Mastering Commit Messages & Pull Request Reviews

A practical guide to better collaboration through clear communication

Clear communication is the backbone of successful software development. Two critical touchpoints where this matters most are commit messages and pull request reviews. Let's explore how to excel at both.

Part 1: Writing Effective Commit Messages

A well-written commit message is a gift to your future self and your teammates. It explains not just what changed, but why it changed.

The Anatomy of a Great Commit Message

Follow this simple structure for clarity:

```
<type>: <subject>

<body>

<footer>
```

Examples:

```
Good:
feat: Add user authentication with JWT tokens

Implemented secure token-based authentication to replace
session-based auth.
This reduces server memory usage and enables better scalability.

Closes #142

Poor:
Fixed stuff
Updated code
WIP
```

Quick Tips for Better Commits

- **Use the imperative mood:** "Add feature" not "Added feature"
- **Keep the subject line under 50 characters**
- **Capitalize the subject line**
- **Don't end with a period**
- **Separate subject from body with a blank line**

• **Explain what and why, not how**

> ■ *Pro tip: If you're struggling to write a concise commit message, your commit might be doing too much. Consider breaking it into smaller, focused commits.*

Part 2: Conducting Effective Pull Request Reviews

Code review is more than finding bugs—it's about knowledge sharing, maintaining consistency, and improving the overall codebase quality.

The Review Checklist

When reviewing a PR, consider these aspects:

• **Functionality:** Does the code do what it's supposed to do?
• **Design:** Is the approach well-structured and maintainable?
• **Performance:** Are there any obvious bottlenecks?
• **Security:** Are there potential vulnerabilities?
• **Testing:** Is the code adequately tested?
• **Documentation:** Are complex parts well-documented?
• **Style:** Does it follow team conventions?

How to Give Constructive Feedback

The goal is to improve the code while respecting the author. Here's how:

1. Be specific and actionable
Instead of: "This is confusing"
Try: "Consider renaming 'process' to 'validateUserInput' to clarify its purpose"

2. Explain your reasoning
"This could cause a memory leak because the event listener isn't removed when the component unmounts."

3. Distinguish between suggestions and requirements
"Must fix: SQL injection vulnerability in line 42"
"Consider: Extracting this into a helper function for reusability"

4. Acknowledge good work
"Great test coverage!" or "Nice abstraction here"

Review Etiquette

• Review promptly—aim for same-day or next-day
• Focus on the code, not the person
• Ask questions when you don't understand
• Pick your battles—not everything needs to be perfect
• Use conventional comment prefixes: MUST, SHOULD, CONSIDER, NIT

Key Takeaways

Great commit messages and thoughtful PR reviews are investments in your team's future productivity. They create a culture of clear communication, shared ownership, and continuous learning. Start small—pick one practice from this guide and implement it in your next commit or review. Your future self and teammates will thank you.

Created on December 02, 2025