Write Docs Devs Love

Ten Tricks to Level Up Your Tech Writing

Mason Egger
Sr. Developer Advocate

@masonegger
MasonEgger
mason.dev

What is Tech Writing?

- Think of that feeling you get when you follow an online tutorial or piece of documentation and the code works on the first run.
- Now think of all the hours spent wasted following broken, outdated, or incomplete documentation.

- What is the difference? What made one good and the other bad?

- Technical Writing is instructional or informative writing that focuses on how to accomplish a task using a specific tool
    - These tools can be hardware, software, and everywhere in between

Why is Technical Writing Important?

- Usually the first impression someone has of your project
  - If they don't like it, *they will leave and find another tool*
- Teaches users about a new project/code they didn't know about
  - When was the last time you found a new project just browsing source code?
    - A projects README is tech writing, so don't count that.
- Teaches users how to use your project ***effectively and safely***
  - Imagine if your vehicle manual said "Fill up the tire with air" instead of "Fill up the tire with air to 35 psi"
- Helps build a community around your project
  - Likely to bring people back
  - People will want to contribute
  - People will tell their friends about the project
  - Ex: DigitalOcean is known for its tutorials

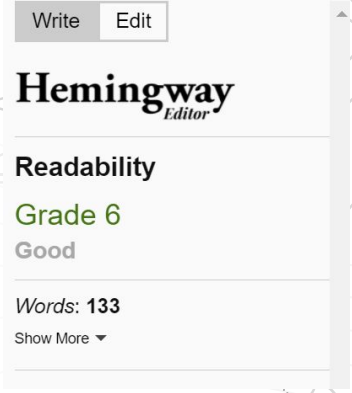
Mason's Top 10 Trips for Improving

- Disclaimer: These are my Tips and Tricks
  - This is what has helped me after spending two years writing tutorials and working with professional editors at DigitalOcean
  - Just because something didn't make it on this list doesn't make it invalid
  - Picking just 10 was *tough*

Tip #10 - Make Your End Goal Clear

- Have a clear, concise goal in your documentation in the first paragraph
  - This library allows you to do X
  - In this tutorial you'll use X and Y to build Z
- If you're writing a tutorial, don't spend 1000 words telling your reader how great the technology is.
  - They already know this, that's why they are reading your tutorial.
  - The reader wants to get something done
- Make it blatantly obvious what your reader will have learned, built, accomplished, understood, etc. by the end of reading the documentation

Tip #9 - Don’t Be Overly Verbose



- Technical documentation should be concise, not a novel
  - If you want to write a tech novel, go for it. They are hilarious
- SAT words are not needed here
  - Always assume the readers of your documentation don’t speak the same language as you
- Aim for a low reading level
  - Tools like Grammerly and https://hemingwayapp.com/ will give you the estimated grade level readability of your writing
    - I aim for 3rd grade reading level. The highest I allow myself to go is 6th

Tip #8 - Use Inclusive Language

- Avoid gendered language and go for more gender neutral pronouns.
  - Don't be afraid to use second person.
    - "Next you will ...."
  - If you're looking for second person plural, I *love* the word y'all
- Avoid using known internet slang that can be viewed as derogatory
  - Noobs, 10x Developers, Dummies
    - "Top 10 Tips for JavaScript Noobs"
- Avoid words that can make someone question or doubt their skills
  - What is "simple" or "easy" to someone might be challenging to someone else
    - "Simply install Python from source"

Tip #7 - Limit Technical Jargon

- *Jargon* - special words or expressions that are used by a particular profession or group and are difficult for others to understand.
- Overuse of Jargon can make it difficult for beginners to grok your content
  - See what I did there? Grok?
- Knowing your audience will help you decide how much jargon you can use
  - Are you writing internal documentation for your team? Probably can use some jargon related to the system
- Assume beginners if you don't know your audience
  - Beginners will appreciate you and experts will skim over it
  - You can also have a statement stating who your documentation is for

Tip #6 - Define ALL Acronyms

- Tech has *way too many* acronyms
  - We even have some acronyms that can mean two different things
- Acronyms can easily scare away readers
  - New learners often feel insecure and intimidated by Tech
    - Acronyms feed this fear
- Many ways of solving this
- Write out the full name of the acronym when you first introduce it
  - “You’ll need to add a record to the Domain Name System, DNS, to …”
- If you plan to use the acronym for the rest of the documentation, say so
  - “You’ll need to add a record to the Domain Name System, DNS, to … We’ll refer to the Domain Name System as DNS for the remainder of this tutorial”
- Define all acronyms used in the tutorial at the beginning or end and link back to them when you use them

Tip #5 - Avoid Memes/Idioms and Regional Language

- Avoid using memes and idioms unless you are positive who your audience is
  - Idioms - a group of words established by usage as having a meaning not deducible from those of the individual words
    - “Pull out all the stops”
    - “Piece of cake”
    - “Costs and arm and a leg”
  - Your 6 coworkers will probably understand these, a global audience of people who might have never seen Spongebob might not
- Avoid using regional language that might confuse native and non-native speakers
  - “Performing this command will totally trash your system”
  - “Don’t use this library, it’s dodgy”

Tip #4 - Use Meaningful Code Samples and Variable Names

- Use examples of real problems your code can solve
  - Readers want to know what problems your code solves, show them
  - Very often readers just skip to code, if you have good examples they may get their entire answer from a single code block
- Use meaningful variable names
  - We all say “Code should be self documenting”, well the documentation should be as well
  - Foo and Bar are useless. They need to go
- Include *everything* that is needed to run the code
  - This includes things like import statements
- Always have a completed copy of the code for copy/pasting
  - If you have a tutorial where you walk the user step by step with little code blocks, have one large codeblock at the end with the completed code

Tip #3 - Don’t Make Your Reader Leave Your Docs

- Avoid sending your reader to many other sites/links
  - Everything necessary to complete task should be in your article
  - Copying a few steps from another set of documentation is better than sending the user to many different sites
- If you are going to make your reader leave your article, have a reason and a way to bring them back
  - Do it at the beginning
  - Have a list of prerequisites at the very beginning that need to be completed before doing the tutorial.
  - Link out to these and direct the reader where to learn/accomplish the prerequisite
  - Don’t just say “Go install Python”. Give them a tutorial that sets them up for success and tell them to come back when they have finished
    - This also allows you to control and write for that specific environment

Tip #2 - Make Your Content Scannable

- Make it easy for the reader to find a single piece of information
  - Beginners tend to read entire posts while experienced users will scan for the information that they need
- Use Headings and SubHeadings to break up content
  - Outline big changes such as a next step or change of context
  - When paired with a Table of Contents it lets users find what they are looking for much easier
- Use consistent style when writing
  - Ex: Make all library names **bold** and all file paths in *italics*
  - The user will quickly pick up on this style and it will make the content more scannable
  - Consistency is key!

Tip #1 - Verify Your Instructions! Test, Test, Test!

- Always verify your instructions and test your work
  - The only thing worse than no documentation is incorrect documentation
- If possible, someone else test your work also
  - Having and editor, peer, teammate, or friend work through the documentation can help you find lapses
- Use a fresh environment for your testing
  - Attempt to remove all bias from your development environment. Your shortcuts, packages, and tools might not be on your readers workstation

Bonus Tip! - Practice, Practice, Practice

- The best way to get better at technical writing is to write
- Set aside a set amount of time daily/weekly to just write
  - You don’t have to publish it.
  - Don’t throw it away. Save it in a folder. You never know when you might want to dust it off

How You Can Get Started in Technical Writing

- Write documentation at work
  - There is always something to be documented
- Start a blog
  - The best part about a blog is it can be freer. You can write tutorials and blog posts
  - Great for practicing
- Contribute to Open Source Projects
  - Hacktoberfest is here, it's a great time to get involved and documentation can be a good place to start
    - It's where I started
- Apply to write tutorials as a contractor
  - DigitalOcean has our Write for DONation Program
    - $300 for the author, $300 to a charity of your choice
    - https://do.co/w4do

That’s all for this time!

- Follow me on Twitter @masonegger
- Slides can be found on my website mason.dev/speaking/remote-work