40

HIDDEN FEATURES · SHORTCUTS · POWER SETTINGS

The Claude Power User Codex.

Forty features buried across Claude Chat, Claude Code, and Claude Cowork that change how you work the moment you discover them. A field guide for serious operators.

CURATED BY

THE AUSTIN AI GUY

Practical AI playbooks for builders & operators

A NOTE FROM THE DESK

Most people use 10% of what Claude can do.

If you have been chatting with Claude through the standard message box and walking away impressed, you are doing the equivalent of using a Ferrari to pick up groceries. It works. It is not the point.

This guide is the result of going through every setting, slash command, hidden toggle, and undocumented behavior across **Claude Chat**, **Claude Code**, and **Claude Cowork** — then ranking what actually changes the way you work versus what is interesting trivia.

What follows is forty entries. Each is short on purpose. The goal is not to teach you everything about a feature — it is to introduce you to it, tell you why it matters, and get out of your way so you can go try it. Read it cover-to-cover in one sitting, or skim the section that fits the tool you use most. Either way, by the end you will have at least five things to try this week.

A few notes before we begin:

HOW TO READ THIS GUIDE

01 Each entry stands alone. Skip around freely.

02 Bold means try this first. Italics are nuance.

03 Anything that requires an upgrade is marked.

04 The features keep shipping — check back.

Let's get into it.

— The Austin AI Guy

IN THIS GUIDE

PART 01

Claude Chat.

Twelve features that change everyday conversations.

The chat interface is where most people meet Claude — and where most people stop exploring. These twelve entries are the difference between having a smart assistant and having a system that knows you, remembers you, and produces work you would actually ship.

The chat box is the front door. Everything below is what's behind it.

---

01 Custom Styles

Settings → Styles holds presets — Concise, Explanatory, Formal, and more — that quietly reshape every reply. The real unlock is building a custom style that matches your voice. Set it once and every conversation starts in the right register without you having to remind it.

■ ***Set it once, never re-prompt your tone.***

02 Projects

Most people miss this entirely. Projects are persistent workspaces with their own instructions and files. Every conversation inside inherits the context automatically. Stop pasting your brief into every new chat — set it once at the project level.

03 Project Knowledge

Inside any project, upload reference documents and Claude reads them at the start of every conversation. Brand guidelines. Product docs. Writing samples. Always loaded. Never forgotten. This is how you make Claude actually know your business.

■ ***The single biggest quality jump for most users.***

04 Custom Instructions (System Prompt)

Project settings let you write instructions that run silently behind every message. "You are my content strategist. Always respond in my brand voice. Never use these words: [list]. Format outputs as: [format]." Treat it like onboarding a new employee — be specific.

05 Artifacts

When Claude produces code, documents, or visual outputs, they render in a side panel you can iterate on independently. Edit, refine, and download without losing the chat context. The chat becomes the conversation. The artifact becomes the deliverable.

06

Memory

Claude now remembers across conversations — preferences, projects, communication style. View and edit what it remembers in Settings → Memory. Curate it like onboarding notes for an employee who never forgets but occasionally over-indexes on something trivial.

■ ***Audit your memory monthly. Trim the noise.***

07

Deep Research

Toggle Deep Research on any query and Claude runs extended searches across dozens of sources before producing a comprehensive report. Think of it as a junior analyst spending two hours on research, delivered in two minutes. Use it for any decision worth more than 30 minutes of your time.

08

File Upload Intelligence

Upload PDFs, images, spreadsheets, CSVs, code files, documents — Claude does not just store them, it reads and understands them. Upload a 50-page report and ask about page 37. Upload a spreadsheet and ask for trend analysis. Treat every file as a queryable object.

09

Image Analysis

Upload any image and Claude sees it with remarkable detail. Error screenshots, whiteboard photos, charts, handwritten notes, receipts, business cards. If it contains visual information, Claude can extract and interpret it. Stop transcribing things by hand.

10

Canvas

Claude generates visuals, diagrams, and interactive components directly in chat. Flowcharts. Comparison tables. Org charts. Working calculators. They render inline and remain interactive — not just images. This is the feature that quietly replaced three tools in my stack.

■ ***Ask for a flowchart of any process you keep explaining.***

11 LaTeX Rendering

For anyone working with math, statistics, or technical content, Claude renders LaTeX equations beautifully. Ask for formulas, derivations, or statistical outputs and they display properly formatted. The difference between "good enough" and "ready to publish."

12 Conversation Branching

Edit any previous message and Claude regenerates the response from that point, creating a new branch. This lets you explore alternative approaches without losing the original thread. Stop opening new chats to try a different angle — branch instead.

■ ***The undo button you didn't know you had.***

PART 02

Claude Code.

Twelve features for serious development work.

This is where Claude stops being a chat product and becomes a teammate. Hierarchy, planning, custom commands, parallel review — these patterns separate developers who use AI from developers whose codebases are shaped by it. If you ship code, the next pages matter.

Stop typing into the terminal like it's a chat. Configure it like infrastructure.

---

13 The CLAUDE.md Hierarchy

Most users have one CLAUDE.md. Power users have three. User-level (~/.claude/CLAUDE.md) for personal preferences. Project-level (.claude/CLAUDE.md) for team standards. Directory-level for module-specific rules. They cascade — most specific wins. Architect your context like you architect code.

■ ***Three levels. Cascading. Specificity wins.***

14 Path-Specific Rules

Create files in .claude/rules/ with YAML frontmatter specifying glob patterns. A rule with paths: ["**/*.test.*"] applies to every test file automatically. Different standards for different file types without cluttering your main CLAUDE.md. This is how big codebases stay sane.

15 Plan Mode (Shift+Tab)

Switch to plan mode before complex tasks. Claude creates a step-by-step plan, shows it for approval, and only executes after you agree. Essential for any task touching multiple files. This is the difference between clean execution and debugging chaos.

■ ***Always plan before multi-file changes. Always.***

16 /compact

Compresses your conversation when context gets long. Claude keeps the important details but frees up window space. Use this the moment Claude starts repeating mistakes or losing track of earlier decisions — that is context exhaustion talking, not the model.

17 /memory

Shows exactly which memory files Claude Code has loaded for this session. If Claude is behaving inconsistently, run this first to check whether the right context is actually active. Half of "why is Claude doing this" debugging starts and ends here.

18 Custom Slash Commands

Build reusable commands in .claude/commands/ (personal) or ~/.claude/commands/ (global). A /review command that runs your code review checklist. A /test command that generates tests following your patterns. Ten minutes to create. Hours saved over a quarter.

19 Git Integration

Claude Code has native git awareness. Commit, push, create branches, write commit messages based on the actual changes it made. "Commit everything with a descriptive message" actually works — and the messages are often better than what you'd write at 11pm.

20 Multi-File Editing

Claude Code reads and edits multiple files in a single operation. Rename a function across your entire codebase. Update an import path everywhere it appears. Refactor a component and update every file that uses it. The kind of work that used to eat half a day.

■ ***Refactors that used to take hours now take minutes.***

21 Test Generation

Point Claude Code at any function or module. "Write comprehensive tests for this." It generates test files that follow your project's conventions — assuming you've set them in CLAUDE.md or path rules — including edge cases and error scenarios you'd probably miss at 2pm on a Friday.

22 The -p Flag

Runs Claude Code in non-interactive, headless mode. Essential for CI/CD pipelines. Without it, your CI job hangs forever waiting for user input that will never come. With it, Claude runs autonomously and returns structured output. This is the flag that makes automation real.

23

--output-format json

Combined with --json-schema, Claude Code returns machine-parseable structured output. Your CI pipeline can automatically parse findings and post them as inline PR comments. The bridge between "Claude helped me" and "Claude is part of the system."

24

Independent Review Instances

The Claude Code session that wrote the code is biased toward its own decisions when reviewing. Always use a separate, fresh session for code review. This is so important the Claude Certified Architect exam explicitly tests it. Treat the reviewer as a different engineer.

■ ***Never let the author review their own work — even an AI author.***

PART 03

Claude Cowork.

Ten features for autonomous, long-running work.

Cowork is the agent layer — where Claude does work for you across files, applications, and scheduled time. The features here are less about individual messages and more about systems. Set them up once and they produce value for months without you in the loop.

The shift from operator to architect happens in this section.

---

25 Sub-Agent Parallel Processing

When Cowork gets a large task, it spins up multiple sub-agents working simultaneously. Tell it to process 20 files and it divides them across 4-5 sub-agents in parallel. What takes 30 minutes sequentially finishes in 6. Frame your big tasks accordingly.

■ ***Batch big jobs. Let it parallelize.***

26 /schedule

Set recurring tasks. Daily briefings. Weekly cleanups. Monthly financial processing. Your computer needs to be on and Claude Desktop open, but the tasks run unattended. If your laptop sleeps mid-run, it auto-resumes when you reopen. This is hands-off automation.

27 Connector Chaining

Combine connectors in a single workflow. "Read my Gmail, check my calendar, pull relevant files from Drive, and create a meeting prep document." Four connectors, one instruction, zero tab switching. The whole point of Cowork lives in this pattern.

28 Plugin Marketplace

Verified plugins at claude.com/plugins give you pre-built capabilities for specific roles. Product management. Marketing. Finance. Legal. Each plugin adds slash commands and skills tailored to that function. Browse it like an app store for your job.

29 Folder Instructions

Drop a markdown file with instructions into any folder. When Cowork works on files in that folder, it reads those instructions first. Different rules for different projects. Different formatting for different clients. Automatic context switching, no manual prompting.

■ ***Context that travels with the work itself.***

30 Sandbox Security

Everything Cowork does runs inside a sandboxed Linux VM. It cannot access files outside the folders you explicitly grant. You control the blast radius. This is why Cowork is safe for production work and why your IT team will let you use it.

31 Browser Bridge

When Claude in Chrome is installed alongside Cowork, the two work together. Cowork delegates web research to Chrome, processes the results locally, and continues the workflow. The best of both worlds — autonomous browsing plus local file operations.

32 Session History

Every Cowork session is logged in full — actions taken, files modified, output produced. Review any past session to understand exactly what happened. Essential for debugging failed automations and for explaining to your boss what the AI actually did.

33 Token Usage Awareness

Cowork tasks consume 3-5x more tokens than regular chat. Batch related tasks into single sessions. Be specific to avoid back-and-forth clarification. Schedule heavy jobs for off-peak hours where throughput is reportedly higher. Token discipline = cost discipline.

■ ***Be specific. Batch ruthlessly. Schedule heavy work.***

34 Plugin Chaining

Combine multiple plugins in a single workflow. Your research plugin feeds into your analysis plugin feeds into your report plugin. Multi-step, multi-capability workflows triggered by one command. This is where Cowork stops feeling like a tool and starts feeling like a team.

PART 04

Platform-Wide.

Six settings that quietly shape everything else.

The final stretch — six platform-level levers that affect every product above them. Model selection, privacy, team sharing, API access. Get these right once and you stop fighting the platform. Get them wrong and every feature in the previous sections works against you.

The settings panel is where the boring decisions live. And the consequential ones.

---

35 Usage Dashboard

Check your token consumption and usage patterns. If you are hitting limits, this shows you exactly where your tokens are going so you can optimize. Most people get throttled and blame the product. The dashboard tells you the truth.

36 Model Selection

Different tasks benefit from different models. Haiku is faster and cheaper for simple work. Sonnet balances speed and quality for most tasks. Opus is the most capable for complex reasoning. Match the model to the task — do not default to the biggest one out of habit.

■ ***Right model, right task. Not always the biggest.***

37 API Access

Your Claude subscription includes API credits. Build custom integrations, connect Claude to your own tools, automate workflows the standard interfaces do not support. This is where Claude stops being a chat product and starts being infrastructure.

38 Team Sharing

On Team or Enterprise plans, share projects, skills, and configurations across your team. Everyone gets the same context, the same standards, the same capabilities. Consistency at organizational scale. This is the feature that turns a personal tool into a company-wide multiplier.

39 Data Privacy Controls

You can opt out of conversations being used for training. Check Settings → Privacy. For any sensitive business work, verify your data handling preferences are set correctly before pasting in anything you would not want to read on the front page of the paper.

■ ***Verify before you paste. Always.***

40 Keyboard Shortcuts

Ctrl+/ shows all available shortcuts. Ctrl+Shift+O opens a new conversation. Ctrl+Shift+C copies the last response. Small efficiencies that compound across hundreds of daily interactions. Spend ten minutes learning them this week. Save hours every month for the rest of your career.

ONE LAST THING

You now know more than 95% of Claude users on earth.

Forty features is a lot. You will not implement all of them. But pick three this week — just three — and your workflow shifts permanently. Pick three more next week. In a month, you will be operating at a level that looks like magic to people still typing into the default message box.

If this was useful, the best thing you can do is share it with the one person on your team who would benefit most. I write practical AI playbooks like this every week — no fluff, no hype, just what is working right now for builders and operators.

GO DEEPER

AUSTINAIGUY.COM

Weekly playbooks. Real workflows. No hype.
Subscribe for the next guide — built for operators
who want to ship faster with AI, not just talk about it.

VISIT · AUSTINAIGUY.COM

SHARE THIS GUIDE ■

Features described reflect Claude products as of publication and may evolve.