How OpenAI ships 70% faster

Best-practices from our conversation with Romain Huet (Head of Developer Experience, OpenAI)

We sat down with Romain Huet (Head of DX, OpenAI) to discuss how OpenAI's team uses Codex. The headline: code is now abundant, and human time is scarce. Below, we're including some takeaways and best-practices from the conversation:

By the numbers (as shared on the episode):

- Codex usage is up ~20x since Aug (overall), per Huet
- OpenAI engineers merge ~70% more PRs/week with Codex in the loop
- 100% of PRs get a Codex review pass before a human sees them
- Long-horizon tasks can run over two weeks (and increasing) without intervention
- With Codex, 4 engineers shipped the Sora Android app in 18 days, end-to-end

Best practice #1: Pick the right surface based on the task - local vs cloud

- Local (IDE/CLI): tight-loop work, UI iteration, small diffs you want to watch in real time.
- Cloud: long refactors/migrations, backlog V0s, and best-of-N (run 4x in parallel, pick a direction).
- Treat Slack/Linear as input: let the agent pick up tasks where work already happens.

Best practice #2: Involve AI-first code review

- Code review rapidly becomes the velocity bollteneck
- Make review execution-based: spin a clean container, run tests, flip feature flags, try edge paths.
- Require a PR footer: commands run + risks + rollback. Include it in your '/pr' slash command or skill.

Best practice #3: Chrome MCP for full-stack; Contex7 for integration

- Chrome DevTools MCP: console errors, perf regressions, mobile breakpoints, caching - the stuff humans skip.
- Playwright / click agents: validate flows end-to-end, not just unit tests.
- Figma -> code: paste a node URL/ID; agent pulls tokens/components and matches the design via screenshots.
- Fresh docs MCPs: Context7 keep agents on current APIs, not training cutoff.

Best practice #4: Codex vs Cursor vs. Claude Code: when to reach for which

- In-flow co-creation (you want to read the code as it changes): Cursor or Codex IDE shines in-editor.
- Offload-and-forget (multi-hour, multi-file, 'this would take me days'): Codex shines on long-horizon tasks; Claude Code too.
- Romain: Codex is relentless + steerable - you can add instructions mid-task without restarting.

Best practice #5: Protect your repo; DevEx now means AgentEx

- Repos are the UX for agents. If the agent can't orient, it will thrash... or worse, hallucinate.
- Treat `AGENTS.md` as onboarding, not documentation: conventions, libraries, boundaries, and "when NOT to create new things."
- Per Romain, OpenAI has ~100 `AGENTS.md` files in it's monorepo, or roughly one per service/team.

Over the next 6–12 months, coding agents will feel less like "chat" and more like delegation. Models are already getting much better at sustained reasoning, maintaining context over long tasks, and using tools (tests, browsers, design systems, docs) to check their own work. The practical implication: instead of trying to specify every edge case up front, teams will define the task plus the verifier (which tests to run, what screenshots or traces to capture), and let the agent iterate until it meets those checks. Repos and workflows will adapt accordingly: lightweight per-service guidance files (e.g., AGENTS.md) become normal to keep agents aligned with conventions; setup and validation collapse into a small number of reliable commands; and tool connections (Figma → code, browser/devtools automation, live docs) become part of the default build loop. Code review shifts in parallel: less time debating diffs in the abstract, more time evaluating execution (tests, traces, screenshots) and making product decisions.

For the original conversation with Romain and to join future ones, please visit https://researchtoruntime.com.