# Pull request responsibilities shared between authors and reviewers

Childhood Cancer Data Lab x Alex's Lemonade Stand

## Objectives

- Cover some common complaints around code review
- Characterize the pull request responsibilities that authors and reviewers share
- Explain how these responsibilities can help alleviate common pain points

## Common complaints about code review

- **It takes too long**, and long-lived branches result in merge conflicts.
- **It's no fun.** It taxes both the authors and the reviewers.
- **No clear benefit** because reviews are too long and are rushed.

Source: Wikimedia Commons

And your team leader and the rest of your team, with systems and processes, can prevent these complaints

https://blog.arkency.com/disadvantages-of-pull-requests/

## Some questions you should ask yourself

- **Who is going to review this?** A good code review will be confident and requires some understanding about the project and problem at hand.[1] But it can also be an opportunity to onboard people to the project. Maybe a pull request would benefit from multiple reviewers playing different roles!
- **What are the stakes?** A quick, one-off analysis for internal purposes needs to be correct, but it doesn't necessarily need to be extensible or efficient.
- **When does this need to be completed?** If you have a code review policy, you should factor in that code review takes time, and something isn't completed until it's merged
- **What is this blocking?** Related to the *when* question – how should review be prioritized?

[1] https://slack.engineering/how-about-code-reviews/

# We ask these questions in service of our goal

Generally, leveraging the benefits of code review, such as

- Raising quality
- Sharing knowledge

While making good, sustainable progress

- Working on the right things
- Not grinding to a halt
- Not overburdening our team

## Shared responsibilities

## Does it accomplish what it needs to accomplish?

Both the author and the reviewer need to understand the purpose of the pull request and who the intended audience of the output is.

Addresses: **Clear benefit**

## Does it meet the required standards?

The author’s job is to put forth something they believe is valid (correct, reproducible, maybe reusable and efficient), and the reviewer’s job is to confirm the work’s validity.

Addresses: **Clear benefit**

## Are the right people reviewing it?

An author should request a reviewer with enough understanding of the problem and project that can verify the work’s validity.

Reviewers should be open and honest about the potential limitations of their reviews.

Addresses: **Clear benefit**

## Don’t be a jerk

Hopefully, this goes without saying. Scientific, collaborative work requires team psychological safety.

Sharing your knowledge with your teammates, either as an author or as a reviewer, can be fun!

Addresses: **No fun**

## Instead...

- Assume the other person on your team is doing a good job! Maybe they know something you don’t.
- Get curious and ask for clarifications.
- Use wording like “us”, “our”, “we” rather than “you”, “your”, and “mine.”
- Remember to point out what you liked or learned from, too.

From https://slack.engineering/on-empathy-pull-requests/

## Keep things moving

Yes, there are consequences to things hanging out in review too long. Some sense of urgency is good.

Does everything need to be addressed now, or can you follow up with new issues and subsequent pull requests instead?

Addresses: **Takes too long**

## Summary

- Code review is a collaborative effort and care must be taken to ensure that it’s a net positive.
- Pull requests don’t happen in a vacuum, so it’s helpful to consider the “bigger picture” like what happens when a PR get merged both as an author and a reviewer.
- A team’s code review practice can benefit from some guidelines. What are some guidelines you would write?