Claude Code’s Built-In Skills & Commands: A Quick Reference

Publisher: Frankly AI **Last checked:** 22 August 2026

Claude Code ships with a set of skills, and a separate set of slash commands, built directly into the tool — nothing to install. Before you build or install anything of your own, it’s worth knowing what’s already there, so you’re not duplicating work Claude Code already does out of the box.

This guide covers both, in plain language: **Part 1** is the built-in skills (broader, often automatic capabilities), **Part 2** is the built-in CLI slash commands (typed directly, one specific action each).

Part 1 — Built-In Skills

Building & designing interfaces

design — Drafts a visual design (a UI mockup, landing page, poster, or similar) as an editable canvas you can then click through and adjust by hand — text, layout, images — without touching code.

dataviz — Applies a consistent design system to any chart, graph, or dashboard you ask Claude to build: color choices, chart-type selection, and styling that holds up in both light and dark themes.

artifact-design — A design-quality pass Claude runs before producing any published web page or document, so the output looks intentional rather than like a generic template.

artifact-diagramming — Guidance Claude uses when a diagram would explain something better than text, and for keeping that diagram readable regardless of the page’s structure.

artifact-capabilities — Governs the extra behaviors a published page can have beyond static content: reading live data, remembering what a visitor does on it, shared state between viewers, or files a visitor can add.

Reviewing & improving code

code-review — Reviews a diff, branch, or pull request for correctness bugs and cleanup opportunities, at a review depth you choose. Can also apply the fixes it finds or post them as comments on a PR.

security-review — Reviews the pending changes on the current branch specifically for security vulnerabilities.

simplify — Reviews recently changed code for opportunities to reuse existing code, cut unnecessary complexity, or improve efficiency, then applies the fixes. Not a bug hunt — that's what code-review is for.

fewer-permission-prompts — Looks back through recent activity for safe, repeated actions that keep triggering permission prompts, and allowlists them so they stop interrupting you.

Running & verifying your work

run — Launches your project's actual app so you (or Claude) can see a change working in practice, not just confirm it passed a static check.

init — Generates a starting CLAUDE.md file that documents an existing codebase, so future sessions have context without you writing it by hand.

Automating & scheduling

loop — Runs a prompt or command repeatedly on an interval, either one you set or one Claude paces itself, useful for polling a status or repeating a check over time.

schedule — Creates and manages cron-scheduled runs (including one-off scheduled runs) that execute automatically without you present.

Configuring Claude Code itself

update-config — Edits Claude Code's own settings — hooks, permissions, environment variables — through plain instructions instead of hand-editing the settings file.

keybindings-help — Customizes keyboard shortcuts inside Claude Code, including multi-key shortcuts.

Reference

claude-api — A built-in reference for the Claude API and Anthropic SDK itself: model choice, pricing, streaming, tool use, and caching. Surfaces automatically any time a question is about Claude, Anthropic, or working with an LLM.

Part 2 — Built-In CLI Commands

Slash commands are typed directly (e.g. `/clear`) and each does one specific thing immediately — different from a skill, which Claude can also trigger automatically based on what you ask for. Claude Code has 60+ built-in commands in total; the ones below are the practical core most people actually use day to day. A few niche ones (mobile hand-off, referral sharing, plan upgrades) are left out as not relevant to building your own system.

Session & context

/clear — Starts a brand-new conversation while keeping your project's CLAUDE.md memory intact.

/compact — Summarizes the current conversation to free up space in a long session.

/resume — Returns to an earlier point in a previous conversation.

/rewind — Rolls code or conversation back to an earlier checkpoint.

/context — Shows how much of the available context window the current session is using.

Model & performance

/model — Switches which Claude model the session uses.

/effort — Sets how much reasoning effort Claude applies to a task.

/fast — Toggles faster, lighter-weight responses.

Planning & large changes

/plan — Enters a planning mode for designing a large or multi-step change before writing any code.

/batch — Breaks a large change into independent pieces that can run in parallel.

/diff — Opens an interactive viewer for the current code changes.

/verify — Confirms a change actually works end-to-end, not just that it looks right.

Configuration

/config — Opens Claude Code's settings.

/permissions — Manages which tools and commands are allowed to run without asking first.

/theme — Sets the light/dark color scheme.

/keybindings — Opens the keyboard shortcuts configuration.

Files & directories

/add-dir — Adds another working directory Claude can access.

/cd — Moves the session to a different working directory.

/export — Exports the conversation as plain text.

Tools & integrations

/mcp — Manages connections to MCP servers (external tools and data sources).

/hooks — Views and manages automated hook scripts that run on specific events.

/ide — Manages IDE integrations (VS Code, JetBrains).

/plugin — Manages installed Claude Code plugins.

/install-github-app — Installs the Claude GitHub App for repository integration.

Diagnostics & help

/doctor — Runs a diagnostic check on your Claude Code setup and flags configuration problems.

/debug — Turns on debug logging to troubleshoot an issue.

/help — Shows help and the full list of available commands.

/status — Shows the current session's status.

Usage & feedback

/cost — Shows token usage and cost for the current session.

/feedback — Reports a bug or sends feedback directly to Anthropic.

Account

/login — Signs in to your Anthropic account.

/logout — Signs out.

Where this fits into your build

Built-in skills and commands handle the mechanics: laying things out, reviewing code, running the app, managing sessions, scheduling a job. What they don't give you is a system — a set of rules for how decisions get made, what's allowed to run without asking, and how the project stays organized as it grows. That's the layer installable skill libraries are built to cover.