Code Quality: Code Reviews

> “Rigorous inspection can remove up to 90 percent of errors before the first test case is run.”
> -Robert Glass

- The person reviewing the code should be an active developer on the code base.
    - Don’t call in people who aren’t writing code on that project to review the code. (Priesthood-based reviews)

How code review is done in effective teams? 540

How?

- Tactical code reviews → where one person writes the code and soon after, another person reviews that code.
- If the teams *believes* in code reviews, they’ll do it.
    - If the teams is forced to do it, they’ll just sign off on the reviews saying that everything looks good, just to get it done.

When?

- Should be done as soon as a task is completed.
- Quality of code review is inversely proportional to the amount of code to be reviewed.

What?

- Review both code and the test.
- Focus on:
    - is the code understandable?

    - is the code readable?
    - are the method/variable names good?
    - are the methods small?
    - do the methods follow good design principles?
    - is the single responsibility principle being violated?
    - is code being duplicated?
- Make sure your code review is motivating and encouraging.
- Give constructive criticism — give positive feedback. Don’t make fun of someone or their code; that is discouraging.

> “Reviews are both technical and sociological, and both factors must be accommodated.”
> -Robert Glass

A good code review includes…

- What you like in the code.
    - Be positive and objective. Encourage the coder to do more of the good that they have done until now.
    - Don’t just tell what’s messed up in the code.
- What needs to be improved. (NOT what sucks, or what’s done bad)
    - Suggest improvements. “Poor variable name” isn’t helpful — suggest improvements/better variable names.
- Questions/suggestions
    - If something isn’t clear — ask.
    - Point them to articles/resources.