

Claude Code Cheatsheet

Daily essentials for maximum productivity

v3.8.0 | January 2026

How to Install

```
npm install -g @anthropic/claude-code
# Verify installation
which claude && claude --version
```

Health Check: `claude doctor && claude mcp list`

Essential Commands

Command	Action
<code>/help</code>	Contextual help
<code>/clear</code>	Reset conversation
<code>/compact</code>	Free up context
<code>/status</code>	Session state + context usage
<code>/context</code>	Detailed token breakdown
<code>/plan</code>	Enter Plan Mode (no changes)
<code>/execute</code>	Exit Plan Mode (apply changes)
<code>/model</code>	Switch model (sonnet/opus/opusplan)
<code>/exit</code>	Quit (or Ctrl+D)

Keyboard Shortcuts

Shortcut	Action
<code>Shift+Tab</code>	Cycle permission modes
<code>Esc × 2</code>	Rewind (undo)
<code>Ctrl+C</code>	Interrupt
<code>Ctrl+R</code>	Retry last operation
<code>Ctrl+L</code>	Clear screen (keeps context)
<code>Tab</code>	Autocomplete
<code>Shift+Enter</code>	New line
<code>Ctrl+D</code>	Exit

IDE Shortcuts: VS Code `Alt+K` | JetBrains `Cmd+Option+K`

File References

```
@path/to/file.ts    → Reference a file
@agent-name         → Call an agent
!shell-command      → Run shell command
```

Permission Modes

Mode	Editing	Execution
Default	Asks	Asks
Auto-accept	Auto	Asks
Plan Mode	✗	✗

Shift+Tab to switch modes

Memory & Settings (2 levels)

Level	Path	Scope	Git
Project	<code>.claude/</code>	Team	✓
Personal	<code>~/.claude/</code>	You (all projects)	✗

Priority: Project overrides Personal

`.claude/` Folder Structure

```
.claude/
├── CLAUDE.md           # Local memory
├── (gitignored)
├── settings.json       # Hooks (committed)
├── settings.local.json # Permissions (not committed)
├── agents/             # Custom agents
├── commands/          # Slash commands
└── skills/             # Knowledge modules
```

Context Management (CRITICAL)

Statusline: `Model: Sonnet | Ctx: 89.5k | Ctx(u): 56.0%`



Watch `Ctx(u):` → `>70% = /compact`, `>85% = /clear`

Sign	Action
Short responses	<code>/compact</code>
Frequent forgetting	<code>/clear</code>
>70% context	<code>/compact</code>
Task complete	<code>/clear</code>

Context Recovery Commands

Command	Usage
<code>/compact</code>	Summarize and free context
<code>/clear</code>	Fresh start
<code>/rewind</code>	Undo recent changes
<code>claude -c</code>	Resume last session
<code>claude -r <id></code>	Resume specific session

Plan Mode & Thinking Depth

Feature	Activation	Usage
Plan Mode	<code>Shift+Tab × 2</code> or <code>/plan</code>	Explore without modifying
OpusPlan	<code>/model opusplan</code>	Opus for planning, Sonnet for execution

Extended thinking uses prompt keywords (not CLI flags):

Prompt Keyword	Thinking Depth	Use For
“Think”	Standard	Multi-component analysis
“Think hard”	Deep	Architectural decisions
“Ultrathink”	Maximum	Critical redesign

Typical Workflow

1. Start session	→ <code>claude</code>
2. Check context	→ <code>/status</code>
3. Plan Mode	→ <code>Shift+Tab × 2</code> (for complex tasks)
4. Describe task	→ Clear, specific prompt
5. Review changes	→ Always read the diff!
6. Accept/Reject	→ <code>y/n</code>
7. Verify	→ Run tests
8. Commit	→ When task complete
9. <code>/compact</code>	→ When context >70%

Quick Prompting Formula

WHAT: [Concrete deliverable]
WHERE: [File paths]
HOW: [Constraints, approach]
VERIFY: [Success criteria]

Example:

Add input validation to the login form.
WHERE: <code>src/components/LoginForm.tsx</code>
HOW: Use Zod schema, show inline errors
VERIFY: Empty email shows error, invalid format shows error

MCP Servers

Server	Purpose
Serena	Indexation + session memory + symbol search
mgrep	Semantic search by intent
Context7	Library documentation
Sequential	Structured reasoning
Playwright	Browser automation
Postgres	Database queries

Serena memory: `write_memory()` / `read_memory()` / `list_memories()`

Check status: `/mcp`

CLI Flags Quick Reference

Flag	Usage
<code>-p "query"</code>	Non-interactive mode (CI/CD)
<code>-c / --continue</code>	Continue last session
<code>-r / --resume <id></code>	Resume specific session
<code>--model sonnet</code>	Change model
<code>--add-dir ../lib</code>	Allow access outside CWD
<code>--permission-mode plan</code>	Plan mode
<code>--dangerously-skip-permissions</code>	Auto-accept (use carefully)
<code>--debug</code>	Debug output

Anti-patterns

✗ Don't

- Vague prompts
- Accept without reading
- Ignore warnings
- Skip permissions
- Negative constraints only

✓ Do

- Specify file + line with @references
- Read every diff
- Use /compact at 70%
- Never in production
- Provide alternatives

6

Commit frequently after each completed task

7

Know what's sent — prompts, files, MCP results → Anthropic

Cost Optimization

Model	Use For	Cost
Haiku	Simple fixes, reviews	\$
Sonnet	Most development, Architecture, complex bugs!	\$
OpusPlan	Plan (Opus) + Execute (Sonnet)	\$\$

Quick Decision Tree

Simple task

→ Just ask Claude

Complex task

→ TodoWrite to plan first

Risky change

→ Plan Mode first

Repeating task

→ Create agent or command

Context full

→ /compact or /clear

Need docs

→ Use Context7 MCP

Deep analysis prompts

→ Use extended thinking

The Golden Rules

1

Always review diffs before accepting

2

Use /compact before context gets critical (>70%)

3

Be specific in requests (WHAT, WHERE, HOW, VERIFY)

4

Plan Mode first for complex/risky tasks

5

Create CLAUDE.md for every project

Common Issues Quick Fix

Problem	Solution
“Command not found”	Check PATH, reinstall npm global
Context too high (>70%)	/compact immediately
Slow responses	/compact or /clear
MCP not working	claude mcp list , check config
Permission denied	Check settings.local.json

Health Check:

which claude && claude doctor && claude mcp list

Author: Florian BRUNIAUX | [Méthode Aristote](#) | Written with Claude

Version 3.8.0 | January 2026