保持代码风格一致性
  - 保证代码易读性
- 提早发现潜在缺陷与 Bug
  - 意外错误（例如拼写错误） 或 结构错误（例如无效代码、算法错误）
- 促进团队内部知识共享
- 在 Review 的时候重建思路，更好地理解系统



Review 什么

- 最近迭代的代码（其他分支与 Master 分支将要合并时）
- 系统关键模块
- 业务较复杂的模块
- Bug 率较高的模块

什么时候 Review

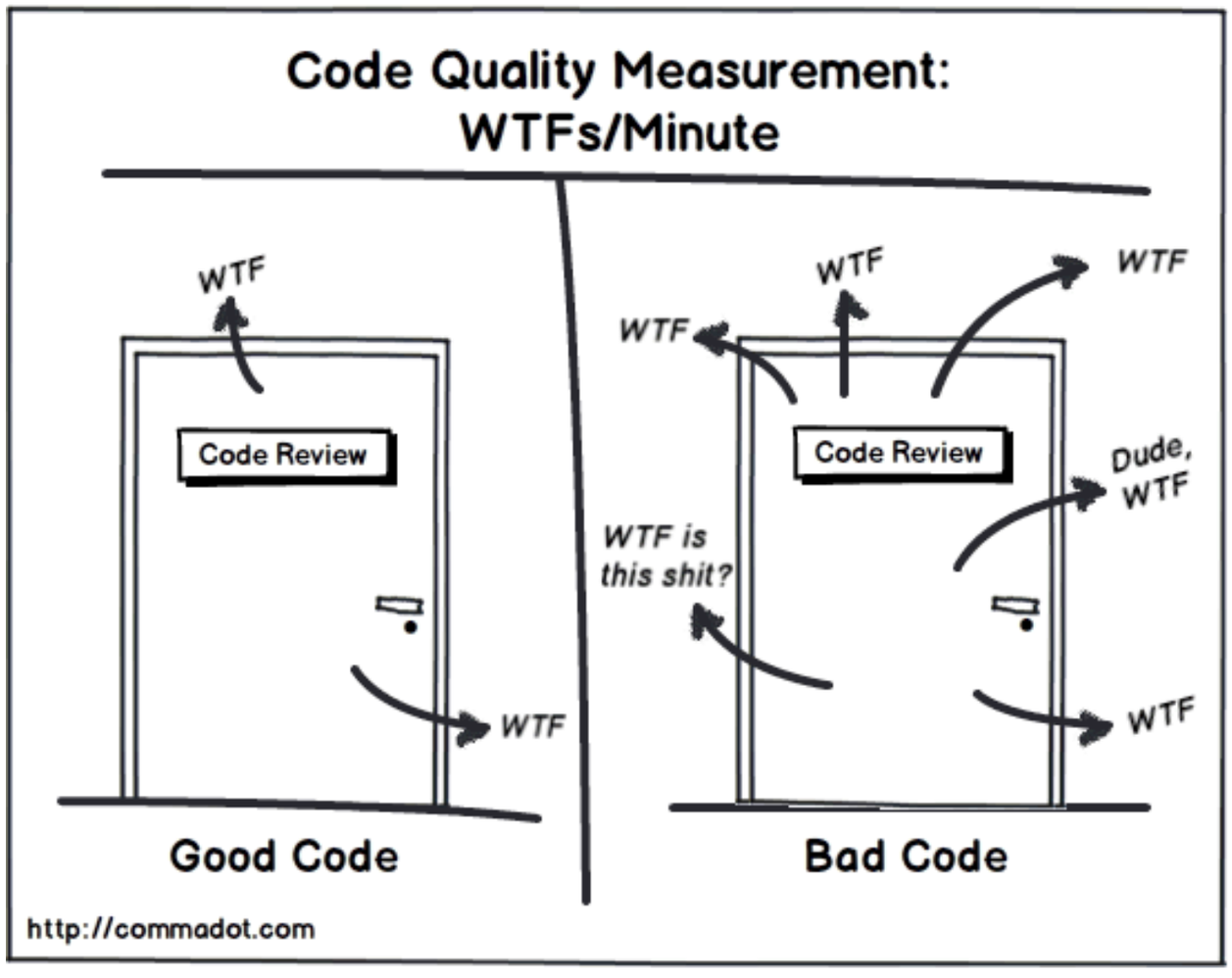
- 晚于测试
- 早于分支合并
- 在正式上线前需要多次 Review
- 细分分支
  - feature/UI-testing
  - feature/UI-js
  - feature/UI-css

2

Code Review 的准备

代码准备

- 确定好 Review 的范围与大小
  - 一个新的 feature
  - 修复了一个 bug
  - 学会分割
- 只提交通过测试，自我审查过的代码



代码提交

- 善用 Commit Message
  - 简单扼要，最好120个字符以内
  - 用祈使句
    - ✅ Fix a bug
    - ✖ Fixed/Fixes a bug
  - 用空行分割段落

```
> BAD. Don't do this.
Make compile again

> Good.
Add jcsv dependency to fix IntelliJ compilation
```


https://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

找到 Reviewer

- 控制在两个 Reviewer 之内
- 最好有一个是一个比较高级别、有经验的工程师

https://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

3

Code Review 的执行

Review 原则

- 意图 (Purpose)
- 代码是否达成了作者的意图?
  - 重构? 修复 Bug? 增加新特性?
- 多问问题
  - 这个函数什么意思?
  - 这个语句为什么这么用?

Review 原则

- 实现 (Implementation)
- 如果是你要解决这个问题，你会怎么做?
  - 如果跟你的想法有差别，为什么?
  - 你可以让这份代码更安全更高效吗?
  - 有没有未被处理的潜在问题?
- 能否把一些重复代码抽象出来?
  - 写成一个函数或类

Review 原则

- 实现 (Implementation)
- 如果是你要解决这个问题，你会怎么做?
- 能否把一些重复代码抽象出来?
- 是否有一些开源代码库可以帮助提升这份代码?
- 这份代码是否遵循了标准?
  - 安全标准
  - 工程标准

Review 原则

- 实现 (Implementation)
- 如果是你要解决这个问题，你会怎么做?
- 能否把一些重复代码抽象出来?
- 是否有一些开源代码库可以帮助提升这份代码?
- 这份代码是否遵循了标准?
- 是否出现了新的依赖?
  - 程序间的依赖越少越好

Review 原则

- 易读性与样式 (Legibility and style)
- 是否能在短时间内读懂代码?
- 是否遵循了代码规范?
- 代码中是否完成? 是否还有TODO?

Review 原则

- 可维护性 (Maintainability)
- 测试是否覆盖了所有的情况?
- 测试是否引入了新的风险?
  - 破坏了代码
  - 破坏了一些临时堆栈

Review 原则

- 可维护性 (Maintainability)
- 测试是否覆盖了所有的情况?
- 测试是否引入了新的风险?
- 这份代码是否破坏了兼容性?
- 代码注释是否清晰?
- 文档是否更新?

Review 原则

- 安全性 (Security)
- 是否符合规定的安全标准?
- 是否检查了所有的薄弱环节?
- 是否考虑到了代码本身的风险?