

Theo’s tsc-rs: ~\$400k of Codex got nowhere,
~\$20k of Opus shipped it; Haiku 5.5 lands as a
cheap subagent model

Coding Agents Alpha Tracker

2026-10-08

**Theo’s tsc-rs: ~\$400k of Codex got nowhere,
~\$20k of Opus shipped it; Haiku 5.5 lands as a
cheap subagent model**

By Coding Agents Alpha Tracker • October 8, 2026

Theo’s agent-built Rust port of TypeScript, the Haiku 5.5 launch and what it means for subagent routing, cross-model adversarial review, and Huntley’s Nix setup for agent sandboxes.

tsc-rs: an agent-built TypeScript compiler, and what the model choice cost

Theo released **tsc-rs** (also called ts-rust), a full Rust rewrite of the TypeScript compiler, type checker and LSP. He describes it as an open-source drop-in replacement for **tsc** (repo) [1]. Agents had been working on it for five months. In his words: “I burned ~\$400k of Codex tokens and got nowhere. Burned ~\$20k of Opus and got there in 2 weeks.” He also says he has not read a single line of the code [2]. He adds that the Opus usage equals about 10 weeks on the \$200 Claude subscription [3].

The port matched its test oracle, upstream TypeScript, closely. Of the first five issues filed, he says four were real upstream TypeScript behaviour that the port reproduced faithfully [4]. He is open about its limits. **bun check** shipped the same day and is “probably the right choice for most apps using Bun,” and tsc-rs is only faster when you use Effect TS checks. He prefers tsc-rs for its drop-in compatibility, WASM readiness and built-in Effect mods [1].

Mario Zechner and Armin Ronacher explained in a separate conversation why projects like this work. If you have “an Oracle that can tell the LLM if what

it did is correct or not, then you basically won.” Bun’s test suite is enough for agents to drive a port with occasional steering [5]. Ronacher had agents implement every format in a serialization library from sample TOML, CBOR, MessagePack and JSON files, then told them to “fuzz the hell out of it.” That found many bugs, and an LLM can reason about cases a fuzz generator misses [5].

Haiku 5.5: Anthropic’s pitch is a cheap subagent

Anthropic says Haiku 5.5 is in Claude Code and the Claude Platform, costs about 75% less to run than Haiku 4.5, and suits use as a subagent under Opus 5.5 or Sonnet 5.5. It suggests “summaries, compactions, or database queries” [6]. Addy Osmani says his team “loves it as a subagent alongside Opus 5.5,” and notes it has an adjustable effort setting [7]. Cursor has it under Settings > Models. Pricing is \$0.10/\$0.50 per million input/output tokens, rising to \$0.50/\$2.50 above 100k input tokens. Sonnet 5.5 cache reads dropped from \$0.20/M to \$0.10/M [8].

Simon Willison found catches that change the routing math: - Up to 100k tokens it costs exactly the same as GPT-6 Luna. Above that, its price rises 5x, while Luna’s goes up only at 272k and only to \$0.20/\$0.75. For long-context jobs, Luna looks like the better deal [9]. - The new tokenizer used about 1.25x as many tokens as Haiku 4.5 on the same long prompt, which amounts to a hidden price increase [9]. - You can’t turn reasoning off. The default is `medium`. To try it: `llm install -U llm-anthropic, llm anthropic refresh`, then `llm -m claude-haiku-5.5 ... -o thinking_effort low` [9]. - Max and Team subscribers now get monthly API credits: \$100 for Max 5x, \$200 for Max 20x, and up to \$500 pooled for Team. They don’t roll over. You can turn off `autoreload` so requests stop when the balance runs out [9].

ThePrimeagen offers an early counterpoint. Dropping Haiku in for Luna, both with reasoning off, took a one-line config change. Haiku “performed significantly worse” and was slower on both passes and failures, and he doesn’t yet know why [10]. Separately, he found OpenAI’s new decision model much faster and more accurate than his previous setup at finding click targets in Omarchy QA screenshots [11]. Benchmark your own automation before switching.

Cross-model adversarial review takes one sentence

DHH’s technique: when you’re driving from Codex, say “Review this with claude”, and when you’re in Claude, say “Review this with codex”. The models know how to start a review through the CLI, take turns and settle an argument. “No magic” [12]. He describes the rest of his setup as “any harness, multiple agents concurrently, barely any skills, and using adversarial reviews” [13]. Theo runs a version of this. Claude spins up Sol subagents and calls Codex through T3 Code to review its work, because he finds OpenAI models “a bit more thorough with their analysis.” He still calls the Claude plan “absolutely mandatory”

for shipping serious engineering work [14]. On the “nerfed” \$200 Codex plan, he argues that cost per task matters more than tokens per dollar. In his own Terminal Bench run, 6.1 Sol at x-high roughly tied Opus 5.5 at about one-thirteenth the cost per task [14].

Audit your agents for logic they keep reinventing

Theo says his agents wrote over 200 bad “watch PR” scripts. His audit prompt: “audit my history with Claude Code, Codex and other agents on this machine. Look for every time I asked for a PR to be watched or babysat.” Then ask how many times the logic was reinvented, how many versions had visible flaws, and roughly how many tokens and dollars were wasted [15]. The production fix in T3 Code took about 30 tries. It replaced CLI calls with direct API calls and switches between GraphQL and REST depending on which uses rate limits more efficiently, with fallbacks. That cut rate-limit usage by over 75% [16].

Huntley: Nix as the shared environment for agent sandboxes

Geoffrey Huntley argues for a single `devenv.nix` as the one source of truth for laptops, CI and ephemeral agent sandboxes. His example stanza provides Rust, Postgres, a rustfmt hook and `prek` “for agent backpressure” [17]. He develops on NixOS and explicitly tells agents to use `sudo`, counting on rollback. With `runNixOSTest`, he puts the whole OS, including multi-machine networking and firewall rules, under test [17]. He also uses an overlay to strip force-push out of the Git binary inside agent sandboxes (`nix-demo`) [17]. The tradeoff he names is that incremental caching is weaker than in Bazel or Buck2 [17]. Separately, he says he is dropping Opus 5.5 because Claude Code’s classifier “is too paternalistic and breaks my flow/development loops” [18].

Smaller items

- **Deep Agents skills:** you can now bind tools to a skill with `metadata.include_tools`. A bound tool stays out of context until the agent reads that skill [19]. Apps can pass `pinned_skills` so a skill like `/meeting-prep` is loaded before the first model call [19]. Setting `skills_metadata=None` makes the next run rescan the skill library [19].
- **Sourcegraph Deep Search** lets Claude Code search code you haven’t checked out. Their demo finds deprecated packages across Kubernetes [20].

Sources

1. X post by @theo
2. X post by @theo

3. X post by @theo
4. X post by @theo
5. Pi Durable: Agents That Survive Crashes — Mario Zechner & Armin Ronacher
6. X post by @ClaudeDevs
7. X post by @addyosmani
8. X post by @cursor_ai
9. Claude Haiku 5.5
10. X post by @ThePrimeagen
11. X post by @ThePrimeagen
12. X post by @dhh
13. X post by @dhh
14. Does the \$200 Codex plan suck now?
15. X post by @theo
16. X post by @theo
17. the world hasn't figured out yet that you can literally just fix everything with a Nix overlay
18. X post by @GeoffreyHuntley
19. Revamping Skills in Deep Agents
20. X post by @Sourcegraph