 By:Mike 17/6/2021

# **How to review someone else’s code: Tips and best practices**

Imagine you have a co-worker or friend who’s asked you to review a piece of code for them. You might be thinking, “Wait, I don’t really know what that means.” Maybe you’re too nervous to say that to your colleague. Not to worry — we’ve all been there.

While your first code review might seem like yet another test of your abilities, it’s actually just another set of eyes on a piece of code to help spot any problems before it gets launched. Even if you’re a new developer or still learning code, you can help people out by reviewing their code.

In the following article, we’ll help to demystify what it means to review code.

**What is a code review?**

A code review is a collaborative process where other members of your development team look over your code to give you:

* Critiques
* Potential improvements
* A list of outstanding questions

In practice, this means that your fellow developers will either add comments within your code or send you a list of questions to answer. They could also sit down with you at your computer and talk through a few suggestions on how to make your code better. It’s a good idea to ask someone who will eventually use the code to help you review it.

**Why is it a good idea to review code?**

Reviewing code helps maintain best practices for the code and keep a certain standard across the code base. Elements of the code, including styling, formatting, design patterns, and naming conventions, should be kept consistent. Having someone who works with the same standards review your code will help keep it formatted correctly and easy to use for other team members.

Code review is also a great way to introduce other team members to the new piece of code going into the code base. You can talk about it and get them up to speed quickly.

Additionally, it helps to minimize or eliminate the “bus factor,” which Mike, the developer from Philadelphia who talks about code review in the video above, describes:

*“The bus factor refers to the problem of having one team member with too much knowledge. The problem is if that team member then got hit by a bus, then no one on the team would know about the code that’s going into the code base.”*

Lastly, a code review is a good idea because it facilitates discussion. When teams are busy or working remotely, communication doesn’t happen as frequently. Scheduling code reviews is a great way to get teammates talking, strategizing, and learning from one another.

**Where do code reviews happen?**

Depending on your relationship with the person asking you to review their code, this could happen in several locations. For example, a fellow learner or programmer friend of yours could simply email you a code file and ask for your comments. In that case, you would put your comments in the code file and send back any questions you have in an email response.

Alternatively, a code review can happen online with a community of programmers. You can help review someone else’s code on websites like:

* Stash
* Bitbucket
* GitHub
* Codecademy Forums

If you take part in a code review on some of these repositories (like Stash or Github), it’s done with a pull request. The original programmer will add you as a reviewer, and then the website will track all changes that you make when you’re completing your code review. You could also be conducting a code review internally at work, either in person or using a version control system that your company has implemented.

**5 easy steps to a successful code review**

To help you get started with reviewing code, we’ve got a simple five-step process to share. Let’s get into code review best practices.

**1. What is the code doing?**

To start, we need to determine why the code was written in the first place. For example, if you’re conducting a code review at work, the purpose of the code could be tied to a change in requirements. It’s key to know the purpose of the code because that context will allow you to deliver the best possible code review with helpful comments.

To make sure that you have a good understanding of what the code is for, here are three questions you can ask yourself (or your fellow programmer):

* Do you have the right background info?
* Do you fully understand why this code is needed?
* Do you understand how this code fits into the project?

**2. Frame your mindset**

Before you get started reviewing the code, you want to get into the right mindset. In a sense, you want to know how you fit into this scenario. What is your level of familiarity with the code? How confident are you in knowing the person who’s written the code and what they are trying to accomplish? This will give you an idea of what feedback you can confidently give to this author.

This step might sound a little vague, so we’ve got a list of questions to help you set up your mindset for the code review:

* How experienced are you with the codebase or language?
* Are you going to be using this code in the future?
* Have you written code similar to this?
* What is the experience level of the author?
* What is your relationship with the author?

**3. Test the code**

Now that we know what the code is for, we can go ahead and test it out. We’re looking to see if the code does what it’s supposed to do. During this step, you can also test out the user interface. For example, if you’re reviewing a website, you could check to make sure that it appears as it should, that the changes are incorporated, and that nothing breaks as you use it.

If you need a couple of questions to help you provide feedback on code function, you can start here:

* Does the code generate warnings or errors?
* Does the code take longer than normal to build or run?
* Do the changes require any additional setup steps?

**4. Inspect the code**

After figuring out if the code *actually* works, we can dive into the code itself and make sure it’s written well. Here, we’re getting into the details of code review best practices. You can offer up advice on how things might be better organized or formatted. For example, you could ask yourself:

* Are correct naming conventions followed?
* Does the code follow DRY principles (i.e., it doesn’t repeat itself)?
* Are new files and folders named appropriately?
* Does the code have error handling?
* Is the code written in a style that's similar to any other code within the same project?

**5. Compile your review**

Just as the code review can take place in several locations, you can also format your code review in a variety of ways. Sometimes you might need to prepare a formal report of your findings and then submit that as a Word document or PDF.

In other situations, if you are using a website like GitHub, you may just add comments to the code. If you have longer questions or feedback, you can add them to a comment thread and allow the conversation to take place all online.

Keep in mind a few tips when you are compiling your code review:

* Beware of nitpicking
* Be respectful
* Be specific and descriptive
* Be open to follow-ups

**Growing as a developer through code review**

One of the great things about [becoming a developer](https://www.codecademy.com/learn/paths/full-stack-engineer-career-path?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=code_review_blog) or learning some aspects of programming for yourself is that you get to learn and work with other people. Code reviews are a common practice, and the discussion they generate will help make you a better programmer, whether it’s your code being reviewed or someone else's code.

“Coding is a team sport,” says Mike. Learning to code should be fun, and learning how to give great code reviews is part of the process. If you’re thinking of becoming a [front-end engineer](https://www.codecademy.com/learn/paths/front-end-engineer-career-path?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=code_review_blog) or a [back-end engineer](https://www.codecademy.com/learn/paths/back-end-engineer-career-path?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=code_review_blog), code reviews are a great way to practice for your new job.