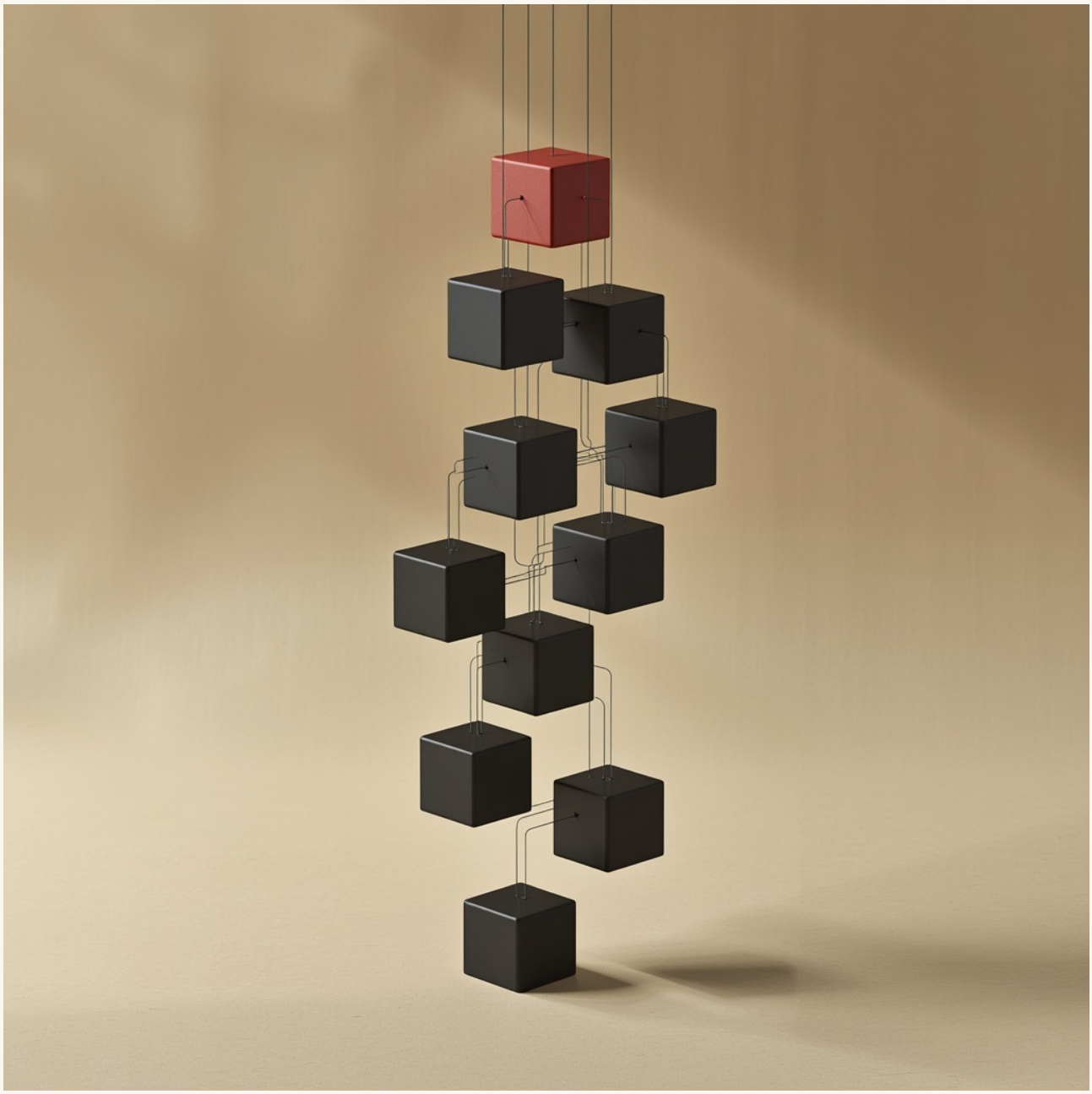
— AI CODING

Code w/ Claude 2026: what actually shipped, what was theater

By **Flowi Editorial** · *May 9, 2026 · 7 min read*

Anthropic's Code w/ Claude developer event landed yesterday. Most coverage will be hot-take. Here's the read on what actually changes for builders.



Anthropic ran their second Code w/ Claude event yesterday. There were no Project Stargate-style announcements, which the live-blog set treated as a disappointment. They're misreading the room. The interesting story is what Anthropic shipped *quietly*, between the keynote talks, that changes how you'll work with Claude Code in production over the next six months.

Here's the practitioner read — what landed, what was theater, what to update your workflow around this week.

What actually shipped

Three things, in order of how much they'll change your day:

1. Background sub-agents that can run for hours. This is the big one and Anthropic buried it. Claude Code can now spawn background agents that operate on a separate execution context, take a defined task, and report back when they're done — minutes or hours later. The classic example: "audit this entire monorepo for security antipatterns and put the results in a markdown file." That used to be a 10,000-token context-eating task that risked drowning your main session. It's now a fire-and-forget call, and the parent session continues unaware.

The implication is bigger than time savings. **You can now build agents that have agents.** A planning agent calls a research sub-agent to dig into a topic; while it's gone, the planner does other work; when the sub-agent returns, the planner integrates the result. That's the multi-agent pattern people have been hand-rolling with the Agent SDK for a year. Now it's a primitive in Claude Code itself.

2. The "skill" system, formalized. Skills were a research preview last quarter. They're now a first-class feature with a defined directory structure, a manifest format, and a repo of community-contributed skills you can install. The spec covers things like: multi-step debugging routines, library-specific code patterns, project-specific conventions. The good news is you can write a skill once and have it apply across every Claude session you start. The trap is that skills can be invoked invisibly, which makes "why did Claude do that?" harder to debug. Read the manifest of any skill you install before letting it run; this is exactly the kind of lever malicious actors will eventually try to abuse.

3. A real plan-mode hand-off. Plan mode in Claude Code now produces a *durable artifact* — a markdown plan file you can review, share with a teammate, and re-feed to Claude tomorrow. Previously the plan lived in conversation context and decayed when the session ended. The hand-off is the missing piece for actual team workflows: senior engineer scopes the plan in a 20-minute session, hands it off, junior engineer (or another agent, or another instance of Claude) picks up the plan and executes. That's the workflow most engineering teams have been waiting for.

What was theater

The "Claude can write your entire startup in 4 hours" demo. It can't. The demo was real but the codebase was a tutorial-grade CRUD app with no auth, no RBAC, no rate limiting, no observability, no error handling, and (critically) no business logic that wasn't representable as a single SQL query. Every one of those is the part where production AI coding still falls over. The demo was honest about being a tutorial, but the headline write-ups won't be.

The "$100B GDP" framing. Anthropic's CEO talked about Claude reaching 100% of global GDP within 21 months. That's not a claim about today; it's a claim about projection curves. It made for a

viral clip. It tells you exactly nothing about whether you should build with Claude Code today versus Cursor versus a homegrown setup. Pick on the basis of the actual feature set you saw in the lab demos, not the keynote rhetoric.

The agentic IDE preview. Anthropic showed an early IDE-integrated experience that looked very polished. It's not shipping in 2026. The team confirmed that, off-stage, when pressed. The demo is real, the timeline is not.

Show the mechanism

Why background sub-agents matter, in concrete terms:

A typical agentic-coding session, before yesterday, ran into a wall around 60–80k tokens of context. Past that, your effective reasoning quality degraded — Claude started forgetting earlier decisions, repeating the same investigation paths, and (most subtly) losing track of which files it had already touched in the current session. The fix was usually a hard restart with a clean context, which lost continuity.

Background sub-agents change the math. Now the heavy investigation work — auditing 500 files, running an entire test suite and analyzing failures, reading a year of git log to understand a refactor history — happens in a separate context, and only the *summary* returns to your main session. That summary might be 200 tokens. Your main session stays in the 30–40k range, which is the comfortable zone for most reasoning tasks.

The mental model: **the sub-agent is a research assistant.** You don't read every paper your assistant reads. You read their two-paragraph summary and the citations they flagged.

The trap nobody is warning about

If you're building production agents that use Claude Code-style sub-agents, watch for **summary loss**.

When a sub-agent returns a 200-token summary of work it did across 50,000 tokens of context, that summary is necessarily lossy. Specifically: it loses *the thing you didn't ask about*. A sub-agent told to "find security antipatterns in this directory" will report security findings — but won't surface a critical performance issue it walked past, even though it spent two hours staring at the relevant code.

This is the same failure mode that plagues human research teams. The senior engineer who delegates a research task only learns what the junior engineer thought to surface. The unknown unknowns stay unknown. With AI sub-agents, the bandwidth gap between what they processed and what they reported is even wider, so the unknowns are even more numerous.

Two mitigations worth adopting:

Re-prompt the sub-agent for orthogonal axes. After a security audit, ask the same sub-agent (or a fresh one, with the same context window) to do a performance audit on the same code. The cost is one extra invocation; the catch rate is meaningfully better.

Keep the sub-agent's working file. Most sub-agent frameworks let you persist the agent's intermediate scratchpad, not just the final summary. Read the scratchpad when the summary feels suspiciously clean. The good stuff is often in the rejected branches of the investigation.

Who should care

- **Engineering teams shipping with agentic coding tools** (Claude Code, Cursor, Aider, Devin, etc.): the background sub-agent pattern is now generally available, and your team workflow should incorporate it within two weeks. The hand-off bottleneck just got smaller; use it.
- **Solo builders running Claude Code on side projects**: the plan-mode hand-off is the under-marketed feature most worth your time. Stop redoing the planning work every time you start a session. Save the plans, version them in git, re-feed them.
- **Tool builders working on agent SDKs**: the skill system is a forcing function. If your agent framework can't load and invoke a Claude-style skill manifest, build that compatibility now. The community library is going to have hundreds of skills within six months, and your framework's value drops significantly if it can't share that ecosystem.

The Code w/ Claude event was a quiet release event with one big feature (background sub-agents) buried under polish theater. That's actually a good signal. Frontier labs talk loudest about the things that aren't quite ready and quietest about the things that just shipped to production. The skills system, the durable plans, the background agents — those are all in production now. Build with them this week.

If you're putting agents into production and trying to figure out the multi-agent coordination patterns, that's the territory we covered in Agent Memory: The 5 Patterns That Ship in Production. The decision tree for sub-agents — what to delegate, how to summarize back, when the parent agent should re-investigate — is the same problem regardless of whether you're using Claude Code or rolling your own.

Originally published on useflowi.app/blog/code-with-claude-2026-what-actually-shipped.
Flowi — the editorial intelligence layer for AI builders.
Daily brief at useflowi.app/blog · Monthly Dispatch at useflowi.app/dispatch.