Code Reviews

They're important

Most Important

As a developer, you’ll be responsible for making both *workable* and *maintainable* code.

It can be easy to defer the latter because of a pressure to deliver something on time, but don’t give into the pressure!

Code Reviews Will Make You Better

Reviewing other’s code will expose you to a new way of thinking, and help you discover different ways to solve problems.

Having your code reviewed will help you learn to tighten it, keep it clean, and be more efficient. It will help you become a better programmer.

Remember What Bill Said

Interviewer: Is studying computer science the best way to prepare to be a programmer?
Bill Gates: No. the best way to prepare is to write programs, and to study great programs that other people have written. [..] *You got to be willing to read other people's code*, then write your own, then *have other people review your code*. You've got to want to be in this incredible feedback loop where you get the world-class people to tell you what you're doing wrong.

Bill Gates cited in: "Programmers at Work: Interviews With 19 Programmers Who Shaped the Computer Industry", Tempus, by Susan Lammers (Editor)

You got to be willing to read other people's code, then write your own, then *have other people review your code*.

- Bill Gates

Be Humble, Be Gracious, Be Open

It can be hard to not be defensive about your code. Trust me, I know.

Keep an open mind with both reviewing and being reviewed – it’s not personal. **Don’t make it be** personal.

What Are You Looking For

The slides that follow are a rough outline of what you should be looking for when you review someone’s code.

It’s not a end-all, be-all, and different organizations will want you to check different things.

This is a general list. It doesn’t cover everything.

Single Responsibility Code

Classes and Functions should do one thing, and they should do that thing well.

If you have to use “and” in a description of your class or function, that could be a sign you need to break it up.

Reduce Duplication

If you find the same chunk of code twice, that might be okay. If you find the same chunk of code three times? That’s a sign that it should be refactored so the common code is a helper function.

Make It Better Than Before

When a segment of code is being modified, if there’s a chance for improvement, it should improved! It can be tempting to “leave it for later,” but you should never give into that temptation.

Guard for Bugs

Could there be a bug hiding in the code?

Check for common things, like off-by-one errors, looping issues (infinite, anyone?), reversed boolean operators – common mistakes you might make frequently.

Error Handling

If something could go wrong, does the code handle it correctly? Are there proper handlers? Have custom errors been used correctly?

Efficiency is Queen

Is the code using the most efficient method?

Is it doing something that could be done with less memory calls?

Is it doing something that could be done with less iterations?

Comments Bring Clarity

Code that does something complex or unusual should have comments explaining what is happening.

An essay is overkill, but nothing is even worse. Make sure that if code can’t be understood on its own, it is commented.

If you have to spend more than 30 seconds figuring out what something does, or why it’s doing it – a comment is needed.

Names Have Meaning

Understanding what a method or variable is doing should come just from the name. Unless you’re concerned about names that are too long, there’s no reason to skimp on a descriptive name.

Syntax, Style, & Formatting

Different companies will have different style and formatting guidelines, and code should conform to those guidelines.

You can use automated tools to help you adhere to those guidelines, but be on the lookout for things that may have snuck through.

Review Yourself, Before You Check Yourself (in)

Review your own comment first, the way you would review someone else’s code.

You won’t catch everything, but you’ll catch the really embarrassing stuff.

Helpful Articles

http://kevinlondon.com/2015/05/05/code-review-best-practices.html

http://www.future-processing.pl/blog/another-code-review-best-practices/

http://smartbear.com/SmartBear/media/pdfs/11_Best_Practices_for_Peer_Code_Review.pdf

https://msdn.microsoft.com/en-us/library/Bb871031.aspx