

Meta’s Muse Wonder enters coding-agent rotations as practitioners describe how they engineer away slop

Coding Agents Alpha Tracker

2026-10-13

Meta’s Muse Wonder enters coding-agent rotations as practitioners describe how they engineer away slop

By Coding Agents Alpha Tracker • October 13, 2026

Muse Wonder arrives with strong agent benchmarks, a confusing split between Full and standard versions, and early quota problems. Separately, poteto, Kent C. Dodds, Huntley, Sanfilippo and ThePrimeagen share concrete ways to keep agent-written code good.

Muse Wonder: use the Full version, and expect mixed results

Meta launched two models. **Muse Wonder** is available on Meta’s API and in Muse Code. **Muse Wonder Full** is only available to Muse Code users. Both have 1M-token context, and Wonder costs \$4 input / \$0.20 cached / \$20 output per 1M tokens. Full has no per-token price yet [1]. On Artificial Analysis’s Coding Agents Index, Wonder Full scores 68 at max effort in Muse Code. That ties Claude Code with Sonnet 5.5 (68) and is just ahead of Opus 5.5 (66). Standard Wonder scores 63 [1].

The two versions are not interchangeable. **Theo** says standard Wonder over the API is “significantly worse” than Full [2]. He notes they use nearly the same number of tokens, but Full matches Opus xhigh on AA while non-Full scores below GPT-6.1 Sol [3]. His working setup pairs the models. He added Muse support to T3 Code, and in Orchestrator V2 he “often [has] Opus ask Wonder for code review, copy work, and architecture review” [4]. In a three-way bug-fix test:

- Opus found the root cause and filed a PR.
- GPT-6.1 Sol wrote 300 lines of tests [5], but its fix would not have worked [6].
- Wonder Full pointed to an existing community PR and suggested merging that instead.

DHH says he has already shipped two Omarchy PRs built with Wonder. He wants faster max reasoning and a proper agent overview in the default harness [7].

Not everyone agrees. **Riley Brown** used it for five hours and went “back to codex and claude code” [8]. **Matthew Berman** found the \$15 plan doesn’t include Full. He then ran out of a \$50 plan in three minutes, with 429 errors [9]. Alexandr Wang later said the cause of the fast quota burn was fixed and reset Muse Code quotas [10]. On cost per task, AA puts Wonder (max) at \$4.52, below Opus 5.5 (\$5.98) but well above GPT-6.1 Sol (\$0.72) [1].

Slop is an engineering problem, so fix it in the repo

Several practitioners made the same argument from different angles: bad agent output is something you engineer away.

- **Make the codebase the memory (poteto).** Give the agent a way to verify its own work. Then turn every nit, mistake and bad pattern you see into a constraint, using lint rules, compiler diagnostics, types and LSPs. “Forget rules and skills. you can add those later.” Better still, refactor so agent anti-patterns become impossible, and pick frameworks that shrink the architectural search space [11].
- **Build a PR pipeline with a safety net (Kent C. Dodds).** Have agents produce high-quality PRs, use agent PR reviewers, and add feature flags plus a regularly tested disaster-recovery process [12]. Only then connect input pipes so agent work starts without you. He says he has averaged 77 merged PRs a day over five days [13].
- **Enforce rules with hooks, not prompts (Geoffrey Huntley).** Updating `AGENTS.md` still leaves drift. His fix is a pre-commit hook that scans the diff for direct observability calls that skip his diagnostics module, fails the commit, and tells the agent to use the approved path [14]. He also says to test skill packs against a no-skills baseline, to delete skills and `AGENTS.md` every couple of months as models improve [14], and to read each lab’s model card for prompting guidance [14].
- **Ask for a simplification pass (Salvatore Sanfilippo, in Italian).** Left alone, AI produces solutions much bigger than the problem, and that complexity makes later AI edits harder too. Once the code works, ask it to cut useless layers and over-generalized functions one at a time [15].
- **Prototype, then rebuild (ThePrimeagen).** He uses AI to build a first version to explore the system’s shape, then designs a deliberate V2. One project is going from about 100k lines to an expected 25–30k because

the tests and structure were better thought out [16]. He also warns that agents tend to pile up flaky end-to-end tests, so decide what actually needs testing [16]. He keeps to one or two active dev tasks at a time [16].

Claude flows: what the workshop added

At AI Engineer NY, Anthropic presenters explained dynamic workflows. The lead agent plans, then writes an orchestration program that launches agents, collects results and manages information flow [17]. You can still pass in pre-configured specialist subagents [17]. Practical points from the Q&A:

- Defaults are low, so ambitious tasks need explicit encouragement to parallelize [17].
- The model doesn't know its budget while running, so set a session budget [17].
- Subagents share a filesystem but can't be told through the API to message each other [17].
- Claude Console lets you inspect the payload passed along every edge of the workflow graph [17].
- In a stock-scanner comparison, flows cost “much more” than a single agent but found more details [17].

Simon Willison builds a feature by voice

Willison used the Codex tab's voice mode in the ChatGPT desktop app with GPT-6 Astra High against his local Django blog. He started with **Start dev server and open in browser** so he could watch a live preview [18]. In about half an hour of talking while cooking, the agent asked clarifying questions [18]. It built a new model, four importers (including Substack's undocumented `/api/v1/archive` endpoint), archive pages and search integration [18]. In PR review he replaced a Git-subprocess importer with an API call, by typing [18]. His verdict: voice is great for multitasking, but typing still wins for details like pasting errors or pointing at specific code [18].

Tools and plans

- **Cursor** is now part of The X Subscription. Paid plans share one usage pool with Grok Build and give twice the coding usage on Grok models. On the new plans, third-party models in Cursor move to API pricing. Existing subscribers can keep their current plan [19].
- **T3 Code nightly** adds an “update everything” button. It updates the app, every environment, and providers like Claude Code and Codex, then resumes your threads [20]. Theo says his Opus-built ts-rust port, now shipped in T3 Code, cut typechecking time by 3x [21].
- **DHH** says he usually prefers agent contributions to Omarchy because agents write more complete bug reports, and that PRs are “mostly just

proofs of concept now” [22]. The flip side is that the backlog is agent-managed, so not every item will be picked up [23].

Sources

1. X post by @ArtificialAnlys
2. X post by @theo
3. X post by @theo
4. X post by @theo
5. X post by @theo
6. X post by @theo
7. X post by @dhh
8. X post by @rileybrown
9. Meta just dropped something insane... (Muse Wonder)
10. X post by @alexandr_wang
11. X post by @poteto
12. X post by @kentcdodds
13. X post by @kentcdodds
14. You’re Making Your AI Dumber (Here’s How to Stop) | Geoffrey Huntley
3/4
15. Open source e AI: nemiciamici
16. It’s not just you
17. How Anthropic’s Dynamic Workflows Coordinate Claude Agents
18. I built a new feature for my blog using my voice
19. X post by @cursor_ai
20. X post by @theo
21. X post by @theo
22. X post by @dhh
23. X post by @dhh