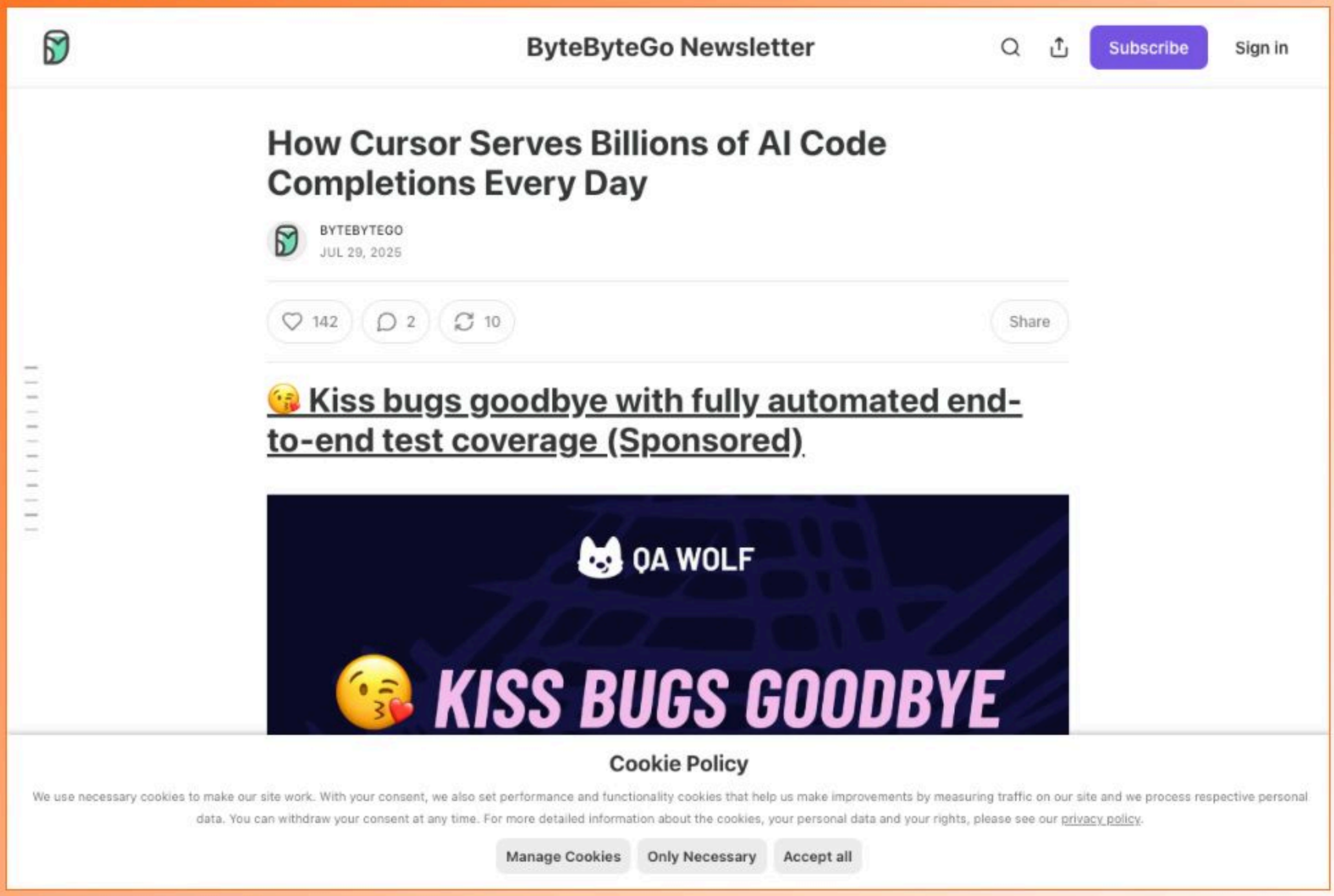
How Cursor Serves Billions of AI Code Completions Every Day

https://blog.bytebytego.com/p/how-cursor-serves-billions-of-ai

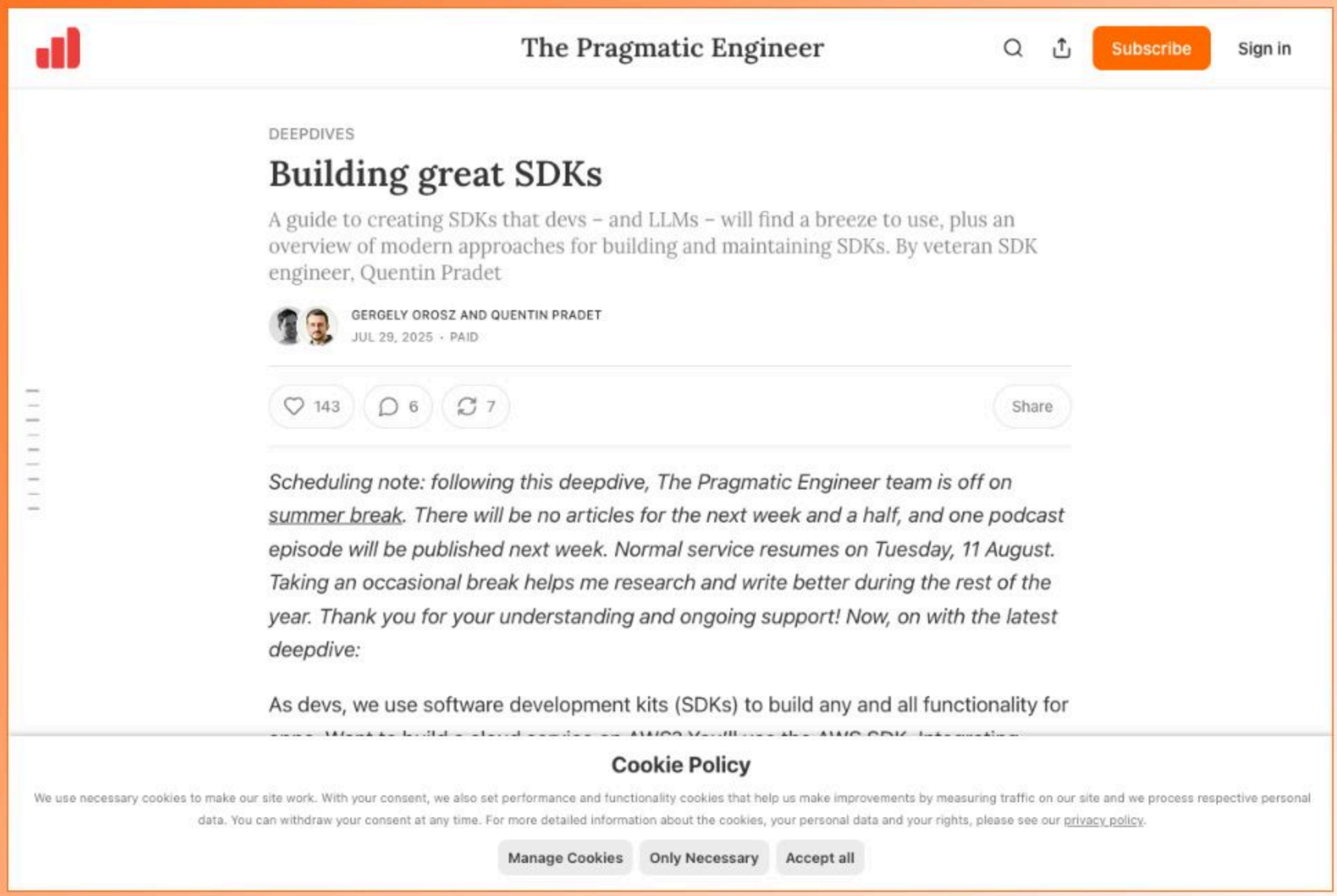


"In this article, we will take a look at the key features of Cursor, how those features work, and the infrastructure stack that powers it."

dariomac.com

Building great SDKs

https://newsletter.pragmaticengineer.com/p/building-great-sdks

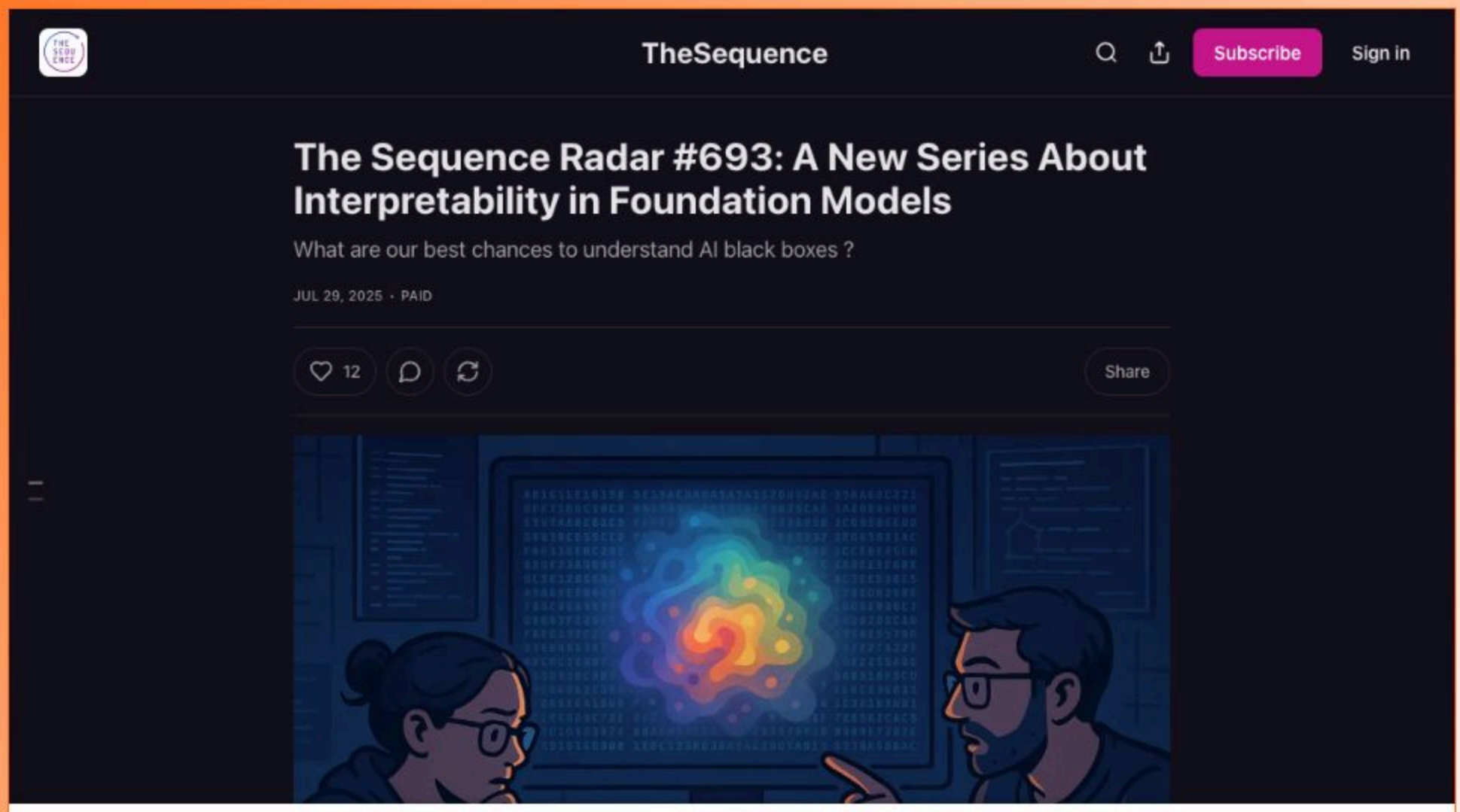


"A guide to creating SDKs that devs – and LLMs – will find a breeze to use, plus an overview of modern approaches for building and maintaining SDKs. By veteran SDK engineer, Quentin Pradet"

dariomac.com

The Sequence Radar #693: A New Series About Interpretability in Foundation Models

https://thesequence.substack.com/p/the-sequence-radar-a-new-series-about



"What are our best chances to understand AI black boxes ?"

dariomac.com

AI is eating the Internet

https://fika.bar/paoramen/ai-is-eating-the-internet-01K10JG1SHGZQHN61HPGWPXN60

AI is eating the Internet

An exploration of the Internet to come

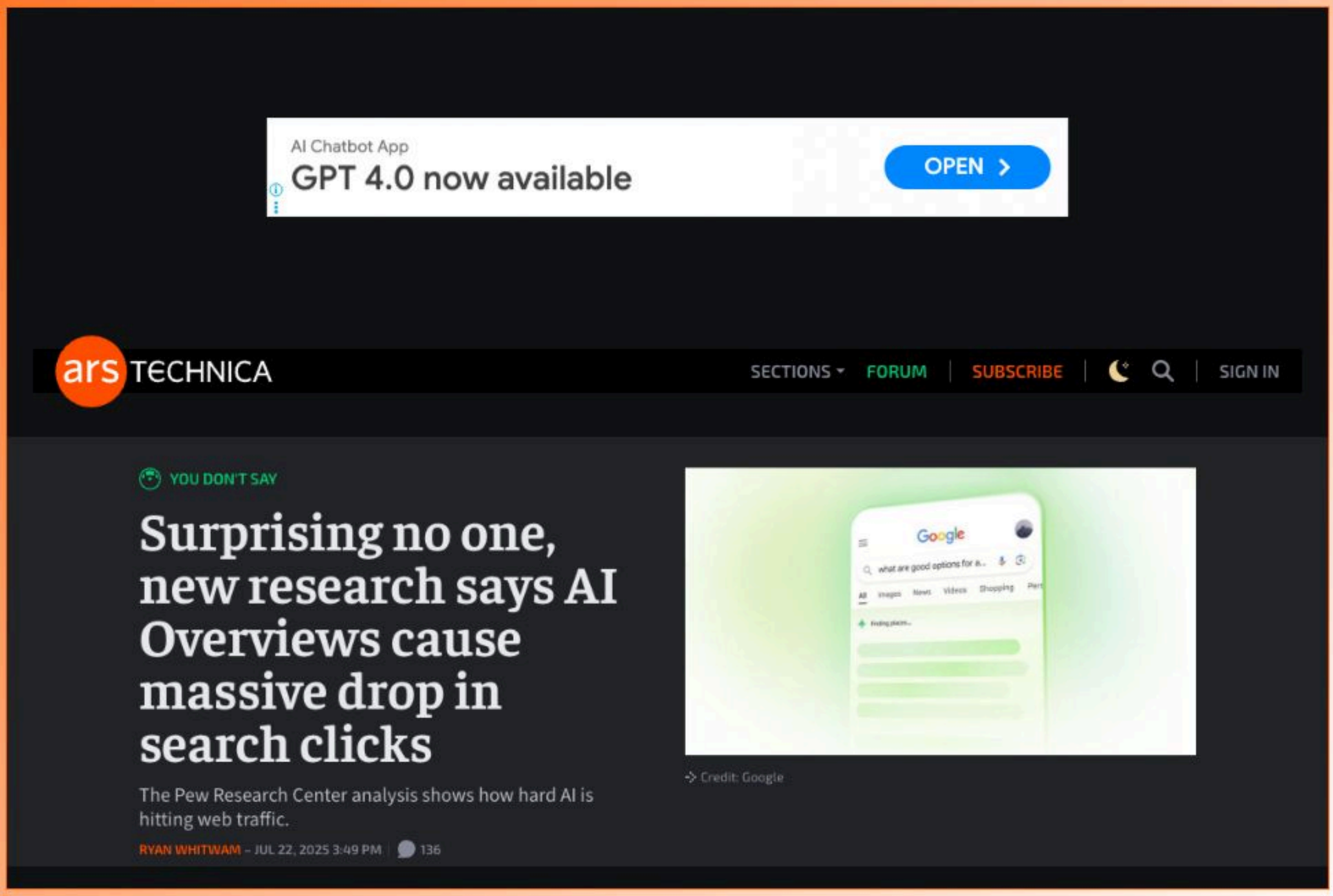
Jul 28, 2025

"You see? Another ad. We were just talking about this yesterday! How can you be so sure they're not listening to us?" – My wife, at least once a week.

"An exploration of the Internet to come"

Surprising no one, new research says AI Overviews cause massive drop in search clicks

https://arstechnica.com/ai/2025/07/research-shows-google-ai-overviews-reduce-website-clicks-by-almost-half/

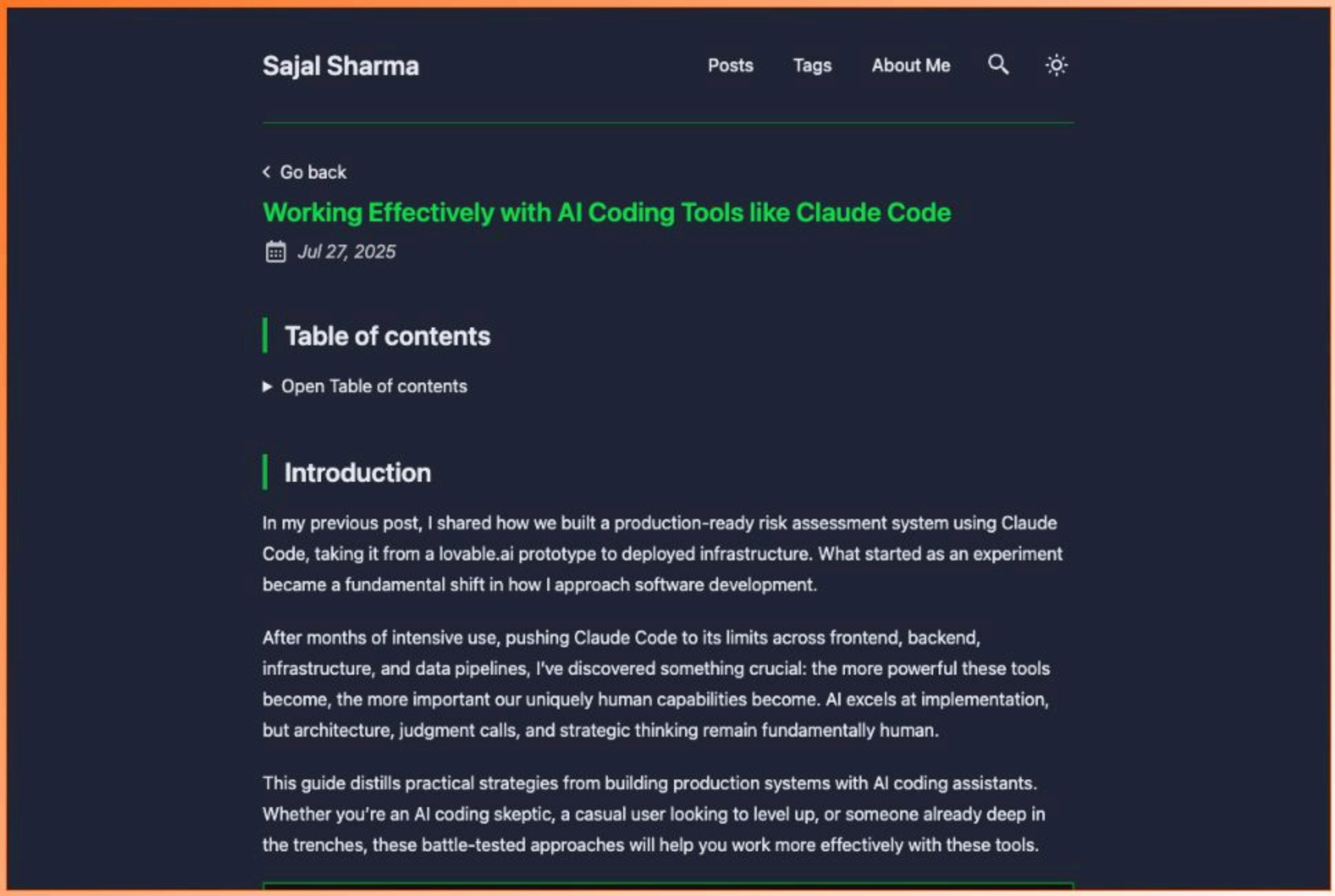


"The Pew Research Center analysis shows how hard AI is hitting web traffic."

dariomac.com

Working Effectively with AI Coding Tools like Claude Code

https://sajalsharma.com/posts/effective-ai-coding/

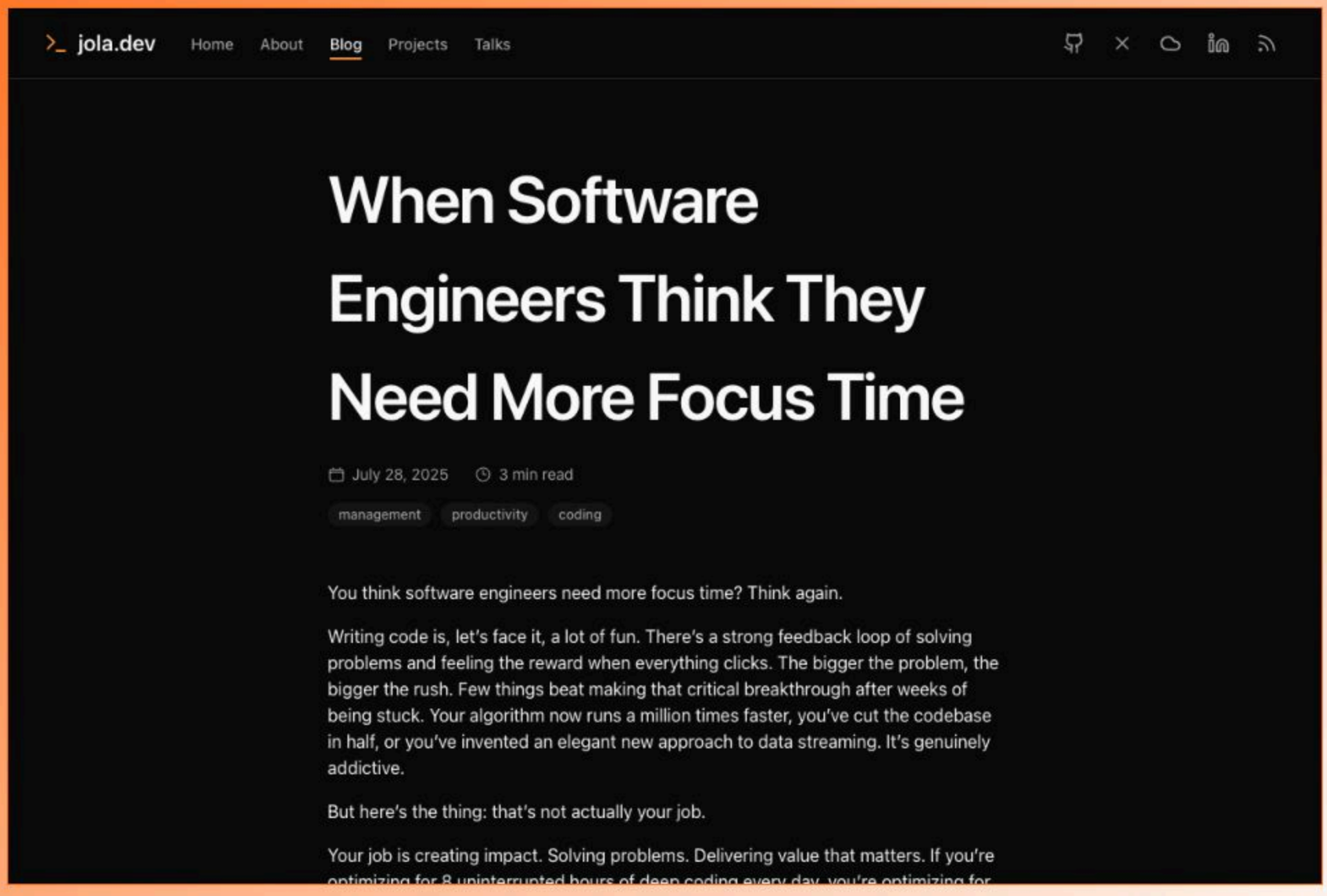


"A practical guide to working effectively with AI coding tools like Claude Code, covering mindset shifts, quality control strategies, and team collaboration workflows for modern software development."

dariomac.com

jola.dev - When Software Engineers Think They Need More Focus Time

https://jola.dev/posts/enough-focus-time

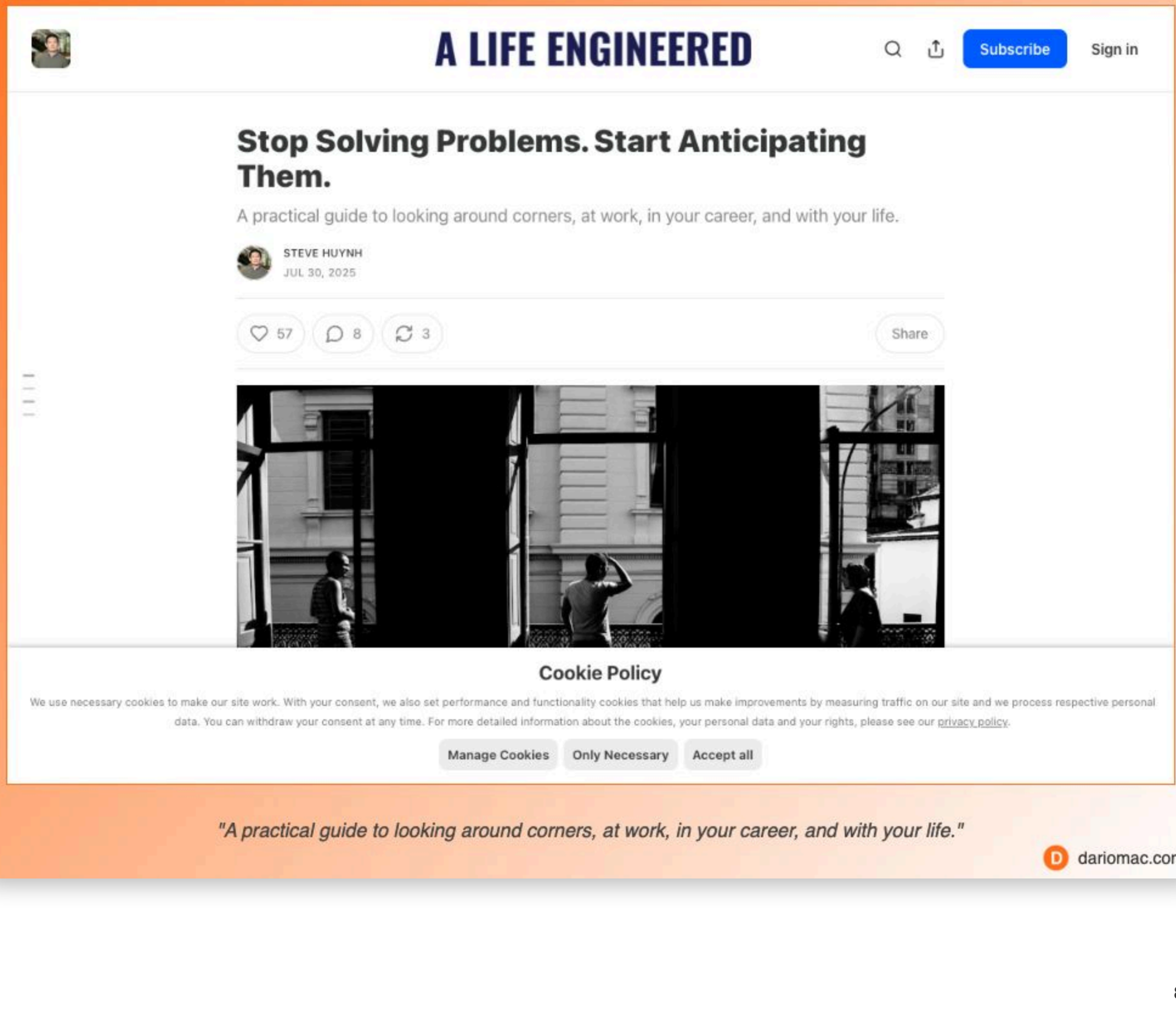


"Johanna Larsson is a software engineer, manager, writer, speaker. With over 14 years of experience in software development and management, she has worked with clients from various industries, including finance, healthcare, and education. She is passionate about creating innovative solutions that solve real-world problems. She is also an advocate for diversity and inclusion in the tech industry. She believes that technology can be a force for good and strives to make a positive impact in the world. She is committed to using her skills and expertise to create a better future for all."

dariomac.com

Stop Solving Problems. Start Anticipating Them.

https://alifeengineered.substack.com/p/stop-solving-problems-start-anticipating

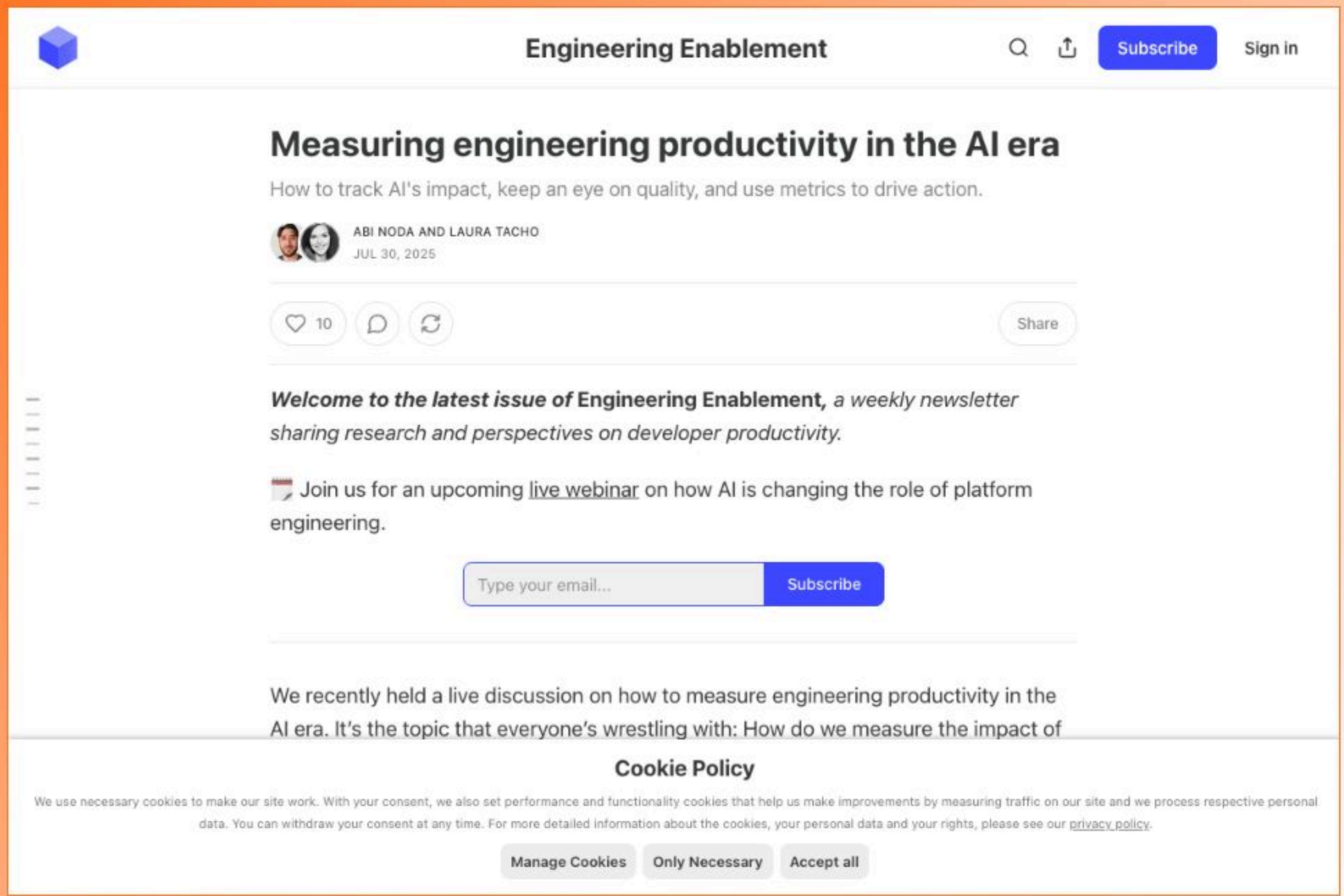


"A practical guide to looking around corners, at work, in your career, and with your life."

dariomac.com

Measuring engineering productivity in the AI era

https://newsletter.getdx.com/p/measuring-engineering-productivity-in-the-ai-era



"How to track AI's impact, keep an eye on quality, and use metrics to drive action."

dariomac.com

Context Engineering for Agents

https://rlancemartin.github.io/2025/06/23/context_engineering

Lance's Blog About

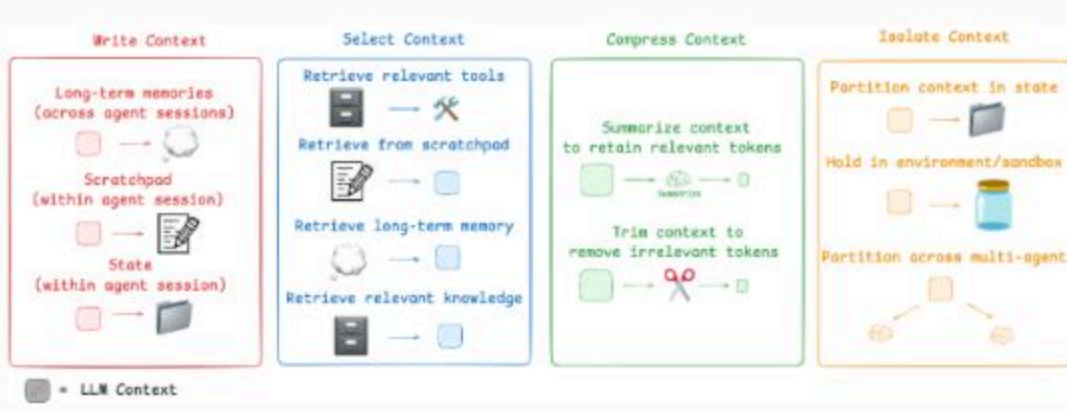
Context Engineering for Agents

Jun 23, 2025

Lance Martin

TL;DR

Agents need context to perform tasks. Context engineering is the art and science of filling the context window with just the right information at each step of an agent's trajectory. In this post, I group context engineering into a few common strategies seen across many popular agents today.



Context Engineering

As Andrej Karpathy puts it, LLMs are like a new kind of operating system. The LLM is like the CPU and its context window is like the RAM, serving as the model's working memory. Just like RAM, the LLM context window has limited capacity to handle various sources of context. And just as an operating system curates what fits into a CPU's RAM, "context engineering" plays a similar role. Karpathy summarizes this well:

"Patterns for managing agent context."

dariomac.com

AI Is Wrecking an Already Fragile Job Market for College Graduates

https://www.wsj.com/lifestyle/careers/ai-entry-level-jobs-graduates-b224d624

THE WALL STREET JOURNAL.

Access blocked.

We detected unusual activity from your device or network.

Reasons may include:

- Rapid taps or clicks
- JavaScript disabled or not working
- Automated (bot) activity on your network (IP 167.62.26.41)
- Use of developer or inspection tools

Need help?
Submit feedback.

ID: 7ba629f3-fee5-7415-10d3-70278ffb859a

"Companies have long leaned on entry-level workers to do grunt work that doubles as on-the-job training. Now ChatGPT and other bots can do many of those chores."

I Know When You're Vibe Coding

https://alexkondov.com/i-know-when-youre-vibe-coding

I Know When You're Vibe Coding

3 minute read

I shouldn't have to care about this. I don't want to care about how someone's code gets into the IDE. Whether you wrote it by hand, copied it from a forum, prompted an LLM, or ran a simulation where monkeys are given infinite time to produce the solution.

I care about what gets merged into the codebase.

When I click that "Approve" button, I've got only a few worries on my mind. Does it produce the correct outcome? Will people understand this next quarter? Will they be able to change it?

But lately, I started noticing things that immediately tell me the code is written by an LLM. No, I don't mean the repetitive comments (I'm fine with those), not even the switch statements they're so fond of.

I know the code was generated because it was written in a way no developer on the team would.

"I shouldn't have to care about this. I don't want to care about how someone's code gets into the IDE. Whether you wrote it by hand, copied it from a forum..."

dariomac.com

Vibe code is legacy code

https://blog.val.town/vibe-code

Blog

Vibe code is legacy code

WED // July 30, 2025 Steve Krouse

Despite widespread confusion, Andrej Karpathy coined "vibe coding" as a kind of AI-assisted coding where you **"forget that the code even exists."**

Legacy code

We already have a phrase for code that nobody understands: **legacy code**.

Legacy code is universally despised, and for good reason. But why? You have the code, right? Can't you figure it out from there?

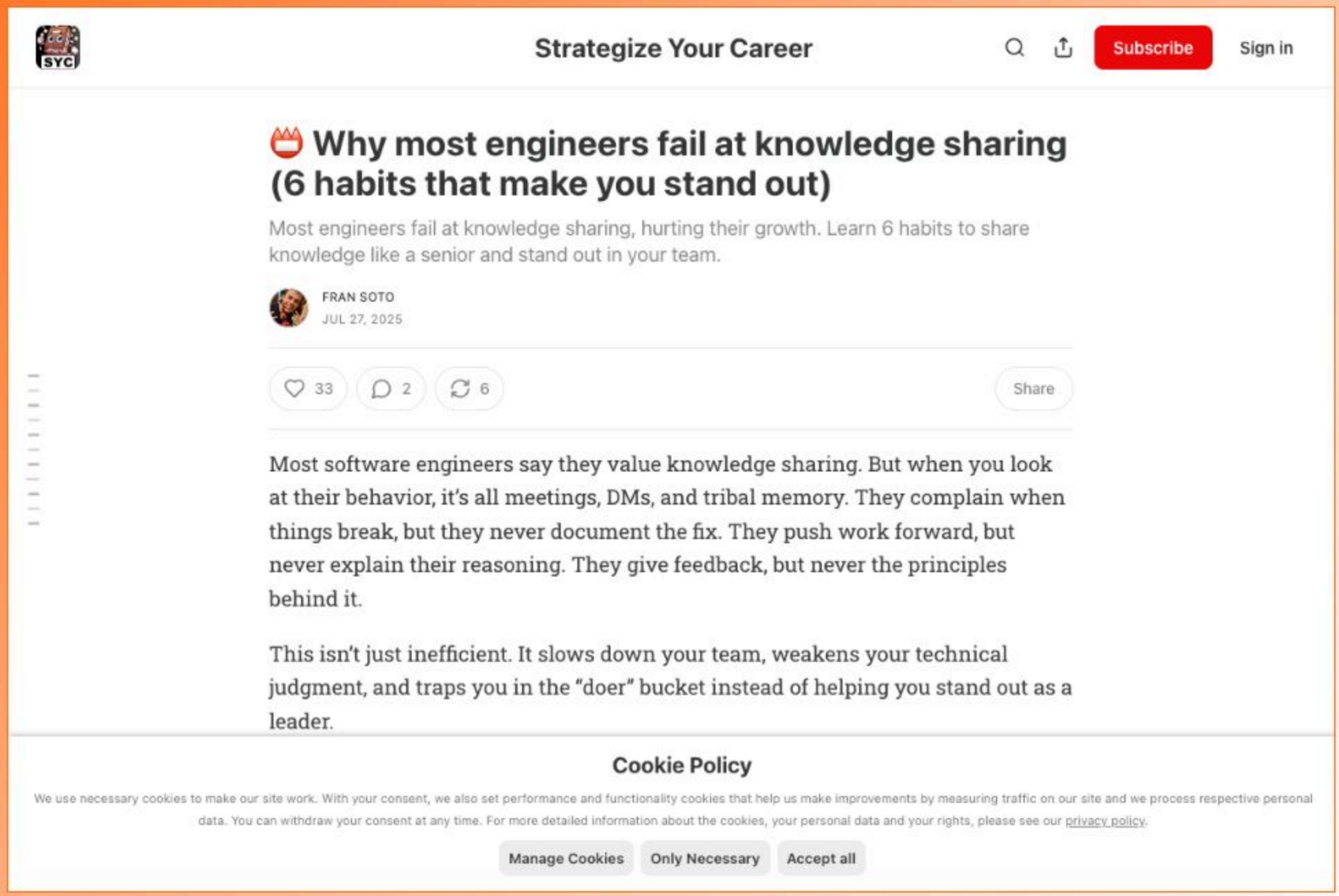
Wrong. Code that nobody understands is tech debt. It takes a lot of time to understand unfamiliar code enough to debug it, let alone introduce new features without also introducing bugs.

Programming is fundamentally *theory building*, not producing lines of code. We know this. This is why we make fun of business people who try to measure developer productivity in lines of code.

"Updates and articles from the Val Town team"

Why most engineers fail at knowledge sharing (6 habits that make you stand out)

https://strategizeyourcareer.com/p/why-most-engineers-fail-at-knowledge

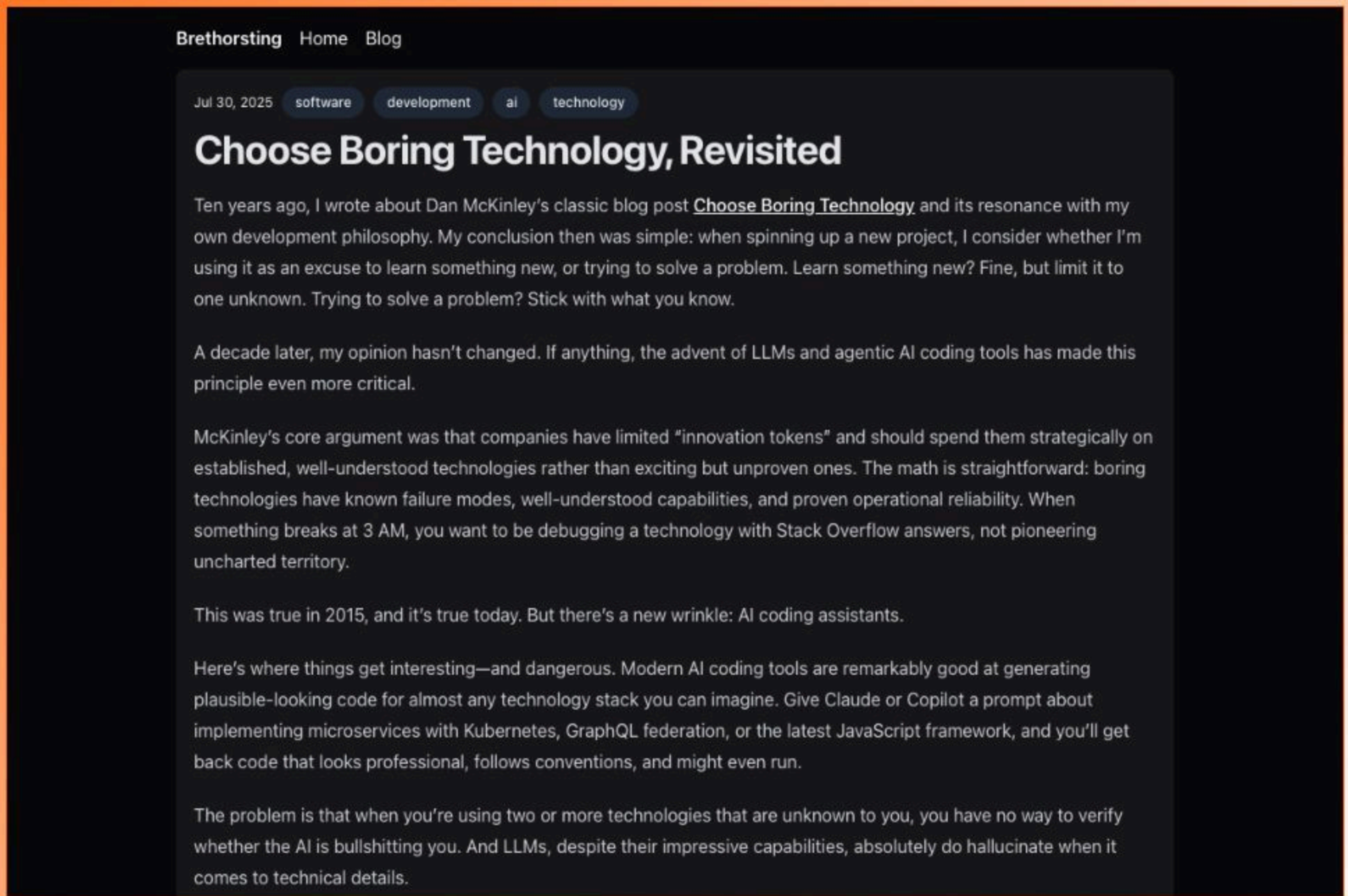


"Most engineers fail at knowledge sharing, hurting their growth. Learn 6 habits to share knowledge like a senior and stand out in your team."

dariomac.com

Choose Boring Technology, Revisited | Aaron Brethorst

https://www.brethorsting.com/blog/2025/07/choose-boring-technology,-revisited



Brethorsting Home Blog

Jul 30, 2025 software development ai technology

Choose Boring Technology, Revisited

Ten years ago, I wrote about Dan McKinley's classic blog post Choose Boring Technology and its resonance with my own development philosophy. My conclusion then was simple: when spinning up a new project, I consider whether I'm using it as an excuse to learn something new, or trying to solve a problem. Learn something new? Fine, but limit it to one unknown. Trying to solve a problem? Stick with what you know.

A decade later, my opinion hasn't changed. If anything, the advent of LLMs and agentic AI coding tools has made this principle even more critical.

McKinley's core argument was that companies have limited "innovation tokens" and should spend them strategically on established, well-understood technologies rather than exciting but unproven ones. The math is straightforward: boring technologies have known failure modes, well-understood capabilities, and proven operational reliability. When something breaks at 3 AM, you want to be debugging a technology with Stack Overflow answers, not pioneering uncharted territory.

This was true in 2015, and it's true today. But there's a new wrinkle: AI coding assistants.

Here's where things get interesting—and dangerous. Modern AI coding tools are remarkably good at generating plausible-looking code for almost any technology stack you can imagine. Give Claude or Copilot a prompt about implementing microservices with Kubernetes, GraphQL federation, or the latest JavaScript framework, and you'll get back code that looks professional, follows conventions, and might even run.

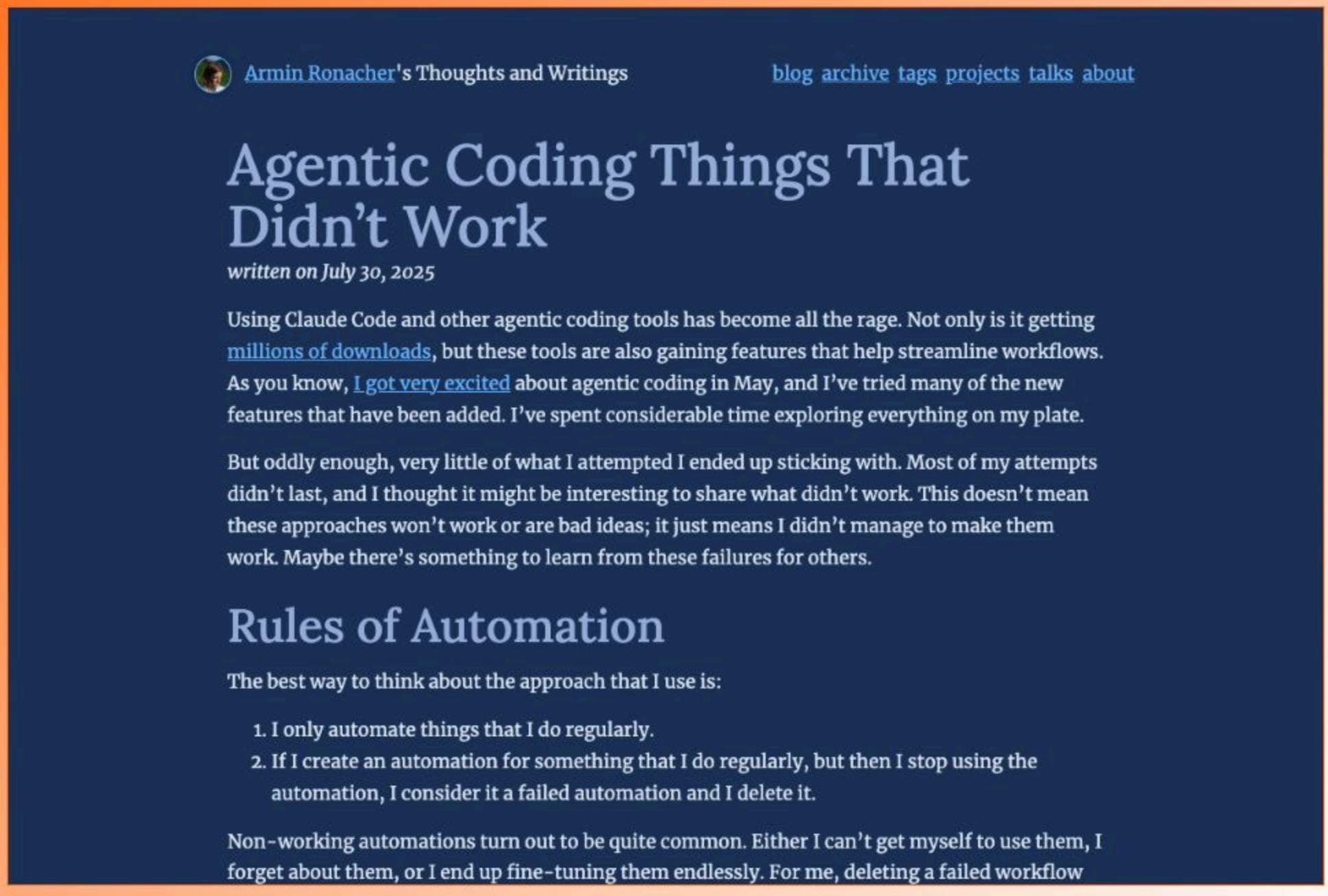
The problem is that when you're using two or more technologies that are unknown to you, you have no way to verify whether the AI is bullshitting you. And LLMs, despite their impressive capabilities, absolutely do hallucinate when it comes to technical details.

"Personal website for Aaron Brethorst - Seattleite, technology leader, photographer, transit enthusiast, erstwhile non-runner."

dariomac.com

Agentic Coding Things That Didn't Work

https://lucumr.pocoo.org/2025/7/30/things-that-didnt-work



Armin Ronacher's Thoughts and Writings

blog archive tags projects talks about

Agentic Coding Things That Didn't Work

written on July 30, 2025

Using Claude Code and other agentic coding tools has become all the rage. Not only is it getting millions of downloads, but these tools are also gaining features that help streamline workflows. As you know, I got very excited about agentic coding in May, and I've tried many of the new features that have been added. I've spent considerable time exploring everything on my plate.

But oddly enough, very little of what I attempted I ended up sticking with. Most of my attempts didn't last, and I thought it might be interesting to share what didn't work. This doesn't mean these approaches won't work or are bad ideas; it just means I didn't manage to make them work. Maybe there's something to learn from these failures for others.

Rules of Automation

The best way to think about the approach that I use is:

1. I only automate things that I do regularly.
2. If I create an automation for something that I do regularly, but then I stop using the automation, I consider it a failed automation and I delete it.

Non-working automations turn out to be quite common. Either I can't get myself to use them, I forget about them, or I end up fine-tuning them endlessly. For me, deleting a failed workflow

"Some of my attempts to make agents work better that just didn't work."

dariomac.com