Stop Writing Bad Commit Messages

Start following best practices for Git commit messages

Devin Soni 👑
Jan 8 · 4 min read ★

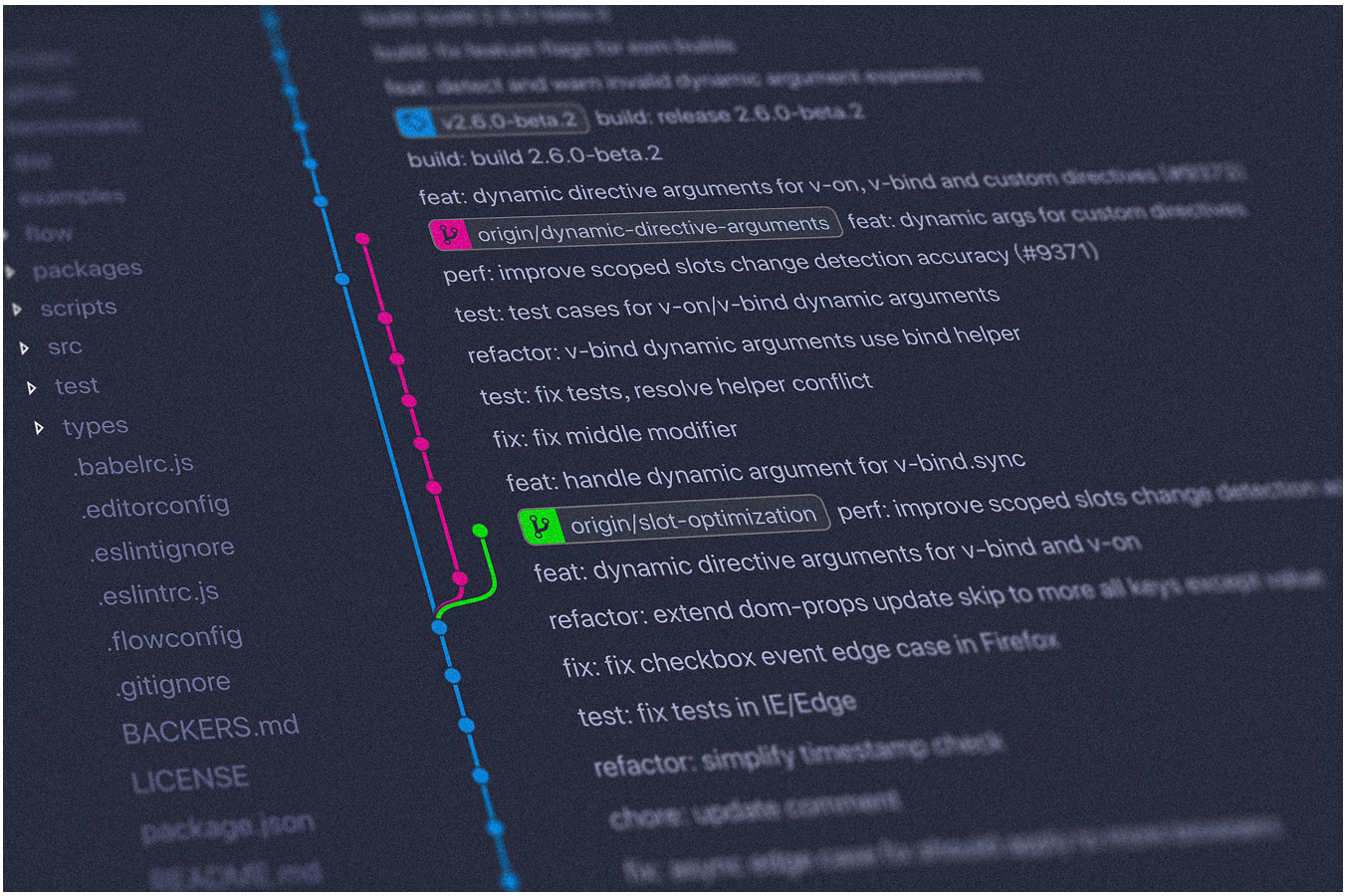

Photo by Yancy Min on Unsplash

We've All Seen It...

You're working on a project and it uses Git for version control.

You've just finished making a change, and you want to quickly update your branch.

So, you open up your terminal, and with a few quick commands, you update your remote branch with your changes.

```
git add .
git commit -m "added new feature"
git push
```

But then you do a bit of testing and find that you have a bug in your implementation.

No worries — you quickly find a fix and make another commit to fix the problem.

```
git add .
git commit -m "fix bug"
git push
```

You repeat this process a few times, and now you end up with a git commit log that looks like:

At the moment, this seems fine to you.

After all, you just worked on it, and you can easily explain what was worked on — even if the messages don't clearly convey it.

. . .

The Problem

A few months pass, and now, another developer is looking back through the changes you made.

They try to understand the high-level details of your changes, but since the commit messages are not descriptive, they cannot glean any information.

They then resort to reading through each commit's diff. However, even after doing so, they still cannot identify the thought process behind the choices that you made in your implementation.

Now, since software engineering is a collaborative process and the `git blame` operation exists, they find out who made these changes and start asking you questions about your implementation.

However, since it was so long ago, you don't remember much. You check back through your commits, and you no longer remember the logic behind the implementation decisions made in that project.

You send your colleague a sad emoji on Slack (😔) and tell them that you can't provide any more information than what they already have.

. . .

Writing Good Commit Messages

Hopefully, the above situation has demonstrated why it is important to write good, informative git commit messages.

In a field as collaborative as software engineering, it is imperative that we make it easy for collaborators to quickly gain context into our work.

Ideally, a good commit message will be structured into three parts — the subject, the body, and the closing line.

Subject line

The subject should be a single line that summarizes your commit's changes.

It should be written in the imperative tense, begin with a capital letter, not end with a period, and be 50 characters or less.

A good subject line will complete the sentence “This commit will …”.

A good commit message, like “add new neural network model to back-end”, nicely finishes the sentence.

A bad commit message, such as “fix bug”, does not complete the sentence very nicely, producing the awkward sentence “This commit will fix bug”.

Body

The body contains the meat of your message and is where you can go into details regarding your changes. Note that for some very small commits, such as fixing a typo, you probably won’t need a body, as the subject line should be informative enough.

In the body, you should go into more details about the changes you are making, and explain the context of what you are doing.

You can explain why you are making these changes, why you are choosing to implement the changes in this particular way, and anything else that would help people understand the thought process behind your commit.

Try not to repeat things that are obvious from the code changes in the diff. There is no need to provide a line-by-line explanation of your changes. Focus on covering more high-level details that may not be obvious from reading the code. The goal is ultimately to provide context into the development process around this change, which primarily concerns its motivations and goals.

Closing line

Finally, the closing line is the last line of your commit message.

This is where you can put useful meta-data regarding your commit, such as JIRA ticket numbers, GitHub issue numbers, co-author names, and additional links.

This can help to link important information together that relates to your change.