链滴

提升编码能力，Google 家的工程实践文档，你不看？远程开发必备!

作者：martinageradams

原文链接：https://ld246.com/article/1588958629846

来源网站：链滴

大家好，我是老王。前端时空的共建者之一，欢迎大家关注前端时空，并且来共同建设我们的前端时社区。

提升编码能力，Google 家的工实践文档，你不看？远程开发必备! - 其它 - 工具 - 分享

Google's Engineering Practices documentation。

这份文档是 Google 团队长期以来的内部项目最佳实践。其目的是帮助开发者更好地进行代码审查工，通过 Code Review 来提升并优化当前项目的代码质量，便于开发人员维护和维护旧项目。

在国内，代码审查一直没有没有被重视，一般都是有了问题才去修修补补。但是，构建一个健壮的大软件应用是必须有代码审查环节的。

如果你的团队开始代码审查环节，但没有正确的方法，这份文档就很适合你了。

我摘抄了一些片段。

首先，它包含两个部分

- 代码审查者指南
- 代码开发者指南

代码审查者指南

本节是基于过往经验编写的 Code Review 最佳方式建议。其中分为了很多独立的部分，共同组成完的文档。虽然您不必阅读文档，但通读一遍会对您自己和团队很有帮助。

- Code Review 标准[1]
- Code Review 要点[2]
- 查看 CL 的步骤[3]
- Code Review 速度[4]
- 如何撰写 Code Review 评论[5]
- 处理 Code Review 中的抵触[6]

另请参阅代码开发者指南[7]，该指南为正在进行 Code Review 的开发开发者提供详细指导。

代码审查者应该关注哪些方面?

代码审查时应该关注以下方面：

- **设计**：代码是否经过精心设计并适合您的系统?
- **功能**：代码的行为是否与作者的意图相同？代码是否可以正常响应用户的行为?
- **复杂度**：代码能更简单吗？将来其他开发人员能轻松理解并使用此代码吗?
- **测试**：代码是否具有正确且设计良好的自动化测试?

- **命名**：开发人员是否为变量、类、方法等选择了明确的名称？
- **注释**：评论是否清晰有用？
- **风格**：代码是否遵守了风格指南[8]？
- **文档**：开发人员是否同时更新了相关文档？

代码开发者指南

本节页面的内容为开发人员进行代码审查的最佳实践。这些指南可帮助您更快地完成审核并获得更高量的结果。您不必全部阅读它们，但它们适用于每个 Google 开发人员，并且许阅读全文通常会很有助。

- 写好 CL 描述[9]
- 小型 CL[10]
- 如何处理审查者的评论[11]

另请参阅代码审查者指南[12]，它为代码审阅者提供了详细的指导。

写好 CL 描述

CL 描述是进行了**哪些更改**以及**为何更改**的公开记录。CL 将作为版本控制系统中的永久记录，可能会长时期内被除审查者之外的数百人阅读。

开发者将来会根据描述搜索您的 CL。有人可能会仅凭有关联性的微弱印象，但没有更多具体细节的况下，来查找你的改动。如果所有重要信息都在代码而不是描述中，那么会让他们更加难以找到你的L 。

首行

- 正在做什么的简短摘要。
- 完整的句子，使用祈使句。
- 后面跟一个空行。

CL 描述的**第一行**应该是关于这个 CL 是**做什么**的简短摘要，后面跟一个空白行。这是将来大多数的代搜索者在浏览代码的版本控制历史时，最常被看到的内容，因此第一行应该提供足够的信息，以便他不必阅读 CL 的整个描述就可以获得这个 CL 实际上是做了什么的信息。

按照传统，CL 描述的第一行应该是一个完整的句子，就好像是一个命令（一个命令句）。例如，“**De ete** the FizzBuzz RPC and **replace** it with the new system.”而不是“**Deleting** the FizzBuzz RPC and **replacing** it with the new system.“ 但是，您不必把其余的描述写成祈使句。

Body 是信息丰富的

其余描述应该是提供信息的。可能包括对正在解决的问题的简要描述，以及为什么这是最好的方法。果方法有任何缺点，应该提到它们。如果相关，请包括背景信息，例如错误编号，基准测试结果以及计文档的链接。

即使是小型 CL 也需要注意细节。在 CL 描述中提供上下文以供参照。

糟糕的 CL 描述

“Fix bug ” 是一个不充分的 CL 描述。什么 bug？你做了什么修复？其他类似的不良描述包括：

- “Fix build.”
- “Add patch.”
- “Moving code from A to B.”
- “Phase 1.”
- “Add convenience functions.”
- “kill weird URLs.”

其中一些是真正的 CL 描述。他们的作者可能认为自己提供了有用的信息，却没有达到 CL 描述的目
。

英文不太好的同学不用担心的，国内的开发者早已将这份指南翻译成了中文。

下面是该项目的完整链接，感兴趣的小伙伴可以看一下。

- **中文版：https://jimmysong.io/eng-practices**
- **英文版：https://google.github.io/eng-practices**
- **GitHub 地址：https://github.com/google/eng-practices**

以上是老王今天的分享，希望大家喜欢。

觉得内容不错的，欢迎点击「**点赞、留言、关注**」三连支持，谢谢各位。

参考资料

[1] Code Review 标准: https://jimmysong.io/eng-practices/docs/review/reviewer/standard

[2] Code Review 要点: https://jimmysong.io/eng-practices/docs/review/reviewer/looking-for

[3] 查看 CL 的步骤: https://jimmysong.io/eng-practices/docs/review/reviewer/navigate

[4] Code Review 速度: https://jimmysong.io/eng-practices/docs/review/reviewer/speed

[5] 如何撰写 Code Review 评论: https://jimmysong.io/eng-practices/docs/review/reviewer/commen
s

[6] 处理 Code Review 中的抵触: https://jimmysong.io/eng-practices/docs/review/reviewer/pushback

[7] 代码开发者指南: https://jimmysong.io/eng-practices/docs/review/developer/

[8] 风格指南: https://google.github.io/styleguide/

[9] 写好 CL 描述: https://jimmysong.io/eng-practices/docs/review/developer/cl-descriptions

[10] 小型 CL: https://jimmysong.io/eng-practices/docs/review/developer/small-cls

[11] 如何处理审查者的评论: https://jimmysong.io/eng-practices/docs/review/developer/handling-c
mments

[12] 代码审查者指南: https://jimmysong.io/eng-practices/docs/review/reviewer/

本文使用 mdnice 排版