Code Review Best Practice

**TABLE OF CONTENTS**

[Foreword 1](#_Toc20965572)

[Thought Bot's Code Review Style Guide 1](#_Toc1389449830)

[Best Practice for Everyone 1](#_Toc1625007961)

[Best Practice While Having Your Code Reviewed 2](#_Toc1712769200)

[Best Practice While Reviewing Code 2](#_Toc732899865)

[Style Comments 3](#_Toc837242296)

[Further Reading 3](#_Toc2119604221)

# Foreword

This best practice is from [*https://github.com/thoughtbot/guides/tree/main/code-review*](https://github.com/thoughtbot/guides/tree/main/code-review)

It is perhaps the best guide to doing and receiving code reviews I've ever seen

Watch a presentation that covers this material from Derek Prior at RailsConf 2015.

<https://www.youtube.com/watch?v=PJjmw9TRB7s>

# Thought Bot's Code Review Style Guide

## Best Practice for Everyone

* Accept that many programming decisions are opinions. Discuss tradeoffs, which  
  you prefer, and reach a resolution quickly.
* Ask good questions; don't make demands. ("What do you think about naming this  
  :user\_id?")
* Good questions avoid judgment and avoid assumptions about the author's  
  perspective.
* Ask for clarification. ("I didn't understand. Can you clarify?")
* Avoid selective ownership of code. ("mine", "not mine", "yours")
* Avoid using terms that could be seen as referring to personal traits. ("dumb",  
  "stupid"). Assume everyone is intelligent and well-meaning.
* Be explicit. Remember people don't always understand your intentions online.
* Be humble. ("I'm not sure - let's look it up.")
* Don't use hyperbole. ("always", "never", "endlessly", "nothing")
* Don't use sarcasm.
* Keep it real. If emoji, animated gifs, or humor aren't you, don't force them.  
  If they are, use them with aplomb.
* Talk synchronously (e.g. chat, screen-sharing, in person) if there are too  
  many "I didn't understand" or "Alternative solution:" comments. Post a  
  follow-up comment summarizing the discussion.

## Best Practice While Having Your Code Reviewed

* Be grateful for the reviewer's suggestions. ("Good call. I'll make that  
  change.")
* Be aware that it can be [challenging to convey emotion and intention online](https://thoughtbot.com/blog/empathy-online)
* Explain why the code exists. ("It's like that because of these reasons. Would  
  it be more clear if I rename this class/file/method/variable?")
* Extract some changes and refactoring into future tickets/stories.
* Link to the code review from the ticket/story. ("Ready for review:  
  <https://github.com/organization/project/pull/1> ")
* Push commits based on earlier rounds of feedback as isolated commits to the  
  branch. Do not squash until the branch is ready to merge. Reviewers should be  
  able to read individual updates based on their earlier feedback.
* Seek to understand the reviewer's perspective.
* Try to respond to every comment.
* Wait to merge the branch until continuous integration (TDDium, Travis CI,  
  CircleCI, etc.) tells you the test suite is green in the branch.
* Merge once you feel confident in the code and its impact on the project.
* Final editorial control rests with the pull request author.

## Best Practice While Reviewing Code

* Understand why the change is necessary (fixes a bug, improves the user  
  experience, refactors the existing code). Then:
* Communicate which ideas you feel strongly about and those you don't.
* Identify ways to simplify the code while still solving the problem.
* If discussions turn too philosophical or academic, move the discussion offline  
  to a regular Friday afternoon technique discussion. In the meantime, let the  
  author make the final decision on alternative implementations.
* Offer alternative implementations, but assume the author already considered  
  them. ("What do you think about using a custom validator here?")
* Seek to understand the author's perspective.
* Sign off on the pull request with a 👍 or "Ready to merge" comment.
* Remember that you are here to provide feedback, not to be a gatekeeper.

## Style Comments

* Reviewers should comment on missed style guidelines. Example comment:
* > Order resourceful routes alphabetically by name.
* An example response to style comments:
* Whoops. Good catch, thanks. Fixed in a4994ec.
* If you disagree with a guideline, open an issue on the guides repo rather than  
  debating it within the code review. In the meantime, apply the guideline.

# Further Reading

* <https://github.com/thoughtbot/guides/tree/main/code-review>
* <https://thoughtbot.com/blog/empathy-online>
* <https://www.youtube.com/watch?v=PJjmw9TRB7s>