Technical Writers Yearbook

Celebrating technical writing in all its forms

Technical Writers Yearbook. Jim Hall, Georgia Harms (eds.). (2025) Technically We Write, supported by Hallmentum LLC.

Table of Contents

Introduction

Technically We Write is a community of technical writers, technical editors, copyeditors, web content writers, and all other roles in technical communication.

We want to celebrate all forms of technical and professional writing. Our readers have a wide range of interests, from writing tools and technologies, editing, usability, web SEO, and anything else you can put under the "technical and professional writing" umbrella.

We are fortunate to have such a talented and engaged community who share their tips, best practices, and "how-tos" about technical writing. We wanted to share a selection of the most popular articles from Technically We Write in this book, as our way to thank and recognize everyone for what they do.

My technical writing story: I love writing (Ashley O’Brien)

My interests in communication, language, and culture are a perfect fit in technical communication.

My journey to technical communication was serendipitous, a perfect blend of chance and fate. It began with a series of poor grades. I knew I was a scientist but I just couldn’t keep up with my peers in the chemistry major. My college had a digital tool students could use to find a new major and I plugged away, just to see options to earn a degree. I uploaded my transcript to the tool and (click!) I could graduate in five semesters with a journalism degree. Or (click!) I could graduate in four semesters with an anthropology degree.

It was then my solution became clear and I discovered I could graduate in two semesters with a Scientific and Technical Communication Bachelor of Science. All the classes I had already taken were required courses for this major.

I read about the long-standing profession of technical communication and the important value it offers both the science and technology disciplines. I scrolled to see that my science courses qualified me for the health science sub-plan. Not only did Technical Writing solve my academic crisis, it was a moment of validation. My interests in communication, language, and culture were front and center of this program.

Watching my Technical Writing path unfold was the best experience in my academic life. I saw my past, present and future in the lessons and practices. My disability didn’t hinder my learning and I was constantly impressed with the vast and intriguing subject matter. I was made for this. The Department of Writing was my home before I knew about the program. The cozy reading space was the perfect landing for a wary student-commuter and the downstairs eating facility filled me like a warm hug from my grandmother.

In the bewilderment of graduation (why didn’t anyone tell me about the costumes?) our program director handed me my diploma. Time stood still in that moment and I reached out in disbelief. I couldn’t believe the day had come. All the years of feeling lost, all the mismatched opportunities, all of the people who didn’t understand the knowledge that interested me. That was all behind me and I was a technical communicator now. I pulled my diploma close, gave my advisor one last hug, and went on my way.

Ashley O’Brien is a design innovator, UX mentor, and mixed methods researcher. Her work combines content strategy with design disciplines to help teams craft inclusive experiences.

Add map-making to your technical writing skills (Chris Hermansen)

Information design elements can make information more clear to your readers. Here's how to add maps to technical documents.

I have written quite a few technical documents over the course of my career. Many of these have been project proposals or project reports. Aside from requiring clear and concise prose, these documents often needed illustrations of various sorts: tables, graphs, images and so forth. And as well, very often, maps.

Because of the particular nature of my work, I have had some experience with making maps since the mid 1980s. I have seen the evolution of map-making software and hardware, and with the improved capabilities of both, improvements in techniques of map-making. Today, there is a tendency to subsume map-making into the broad topic of "data visualization"—and while reasonable in some ways, this does not do justice to the rich legacy of cartography down through the ages. I have also seen many documents that should have included maps but did not. While there may be documents including gratuitous maps, I haven't yet encountered them; possibly because making maps requires specific know-how.

I hope to encourage technical writers to consider whether the project at hand would be better with a map, and therefore whether it might be useful to learn something of map-making in order to create higher-value products. Based on my experience, writing professionals may find that the only person on the team who knows how to make maps has the inside edge for picking up the best assignments.

Why make maps?

Would-be map-makers who cannot articulate the point of the proposed map clearly and in a few words are doomed to failure. Like any document, a map needs a purpose; that purpose may be navigation, or general description of an area, or a reference that accompanies other information such as to show relative locations of points referenced in that other information. Generally, when making a map to accompany a document, that map will seek to show two things:

1. The main focus of interest presented. For example, a document proposing the construction of a building development would normally be accompanied by a map showing the location and extent of the building development.
2. Other features that somehow relate to the main focus of interest. For example, roads that provide access to the building development, location of electric lines, water, sewer, pedestrian areas, parks, parking, transit.

One of the advantages of making a map designed to show the spatial relationships between the main focus of interest and other related features is that the map can specifically incorporate all the relevant material and avoid the irrelevant. For example, it may be worthwhile to show certain nearby properties: restaurants that serve lunch but not late evening dining; or reproduction, mailing and courier services but not grocery stores or pharmacies.

Another advantage of making a custom map for a document is that existing map images or mapping services located by searching the Internet may have usage restrictions that prevent their use within the project at hand. While a certain amount of "quotation with attribution" is generally permitted by copyright law, the nature of maps make "quotations" virtually impossible. And even when an existing map without relevant usage restrictions is found, it may be designed to be reproduced at a different scale and size and look terrible when incorporated into the project.

My map-making workflow

My typical workflow for map-making looks like this:

Planning the project:

1. what should the map present or its theme—for example, rural communities in a study area that face a shortage of water
2. where can the information required for the map, the thematic data, be obtained

3. where is the map to be reproduced, and what restrictions does that place on its symbolization and contents—for example, in document that can be printed on letter or A4 sized paper in color or black and white
4. what additional reference data should be presented—for example, known rural water supply locations, transportation network, water network, topography, major urban centers, and political boundaries
5. what restrictions encumber the use of the various data sources
6. what map-making software is available, and what do I need to learn before use
7. what restrictions apply to other aspects of the map production—for example, font licensing, copyrights on source information such as images or text
8. what cartographic "standards" and "best practices" are applicable in this situation

Executing the project:

1. obtain permission to use when necessary in order to be able to publish the map, including preparation of acknowledgments
2. prepare a summary of sources and other data used, applicable restrictions, and permissions granted
3. obtain the data required
4. organize the data in a project folder on the workstation or in a spatial database on a server
5. develop a prototype map layout that meets the restrictions created by the reproduction format
6. choose a map projection suitable for the reproduction format
7. develop the symbolization of the data layers for the thematic and reference data
8. ensure that the legend or other reference space properly incorporates all required acknowledgments
9. review the draft map layout that incorporates the data layers, both on-screen and in the final A4 or letter layout

10. produce the final map layout for use in the target document

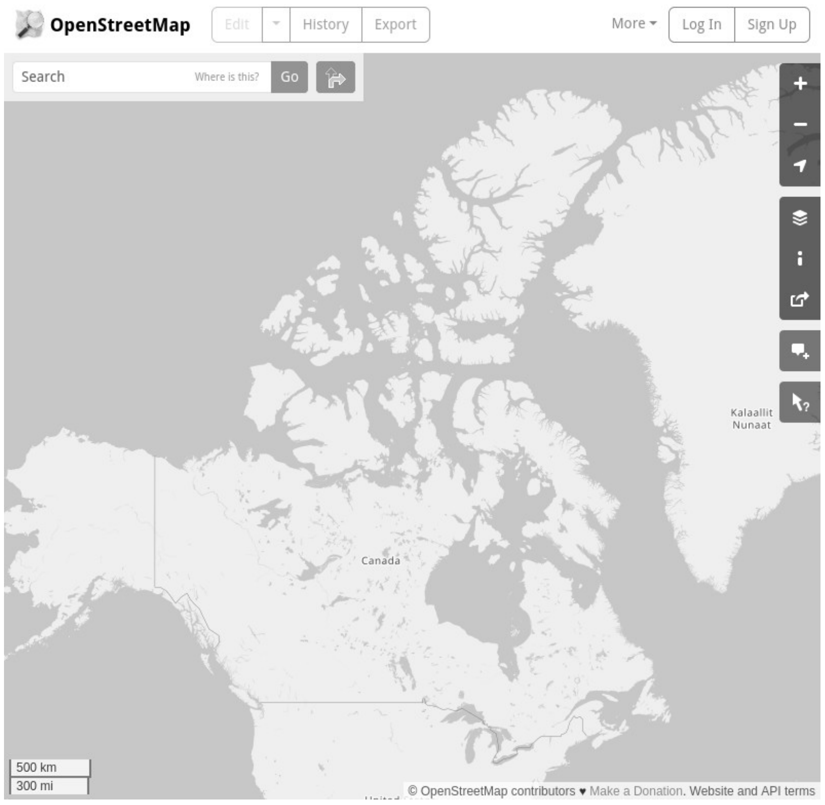
Key takeaways for map-making

To someone unfamiliar with making maps, the workflow described above might seem arcane and daunting. Maybe it seems easier just to use Google Maps or OpenStreetMap to "show some stuff in the surrounding area." And in some cases, that approach could be sufficient, maybe even the best; but certainly not in general. Let's review some of the core concepts, using OpenStreetMap as our review tool:

What is map data, map layout, or map projection?

Maps represent data that includes, or implies, location information. Bring up the OpenStreetMap website in your browser and look at the hyperlink it provides. Those numbers at the end (in my example: 71.34 and -96.82) are the latitude and longitude of the land mass shown in the center of the screen.

```
https://www.openstreetmap.org/#map=3/71.34/-96.8
```

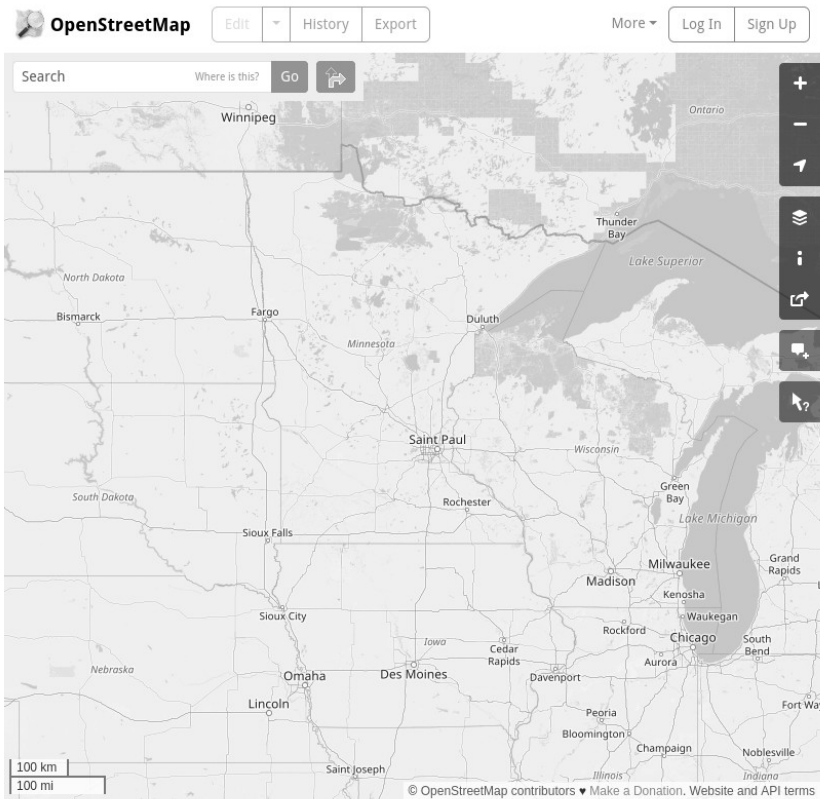


Viewing 71.34 and -96.82 in OpenStreetMap

Put your mouse on the screen over the land or water; you should see the mouse pointer transform into a hand, which you can use to grab—by clicking and holding down the left mouse button—the point over which you are hovering and drag it to some other spot. You can also use your mouse wheel to zoom further in or further out on the map. With a bit of zooming and dragging, I can get Saint Paul, Minnesota in the center of my map window. At this point, those numbers shown in the browser navigation bar should look something like this:

```
https://www.openstreetmap.org/#map=6/45.290/-93.647
```

We can also see that Rochester is southeast of Saint Paul; Duluth is north-northeast, and so on.



Viewing St Paul, Minnesota in OpenStreetMap

We can think of these locations as a particular kind of map data or point data, and the website is transforming that point data from the latitude and longitude locations (in decimal degrees) to pixel locations on the screen. This type of transformation (for example, from latitude-longitude position to pixel position) is a map projection.

Now fiddle around some more with zooming and dragging until you have Saint Paul again located near the center of the screen, with Fargo, North Dakota shown in the upper left and Milwaukee, Wisconsin shown in the lower right. As you compose this view, you are working on the map layout. A map layout is a combination of all the themes (cities, roads, water, states, provinces, etc) that are shown on the map, plus the way in which those themes are shown: roads as red lines, cities below a certain size as small circles… as well as the visual priority of these: labels over lines over areas, the sizes of text, what is named, what is the scale (one cm on the screen equals one km on the Earth), and so forth.

OpenStreetMap determines quite a bit of this map layout for you, but in the toolbar on the right side of the screen there is a tool to select "layers" of spatial data, such as the transportation group of layers.

Applications such as OpenStreetMap are wonderful ways to explore map data by zooming and panning and selecting layers, but they only go so far. A cartographer who wants to be in complete control of their map data, projection and layout needs to use the mapping capabilities provided by a geographic information system.

What are typical usage restrictions or acknowledgements?

Notice the "Copyright" menu item on the top bar of the OpenStreetMap website. If you click on that and read over the Copyright and License information, you'll note the terms of use allowed by OpenStreetMap, especially this paragraph:

> "You are free to copy, distribute, transmit and adapt our data, as long as you credit OpenStreetMap and its contributors. If you alter or build upon our data, you may distribute the result only under the same license. The full legal code explains your rights and responsibilities."—from the OpenStreetMap Copyright and License document.

Almost all map data and maps have some kind of licensing agreement attached to them. Maps themselves are usually covered under copyright law, in the same way textual information is. Map data is often provided subject to terms of licensing, in the same way software is. While copyright is pretty straightforward, licensing agreements can be convoluted to the point of incomprehensibility. Moreover, someone who obtains data under the licensing agreement available in 2015 needs to be aware of any clause in that agreement that binds the user to tracking future changes to the licensing agreement.

Cartographic standards and best practices

Many organizations that produce maps also provide documented standards for map production within their organizations. For example, the US Federal Geographic Data Committee, the Geomatics Canada CanTopo Map Standards and Specifications 1:50,000 (PDF) or the EU Statistical Cartography 2022 standards document all provide useful guidelines for cartographic standards and mapping best practices, and there is an endless list of other authorities.

In terms of a general introduction to cartography, the book *Elements of Cartography* (6th edition, by A.H. Robinson, J.L. Morrison, P.C. Muehrcke, A.J. Kimerling and S.C. Guptill, 1995, John Wiley & Sons) is arguably the classic reference work on cartography.

Other map-making tools

There are quite a number of tools useful in the process of map-making. Probably the two most obvious suggestions are QGIS (a free and open source geographic information system) and ESRI ArcGIS. Many organizations that collect and manage their own data and create maps and other geographic data visualization products use ArcGIS, as ESRI was the first generally useful geographic information system and has maintained a leadership position in the field ever since. However, QGIS is extremely capable, has a strong developer and user community and a great deal of reference material available, and is certainly a viable alternative to ArcGIS.

Some professional cartographers use tools such as Adobe Illustrator to produce final published maps based on output from a geographic information system.

Organizations with a substantial interest in managing their spatial data—for example, by maintaining an up-to-date central database—will select a spatially-enabled database management system, such as PostgreSQL with the PostGIS extension for this purpose. This kind of spatial database also provides geometry and geography functionality as part of the query language, permitting scripting of complex queries that incorporate location information into the analysis process. Some idea of the scope of these kinds of analysis can be gained by reviewing the standards published by the Open Geospatial Consortium.

Maps help readers understand data

Map-making today can be as simple or as sophisticated as the map-maker may desire. But doing anything beyond visualizing well-known information such as transportation, cities, and water bodies, and especially making thematic maps that strongly support a great technical report, can quickly pull the map-maker into the deep end of the cartographic pool.

For university or technical school students, many geography departments provide introductory courses in geographic information systems, which can be a wise investment in a future career in technical communication, particularly as post-secondary institutions usually have access to a range of open source and commercial software that would be difficult to assemble on one's own.

No matter the approach taken, the other important aspect of map-making is finding the data needed to support the work. Countries, states, cities, governmental organizations and NGOs offer a bewilderingly large range of data that may seem relevant, and only careful examination will prove whether the data is fit for purpose.

Chris Hermansen "I have been a full-time Linux user since 2005, a full-time Solaris and SunOS user from 1986 through 2005, and UNIX System V user before that. On the technical side of things, I have spent a great deal of my career as a consultant, doing data analysis and visualization; especially spatial data analysis."

8 tips for pitching articles to editors (Damon Garn)

Pitching article ideas to publishers is critical to a successful writing business.

I struggled with pitching articles to editors for a long time before finally dedicating some energy to establishing the best process. I want to share my thoughts with all the potential authors out there.

One essential aspect of my business is writing informative, tutorial-style articles for the IT industry. I contribute to many online publishers, and I've fine-tuned my approach to pitching. Your pitching goal should be to make it easy for the editor to say Yes.

Here are a series of tips for effectively and efficiently pitching article ideas to editors. Read to the end to find a sample pitch for this article!

Learn the publication's cadence

It may take some time, but learn the publication's cadence and organization. One of my key customers organizes their budget and schedule on a quarterly basis. I've learned to approach them with ideas just before a new quarter begins. Find out if the publication does collections of similar articles and pitch to those. For example, I was assigned five articles on Linux backup tools as part of a single collection in the summer of 2023.

Once you've submitted a few well-polished articles on time, remember to ask for other contacts within the company for whom you can write.

Provide the why and context

Give the editor background information on why the content is essential and unique. Provide context around the subject, especially if it's a critical topic that isn't prevalent in the news. It's easier for the editor to accept the piece if they understand who their readers will be and why those readers care about this subject.

Your outline becomes your pitch

My pitches are a summary of my article outline. I rarely have a fully developed outline when I make the pitch, but I always have a general idea of the article's structure. This allows me to estimate the word count and the effort I'll need to invest, all tied to the article's cost. In some cases, I've identified topics that should become two articles for readability and efficiency.

Estimating your effort is critical. I defeated myself with a few early pitches that were not well thought out. I proposed topics that I thought would be straightforward but turned out to be extremely difficult and time-consuming. Those topics are fine but be sure to plan time for them.

Schedule follow-ups

Create a follow-up calendar. Editors are busy folks! You may recognize the brilliance of your idea but other pitches flow across their desks daily. You might get lost in the shuffle, so don't hesitate to reach out a few weeks later to ask whether they are interested in your pitch. Just don't bug them by following up too soon.

You're also busy, so calendar reminders help you keep track of your queries.

Provide the details

Remember to provide relevant details. My pitches include one to two short paragraphs describing the topic, providing context, and explaining why it's important. I am also careful to include a paragraph estimating the word count and target audience. Finally, I remind the editor that I provide screenshots and command snippets. You could include the cost here, too.

Here's an example:

> This article is __ words and includes screenshots and command snippets as appropriate. The target audience is __ of any level.

The goal is to make it easy to say Yes.

Use a pitch template

I view pitching as unbilled hours (though I recognize the work pays off later). I need the pitching process to be as efficient as possible, so I created a template in Microsoft Word. It includes title headers and sample language (such as word count and target audience). All I must do is fill in the details. I place my contact information and date at the top, too:

> Article pitches for Organization, Q_ 202_
>
> Cogspinner Coaction, LLC, Damon M. Garn *cogspinnercoaction@gmail.com*
>
> **Article 1**
>
> Article content…
>
> This article is __ words and includes screenshots and command snippets as appropriate. The target audience is __ of any level.
>
> **Article 2**
>
> Article content…
>
> This article is __ words and includes screenshots and command snippets as appropriate. The target audience is __ of any level.

Pitch templates keep your queries consistent and easy to read.

Track your pitches and avoid repeats

Develop a tracking system for your pitches. I badly embarrassed myself some years ago by resending a series of pitches to an editor a few months after the initial query. She replied that she'd already declined these ideas and wanted to know why I wasted her time resending them.

I use Microsoft Excel to manage my pitches. I will write a separate article with details on pitch tracking in the future. For now, ensure you have some method of avoiding repeated queries to the same editor.

Remember to track accepted articles!

Articles are not rejected; they are declined

I've periodically dabbled in writing fiction, and some years ago, I learned a great mental trick. Upon being down and frustrated over a good story being rejected multiple times, a fellow author suggested a change in terminology. Pieces are not "rejected" but rather "declined." This term is gentler and more accurate. When an editor declines a piece, it may not be a reflection of that article's quality. Instead, it indicates the piece isn't a good fit for the publication at the moment.

Not only does that word change help maintain your enthusiasm, but it reminds you to submit the pitch to other editors, where it may fit better with their current initiatives.

Making a great pitch

It goes without saying that you should check your pitches for spelling and grammar. This may be the editor's first exposure to your work.

Pitching article ideas to publishers is critical to a successful writing business. I hope the above tips provide you with approaches you can apply to your own writing endeavors. Please comment with your own pitching tips.

Here is a sample pitch for the article you just read:

> Article pitch for Technically We Write
>
> Cogspinner Coaction, LLC, Damon M. Garn *cogspinnercoaction@gmail.com*
>
> **8 Tips for Pitching Articles to Editors**
>
> Many new authors are breaking into tutorial and how-to writing. This article demonstrates specific and simple tips for authors pitching article ideas to editors. The goal is to explain basic techniques to make the process more consistent and efficient. Readers benefit from ideas they can apply immediately.
>
> This article is 800 to 1000 words and includes screenshots and command snippets as appropriate. The target audience is technical writers and authors of any level.

Damon Garn spent twenty years in the classroom, in-person and virtual, delivering technical training on Windows, Linux, security, and CompTIA products. In recent years, he left the training industry and now provides freelance writing and editing services for companies.

How and why I write articles (David Both)

Share what you know by writing about it, including all the mistakes and problems, to help others through the same issues.

I've written a lot of articles over my more than fifty years in IT, most of them since I started using Linux in 1996. Without doing an exact count, I can safely say that the total is over 150 articles.

Over the years, people have asked me how I decide what I'm going to write about and about the process I use. This is my process:

Choosing a subject

Although an image of a writer wracking their brain to come up with an appropriate subject for an article comes to mind, that's not how it works—for me at least. I have little trouble finding things to write about. Sometimes the problem is sorting through the possibilities to choose the ones that belong high on my list. I very seldom discard a potential article but they can get bumped down in priority.

For example, this article was conceived during an email exchange with Jim Hall. Jim had already interviewed me about the process I use for writing my books. In order to maintain the focus of that article on my books, we didn't even touch on my articles. So when he shared a prompt for articles for Technically We Write, I suggested that I write an article about writing articles.

Editors send out requests for articles to their writers about specific subjects. This is a common practice and the requests can range from looking for a single article on a single subject to something more general like, "Pi day is coming up. Please write an article about how you use your Raspberry Pi." Or something similar. Other requests I've received included articles to celebrate the birthday of the Linux kernel, the top *x* software packages for any of a number of types of software like accounting or office suites, and much more.

I have sometimes written articles as a result of those requests. However I have a foolproof method for picking subjects. It usually starts with my mistakes—or at least with the problems I encounter while doing my systems administration job.

Yes, mistakes and problems. I have made many mistakes and encountered lots of problems. All of which I have had to resolve myself.

Those problems, including and especially the ones I created for myself, and their resolutions are the best source of my articles. I find that if I have encountered a problem, others will too—and they'll look for a solution.

Getting started

But I didn't start out by writing articles.

At first, I started keeping notes recording the problems, how they came about—especially the self-inflicted ones—and how I fixed them. These were simple notes with a description of the problem and usually a bit about how I performed problem determination. I also described the details of how I fixed it including any failures.

I started this in a paper notebook I kept many years ago. As more notekeeping options became available for my computer, I moved first to a note-taking application and then to a simple database. At that time, I was working in the IBM PC Company support center, so I started sharing my database with others over the simple network we had.

After leaving IBM, I continued to keep notes on my PC using various tools. This eventually led to a large Lotus Notes database, primarily about IBM hardware and OS/2, which I shared with my consulting customers.

Eventually, I realized that OS/2 was stagnating. I decided to switch to Linux because that seemed to me, in 1996, to be the best and most exciting software around. There's a long story about how that all happened, but I won't bore you with it here. The point is that I started doing with Linux what I had done previously with IBM's OS/2 operating system.

Getting them published

When I realized that other people might encounter the same problems with Linux I had, I began publishing those notes on my own websites. In 2013, I went to my first All Things Open conference and got in contact with the editors at Opensource.com. That was a great place to publish my articles, and I started writing for them.

Because my notes were just that—collections of notes—I needed to add some context to make them more understandable to readers. So I added stories about how I encountered the problems and how I performed problem determination to locate the root cause. I also added more readable descriptions of creating and implementing the solution. This tied it all neatly together in a readable narrative.

Those articles ranged from fun to deeply technical. But most tended to be based on my own experiences.

I became friends with many of the authors and staff at Opensource.com. We shared a lot of things, including our love for Linux and open source software as well as our deep commitment to sharing our knowledge.

When Red Hat stopped adding new content to Opensource.com, I started publishing my own articles—as well as those of other former Opensource.com contributors—on my own website, Both.org.

Final thoughts

I have many things to write about. That's in part at least, due to the fact that I am a hands-on person. This is kind of the nature of Linux system administrators: we like to play with the toys. It also provides us with a steady stream of fun things to write about.

No matter how trivial something seems to you, there are always plenty of people who want to read about it. So if you find yourself thinking you should take notes about something you just did, or that others might benefit from your newly gained experience, go ahead and write the article. Then submit it to an appropriate place—or just start your own website if you can't find a suitable venue.

David Both is an Open Source Software and GNU/Linux advocate, trainer, writer, and speaker who lives in Raleigh North Carolina. He has written articles for magazines including *Linux Magazine*, *Linux Journal*, and Opensource.com. He currently has five books published with Apress: *The Linux Philosophy for SysAdmins*, a self-study training course in three volumes, *Using and Administering Linux: Zero to SysAdmin*, released in late 2019, and *Linux for Small Business Owners* with Co-author Cyndi Bulka.

Mastering Markdown with MarkText (Don Watkins)

MarkText is a text editor that aims to make writing Markdown files easier.

A few years back, I learned about Markdown. I had never heard of it before, but Markdown was a familiar skill to many of my writing colleagues. Yet it was uncharted territory for me.

While its merits were extolled by many, I remained uncertain about the necessity of acquiring Markdown skills. My inherent curiosity, however, drove me to delve into the realm of Markdown, investigating how it could integrate into my writing endeavors.

What is Markdown

Markdown is a universal method for composing text, employing concise notation to apply to style. For instance, rather than relying on a button click in a desktop word processor to emphasize a word, you envelop the word with two asterisks, such as `**word**` to make a word bold.

Markdown holds a significant edge in its reliance on intuitive notations, often drawing from our ingrained habits. Employing asterisks for emphasis, and utilizing characters to distinguish headlines, these practices align seamlessly with our natural inclinations.

As I underwent the learning process, I found an excellent Markdown cheat sheet online and discovered that I could write Markdown in any simple text editor like Nano, Vim, or GNOME Gedit. While it is technically possible to use almost any text editor to write Markdown, it is much more powerful to use an editor specifically designed to output Markdown formatted documents.

MarkText for Markdown

I stumbled upon MarkText, an open source platform equipped with features that streamline Markdown writing while presenting an unobtrusive interface. This tool boasts six themes, comprising three light and three dark options. I find the dark themes more comfortable to work with. Notably, the user documentation is comprehensive, and a dedicated resource for Markdown syntax assistance is also available.

MarkText presents a clean, minimalistic interface with a real-time preview feature. It accommodates several Markdown specifications, including Commonmark, GitHub Flavored Markdown, and Pandoc Markdown. Its official website shows MarkText supports Markdown enhancements like KaTeX, front matter, and emoji usage. The application is capable of generating both HTML and PDF output files.

Within MarkText, you'll find diverse editing modes such as typewriter mode, source code mode, and focus mode. Incorporating images is effortlessly achieved by copying and pasting them directly from the clipboard.

For added convenience, a pop-up situated in the upper-left corner of the MarkText interface provides a continuous tally of the characters and paragraphs that have been entered. This proves particularly advantageous for writers.

Saving files is a straightforward task accessible via the upper-left menu of the MarkText window or by employing the Ctrl+S shortcut. Remarkably, the menus within MarkText bear a friendly and recognizable resemblance to those found in fundamental text editors or word processors, creating a sense of familiarity for users.

The versatility of MarkText truly impresses me, as it effortlessly accommodates many formats through simple keystroke shortcuts. These include table blocks, diagrams, inline formats, math formula blocks, and other code blocks.

Try MarkText for yourself

MarkText is an open source project under the MIT license. You can download the latest version from GitHub. You can acquire MarkText for your respective operating system through the following:

- Linux
- macOS

- Windows

Alternatively, on macOS, you can install MarkText using Homebrew:

```
brew install --cask mark-text
```

On Windows, you can install MarkText through Chocolatey:

```
choco install marktext
```

MarkText continually seeks the support of sponsors and developers. The project provides a contributor guide for those interested in contributing. Furthermore, you can back the project on Patreon and Open Collective.

Don Watkins is an educator, entrepreneur, open source advocate, life long learner, Python teacher. M.A. in Educational Psychology, MSED in Educational Leadership, Linux system administrator.

Different keyboard sizes (John Hall)

Explore several different configurations in full-sized and popular compact keyboards.

The most common interface for technical writing is the keyboard. But despite using a keyboard for so many hours in the day, you might not know much about the keyboard you use, or why it's different from another keyboard. In this article, let's explore several common keyboard configurations and how they are arranged.

Keyboard configurations

A full-sized keyboard is the most common type of computer keyboard found in offices. A full-sized keyboard typically consists of 104 keys, which includes a row of function keys labeled F1 through F12 (commonly called the "F row"), arrow keys, navigation keys (including Home, End, Page Up, and Page Down keys) and a number pad.

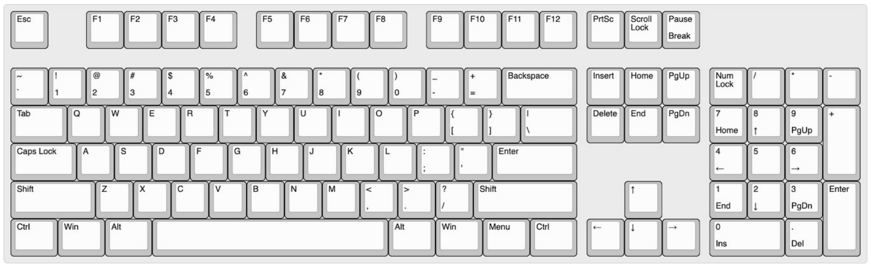
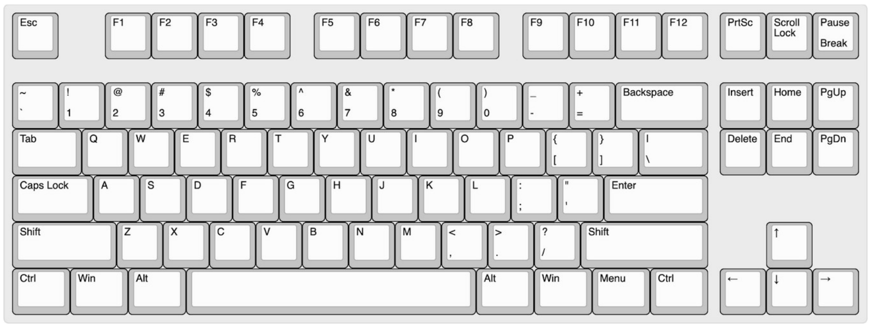

Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 PrtSc Scroll Lock Pause Break
~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace Insert Home PgUp Num Lock / * -
Tab Q W E R T Y U I O P { [}] | \ Delete End PgDn 7 Home 8 ↑ 9 PgUp +
Caps Lock A S D F G H J K L : ; " ' Enter 4 ← 5 6 →
Shift Z X C V B N M < , > . ? / Shift ↑ 1 End 2 ↓ 3 PgDn Enter
Ctrl Win Alt Alt Win Menu Ctrl ← ↓ → 0 Ins . Del

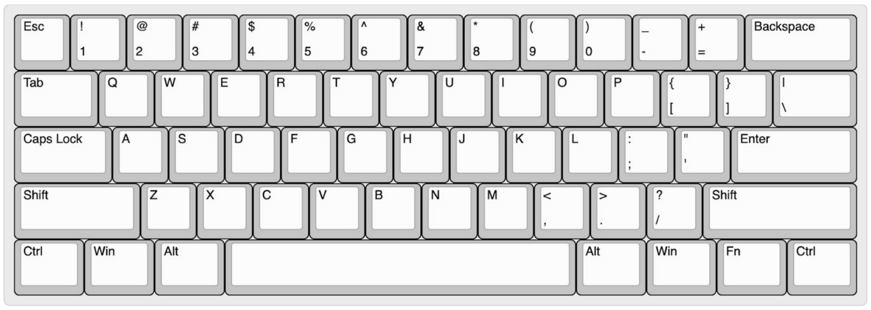
Image: Full-sized keyboard

If you do not need the number pad, then consider a TKL keyboard. A TKL is the same as a full-size keyboard but omits the number pad on the right side. The number pad, often referred to as a "ten key," is precisely what the "TKL" acronym signifies: Ten Key Less. By giving up the number pad, TKL keyboards gain the benefit of a more compact layout, which allows for greater freedom of movement for the mouse and conserves desk space.


Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 PrtSc Scroll Lock Pause Break
~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace Insert Home PgUp
Tab Q W E R T Y U I O P { [}] | \ Delete End PgDn
Caps Lock A S D F G H J K L : ; " ' Enter
Shift Z X C V B N M < , > . ? / Shift ↑
Ctrl Win Alt Alt Win Menu Ctrl ← ↓ →
Image: TKL keyboard

A 60% keyboard is even more compact than a TKL keyboard; it is a TKL without the F row, navigation cluster, and arrow keys. The reduced footprint of 60% keyboards frees up more desk space. However, this means users need to access those keys by using the Fn key. For instance, to access the F2 key, users would need to press Fn+`2`.

It seems that each keyboard manufacturer has its own interpretation of how users should access arrow keys on a 60% keyboard. For instance, they may require you to press the Fn key in combination with the `L`, `<`, `>`, and `?` keys to navigate; alternatively, they might designate the `IJKL` keys for this purpose, or even the `UHJK` keys. However, a customizable firmware will let you control how to access these key functions.


Esc ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace
Tab Q W E R T Y U I O P { [}] | \
Caps Lock A S D F G H J K L : ; " ' Enter
Shift Z X C V B N M < , > . ? / Shift
Ctrl Win Alt Alt Win Fn Ctrl
Image: 60% keyboard

Some people do not like the minimalist layout of 60% keyboards and want dedicated arrow and navigation keys. Moving up in size, a 65% keyboard is a 60% keyboard that adds arrow keys and a column of navigation keys while only being one column wider than 60% keyboards. In order to squeeze in the arrow keys, the right Shift key is shortened.

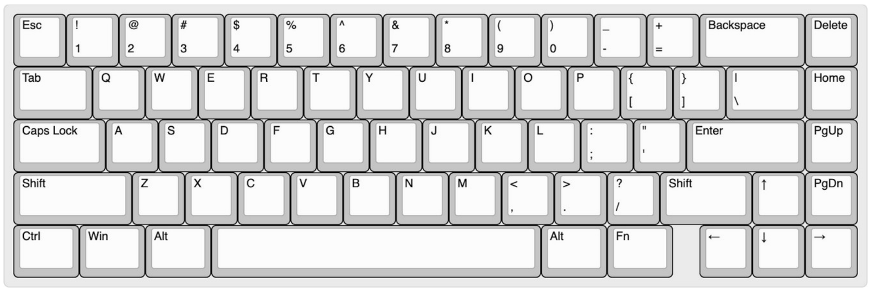


Image: 65% keyboard

A 75% keyboard is a 65% keyboard with an F row. It is still compact like a 65% keyboard but adds just a row in height. It retains a compact footprint while offering more keys and functionality compared to its smaller counterparts. This size is popular with people who still want their F row but want something smaller than a TKL.

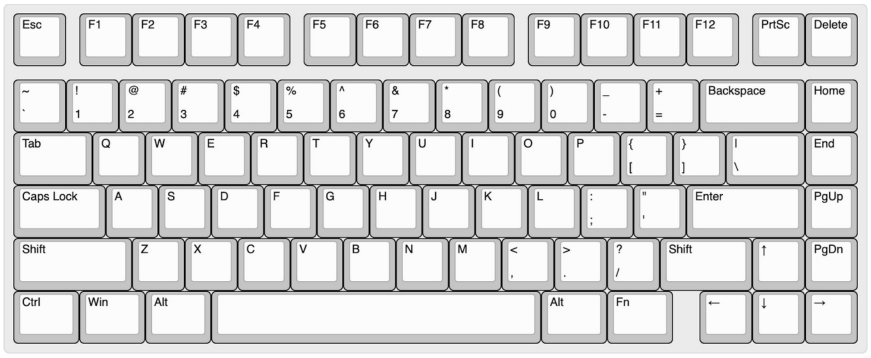


Image: 75% keyboard

A 96% keyboard is a 75% keyboard with a number pad. These keyboards are commonly called an “1800 compact” after the Cherry G80-1800 keyboard. It is slightly smaller than a full-sized keyboard but larger than a TKL keyboard. It offers a balance between a space-saving design and the functionality of a full-size layout, making it an attractive option for users who desire a compact yet fully featured keyboard for everyday typing and productivity tasks. However, to make room for the number pad, the navigation keys are rearranged, and the zero key on the number pad is shortened. This may cause you problems if you are used to a wider zero key.

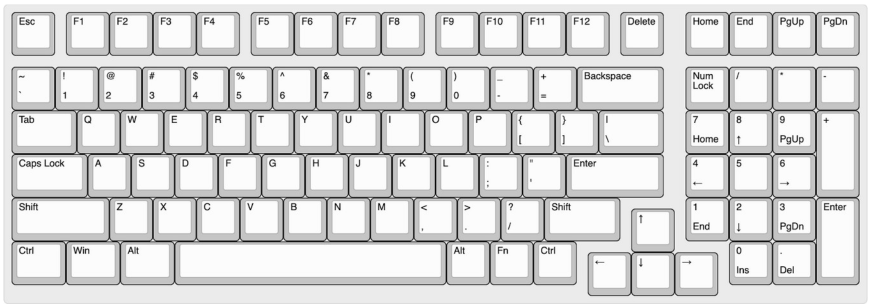

Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 Delete Home End PgUp PgDn
~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace Num Lock / * -
Tab Q W E R T Y U I O P { [}] | \ 7 Home 8 ↑ 9 PgUp +
Caps Lock A S D F G H J K L : ; " ' Enter 4 ← 5 6 →
Shift Z X C V B N M < , > . ? / Shift ↑ 1 End 2 ↓ 3 PgDn Enter
Ctrl Win Alt Alt Fn Ctrl ← ↓ → 0 Ins . Del

Image: 96% keyboard

These are the most common keyboard sizes. Of course, there are other variations. For example, a 40% keyboard is a 60% keyboard that omits the number row. With this layout, all the keys on your keyboard are no more than one column or one row away.

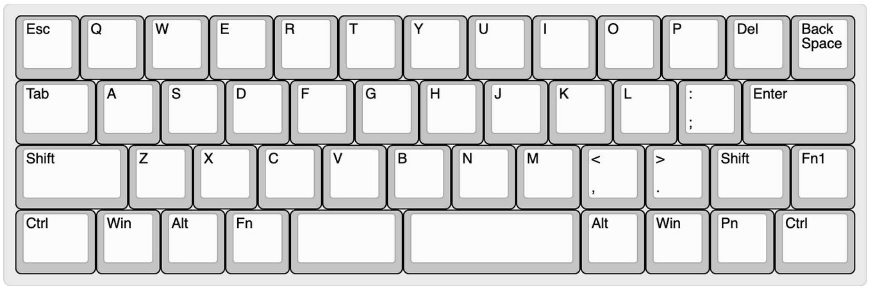

Esc Q W E R T Y U I O P Del Back Space
Tab A S D F G H J K L : ; Enter
Shift Z X C V B N M < , > . Shift Fn1
Ctrl Win Alt Fn Alt Win Pn Ctrl

Image: 40% keyboard

A variation of this size is a 40% ortholinear keyboard. An ortholinear keyboard has straight lines between rows and columns.

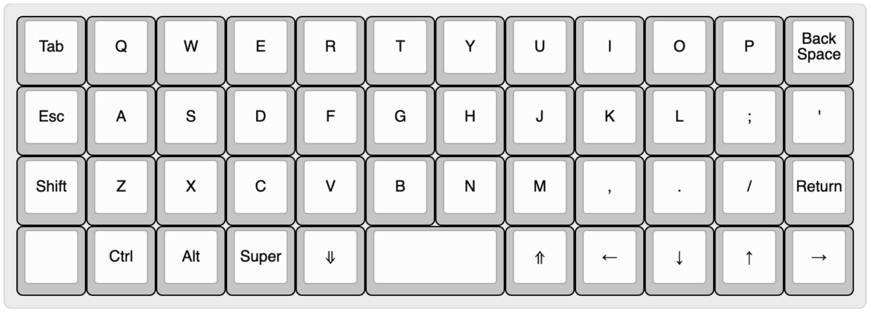

Tab Q W E R T Y U I O P Back Space
Esc A S D F G H J K L ; '
Shift Z X C V B N M , . / Return
Ctrl Alt Super ⇓ ⇑ ← ↓ ↑ →

Image: 40% ortholinear keyboard

Split Keyboards

Your shoulders are wider than your keyboard. For example, the average shoulder width of males is between 16 inches and 21 inches. As a result, you bend your forearms at the elbows inward towards your keyboard. But to properly type on the keys, you have to bend your wrists outward in the direction of your little finger.

This is known as ulnar deviation and requires tensing your muscles. Try holding your arm out in a “stop” gesture. Easy, right? Now hold it that way for two hours straight. If you type nonstop for long periods, you could start feeling cramps or other wrist pain.

In contrast, a split keyboard rearranges your keyboard to keep your wrists straight. Ideally, the two halves would be totally separate so you can place the two halves further apart and/or angle them.

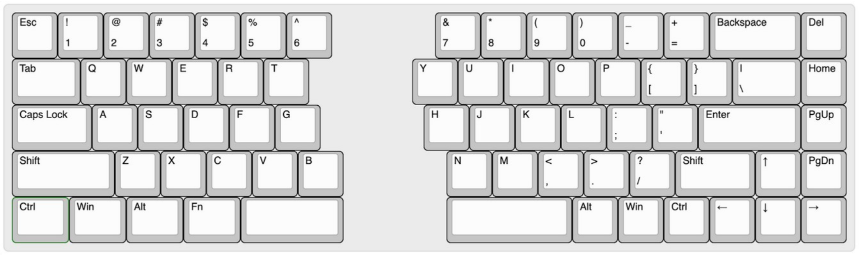


Image: Split keyboard

Keycap sizes

The keyboard keys themselves are switches of some kind, covered by a keycap. Keycap sizes are described in “units,” and 1 “unit” (written as 1u) is the size of a letter or number key. Keycap sizes are multiples of 0.25u. For example, the Tab key is typically 1.5u; Caps Lock is 1.75u; Backspace is 2u; Enter is 2.25u; and so on.

The right Shift key is normally 2.75u. However, on 75%, 65%, and other keyboard layouts that squeeze in the arrow keys, the right Shift key is shortened to 1.75u.

The standard bottom row of modifiers like Ctrl and Windows are 1.25u. However, you will also see keyboards with 1u or 1.5u modifiers to the right of the spacebar, especially on 75%, 65%, and 96% (1800 compact) layout keyboards.

An essential tool

The keyboard is an essential but often overlooked tool in technical writing - and not all keyboards are the same. These are the most common keyboard examples showing different layouts. Whether you want a full-size keyboard, with or without the ten key pad, or a more compact keyboard that you can take with you, you can find the keyboard that matches your personal preferences. Choose the keyboard that’s right for you.

John Hall is a senior systems engineer. He also loves keyboards. You can find him on Reddit and other social media, where he shares advice about various keyboards.

Learn concepts, they'll teach the tools (Robin Bland)

Don't worry about learning the "wrong" tool. Learn the concepts and those skills will transfer to other, similar tools.

For anyone who wants to transition their career into technical writing or technical communication, the variety of available tools can feel overwhelming. Which tool should you learn if you want to get hired? The concern is that if you don't learn the right tool, you might not get hired.

I've asked this question of many working professionals. For example, I recently attended a workshop with technical writers and editors and heard the same theme: pick a tool, learn it, and those concepts will carry forward to other specific tools at another organization.

HTML as a first step

If you want to start learning a digital writing technology, I recommend exploring HTML, the Hyper Text Markup Language. HTML is used everywhere and it's not too difficult to learn. I can summarize the essentials in these statements: Tags are enclosed in angle brackets like `<p>`. And if you "open" a tag, you need to "close" it, such as `<p>` to start a paragraph and `</p>` to end it.

Once you understand the basics of HTML, you are in a great position to learn how to create content in a web content management system such as Drupal or Wordpress. The content management system does all the work of organizing your pages and automating the workflow; you just need to create the content.

Web content management platforms usually let you type content into pages using a tool that feels like a lightweight word processor. So you don't really need to write HTML by hand, although knowing a little HTML will be helpful when you need to fix a web page when the web content system eventually messes up your formatting.

HTML then XML

With a basic understanding of HTML, it's a "step to the left" to learn XML, the eXtensible Markup Language. XML is just data that uses a similar markup syntax to HTML. Tags are enclosed in angle brackets. If you open a tag, you need to close it.

Other markup systems are based on XML. For example, technical writers who use Docbook or DITA are actually creating XML files. It's just that the XML rules are different depending on the markup system.

With a little knowledge of XML, you can also pick apart other file types to see what's in them. The EPUB document format for ebooks is a zip file with XML metadata and HTML content. You can rename any EPUB document to use a `.zip` extension, then extract it to examine the contents.

Open source tools are a great start

If you want to build experience in technical writing, you don't need to buy expensive tools. Many organizations use open source tools like LibreOffice to create documentation. And even those that use some other office software such as Microsoft Word use the same features as LibreOffice such as styles.

I encourage aspiring and new technical writers to download and explore LibreOffice Writer. Experiment with creating different kinds of documents, including memos, instructions, booklets, and manuals. Use styles for everything: paragraph styles for headings, subheadings, and indented paragraphs, lists, and other interior content; character styles for "inline" formatting such as italic and bold text; and page styles to adjust the page size, margins, and other dimensions.

If you use styles throughout your content, you can always update a style later and the word processor applies the new formatting throughout the entire document. For example, a technical editor may ask you to display bold and italic text in a color so it stands out from the rest of the text during review. Update the "Emphasis" character style to use blue, and every instance of italic text becomes blue. Or you might decide later that "Heading 1" section headings should start on a new page; update the style to insert a page break before the heading, and the word processor updates the document.

These features are not specific to LibreOffice, but if you learn how to use LibreOffice well, you will be able to use styles in other word processors such as Microsoft Word.

The “path” to access these features may be different across LibreOffice or Word, but the functionality is the same.

They’ll teach the tool

When organizations hire a new technical writer, they may ask about specific tools that they use. An organization that relies on Microsoft Word will ask interview questions about Word. But these same organizations will also want to hear from applicants that are strong in LibreOffice Writer. The same skills apply to both; if you know how to leverage styles to create technical documents in LibreOffice, you’ll be able to do the same in Word.

Similarly, an organization that builds web pages using Drupal will also want to interview applicants who are familiar with Wordpress. Both are web content management systems. If you know how to create and manage websites using Wordpress, you can do the same in Drupal.

The key is to learn concepts and understand how to apply those concepts everywhere. Knowing the concepts and how to apply them effectively in one tool opens the door to working with other, similar kinds of tools. And the organization will teach you how to use those specific tools.

Robin Bland is a technical writer in the Minneapolis/St Paul area.

5 things you didn’t know about styles in your word processor (Seth Kenlon)

Styles are a power feature in technical writing. Learn five ways you can leverage styles to work for you.

You probably add “style” to a document you’re writing without even thinking about it. When you make a word bold or italics, you’re adding style. When you create a heading above a paragraph, you’re adding style. Style controls how your document looks, and it’s important because ideally it also helps your reader process the information you’re hoping to convey. For example, I’ve written this article as a list of clearly-defined concepts in an attempt to help you (the reader) remember the principles I’m hoping to teach. So that each principle stands out, I’ve made each item in my list a sub-heading in the article. These styles help the reader to navigate the content. Some people will read just the subheadings, while others will read the full text. The ability to skip around a document is a feature of how the article is styled.

Style can affect the way you write, because it affects the way people consume the information you’ve written. Here are five things you can do to put style to work for you.

Use the style menu

Try this experiment: Have a friend or your computer read a random document to you, with your eyes closed. Try to guess which words were written in bold or italics and which phrases are sub-headings (and what level of sub-heading they are). Depending on the document and the reader and your interpretation of what you hear, it’s not impossible to get right, but it requires concentration.

A computer doesn’t have that benefit. While some AI may be able to differentiate between a header and a body paragraph, the point is that no computer or human has to reverse engineer the structure of your document. You can make it explicit with the Style menu in your word processor.

The Style menu is presented as options that quickly change the appearance of text. Set a top-level heading to Heading 1 and it becomes big and bold, the obvious title of your document. Set a sub-heading to Heading 2 and it becomes less big and bold, an obvious delimiter between sections of text. It makes your document easier to read, but just as importantly it changes the markup, or how your document is encoded.

Depending on what word processor you use, the markup might be XML or LaTeX or HTML or something else entirely. Whatever format it is, that markup is what the computer reads. Markup tells the computer your intent, and it ensures that your intent carries over from one file type to another. The computer doesn't have to guess whether that lone short sentence is meant to be a section heading or a paragraph. The markup makes it clear.

Avoid direct formatting

When you're not used to writing with style, it can be difficult to break the habit of just making the document look the way you want it to look without using the Style menu. You might, by old habits, set a block of code to a monospace font without using a Preformatted Text or Code style, or you might create a quick section header by adjusting the boldness and size of a sub-heading.

I think of these as "rogue" style changes. They're quick sleight-of-hand tricks you do to get the job done. The result appears to be the same, so what's the harm?

There is no harm in rogue style changes until you export your text to a new format. This direct formatting doesn't carry over because there were no styles defined. All text, whether it's code or heading or the body of the document looks exactly the same and, worse yet, is treated as equal by the computer, post-processors, and screen readers.

If you catch yourself trying to adjust your text so it looks good, take a moment to ask whether it's something you can do as a style instead.

Think in document blocks

One way to train yourself to use styles is to think in terms of block elements. If you've hit Return or Enter on your keyboard while typing, you've probably just changed into a new, imaginary “block” element. Each paragraph is a block. Each heading is a block. A code sample is a block.

Block elements are as easy for a computer to identify as they are for you, and so they’re easy to style. Your word processor probably already provides an array of definitions for common block elements, including a paragraph style, several heading styles, code block styles, and maybe more. Use these existing styles to customize what your block elements look like, and then update the style so that your customization becomes the new default.

Keep your custom styles updated

In LibreOffice, the main Style menu is located in the top left toolbar. It’s a drop-down menu that can apply a style to text, and which can be updated to match existing text. Here’s how to create a new style in three easy steps, using a sub-heading as an example:

1. Type your sub-heading, and set it to Heading 2 from the Style menu.
2. Customize the sub-heading using the usual text font, size, and colour tools.
3. Select your styled sub-heading, and click the Style menu again. Click the disclosure triangle to the right of Heading 2 and select Update to Match Selection

Within this document, all Heading 2 elements now match the style you created, and any new ones you create will match it, as well. It’s that easy.

Separate style from content

The separation of style and content has been a vital principle in modern writing for decades now, but it’s not something that tends to occur to us writers naturally. Writers use words to express ideas, and sometimes the way those words are laid out is just as important, at least in the writer’s view, as what the words are.

However, in reality you can’t control how your reader ingests the data contained in your words. Audio books don’t traditionally stop to describe metadata, like the appearance or page layout, of the words being read. Blind readers never see the words the way sighted readers do, but they get the same data unless that data isn’t well-structured.

In technical writing, especially, it’s important that the information you write can be parsed no matter the delivery mechanism. And structured writing can benefit your

SEO. For example, search engines can classify structured writing, and it's thanks to semantics like this that the Internet knows a business's phone number from its post code.

What is style?

We live in a golden age of word processing. You have your choice of office suite, including word processor, spreadsheet, page layout, and diagrams and other drawing tools. You can write a document in one format, and export it as PDF or EPUB or HTML or other document types. Leveraging styles may not make the writing easier, but it can make the logistics trivial. The "invisible" semantics of style helps you convey information that's easier for both your reader and computers to parse.

Seth Kenlon is a Unix and Linux geek, open source enthusiast, and tabletop gamer. Between gigs in the film industry and the tech industry (not necessarily exclusive of one another) he likes to design games and hack on Java or Lua code (also not necessarily exclusive of one another).

Getting started with web accessibility (Shareen Mann)

Build accessibility into your projects to ensure everyone can access it. Start with alt text.

If you're new to the idea of accessibility in web design, or if you're looking to enhance your existing understanding of the subject, the World Wide Web Consortium (W3C) has done fantastic work developing guidelines through its Web Accessibility Initiative (WAI).

If you're brand new to the topic, the WAI's principles of accessibility can seem a bit intimidating. But getting started with website accessibility doesn't have to be complicated, and a super easy place to start is by adding alternative text ("alt text") to your graphics and other images.

Alt text is the description a screen reader will use when relaying the page to a user with a vision impairment or reading disability. If you've ever had your car or cell phone read a text to you while driving, you may have heard the emojis pronounced as words. That's alt text at work.

> "Don't forget to pick up milk smile smile smile. I'm making cookies tada clap tada."

Depending on the software you use, you may be able to add alt text to your photos before uploading them. Web content management systems generally make this pretty easy to find (often under Edit in your photo catalog). If that doesn't work for you, or if you enjoy working directly with the HTML code, it's a simple alt attribute in your `<img>` tag.

```
<img src="dog.gif" alt="My Shetland sheepdog">
```

If you want to start editing your web content for accessibility, but aren't sure how to go about it, an excellent starting point is Brenda Barron's article "How to Make Your

Website Accessible to People with Disabilities.” She covers the basics and provides some best practices, starting with an understanding of how disabled people access and interface with the Internet. This can really help form a foundation to build on as you continue to explore this topic.

Once you know what you’re doing, incorporating accessibility from the conceptualization or design stage can become second nature. You don’t have to get to that level of understanding all at once, either. Building your understanding one attribute at a time can get you there just as surely.

Shareen Mann has been writing and editing in a variety of fields for over 25 years. Shareen can be hired as a part-time freelancer, contractor, or casual employee for a wide range of writing and editing tasks. You can find her at The Write Mann: *thewritemann.com*

Passive voice can be used by you (Siobhan Climer)

You don't have to completely eliminate passive voice in your writing. Use passive voice to shift focus to what's most important.

Passive voice and active voice are two different approaches to communication. Neither is inherently bad or good. Both are tools for communication. As a writer your decisions on how you communicate should be intentional.

Voice, active or passive, is one tool you have with which to work. Choose active or passive voice in line with the goals of your communication. It doesn't matter whether you're a speech writer for the president of the United States or a copywriter for the local HVAC company, voice is a critical tool for writers.

Passive voice and active voice

Passive voice deemphasizes the subject, instead placing the emphasis on the object of the sentence.

Passive voice: *The sandwich was eaten by the boy.*

- Subject: the sandwich
- Verb: was eaten
- Agent: the boy

You could even remove the agent, "by the boy," and be left with a very ominous statement:

Passive voice: *The sandwich was eaten.*

This raises so many questions: By whom? When? And that's exactly where the strength in passive voice lies. It is vague; sometimes, as a writer, you want to be vague.

Active voice, on the other hand, puts the action and the subject at the forefront of the reader's mind:

Active voice: *The boy ate the sandwich.*

- Subject: the boy
- Verb: ate
- Object: the sandwich

The opposite of passive voice is active voice. Active voice is where the subject of the sentence does the action. There is no agent and ambiguity; it is clear who did what and to whom: The boy ate the sandwich.

Why shouldn't you use passive voice?

One of the first rules you'll learn as a writer is to eradicate passive voice. Why? Passive voice is inherently vague. It separates the action from the subject of the sentence.

In the example of the boy and the sandwich and the eating, you have four parts:

1. Subject
2. Verb
3. Object
4. Agent

If you haven't thought about parts of speech in a hot second—and only weirdos like me do this on the regular—here's your cheat sheet:

- Subject: The person or thing about whom the statement is made.
- Verb: A word that indicates a physical action, mental action, or a state of being.
- Object: A noun or pronoun affected by a verb or a preposition.
- Agent: Noun phrase or pronoun that identifies the person or thing which performs an action in a sentence.

Once you introduce an agent into your sentence, you're complicating the grammatical structure. It's less clear. It's also usually wordier and longer. And most of the time, that isn't what you want in your writing. When you use active voice, it's clear who is doing what.

Active or passive voice shifts the focus in your sentence

Active voice: *The boy ate the sandwich.*

Most of the time, we'd weigh the value of the person far higher than the sandwich. Imagine this scene: a boy enters a café and orders a sandwich between his wandering of some forlorn cityscape. He eats it. We're focused on the boy: what he tastes, how he feels, and what he's thinking.

But when you use passive voice, you take the action off of the true subject and manipulate the sentence, making what is really the object of the sentence—the sandwich—into the subject.

Passive voice: *The sandwich was eaten by the boy.*

All eyes are on this sandwich in this passive voice construction. Now, we, as readers, are empathizing with the sandwich. The focaccia loaves with mozzarella and provolone, genoa salami and pepperoncini. The way the crust is seasoned with herbs. Who made this sandwich? Why do we care if it is eaten or not? Should someone else have eaten it?

Passive voice has changed the focus of the sentence for the reader. No longer do we care about the eater; we care about what was eaten: the sandwich.

Writers (usually) prefer active voice because it's clearer

Most of the time, as a writer, you want to write with clarity. Here's the value. Here's why you should care. Here's the learning you need to know. Active voice is clearer, less wordy, more descriptive and easy to understand, and more personable than passive voice.

Most of the time, you want to be clear. And that's why most writing advice adheres to the "ban passive voice" approach. If you're placing bets, then banning passive voice would be the smart choice.

Most of the time it's about the boy. But sometimes you might want to emphasize something else.

When can you use passive voice?

Use passive voice to emphasize different ideas in your writing. Because sometimes you want or need to obscure something.

Maybe there's a fault in a product or service you'd rather not dwell on. Or maybe a geopolitical situation requires some nuance around who's involved and how. Or maybe you'd rather tell a caller what to expect as opposed to lecturing them on what you're going to do.

Whatever the reason, passive voice can be invaluable when you're trying to emphasize different aspects of an idea.

> "The passive voice is meant for sentences where you need to emphasize the target of an action or the action itself rather than who or what is performing the action." —Grammarly

When might you want to focus less on who/what performs an action? When the action itself is more important, or when you want ambiguity, or when you want to delay mentioning the agent (perhaps for effect or to create surprise).

Here's an example from Dan Broetzel on Medium where passive voice makes copy better:

Active Voice: *Sorry, system upgrade work could mean extended call wait times.*

- Subject: system upgrade work
- Verb: mean

Passive Voice: *Sorry, you may have to wait a little longer for us to answer your call at the moment, as we're upgrading our system.*

- Subject: you
- Verb: wait

The goal of this communication is to relate how you, our customer, are affected by the subject of our message, the system upgrade. Not that the system upgrade is causing longer waits. You probably don't care why there are longer waits, you just want them to end.

Yes, the second example is longer. It's wordier. It's less clear what's going on behind the scenes at our company, and that's all okay. Because the second example does a better job of empathizing with the actual character (you, the customer).

So, before sending passive voice to its grave, remember you might be relying on it sometime soon to shift perspectives and focus on your copy!

Siobhan Climer is the founder of the Lore School of Writing. "I founded Lore because I believe writers have had a cruddy time building careers. It's not an equitable road. It's not straightforward or direct. At Lore our goal is to help writers skip over the decade of painful learning. We're passing down the hard-won lessons so you can get right to the fun parts."

About the Editors

Jim Hall is an open source software advocate and technical writer. At work, Jim is CEO of Hallmentum, an IT executive consulting company that provides hands-on IT Leadership training, workshops, and coaching.

Georgia Harms is an undergraduate student at the University of Minnesota Twin Cities. She is studying technical Writing and communication and her interests are biology and ecosystems. Georgia will graduate with her BS in technical writing in spring of 2026.

We'd love to share your story! Your article can be about anything related to technical communication, such as tools, tips, how you got started, what you've learned, or other topics in technical communication. Join us at *technicallywewrite.com* to share your article idea.