



Prevent cognitive debt by manually retyping LLM-generated code

2 Aug 2026 at 2:53PM IST

Despite [what I said in April](#), I'm still using coding assistants on my personal projects.

Using them to one-shot entire features leaves me unsatisfied and disoriented, but I do enjoy using them to fast-forward through the boring parts of my projects.

However, allowing my coding assistant to roam free in my projects leaves me with a colossal amount of cognitive debt. I might hate the idea of poring over the Django documentation to figure out how to add tagging to my website, but I still fundamentally want to understand how it works. Just because a problem is boring doesn't mean I want to fully offload my understanding of the solution to a machine.

Of course, I could review every single line of code the LLM produces. That's what most developers are expected to do in this cursed year of 2026. Robots raise PRs, humans review them. It's a brave new world.

But I don't enjoy reviewing AI-generated PRs. Poring over hundreds of lines of overly-defensive, badly-commented, subtly incorrect code is not fun. I might grudgingly do it for an employer—while making sure said employer becomes an ex-employer as soon as possible—but I'm sure as hell not doing it for my

personal projects. Personal projects must be fun above all else. The joy of working on personal projects comes from the process, not from the outcome.

So what's a boy to do? How do I offload the boring work to LLMs without ceding control of my own work and cognition to the slop machine?

I've come up with a solution that's grossly inefficient and perhaps slightly comical: I ask my coding assistant to generate code in the chat, then manually make all the edits myself.

I have these instructions in all the agents files in my personal projects:

I want to understand every line of code that goes into this project. Never create, edit, move, rename, or delete project files unless I explicitly ask you to do so. Instead, show me every proposed edit in the chat so I can type it in manually.

Do not run commands that modify project files, install dependencies, or change repository state unless I explicitly request that action. Instead, show me those commands in the chat so I can run them manually.

I'm an experienced developer. Do not explain syntax, APIs, programming concepts, or implementation details unless explicitly asked.

Using LLMs this way allows me to work faster than not using LLMs at all, but I'm still slower than those who are willing to allow the machine to think for them. Instead of being 10x faster, I'm probably only 2x faster. But what I lose out on in terms of speed, I gain in terms of a deeper understanding of my code.

As I manually type every single line of LLM generated code into my editor, I build up a mental model of how it works and fits into my existing codebase. If

I don't understand an API or algorithm, I can stop to look it up, or just ask the LLM to explain it.

Typing the code myself forces me to slow down, which means I'm more likely to detect hallucinations or bad design choices the LLM might have made. I can clean up the code as I go, reorganizing it, refactoring it, adding comments, and generally adapting it to my own taste.

Most importantly, this workflow allows me to build a *spatial map* of my codebase. I know where every bit of functionality lives in the codebase. When I need to make a change, I know exactly where I need to make it. It not only helps me work faster within my projects, it also makes it easier for me to better prompt and instruct the LLM in the future.

When I was learning to code as a teenager, experienced programmers would often tell me to never copy and paste code into my projects. If I was learning from a book, I was advised to copy all the examples into my computer and make sure I could run them. If I was learning from a blog post or forum answer, I was advised to type it out and adapt it to my codebase so I understood it completely.

Manually typing LLM-generated into my codebase feels like the exact same learning process. It might not be the most efficient way to work with an LLM, but I value comprehension over productivity. I've been doing this for a few months now, and it's been working well for me. I plan to continue using this workflow for as long as I can.

I fear the software industry is taking on a large amount of cognitive debt that we'll have to pay back very soon. There will come a time when we no longer understand how large parts of our digital infrastructure are put together. I might not personally be able to change the course of the entire industry, but I can at least make sure I completely understand the software I put out into the world. Anything else would be professional malpractice.

© Ankur Sethi 2025–2026

[MASTODON](#)

[BLUESKY](#)

[EMAIL](#)

[RSS FEED](#)