**Good Commit Messages**

Let's take a quick stroll down Stickler Lane and ask the question:

*How do I write a good commit message? And why should I care?*

These are *fantastic* questions! I can't stress enough how important it is to spend some time writing a *good* commit message.

Now, what makes a "good" commit message? That's a great question and has been [written about](https://chris.beams.io/posts/git-commit/) [a number](https://medium.com/@preslavrachev/what-s-with-the-50-72-rule-8a906f61f09c#.jwprsco0n) [of times](http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html). Here are some important things to think about when crafting a good commit message:

**Do**

* do keep the message short (less than 60-ish characters)
* do explain *what* the commit does (not *how* or *why*!)

**Do not**

* do not explain *why* the changes are made (more on this below)
* do not explain *how* the changes are made (that's what
* do not use the word "and"
  + if you have to use "and", your commit message is probably doing too many changes - break the changes into separate commits
  + e.g. "make the background color pink *and* increase the size of the sidebar"

The best way that I've found to come up with a commit message is to finish this phrase, "This commit will...". However, you finish that phrase, use *that* as your commit message.

Above all, ***be consistent*** in how you write your commit messages!

**QUESTION 1 OF 3**

Reviewing the guidelines on what makes a good commit message, is the following commit message good?

"Update the footer to copyright information"

* Yes    No

**SUBMIT**

**QUESTION 2 OF 3**

Is the following a good commit message?

"Add a

tag to the body"

* Yes    No

**SUBMIT**

**QUESTION 3 OF 3**

Is the following a good commit message?

"Add changes to app.js"

* Yes    No

**SUBMIT**

**Explain the *Why***

If you need to explain *why* a commit needs to be made, you can!

When you're writing the commit message, the first line is the message itself. After the message, leave a blank line, and then type out the body or explanation including details about why the commit is needed (e.g. URL links).

Here's what a commit message edit screen might look like:

*[Code editor showing the commit message edit window. A message has been typed, followed by a blank line, followed by the body of the commit.](https://classroom.udacity.com/nanodegrees/nd104-ent/parts/e325d5cc-ccd1-4ca8-a562-7eed09ab76c7/modules/a141ebc6-acee-494b-927f-2016d6fe0f9c/lessons/950113cc-8d4b-4346-b03f-8872ded1c0a6/concepts/0186fd0c-3929-4f0d-82bf-999ab24df61b#)*

This details section of a commit message \_is\_ included in the

Only the message (the first line) is included in

**Udacity's Commit Style Requirements**

As I've mentioned, there are a number of ways to write commit messages. If you're working on a team, they might already have a predetermined way of writing commit messages. Here at Udacity, we have our own standard for commit messages. You can check it out on our [Git Commit Message Style Guide](https://udacity.github.io/git-styleguide/).

If you haven't chosen a commit message style, feel free to use ours. But if you're working on an existing project, use their existing style; it's much more important to be consistent with your actual team than to be consistent with us!

**Git Diff Up Next!**

In the next section, we'll look at a new tool (with a familiar output!). This tool will tell us what changes we've made to files *before* the files have been committed!