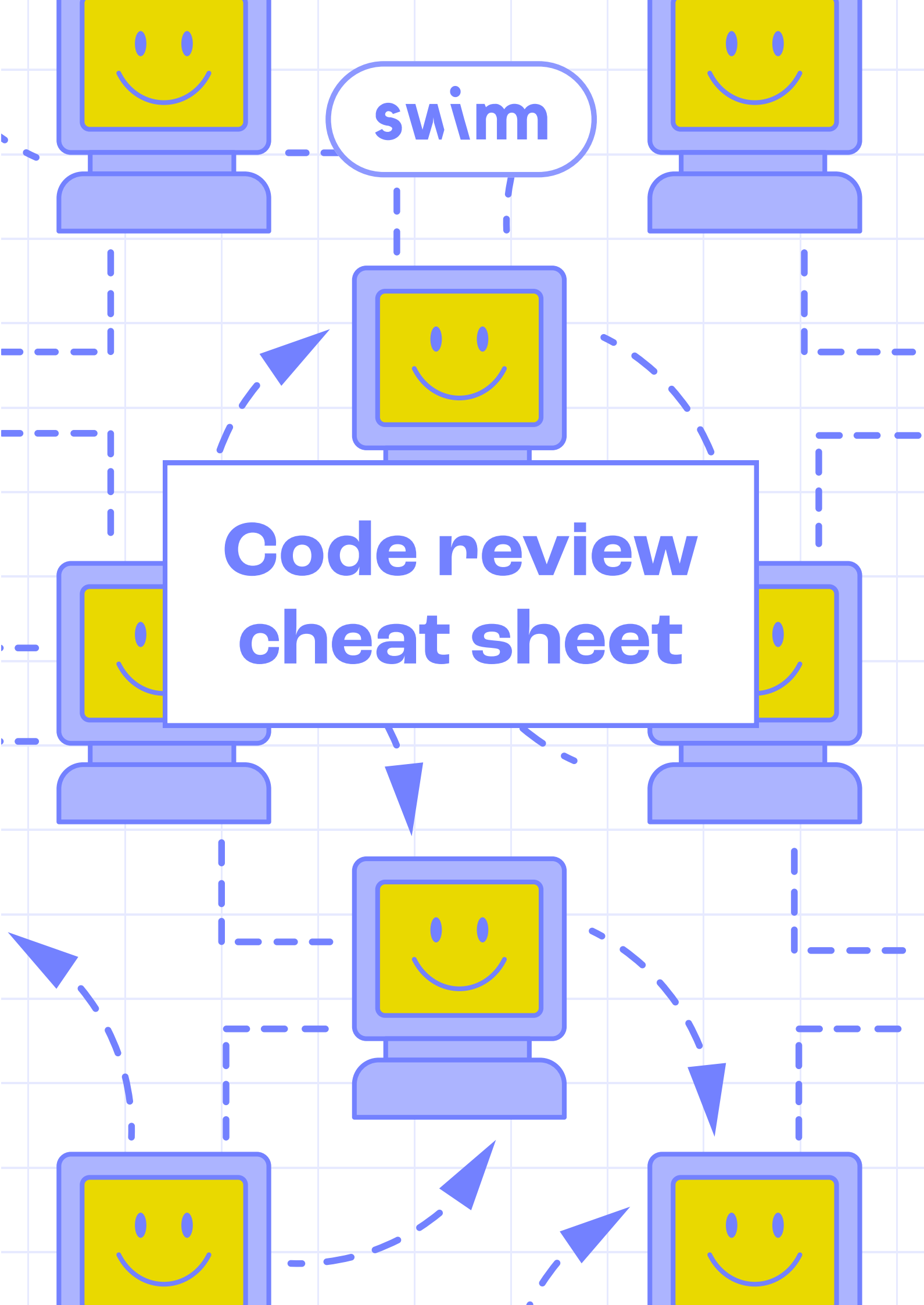

swim
Code review
cheat sheet

Table of Contents

01 Keep pull requests small 3

02 Create a code review checklist 4

03 Introduce code review metrics 5

04 Foster a positive feedback culture 6

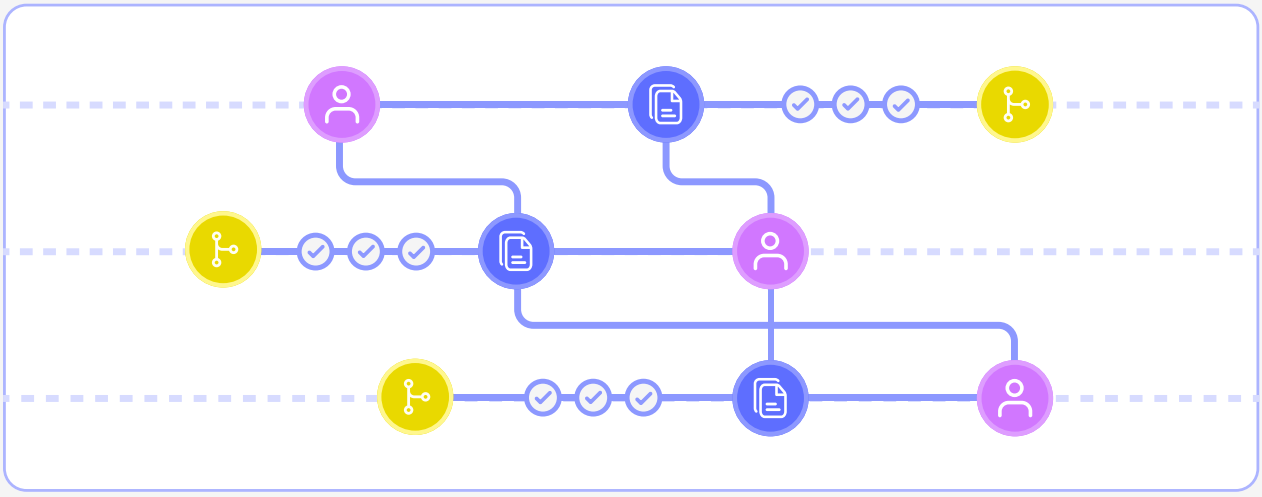
05 Documentation is always helpful 7

06 Conclusion 8

Introduction

In our somewhat humble, yet unsurprising opinion, code documentation is incredibly important. The need to understand code happens all the time – whether onboarding to a new team or working on an area of the codebase that developers are unfamiliar with. Without proper documentation, we offer a sincere and heartfelt, “good luck!”.

> **But you know what else is hard? Code review. Does documentation make it easier to establish rules and procedures, for both the reviewer and reviewee?**



Absolutely, yet the fact remains that code reviews are essential processes that no dev pipeline can go without.

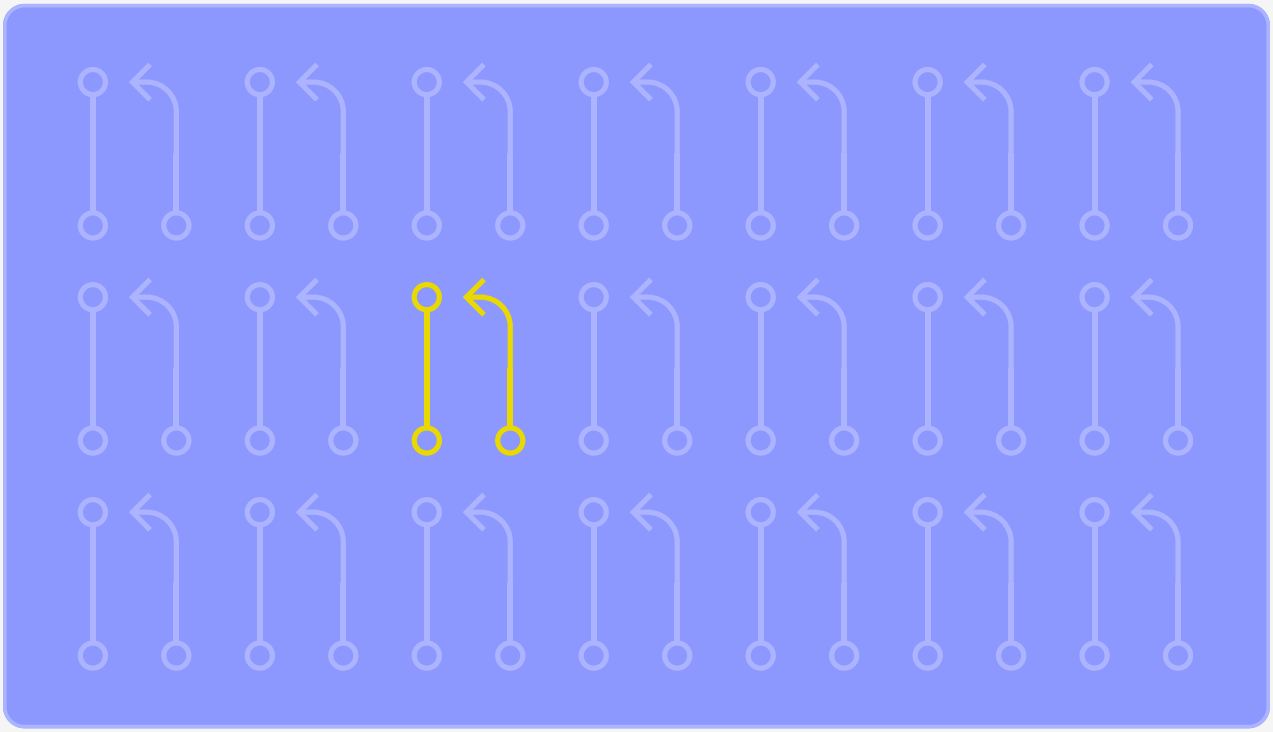
Now, what might best practices for code review entail? We sat down with Swimm’s CTO, Omer Rosenbaum to answer that very question, and came up with these 5 tips:

Best practices for code reviews:

01

Keep pull requests small

Smaller deployments are easier to design, test, review, and merge. A common rule of thumb is that 100 lines of code is a reasonable size for a pull request, while 1000 lines is too big. The number of files a distribution changes also affects its perceived size - for example, 100 lines of change in one file is different from the same number of lines spread across multiple different files.



Breaking large changes into smaller chunks is almost always possible and makes many aspects of the development process easier, code reviews included. With more practice, reviewers and reviewees can become proficient at finding the smallest possible increment of a product. Note that feature gates or feature flags may be required to release unfinished product features alongside existing features.

02

Create a code review checklist

Conceptually, a code review checklist is a set of predefined questions and rules that reviewers and reviewees should follow during the code review process. Code review checklist can help all involved take a structured approach to the quality checks they need before submitting or approving code in their codebase. Checklists may include:

- **Readability:**
  Identifying redundant or unclear comments in the code
- **Security:**
  Identifying weaknesses that expose the code to cyberattacks
- **Test coverage:**
  Identifying the need for more test cases
- **Architecture:**
  Identifying issues like encapsulation and modularity of the code
- **Reusability:**
  Identifying whether the code properly reuses components, functions, and services

03

Introduce code review metrics

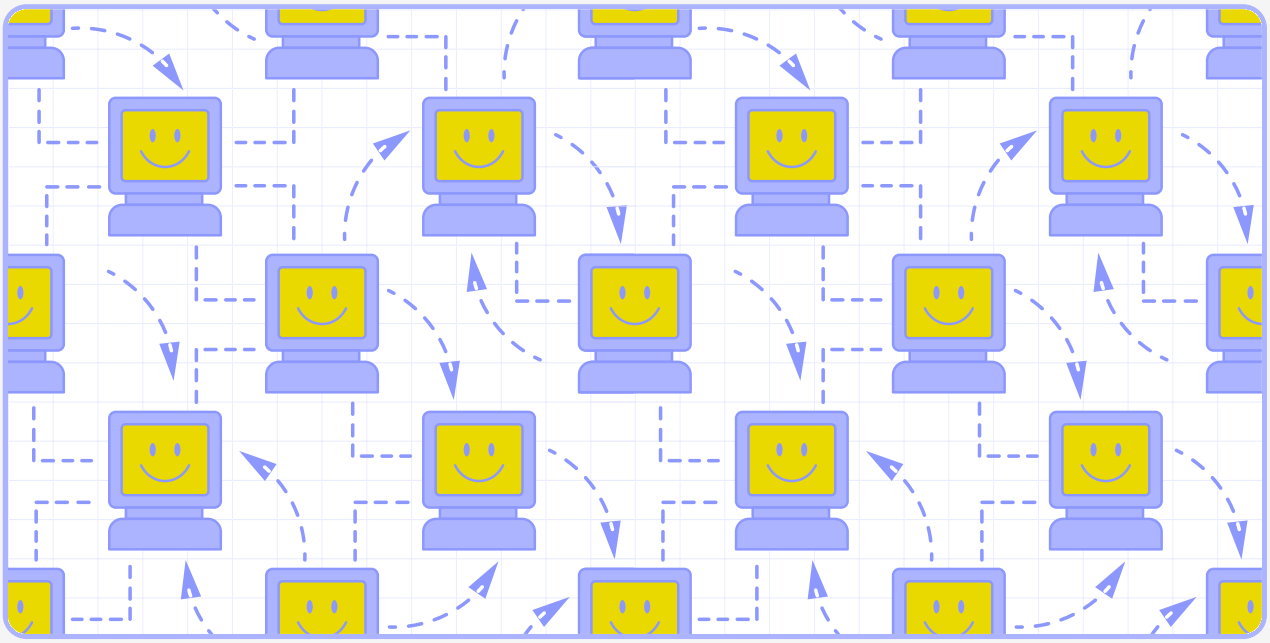
Reviewers cannot modify someone else's code without measuring its quality. Objective metrics help determine the effectiveness of reviews, analyze the impact of changes on processes, and predict how long it will take to complete a project. Commonly used metrics include:

- **Inspection rate:**
  The rate at which a team reviews a given amount of code, which is lines of code (LoC) divided by total review time. If a code review takes a long time, you may need to address readability issues in the codebase.
- **Defect rate:**
  How often defects are identified. Calculated by dividing the number of defects by the time spent reviewing them. This metric helps determine the effectiveness of the testing process. For example, if developers are slow at finding defects, they may need better testing tools.
- **Defect density:**
  The number of defects identified in a given amount of code. It is calculated by dividing the number of defects by thousands of lines of code (kLOC). Fault density helps identify which components are more prone to failure than others, so more resources can be allocated to the weaker component.

04

Foster a positive feedback culture

Effective communication is difficult to master. For reviewers, giving feedback on a colleague's work is challenging, both intellectually and emotionally. Below is a list of suggestions for improving code review discussions:



- Always provide feedback on the code, not the author.
- Focus on the most important problems within the code, not all the problems.
- Admit that there are multiple correct solutions to every problem.
- Give objective reasons, not emotions, for a problem in the code.
- Always try to provide positive feedback. This can make it easier for the code author to accept negative feedback, and can be valuable in itself by reinforcing best practices.

05

Documentation is always helpful

Obviously, if you're reviewing code, you won't be actively documenting. However, code reviews are commonly known to be bottlenecks within the development process. So, when reviewers have an understanding of the code's intention, complete with diagrams and methodologies, the process can be sped up dramatically - and is best presented via documentation. And on that note, the best documentation includes the following:

- Notes on what was being worked on, when, and by who
- Context that explains why it was written in the way it was written
- Any relevant diagrams or visual aids
- Full text explanations of how the code was written

06

Conclusion

Code reviews are incredibly important when it comes to maintaining a working product or platform. Thanks to the best practices outlined above, reviewers and reviewees will find their processes more efficient than ever. That said, no best practice can be a replacement for solid documentation.

But, even if you have a documentation process in place, can you honestly say that it isn't a hassle? Furthermore, does it distract reviewers and reviewees from doing the one thing they need to do - code? If you said yes to any of that, then that's where Swimm comes in.

Now, dev teams can improve their workflows with a platform that can generate and store in-context code documentation, which in turn streamlines collaboration, and empowers teams to work more efficiently and productively.

---

swimm

Intrigued and want to learn more about Swimm?

Sign up for a community demo today.

Sign up