Google

*Testing on the Toilet Presents...* ***Healthy Code on the Commode***

# Respectful Reviews == Useful Reviews

While code review is recognized as a valuable tool for improving the quality of software projects, **code review comments that are perceived as being unclear or harsh can have unfavorable consequences**: slow reviews, blocked dependent code reviews, negative emotions, or negative perceptions of other contributors or colleagues.

**Consider these tips to resolve code review comments respectfully.**

### As a Reviewer or Author:

- **DO: Assume competence.** An author's implementation or a reviewer's recommendation may be due to the other party having different context than you. Start by asking questions to gain understanding.
- **DO: Provide rationale or context,** such as a best practices document, a style guide, or a design document. This can help others understand your decision or provide mentorship.
- **DO: Consider how comments may be interpreted.** Be mindful of the differing ways hyperbole, jokes, and emojis may be perceived.

| **Author Don't:** | **Author Do:** |
|---|---|
| I prefer short names so I'd rather not change this. Unless you make me? :) | Best practice suggests omitting obvious/generic terms. I'm not sure how to reconcile that advice with this request. |

- **DON'T: Criticize the person**. Instead, discuss the *code*. Even the perception that a comment is about a person (e.g., due to using "you" or "your") distracts from the goal of improving the code.

| **Reviewer Don't:** | **Reviewer Do:** |
|---|---|
| Why are you using this approach? You're adding unnecessary complexity. | This concurrency model appears to be adding complexity to the system without any visible performance benefit. |

- **DON'T: Use harsh language**. Code review comments with a negative tone are less likely to be useful. For example, prior research found very negative comments were considered useful by authors 57% of the time, while more-neutral comments were useful 79% of the time.

### As a Reviewer:

- **DO: Provide specific and actionable feedback.** If you don't have specific advice, sometimes it's helpful to ask for clarification on why the author made a decision.

| **Reviewer Don't:** | **Reviewer Do:** |
|---|---|
| I don't understand this. | If this is an optimization, can you please add comments? |

- **DO: Clearly mark nitpicks and optional comments** by using prefixes such as 'Nit' or 'Optional'. This allows the author to better gauge the reviewer's expectations.

### As an Author:

- **DO: Clarify code or reply to the reviewer's comment** in response to feedback. Failing to do so can signal a lack of receptiveness to implementing improvements to the code.

| **Author Don't:** | **Author Do:** |
|---|---|
| That makes sense in some cases but not here. | I added a comment about why it's implemented that way. |

- **DO: When disagreeing with feedback, explain the advantage of your approach**. In cases where you can't reach consensus, follow Google's guidance for resolving conflicts in code review.

**More information, discussion, and archives:**

**testing.googleblog.com**