OFFICIAL RESOURCE FROM

PROMPTSLOVE

Claude Code Shortcuts Cheat Sheet

Every Shortcut · Every Command · Every Flag

80+ Slash Commands | 60+ CLI Flags | Vim Mode | 2026 Edition

promptslove.com

CLAUDE CODE REFERENCE GUIDE · 2026 · ALL RIGHTS RESERVED

SESSION & NAVIGATION CONTROLS

Shortcut	What It Does
`Shift+Tab`	Cycle permission modes: Default → AcceptEdits → Plan → Auto → BypassPermissions
`Esc`	Interrupt Claude mid-response. Keeps all work done so far
`Esc + Esc (double)`	Clear input & save draft to history; or open rewind/checkpoint menu when input is empty
`Ctrl+C`	Cancel current operation. Second press exits Claude Code. Cannot be rebound
`Ctrl+D`	Exit Claude Code (EOF signal). Cannot be rebound
`Ctrl+O`	Toggle verbose transcript viewer (shows tool usage, MCP call details)
`Ctrl+R`	Reverse search through command history (interactive)
`Ctrl+T`	Toggle the task list in the terminal status area
`Ctrl+B`	Background running tasks or bash commands. Tmux users press twice
`Ctrl+X Ctrl+K`	Stop all running background subagents. Press twice within 3 seconds to confirm
`Ctrl+L`	Redraw the screen. Keeps input and history. Use when display becomes garbled
`Ctrl+G or Ctrl+X Ctrl+E`	Open current prompt in your default external text editor
`Ctrl+V`	Paste image from clipboard (Alt+V on Windows/WSL; Cmd+V on iTerm2)
`Alt+P / Option+P`	Switch model without clearing your current prompt (Win/Linux / Mac)
`Alt+T / Option+T`	Toggle extended thinking mode
`Alt+O / Option+O`	Toggle fast mode
`Left / Right arrows`	Cycle between tabs in permission dialogs and menus
`Up / Down arrows`	Navigate command history (moves cursor in multiline input first)
`Ctrl+S (in history search)`	Cycle history scope: this session → this project → all projects

Mac Option Key Setup: Alt/Option shortcuts require Option configured as Meta. iTerm2: Profiles → Keys → Left/Right Option → "Esc+". Apple Terminal: Settings → Profiles → Keyboard → "Use Option as Meta Key". VS Code: add `"terminal.integrated.macOptionIsMeta": true`

QUICK INPUT PREFIXES

Prefix	What It Does
`/ at start of input`	Opens command/skill menu. Type letters to filter
`! at start of input`	Shell mode: runs command directly and adds output to session context
`@ anywhere`	Triggers file path autocomplete in your prompt

MULTILINE INPUT METHODS

Shortcut	Works In
`\ + Enter`	All terminals (universal)
`Ctrl+J`	All terminals (universal)
`Shift+Enter`	iTerm2, WezTerm, Ghostty, Kitty, Warp, Apple Terminal, Windows Terminal (run /terminal-setup for VS Code, Cursor, Alacritty, Zed)
`Option+Enter`	Mac with Option as Meta enabled

TEXT EDITING (READLINE-STYLE)

Shortcut	What It Does
`Ctrl+A`	Move cursor to start of current line
`Ctrl+E`	Move cursor to end of current line
`Ctrl+K`	Delete from cursor to end of line (stores for paste)
`Ctrl+U`	Delete from cursor to line start (stores for paste)
`Ctrl+W`	Delete previous word (stores for paste)
`Ctrl+Y`	Paste previously deleted text
`Alt+Y (after Ctrl+Y)`	Cycle through paste history
`Alt+B`	Move cursor back one word
`Alt+F`	Move cursor forward one word

HISTORY SEARCH (CTRL+R MODE)

Action	How To
`Activate`	Press Ctrl+R
`Cycle older matches`	Press Ctrl+R again
`Cycle scope`	Ctrl+S (session → project → all)
`Accept + keep editing`	Tab or Esc
`Accept + execute`	Enter
`Cancel`	Ctrl+C or Backspace on empty

KEYBINDINGS CONFIGURATION

Run **/keybindings** to create or open `~/.claude/keybindings.json`. Changes hot-reload without restarting Claude Code.

Cannot be rebound (hardcoded):
Ctrl+C · Ctrl+D · Ctrl+M

TRANSCRIPT VIEWER (OPEN WITH CTRL+O)

Key	What It Does
`?`	Toggle keyboard shortcut help panel
`{ / }`	Jump to previous / next user prompt
`Ctrl+E`	Toggle "show all content" mode
`[`	Write full conversation to terminal scrollback
`v`	Open conversation in $VISUAL or $EDITOR
`q / Ctrl+C / Esc`	Exit transcript view

/BTW SIDE QUESTION OVERLAY

Ask a quick question without adding it to conversation history. Use `/btw your question here`

Key	What It Does
`Space / Enter / Esc`	Dismiss answer, return to prompt
`Up / Down`	Scroll the answer
`Left / Right`	Step between earlier /btw answers
`c`	Copy answer to clipboard as raw Markdown
`f`	Fork into new session (inherits full context and Q&A)
`x`	Clear earlier /btw exchange list

PERMISSION MODE REFERENCE

Mode	Behavior
`default`	Asks before every file edit and shell command
`acceptEdits`	Auto-approves file edits; still asks for shell commands
`plan`	Explore and propose only; no code execution
`auto`	Intelligent permission routing
`bypassPermissions`	Skip all prompts (sandboxed environments only)

Official Docs: code.claude.com/docs/en/commands

SESSION & CONTEXT MANAGEMENT

Command	What It Does
`/clear`	Clear conversation history. Aliases: /reset, /new
`/compact [focus]`	Summarize conversation to free context window with optional focus
`/context`	Visualize context window usage with optimization suggestions
`/cost`	Show token usage and session spend
`/usage`	Show plan limits and rate-limit status
`/btw <question>`	Quick side question without conversation bloat
`/resume [session]`	Reopen previous session by ID or name. Alias: /continue
`/rewind`	Roll back to checkpoint. Aliases: /checkpoint, /undo
`/export [filename]`	Export conversation as plain text
`/copy [N]`	Copy last assistant response to clipboard
`/rename [name]`	Rename the current session
`/branch [name]`	Fork conversation to new branch. Alias: /fork
`/goal [condition]`	Set a goal; Claude keeps working until met

PROJECT SETUP & MEMORY

Command	What It Does
`/init`	Initialize project with CLAUDE.md guide
`/memory`	Edit CLAUDE.md, toggle auto-memory, view entries
`/permissions`	Manage allow/ask/deny rules. Alias: /allowed-tools
`/config`	Open configuration interface. Alias: /settings
`/add-dir <path>`	Grant Claude access to an additional directory
`/keybindings`	Open or create custom keybindings file

PLANNING & EXECUTION

Command	What It Does
`/plan [description]`	Enter plan mode (explore only, no edits executed)
`/effort [level]`	Set effort: low, medium, high, xhigh, max, auto
`/model [model]`	Switch model and save as default for new sessions
`/diff`	Open interactive diff viewer for uncommitted changes
`/fast [on/off]`	Toggle fast mode on or off
`/advisor [model/off]`	Enable second model as advisor at key decision moments

CODE QUALITY & REVIEW

Command	What It Does
`/code-review [effort]`	Review diff for bugs. --fix applies findings; --comment posts to GitHub PR
`/review [PR]`	Read-only review of a GitHub pull request
`/security-review`	Analyze pending changes for security vulnerabilities
`/simplify [target]`	Parallel agents review changed code for cleanup
`/ultrareview [PR]`	Deep multi-agent cloud code review

GITHUB & CI INTEGRATION

Command	What It Does
`/install-github-app`	Walk through GitHub App + Actions workflow setup
`/install-slack-app`	Install Claude Slack app via OAuth flow
`/autofix-pr [prompt]`	Spawn session to watch PR and push fixes when CI fails
`/pr-comments [PR]`	Fetch and display GitHub PR comments (needs gh CLI)

PARALLEL AGENTS & BACKGROUND WORK

Command	What It Does
`/batch <instruction>`	Decompose large changes across parallel worktree agents
`/agents`	List and manage agent configurations
`/tasks`	View everything running in background. Alias: /bashes
`/background [prompt]`	Detach session to run as background agent. Alias: /bg
`/loop [interval] [prompt]`	Run prompt repeatedly while session stays open. Alias: /proactive
`/schedule [desc]`	Create, update, list, or run cloud scheduled tasks. Alias: /routines
`/fork <directive>`	Spawn subagent that inherits full conversation (v2.1.161+)

TOOLING & ENVIRONMENT

Command	What It Does
`/mcp`	Manage MCP server connections and OAuth auth
`/hooks`	View hook configurations for tool lifecycle events
`/skills`	List available skills in this session
`/plugin`	Install and manage Claude Code plugins
`/voice [hold/tap/off]`	Toggle voice dictation or set input mode
`/statusline`	Configure Claude Code's terminal status line
`/color [color]`	Set prompt bar color: red/blue/green/yellow/purple/orange/pink/cyan
`/terminal-setup`	Configure Shift+Enter for VS Code, Cursor, Alacritty, Zed
`/theme`	Change color theme (light, dark, daltonized, ANSI)
`/sandbox`	Toggle sandbox mode for isolated execution
`/remote-control`	Make session available for remote control. Alias: /rc
`/teleport`	Pull a cloud session into your local terminal. Alias: /tp

DIAGNOSTICS & HELP

Command	What It Does
`/help`	Show all available commands
`/doctor`	Diagnose installation issues. Press f to fix problems
`/feedback`	Submit feedback or report a bug. Alias: /bug
`/status`	Show account, model, working directory, version
`/release-notes`	View full changelog in interactive version picker
`/insights`	Generate report analyzing your Claude Code sessions
`/stats`	Visualize daily usage and session history

AUTHENTICATION & ACCOUNT

Command	What It Does
`/login / /logout`	Sign in or sign out of your Anthropic account
`/upgrade`	Open upgrade page to switch to a higher plan tier
`/usage-credits`	Configure credits to keep working at rate limits
`/privacy-settings`	View and update privacy settings (Pro/Max plans)
`/mobile`	Show QR code for Claude mobile app. Aliases: /ios, /android
`/desktop`	Continue session in Claude Code Desktop app. Alias: /app
`/exit`	Exit the CLI. Alias: /quit

COMMAND ALIASES QUICK REFERENCE

Alias	Points To
`/allowed-tools`	/permissions
`/app`	/desktop
`/bashes`	/tasks
`/bg`	/background
`/bug, /share`	/feedback
`/checkpoint, /undo`	/rewind
`/continue`	/resume
`/cost, /stats`	/usage
`/new, /reset`	/clear
`/proactive`	/loop
`/quit`	/exit
`/rc`	/remote-control
`/routines`	/schedule
`/settings`	/config
`/tp`	/teleport

Official Docs: code.claude.com/docs/en/cli-reference

CORE LAUNCH COMMANDS

Command	What It Does
`claude`	Start an interactive session
`claude "query"`	Start session with initial prompt
`claude -p "query"`	Print mode: send query and exit (no interactive session)
`cat file | claude -p`	Process piped content non-interactively
`claude -c`	Continue the most recent conversation in this directory
`claude -r "id" "query"`	Resume a specific session by ID or name
`claude update`	Update to the latest version
`claude --version / -v`	Print current version and exit

MODEL & OUTPUT FLAGS

Flag	What It Does
`--model / -m`	Set model: sonnet, opus, haiku, fable, or full model ID
`--fallback-model`	Fallback model when primary is unavailable or overloaded
`--print / -p`	Non-interactive print mode (exits after response)
`--output-format`	Format: text (default), json, stream-json
`--input-format`	Input format: text or stream-json
`--verbose`	Enable verbose logging with full turn-by-turn output
`--debug`	Enable debug-level logging
`--debug-file <path>`	Write debug logs to file (implies --debug)
`--no-color`	Disable colored output for log files or pipes

SESSION FLAGS

Flag	What It Does
`--continue / -c`	Load most recent conversation in current directory
`--resume / -r`	Resume session by ID or name (or open interactive picker)
`--fork-session`	Create new session from existing one instead of resuming
`--name / -n`	Set display name for the session
`--session-id`	Use specific session ID (must be valid UUID)
`--no-session-persistence`	Disable saving the session (print mode only)
`--from-pr`	Load context from pull request URL or number

SYSTEM PROMPT FLAGS

Flag	What It Does
`--system-prompt`	Replace the entire system prompt with custom text
`--system-prompt-file`	Load system prompt from file (mutually exclusive with --system-prompt)
`--append-system-prompt`	Add text on top of the default system prompt without replacing it
`--append-system-prompt-file`	Append file contents to default system prompt

SHORT-FORM FLAG REFERENCE

Short	Long	Purpose
`-p`	`--print`	Non-interactive print mode
`-c`	`--continue`	Continue last session
`-r`	`--resume`	Resume session by ID
`-m`	`--model`	Set model
`-v`	`--version`	Print version
`-w`	`--worktree`	Git worktree session
`-n`	`--name`	Name the session

TOOL & PERMISSION FLAGS

Flag	What It Does
`--allowedTools`	Pre-approve specific tools without prompts. Supports glob patterns: "Bash(git:*)" "Read"
`--disallowedTools`	Block specific tools for the session: "Bash(rm:*)" "Edit"
`--permission-mode`	Set mode at launch: default, plan, acceptEdits, dontAsk, bypassPermissions, auto
`--dangerously-skip-permissions`	Skip all permission prompts. Use only in sandboxed environments
`--permission-prompt-tool`	Use MCP tool to handle permission prompts programmatically

CONTEXT & DIRECTORY FLAGS

Flag	What It Does
`--add-dir`	Add working directories for file access. Can repeat for multiple paths: --add-dir ../lib ../apps
`--mcp-config`	Load MCP server configurations from a JSON file or inline string
`--strict-mcp-config`	Use only MCP servers from --mcp-config, ignore all other MCP configuration

AUTOMATION & LIMITS FLAGS

Flag	What It Does
`--max-turns <n>`	Limit agentic turns before stopping (print mode only)
`--max-budget-usd`	Cap cost budget in USD for the session
`--json-schema`	Enforce JSON schema on output — eliminates brittle text parsing
`--bare`	Skip auto-discovery of CLAUDE.md, hooks, skills, plugins, MCP. ~10x faster startup for CI
`--safe-mode`	Start with all customizations disabled for troubleshooting (v2.1.169+)

CI / AUTOMATION PATTERNS

Fastest CI pattern:

```
claude -p --bare --output-format stream-json "your query"
```

Structured JSON output:

```
claude -p --bare --output-format json --json-schema
'{"type":"object","properties":{"risk":{"type":"string"}}}'
"Review this diff"
```

The `--bare` flag skips CLAUDE.md, skills, plugins, hooks, and MCP auto-discovery, making startup roughly 10x faster for CI pipelines.

PARALLEL WORK FLAGS

Flag	What It Does
`--worktree / -w`	Start session in isolated git worktree. Changes don't affect main branch
`--tmux`	Create tmux session for the worktree (--tmux=classic for traditional tmux)
`--background / --bg`	Start as a background agent and return immediately. Prints session ID
`--agent <name>`	Specify a custom agent for the session
`--teammate-mode`	Set how team agent teammates display: in-process, auto, tmux, iterm2

CLAUDE.MD MEMORY FILE LOCATIONS

Scope	File Location
Org policy	/Library/Application Support/ClaudeCode/CLAUDE.md
User (all projects)	~/.claude/CLAUDE.md
Project (team)	./CLAUDE.md or ./.claude/CLAUDE.md
Local (gitignored)	./CLAUDE.local.md
User rules	~/.claude/rules/*.md
Project rules	./.claude/rules/*.md

HOOKS CONFIGURATION

Hook Event	Best Use Case
`SessionStart`	Load dynamic context at session start
`PreToolUse`	Log bash commands before execution; validate inputs
`PostToolUse`	Auto-format files after Write/Edit; lint after edits
`PermissionRequest`	Route prompts to Slack or another review system
`Stop`	Run checks after tasks; nudge Claude to continue
`PostCompact`	Re-inject critical instructions after compaction

Sound on completion (macOS):

```
{"hooks":{"Stop":[{"hooks":
[{"type":"command","command":"afplay /System/Library/Sounds/
Pop.aiff"}]}]}}
```

Enable: /config → Editor mode | or set "editorMode": "vim" in ~/.claude/settings.json

MODE SWITCHING

Key	From	Action
Esc	INSERT/ VISUAL	Enter NORMAL mode
i	NORMAL	Insert before cursor
I	NORMAL	Insert at line start
a	NORMAL	Insert after cursor
A	NORMAL	Insert at line end
o	NORMAL	Open new line below
O	NORMAL	Open new line above
v	NORMAL	Character-wise visual
V	NORMAL	Line-wise visual

NAVIGATION (NORMAL)

Key	Action
h / j / k / l	Left / down / up / right
w / e / b	Next word / end of word / prev word
0 / $ / ^	Line start / end / first non-blank
gg / G	Beginning / end of input
f{char} / F{char}	Jump to next / prev char occurrence
t{char} / T{char}	Jump to just before / after char
; / ,	Repeat f/F/t/T / in reverse
/	Open history search (Ctrl+R)

EDITING (NORMAL)

Key	Action
x	Delete character
dd / D	Delete line / to end of line
dw / de / db	Delete word / to end / back
cc / C	Change line / to end of line
cw / ce / cb	Change word / to end / back
yy / Y	Yank (copy) line
yw / ye / yb	Yank word / to end / back
p / P	Paste after / before cursor
>> / <<	Indent / dedent line
J	Join lines
u	Undo last change
.	Repeat last change

VISUAL MODE

Key	Action
d / x	Delete selection
y	Yank (copy) selection
c / s	Change selection
p	Replace with register contents
r{char}	Replace each char with {char}
~ / u / U	Toggle / lowercase / uppercase
> / <	Indent / dedent selected lines
J	Join selected lines
o	Swap cursor and anchor

TEXT OBJECTS (WITH D, C, Y)

Object	Selects
iw / aw	Inner / around word
iW / aW	Inner / around WORD (whitespace-bounded)
i" / a"	Inner / around double quotes
i' / a'	Inner / around single quotes
i(/ a(	Inner / around parentheses
i[/ a[	Inner / around brackets
i{ / a{	Inner / around braces

VIM SETTINGS & NOTES

Enable via settings file:
`"editorMode": "vim"`
in ~/.claude/settings.json

Or use: /config → Editor mode

Note: Ctrl+V visual block mode is NOT supported. Use v for character-wise or V for line-wise visual selection.

Ctrl keys pass through to the application keybinding layer even in vim NORMAL mode.

KEYBINDINGS JSON REFERENCE

Run **/keybindings** to create `~/.claude/keybindings.json`

Changes hot-reload without restart.

Cannot be rebound:
Ctrl+C · Ctrl+D · Ctrl+M

PARALLEL SESSIONS WITH WORKTREES

Start isolated session:
`claude --worktree feature-auth --tmux`

Changes happen in a separate git worktree branch. Your main branch stays untouched. Use **/color** and **/rename** to identify each parallel pane.

Use **/batch** for large-scale changes. It decomposes the task into independent units, sends each to a separate worktree agent, and merges results automatically.

CONTEXT HYGIENE

Use **/clear** between unrelated tasks. Context bleed from an earlier conversation causes wrong assumptions in the next one.

Use **/compact "keep only the auth changes"** with a specific focus instruction. It tells Claude what to preserve in the summary, so important context survives compression.

Check **/context** before starting a long task. Compact before you hit limits, not after Claude loses critical context mid-task.

CLAUDE.MD BEST PRACTICES

Run **/init** to auto-generate CLAUDE.md from your codebase.

Use `@path/to/file` inside CLAUDE.md to import other files (max 4 hops deep).

HTML comments `<!-- notes -->` get stripped from Claude's context — use them for maintainer notes.

CLAUDE.md survives **/compact**. After any correction: tell Claude "Update your CLAUDE.md so you don't make that mistake again."

MOBILE & REMOTE ACCESS

/teleport or `claude --teleport` moves your session between devices.

/remote-control lets you control a local machine session from mobile or web browser.

/desktop continues your current session in the Claude Code Desktop app (macOS and Windows).

VERIFICATION LOOPS

Give Claude a feedback mechanism to verify its own work. Browser automation via the Chrome extension, test suites via bash, and screenshot diffing all work as verification tools.

Prompts that push self-verification:
"Prove to me this works"
"Run the tests and fix any failures"
"Knowing everything you know now, is this the right implementation?"

Use **/simplify** after changes to run parallel cleanup agents over the modified code.

OFFICIAL DOCUMENTATION LINKS

Topic	URL
Keyboard Shortcuts	code.claude.com/docs/en/interactive-mode
Custom Keybindings	code.claude.com/docs/en/keybindings
Slash Commands	code.claude.com/docs/en/commands
CLI Reference	code.claude.com/docs/en/cli-reference
Memory / CLAUDE.md	code.claude.com/docs/en/memory
Hooks	code.claude.com/docs/en/hooks
GitHub Actions	code.claude.com/docs/en/github-actions
Settings	code.claude.com/docs/en/settings
Official Cheatsheet	support.claude.com (search "cheatsheet")

MCP TIPS

Commit `.mcp.json` to git for team-shared MCP server configs. Each connected server consumes context tokens, so remove unused servers: `claude mcp remove <name>`

Add MCP servers in `~/.claude.json` for personal use across all projects, or in project settings for team sharing.

PROMPTSLOVE

The World's Largest AI Prompt Library

Get 20,000+ AI Prompts for Every Tool and Task

ChatGPT, Claude, Midjourney, Stable Diffusion, and every major AI tool. Expertly crafted prompts that actually work, organized by use case, and updated weekly.

20,000+	50+	100k+
AI PROMPTS	AI TOOLS	USERS

- ChatGPT prompts for writing, coding, business, marketing, and creative work
- Midjourney and image AI prompts with style references and parameters
- Claude Code prompts, workflows, and custom slash commands
- New prompts and cheat sheets added every week

members.promptslove.com/login

promptslove.com · Your AI Prompt Library