# Documentation

Watched 2 videos in our breakout rooms:

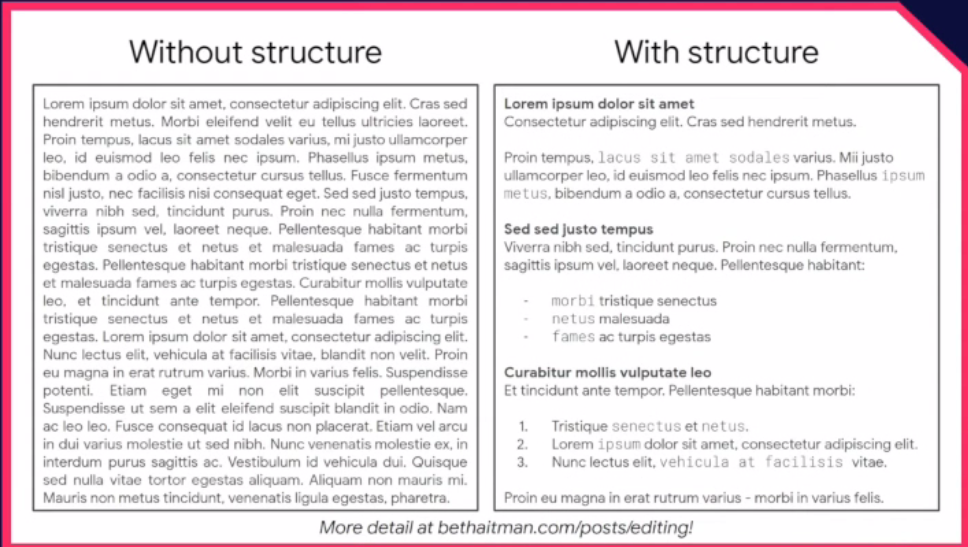
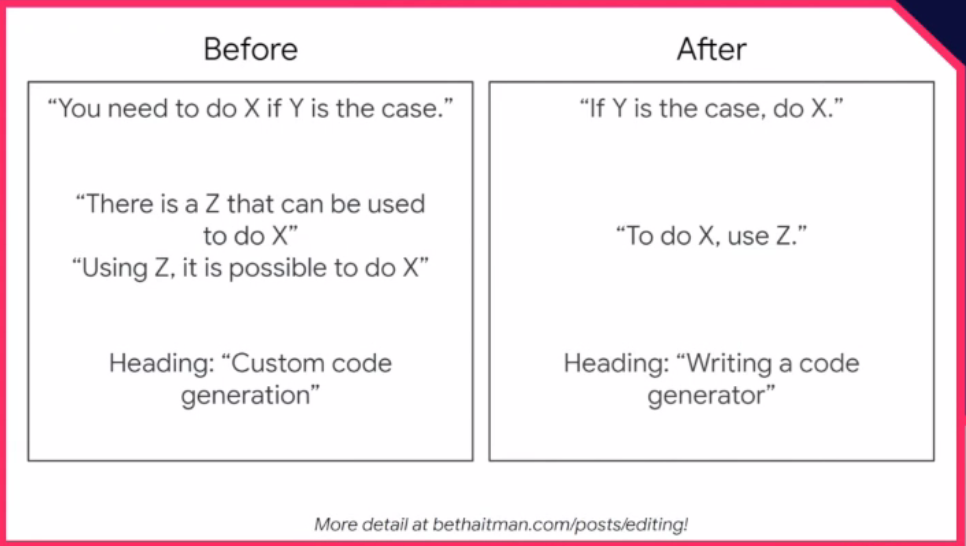
<https://www.youtube.com/watch?v=R6zeikbTgVc> – Writing effective documentation

<https://www.youtube.com/watch?v=uPMxUnBjGG8> – Top 5 ways to document your code

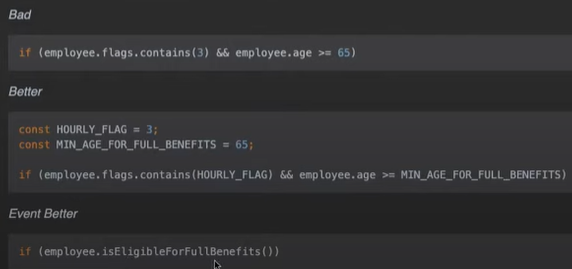
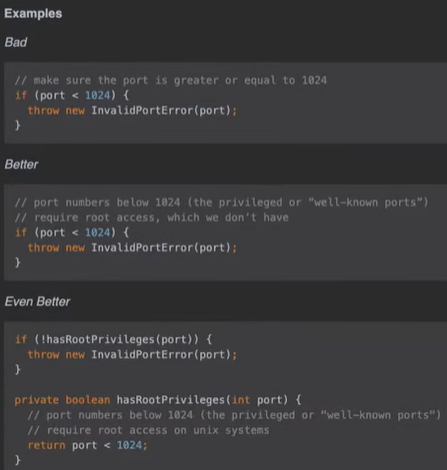
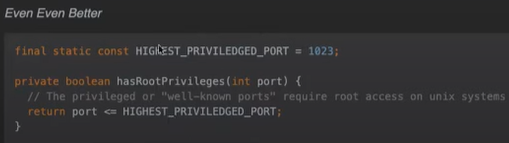
Below are my takeaways from the videos.

## Writing effective documentation notes:

5 tips for writing effective documentation:

* Start with what the reader needs
  + Help them achieve a goal
  + Could be narrow, such as “How do I setup this CLI” or broad like “I’m new to the team, how do I use this whole architecture?”
  + For each feature, ask yourself – “Does the read need to know about that?”
* Write less!
  + This makes it easier for the reader to find what they need
  + Remember – you don’t need to cover everything!
  + Write documentation that reflects what you can commit to (don’t document things you aren’t going to maintain)
  + Make sure it adds value
* Write the outline first
  + Think about the questions from point 1 above – use those as headings for your docs and fill them in
  + This will help structure
* Rubber ducking works for docs too
  + (This is talking it through out loud)
  + We talk more than we write, so talking it out loud can help express it better
  + Can help unblock and make better quality docs
  + A friendly, conversational style will help users take it in more
* Write readably
  + Not just for docs writing!
  + Give it structure: 
  + Keep your paragraphs nice and short (1 idea per paragraph)
  + Keep sentences to less than 25 words as a guideline
  + 
* There’s a 6th meta point - It’s not just about documentation
  + Docs is one part of the process to help the user
  + If you find as you’re writing that you’re trying to document around a problem, try and fix the thing so the problem isn’t there anymore!

## 5 ways to document your code notes:

* Use the tools your language provides
  + A.k.a Self-documented code (code describes itself)
  + Using the language to express the intent of the code clearly. For example using meaningful:
    - Directory structure
    - File/class/method/variables names
  + Use common standards/patterns that are familiar
  + The main goal for this is to allow devs to understand the code at a glance
  + Examples:
    - Bad: var n = args[0]
      * N means nothing to anyone
    - Better: var username = args[0]
  + 
* Comments
  + Comments should be used only to explain the intent behind code that cannot be refactored to explain itself
  + Mostly used for providing additional context, i.e. answering the WHY not the WHAT
  + Example:
    - 
    - 
* Tests
  + Tests document how the code should behave, not just that it “does what it does”
  + Test are examples of how to use an API, with assertions
  + Tests usually also give examples of edge case scenarios
* Git Commit Log
  + Most misunderstood
  + Gives more context about why a change is going into the codebase
  + Answers the “why is this done like that??” question
  + Flexible in terms of the amount of content, lives with the code with no clutter
  + Commit message can never get out of date
  + How to write a git commit: [https://chris.beams.io/posts/git-commit/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmZpTkgzaHZuYm9PY1pBcU9HbUdBRmd0T0tXQXxBQ3Jtc0trTERyRTFRUGVvdHFFUWV2bVJoaFRNaVhSaHJqaEl2WHFRUkVPOEtOTDd0a3kyN3ZVSkczNmZVYjBsSGZoaUpxelUxbjRZcUtxQWlzX2FZRW42cWtMZ3J0Z2xLTElKcDFBWFFhZXY1dEl0MGlvV3dtMA&q=https%3A%2F%2Fchris.beams.io%2Fposts%2Fgit-commit%2F&v=uPMxUnBjGG8)
* Readme and Wikis
  + Everything that is not directly code-related lives here (e.g. architecture, structure, how user can use it etc.)
  + Higher level context such as structure of code, coding standards, design choices etc.
  + This can help you better navigate the different components of the codebase

We were then tasked with documenting our projects from last week.

Other resources provided by Arshi in Slack that we haven’t visited:

* [Atlassian - Building Better Documentation](https://www.atlassian.com/software/confluence/documentation" \t "_blank)
* [Write the Docs Guide](https://www.writethedocs.org/guide/writing/beginners-guide-to-docs/)
* [README Editor with Template Sections](https://readme.so/editor)
* [How to Write a Good Documentation: Home](https://guides.lib.berkeley.edu/how-to-write-good-documentation)
* [Best Practices When Documenting Your Code for Software Engineers](https://betterprogramming.pub/best-practices-when-documenting-your-code-for-software-engineers-941f0897aa0)
* [Mastering Markdown](https://masteringmarkdown.com/)