RUBY/RAILS

Hardly Promethean

By Jared Norman
published September 28th, 2026

Last week I published What About Rails, a dive into DHH's Rails World keynote. Smarter people than me had interesting things to say about it:

> *You have the most powerful tool you ever had, you have become a 1000x maker, and you can't think of how to make your stack 10x better?*

José Valim poses an excellent question. I tried to find an answer.

The Bottleneck Isn't Gone

> *They're not gonna be web apps much longer. They're gonna be native applications, because the price of developing those things has gone to damn near zero.*

The move to native apps for the frontend and Rust on the backend isn't about any particular technology. It's about cost. DHH isn't the first to make this case.

Back in August, Dan Luu posted There's no reason for software to be slow anymore. In it, he argued that the cost of specialized performance work has dropped so significantly (because LLMs) that it's now cheap enough for almost anyone to do.

Luu is *much* more careful than DHH. He points out that agents overfit benchmarks, that they are poor at experimental design (without human assistance), and that the time to get a *rigorous* result hasn't dropped, just the time to get an *interesting* one.

Shortly after, Varun Gandhi posted a response of his own, titled There continue to be reasons for software to be slow. He points out the shape of the argument: X cost too much, LLMs divide the cost by a large number, so people will now do X. Gandhi argues that while this holds true for people like Luu (experts working on their own projects), those cases are rare.

Substitute the Rust backend, six native apps, or a CLI by last Friday for X and you get DHH's keynote pitch. DHH *is* an expert working on his own product, at a company he controls. This is the kind of scenario that Gandhi argues is most likely to work. There's not even a manager to squeeze the budget here. If it works anywhere, it works here. He took a best-case result and generalized it to "virtually all programmers, virtually all companies, by December."

Gandhi's most useful point is that writing[1] the code was never the dominant cost. We also have to consider shipping the changes, maintaining them, and avoiding regressions. DHH's experience with Hey Next is a week old. It's not even a production system yet.

He showed us this himself. Basecamp 5's "Swiss cheese" architecture was born out of the reality that code was cheap, but coordination wasn't. Gandhi tells a version of the same story: Bun's LLM-assisted fork of Zig that compiles 4x faster but can't be upstreamed, because no one[2] wants a non-deterministic compiler.

Removing a bottleneck doesn't remove the queue; it just shows you where the next constraint is. With LLMs, we're moving the bottleneck one step to the right, from writing code to everything that happens after. DHH's solution is to skip it.

Intolerance

> *Now, part of that is that these programming languages like Rust are tremendously verbose and unappealing for humans to look at. So I don't, and I allow the agent to just spit out more than was necessary, in a way I would never tolerate from my Ruby code.*

One of Gandhi's reasons the cost argument fails is that people's tolerance goes up. When work is asynchronous and agent-driven, we're no longer face-to-face with slower git, laggier autocomplete, and longer builds. There's no human sitting there waiting. If you still care about these things, you probably hate this.

DHH skips it all. You hand the task off "like you would a coworker" and "go back and review when there's something ready." "Review" doesn't mean code review here; it means checking whether the button does the thing.

Maybe that's okay for his personal one-shot projects. It sounds like it's working, a week into Hey Next. But his tolerance going up doesn't raise anyone else's. He's free to not care what's in the Hey Next box, but lots of people care what's in the Rails box. So much for "virtually all programmers."

The Scarcity Is Still Here

So, back to Valim's question. You have infinite tokens. You're a 1000x maker. You can create anything. Could you not find something, *anything* to create for Rails?

This was never really about the budget. An increase in velocity doesn't change priorities. Everything DHH built this year, he wanted for himself. Nothing he's building needs Rails to be better, so he hasn't made it better.

> *We can now want everything. We can now get everything.*

So, what is this everything? Turns out it's a calculator. And a video editor. And some presentation software. Yet another Linux distro. And a rewrite of his own product. He's been handed *unlimited* tokens, and this is all he could dream up. There's some scarcity here. Scarcity of ideas.

Look at that list. Not one single new idea. A microcosm of the industry right now. DHH's wants are on display, and he wants nothing that isn't his and nothing that didn't already exist.

LLMs are exceptional at making things that already exist. Luu admits this; earlier models overfit to the point of comedy[3]. A calculator is a safe ask. In 2004, Rails wasn't. It was novel, and celebrated for it.

DHH's keynote has this backwards. The era of hand-written code isn't some charming thing we've outgrown. Before LLMs, executing on an idea took a hell of a lot more legwork. But you needed a spark first, and you still need it now.

In 2005, "Look at all the things I'm not doing" was a boast. He replayed it this year for the parallel. From where I'm sitting, the thing he's no longer doing is coming up with new ideas.

---

1. "Writing" includes design and debugging, not just typing. ↩

2. Well, except for Bun, whose fork only has to work for them. ↩

3. Further reading: Like Humans, AI Can Jump to Conclusions, Mount Sinai Study Finds ↩

Related Posts

RUBY/RAILS

August 13th, 2025

Undervalued: The Most Useful Design Pattern

Let's explore how we can use data and value objects with the factory method pattern to help decouple the components of our software.

Read post →

RUBY/RAILS

October 10th, 2025

Announcing Burg.rb

I made a web framework for Ruby. Well, not really. But kind of.

Read post →

Skill Issue Newsletter

You want the really good shit? Skill Issue is where you'll find my half-baked ideas about software development, testing, Ruby, and whatever else I'm thinking about.

You can unsubscribe at any time, but you probably won't.

Enter your email address →

CONTACT

Contact Me

JARDO.DEV

About

Blog

Speaking

TOPICS

Software Development

Ruby/Rails

Random

Solidus

Fediverse

SOCIAL

RSS Feed

Mastodon

Bluesky

GitHub

LinkedIn

X

© 2026 Jared Norman. All rights reserved.