Chapter 11. Powering Up Your Productivity

> Focus on being productive instead of busy.
>
> Tim Ferriss, American entrepreneur, investor, and author

You may have heard of the so-called 10× developer, the software engineer who is thought to be 10 times as productive as an average developer. Whether they truly exist is debatable, but it's undeniable that there *are* people who get tasks done faster and more efficiently than others. To become one of them, always look for ways to improve—starting with your developer toolkit, code editor, command line, and all those little utilities senior engineers seem to know about. This chapter will help you customize your development environment and become a more productive software engineer. You'll learn how to take control of your editor, how to navigate the command line with ease, and why mastering keyboard shortcuts is a superpower.

Optimizing Your Development Environment

Early developers had to make do with fairly rudimentary tools. Luckily, you have a veritable plethora of amazing, full-featured editors like IntelliJ IDEA and VS Code. But installing a tool isn't the end of your journey; it's only the beginning.

Warning

Feel free to explore different settings, but take care not to let it become a time sink. If you aren't careful, you might find yourself tinkering with fonts and themes when you really should be working. It is wise to timebox your experiments. Taken too far, customizations can also make it nearly impossible to pair or work effectively with others. Nate once tried to pair with a developer who could work only inside Vim. Using a specialized keyboard. With the Dvorak settings. It did not go well.

Know Your Development Tools

In many professions, you are responsible for your tools. There is no communal knife block in a professional kitchen; the chefs are responsible for bringing their own knives to the job. Mechanics often have their own toolbox with their preferred tools organized to their standards. In many professions, purchasing the required tools is the cost of entry to the trade. While your company may cover the cost of software licenses, you're still responsible for your toolkit, and you shouldn't be afraid to make it your own.

As a software engineer, you should take ownership of your development tools. Modern IDEs encourage you to create the perfect environment through the

selection of the right set of plug-ins. You must learn the instruments of your craft. That doesn't mean you need to know every nook and cranny and every obscure command or menu item. But you should know the things you use day in and day out like the back of your hand.

Your goal is to stay in the flow state as described by Mihaly Csikszentmihalyi in *Flow: The Psychology of Optimal Experience* (Harper & Row). *Flow states* are periods of deep concentration where you are completely absorbed in the tasks at hand. Code will seem to magically appear on the screen. Stopping every few minutes to look up a command or reach for the trackpad when you could have used a keyboard shortcut can disrupt flow and break your concentration. (You'll read more about keyboard shortcuts later in this chapter.)

> The code just kind of flows from brain to IDE and, before you realize it (since you know your IDE's key combinations by heart), you're building/testing/running.
>
> Mark Heckler, author of *Spring Boot: Up and Running*

Take time to learn more about your editor and your source code management tool of choice. You'd rather not be the person who causes the team to lose three weeks of work by inadvisedly forcing a push to the main branch.

Modern IDEs boast immense capabilities, making it nearly impossible to stumble upon all their functionalities. A more effective approach is to commit to learning a new feature weekly. For example, as we've mentioned in Chapter 2, JetBrains IntelliJ IDEA offers a helpful 'tip of the day' on launch. This is a great way to discover shortcuts, tools, refactoring, debugging, and plugins you might not have encountered on your own. Look for comparable features in other IDEs and learn something new!

Build Your Own Lightsaber

When you get behind the wheel of a car, you adjust the seat, the steering wheel, and the mirrors to fit you; using the proverbial factory defaults wouldn't be comfortable (or safe). In the same vein, you should customize your development environment to suit *your* unique needs. Your tools have an array of settings, so you owe it to yourself to spend some time optimizing them for your preferences. Think of it as building your own lightsaber; you want your tools to fit your hands, your personality, and your needs. Never be afraid to make your environment your own.

You will spend hours upon hours staring at code in your editor. While the default font might be great, your eyes will thank you for taking some time to explore your options. A different font not only might look better to you but also can offer another way to customize your environment to reflect your personality.

There is no shortage of excellent monospace options to pick from. A quick search of the internet will surface many lists of best monospaced fonts for coding and programming. Perform an experiment: try one out for a day or two and see what you think. Don't be afraid to try another. There's also nothing magical about 11- or 12-point font; if you find that something else fits your eye better, change it!

Most editors have countless themes that alter the appearance of nearly everything in the tool. Many developers love dark mode, as staring at a very bright display all day can be hard on your eyes. That said, dark themes may or may not fit your eye, so try a few until you find a good fit.[1] Some operating systems will automatically adjust the display brightness to your surroundings and/or the time of day which,

again, can reduce eye strain. Regardless of your preference, you should modify themes and find what works best for you.

Don’t stop with your editor either. Odds are your operating system of choice can be tuned to your exact needs; the default settings may not be right for you, so tweak them to suit your preferences. Try putting the dock in a different place, change the magnification, or update the background image to something that resonates for you.

An entire ecosystem of free or low-cost utilities exist to unlock any number of features to customize your working environment. From clipboard utilities to more granular sound control to apps that let you rearrange your menu bar, it’s highly likely that removing some friction from your day is just a short install away. If you see something you don’t recognize in a colleague’s menu bar, ask them about it; such a discovery could save you hours a month just by asking. An internet search will lead to any number of tools that you can add to your utility belt!

Don’t forget that you can customize your physical environment too. Adjust your chair and monitor to a comfortable height that puts your body in a neutral position. Some people prefer a standing desk, and there are many excellent options that allow you to change your position throughout the day.[2] If your company has an ergonomics department, ask it to perform an assessment of your workspace and don’t be afraid to request a better chair or an adjustable desk. If you work from home, your company may offer a stipend to purchase equipment, but if it doesn’t, you should invest in a quality desk and chair; your back will thank you.[3]

Eye strain is a real thing, and staring at screens all day can be problematic. Take breaks throughout the day to focus at distances other than your monitor. If you wear glasses, make sure your optometrist knows you spend significant time staring at screens. From different coatings to special lenses, there are many options that can ease eye strain. Some developers even have a separate set of frames with a modified prescription for use when coding.

Buy a Better Keyboard

Speaking of keyboards, don’t be afraid to purchase a better one. Odds are, whatever your company bulk-purchased isn’t right for you. An ergonomic split keyboard such as the infinitely customizable ErgoDox EZ can make a huge difference in your daily life. With the ability to choose the exact keyswitch you like, you should use whatever works best for you. Many companies will pick up some or all of the cost if you ask.[4]

Leverage the Power of the Command Line

Before the advent of windowing systems and “what you see is what you get” computer environments, computers were purely text-based systems that required arcane, cryptic, and often unforgiving commands. You should absolutely take advantage of today’s modern computing systems, but as a software engineer, you should be comfortable with the command-line interface (CLI) as well.

While the command line can be intimidating at first, leveraging the CLI quickly becomes a superpower and a force multiplier. Many common tasks can be done far more efficiently via a few keystrokes. Spend a few minutes a week trying out new commands.

Tip

Don’t forget, the manual is baked into the command line! If you’re not sure what something does, just look it up with the `man` command.

Shell commands are your friend. You are probably familiar with things like `cd`, `pwd`, `ls`, and `mkdir`, but there are so many more at your disposal. From the command line, you can easily launch an application via `open`. Commands like `cat`, `cut`, `grep`, and `pbcopy` can save you valuable time. Of course, these can all be combined using the pipe (|) command. Pipe allows you to string together simple, single-purpose tools to get more complicated outputs. For example, have you ever wanted the URL for a Git repo? `cat .git/config | grep url | cut -f2 -d= | pbcopy` to the rescue.

Have you ever wanted to make your computer talk to you? Try this:

```
say I can’t let you do that Nate.
```

You can also use other CLI tools. Have you ever wanted `cat` but with syntax highlighting? Give `bat` a try. Would you like nice diffs? Take a look at `diff so fancy`. Do you spend a lot of time staring at JSON? Consider adding `fx` to your toolbelt. Would you like `ls` but with better colors? Try installing `exa`.

Many developers take the time to create shell aliases for things they do often at the command line. By simply adding a line or two to your shell’s configuration file (for example, `~/.bashrc` for bash or `~/.zshrc` for Z shell), you can save yourself countless keystrokes. These are very helpful until you’re working on a machine that doesn’t have your specific modifications on them, so use them wisely. Take advantage of them for things you do often or things that are lengthy to type. For example:

```
alias k=kubectl

alias d='docker'

alias dc='docker compose'
```

Don’t forget about shell history either. The shell history is one of the most underrated aspects of working at the command line, enabling you to scroll up through your recent commands and press Tab to autocomplete anything you’ve done recently. The `history` command will give you a list of what you’ve previously entered, and Ctrl-R will allow you to perform a reverse search of your command history.

All of that is tunable: you could increase history size or even ignore certain commands. There are any number of shell hacks that you can take advantage of. Add in a fuzzy finder to your setup and there’s a decent chance you may never type a command more than once again!

Finally, don’t limit yourself to your operating system’s default shell. There are many excellent options available to you, such as Oh My Zsh. In addition to many helpful built-in functions, Oh My Zsh has thousands of plug-ins, themes, and helpers.

Harness the Power of Keyboard Shortcuts

Modern operating systems leverage the power of the pointer, be that a mouse, a trackpad, or your own finger. While that has certainly made computers far more accessible, your goal should be to keep your hands on the home row of your

keyboard.[5] Learn your keyboard shortcuts. They're not only faster and more efficient but also save you the time of constantly moving your hand to a mouse or a trackpad. Using shortcuts also reduces strain on your wrists and hands.

Tip

If you don't touch type, you should learn to do so. Seriously. The smartphone taught an entire generation how to type with their thumbs, and while that is useful when conversing with your friends, as a developer you should use all of your digits when writing code. Touch typing is faster and easier on your hands, eyes, and neck. Odds are you didn't learn typing in school, but there are many online options that have gamified the process. Please learn to touch type; you'll thank us, we promise.

Mastering keyboard shortcuts separates beginners from seasoned pros, and they quickly become a force multiplier. While you should learn the keyboard shortcuts for your editor of choice, you should also learn them for your operating system.

Some shortcuts are so hardwired that your hands just do them without you even thinking. Most computer users are well acquainted with options like Cmd-C and Cmd-V, but far more are at your disposal. How many browser windows do you have open right now? If you want to see all the windows of the frontmost app, Ctrl-down arrow will do the trick. Need to hide the frontmost application? Cmd-H to the rescue. In IntelliJ, if you pick any two files and hit Cmd-D, you will see a diff of those two files. This works on JARs and ZIPs as well.

Many applications and operating systems work with the Emacs key bindings, many of which are incredibly helpful, especially when it comes to navigating text (like code files, for example). Moving forward or backwards by a single character can be accomplished with Ctrl-F and Ctrl-B. You can also move forward or backward by an entire word by using a Meta-F and Meta-B, respectively.[6] You can move to the beginning or end of a line via Ctrl-E and Ctrl-A. A few minutes learning a handful of Emacs key bindings could save you significant time.

Emacs Versus vi

If you really want to start a battle royale in a project room, debate Emacs versus vi. Seriously, it makes tabs versus spaces seem quaint by comparison. Suffice it to say, "Emacs is a great operating system, lacking only a decent editor," while "Vim is a great editor, utterly lacking a decent window manager." You've been warned.

That said, learning vi basics can be incredibly useful. In some instances, it may be the only editor you have available. And again, learning Emacs key bindings can make you a file-navigating savant.

Even the most used keyboard shortcut combination, Ctrl-C, Ctrl-V, can be optimized. Consider using a tool like Pastebot, as shown in Figure 11-1, which gives you access to your paste history. How often have you needed to copy five things from one application to another? Did you enjoy flipping between windows? Probably not. A tool like Pastebot allows you to batch your copy and pastes, which can save you considerable time. Pastebot also allows you to go back in your history (be careful with passwords and other sensitive information) to repaste something you copied earlier in the day.

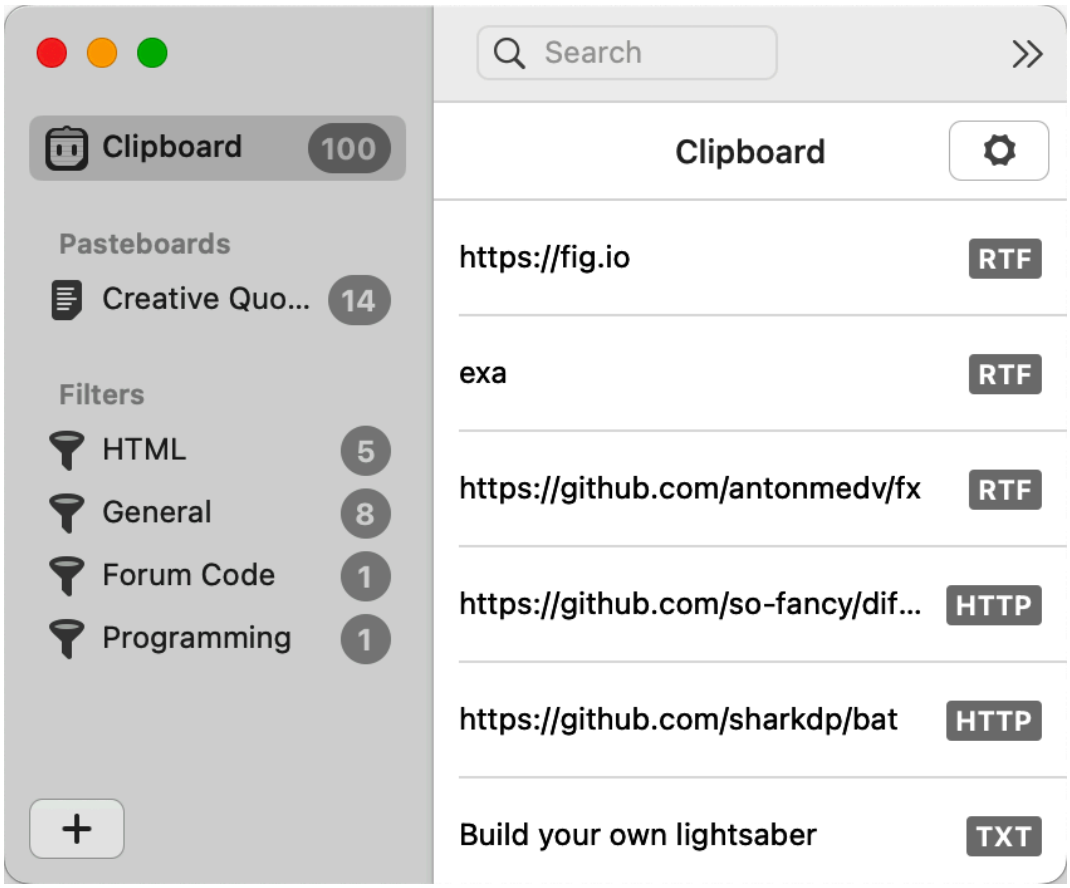

Figure 11-1. Pastebot gives you access to your clipboard history

Text expanders are another excellent productivity tool. Standalone tools like TextExpander can massively boost productivity. Many operating systems also have built-in text replacement (see Figure 11-2) that is completely tunable by you. Again, little savings add up over time.

As you can see, your environment is ripe with customization options. Don't settle for the defaults: you shouldn't bend to your tools; they should conform to you. The time you take tweaking your environment is an investment in your productivity.

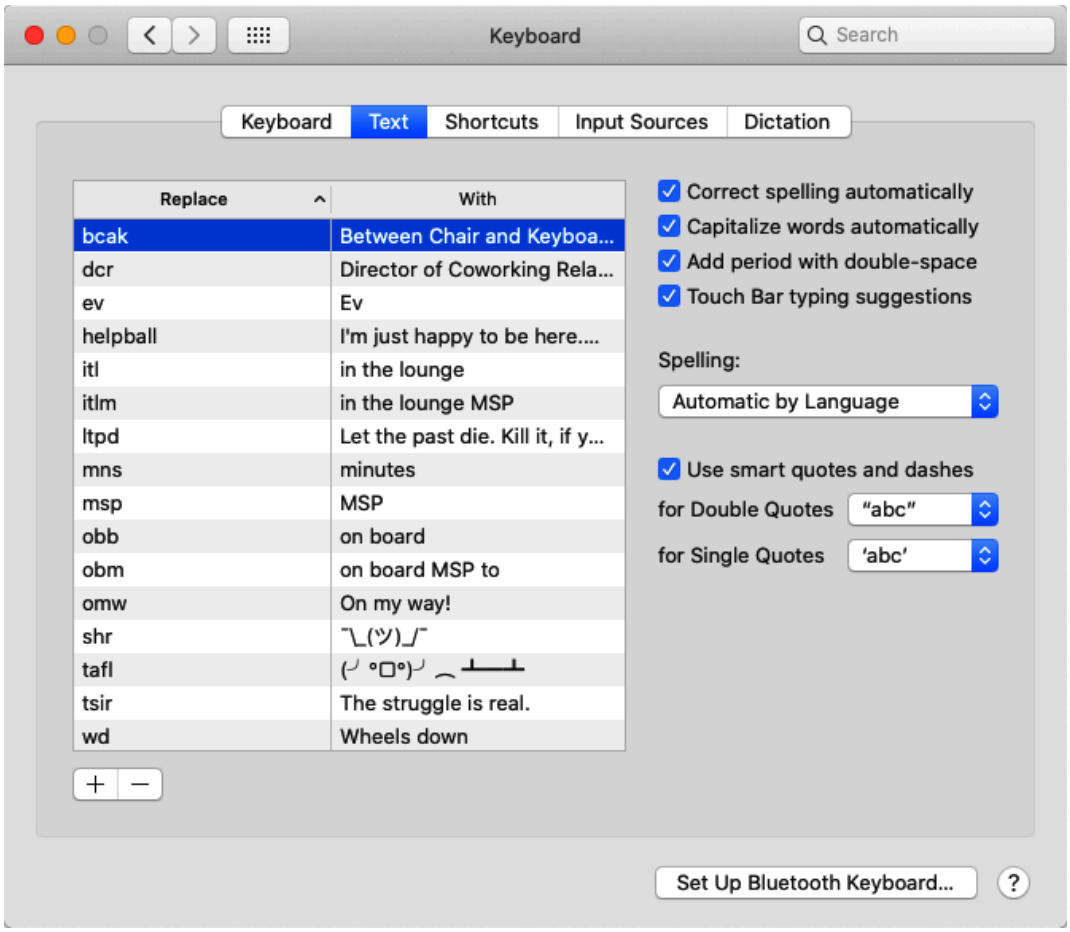

Figure 11-2. Text replacement options can improve your productivity

Strategic Automation

You spend all day writing code for others, so don’t be afraid to write some to help yourself. A few lines of code could ultimately save you hours and hours of work, and it also provides a really good excuse to play around with other languages and techniques. Many developers have a love-hate relationship with regular expressions, but learning even just a little bit of regex can be incredibly helpful.

Never forget, in computer science there are only three numbers. There’s something you do zero times, something you do once and only once, and then *n*. If you do something more than once, you should expect you’re going to do it countless times. You should consider taking the time to automate anything you do more than once.

Warning

The pursuit of automation can be a dangerous time sink. Sometimes, writing the code can take longer than the task itself. If a task really is just a one-off you’re unlikely to repeat, automating it away isn’t time well spent. Consider waiting until you’ve done something a handful of times before you try to automate it out of existence, and even then, be sure to timebox your effort. At the end of the day, use your best judgment.

To this end, write helper scripts. Thirty lines of Python could save you hours. The moral of the story is automation is your friend; let the computer do more work for you. And if you don’t feel like writing the script yourself, you could always let AI

write one for you! You should still read and test the code, but the barrier to automation is lower than ever.

The Perpetual Pursuit of Productive Habits

The sheer volume of shortcuts, utilities, and commands is practically endless, so learning them can seem overwhelming. Don't attempt to pick them all up in a month or two; it is an ongoing process. The goal is to improve a little bit each day; try to learn a new thing every week. This method is more approachable and sustainable, and it also makes it more likely you'll retain what you're learning.

Collaborative Learning

One of the best sources of tips and tricks is your coworkers. Get in a habit of asking your teammates about their productivity practices. If a colleague uses a tool, command, or shortcut that you haven't seen before, ask them about it. Don't feel bad that you aren't familiar with it. By the same token, if you see a teammate doing something the hard way, take the time to show them the simpler approach. To reinforce the concept, take the time to repeat it three or four times before moving on. If your coworker doesn't know how to get started with one of your favorite tools, help them get up and running. Doing so not only further ingrains your knowledge, but also enables you to have a familiar environment at your disposal.[7]

Commit a few minutes each week to learning something new about your tools, and consider sharing what you've learned with your friends and colleagues. If you are pair programming and your pair is using a tool or utility that's new to you, ask them to teach it to you. If your pair doesn't use a shortcut that you know, teach it to them. Consider having a semi-regular lunch-and-learn session as a forum for people to share cool things they use; don't assume everyone knows all the things you know. They probably don't. By the same token, don't skip that session or webinar about a tool you know well; don't assume there's nothing new for you to learn.

This approach extends to your language of choice. Languages evolve,[8] so your coding has to as well. When a new version drops, study the release notes and take the time to explore new features. Teach others what you've learned and be aware of opportunities to put those new approaches into practice. Lead a study group: the see one, do one, teach one approach is effective.

Personal Knowledge Management

Technology changes at an incredibly rapid pace, meaning you will be inundated with information. From podcasts to blog posts to videos to tutorials, it behooves you to have a personal knowledge management strategy. It needn't be complex or involve dedicated tooling, but you should have a repository where you can stash interesting bits of information for later use. You need to be prepared to capture and organize the flow of information in your world. Nothing is quite as frustrating as *knowing* you had the exact answer to a question, but you just don't remember where it is.

Take notes. You *will* run into that problem again. So will a teammate. Write it down. You won't remember for next time. Offload it from your brain—external

storage for the win. Doesn't matter how, doesn't matter if it's formal. A binder clip with random scraps of paper? That works too, though electronic variations are definitely easier to search, and paper can be more challenging to share the results with colleagues.

Some developers keep a note or document on their laptop. Others prefer a simple notebook like those from Field Notes or Moleskine along with a pocket-sized pen like the Mark Two or Space Pen. Many opt for a more technical solution like Org Mode, Evernote, or Notion. Consider as well how you want to handle the various web-based resources you will unearth throughout the day. Syncing your (organized) bookmarks across devices is one option, but you can also use dedicated bookmark managers like Pinboard.

Notion

Dan here. As a software engineer and someone who loves staying organized, I have experimented with numerous knowledge management tools, and Notion is my favorite. I love it because it gives me the flexibility to use it as little or as much as I need. Here are just some ways that I use Notion as a software developer:

Learning
: When learning a new language, framework, or tool, I consolidate all my research in Notion. By combining notes, code snippets, images, and resources, I accelerate my learning process.

Documentation
: For documenting new processes or features, I start in Notion. Its free-form editor supports Markdown while providing a real-time preview, enhancing my documentation workflow.

Checklists
: I store various checklists in Notion, such as my code review checklist. It helps me ensure I'm covering all necessary points during a review.

Note taking
: Notion serves as my central hub for notes, allowing me to structure them in a way that works best for me. The mobile app keeps my notes accessible on the go, while the Notion Clipper browser extension helps me save items for later reference.

The real power of Notion lies in its versatility and customizability. It enables me to create a personalized workspace that adapts to my unique needs and workflows. Notion serves as my second brain, and I couldn't imagine life as an engineer without it.

That said, there's no shortage of tools in this space, and while Notion is a vital part of my workflow, your mileage may vary. You should use what works best for you.

Attending conferences is an invaluable way to learn new skills and hone existing ones. But you'll want to capture what you've learned for future study. Many conferences post videos of the presentations after the event. Take some time to identify the talks you're most interested in and curating your own playlist of videos for simpler playback or recall later. Some developers will even treat these talks like a podcast, queuing them up to listen when they take a walk, work out, or are commuting. You may even be able to listen at greater than 1.0 speed, allowing you to consume more content faster.

Whatever approach you take, make sure it isn't locked to just one device; it should be portable and work in a mobile context. You never know when the muse will

strike, giving you a key insight for a problem at work—be prepared to capture it. You never know when a friend will recommend an amazing book or podcast—be prepared to capture it.

Keep It Simple

Nate here. There's nothing wrong with simple. I had a colleague early in my career who had a bunch of pieces of scrap paper held together with a binder clip. He'd scrawled various commands, tips, instructions, and other ephemera gathered over the years. When you asked him a question, there was a high likelihood he'd page through his scrap paper notes and find the exact bit of information you needed. Although he had a simple knowledge management system, it worked for him, and that's what matters.

Wrapping It Up

A good software engineer takes control of their toolkit. Just as you'd spend a few minutes adjusting the seat, mirrors, steering wheel, and temperature in a new car, your development environment should fit you and your needs. Don't just accept the default settings. Make your editor and your operating system your own. Power users use power tools: keyboard shortcuts and the command line can be force multipliers for you. You'd be amazed at how much a free or low-cost utility can change your day, so don't be afraid of trying out something new.

Keeping track of all these shortcuts and utilities can be overwhelming, so don't rely on memorization. Hone your knowledge management practice as you grow your toolkit. Learn from, and teach, your colleagues.

Putting It into Practice

Experiment with new ways to make yourself more productive but take care not to let your experiments become time sinks. Take a repetitive task and see if you can automate away the toil. Consider evolving an existing automation to cover something more significant. Share what you learn with your colleagues.

Commit to learning one new keyboard shortcut a week for the next few months.

Take 15 minutes on a Friday afternoon to peruse the documentation for your favorite editor. What did you learn that you didn't know before? Turn on the tip of the day feature for your editor or terminal: what feature surprised you?

Examine your knowledge management approach: is it working for you? If not, try something different. It doesn't need to be complex or involve fancy tools; it just has to work for you.

When you learn a new command, write it down! Leverage an external brain.

Additional Resources

- *Flow: The Psychology of Optimal Experience* by Mihaly Csikszentmihalyi (Harper & Row, 1990)
- *The Productive Programmer* by Neal Ford (O'Reilly, 2008)

- *Building a Second Brain: A Proven Method to Organize Your Digital Life and Unlock Your Creative Potential* by Tiago Forte (Simon Element / Simon Acumen, 2022)

- *The Passionate Programmer*, 2nd Edition, by Chad Fowler (Pragmatic Press, 2009)

[1] Once you've found a great theme, you may want to change a thing or three, you don't have to settle for off-the-rack settings.

[2] If you find yourself dealing with strange aches and pains, relief could be as simple as changing from sitting to standing throughout the day.

[3] While the price of a high-end chair and desk may give you sticker shock, quality gear lasts a long time.

[4] Don't wait for the pain of a repetitive strain injury. If you are facing issues, a better keyboard can work wonders.

[5] *Home row* is the row of keys your fingers naturally rest on when touch typing, and most keyboards include a tactile marker for where your index fingers should rest.

[6] Meta is usually mapped to Alt, Caps Lock, or Escape, depending on your operating system.

[7] Teams may decide on a basic level of tool consistency to ensure ease of pairing.

[8] Many languages and frameworks encourage community contributions. Don't be afraid to shape the future of your environment by getting involved.