

# Claude Code Cheatsheet

---

Daily essentials for maximum productivity

Florian BRUNIAUX

March 2026

v3.29.2



# Table of contents

1	How to Install .....	2
2	Essential Commands .....	3
3	Keyboard Shortcuts .....	4
4	File References .....	5
5	Features Méconnues (But Official!) .....	6
6	Permission Modes .....	7
7	Memory & Settings (2 levels) .....	8
7.1	.claude/ Folder Structure .....	8
8	Context Management (CRITICAL) .....	9
8.1	Context Recovery Commands .....	11
9	Plan Mode & Thinking .....	12
10	Typical Workflow .....	13
11	Quick Prompting Formula .....	14
12	MCP Servers .....	15
13	CLI Flags Quick Reference .....	16
14	Anti-patterns .....	17
15	Cost Optimization .....	18
16	Quick Decision Tree .....	19
17	Common Issues Quick Fix .....	20

# 1 How to Install

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

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

## 2 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>/insights</code>	Usage analytics + optimization
<code>/teleport</code>	Teleport session from web
<code>/tasks</code>	Monitor background tasks
<code>/fast</code>	Toggle fast mode (2.5x speed, 6x cost)
<code>/debug</code>	Systematic troubleshooting
<code>/remote-env</code>	Configure cloud environment
<code>/simplify</code>	Detect over-engineering in changed code + auto-fix
<code>/batch</code>	Large-scale refactors via 5–30 parallel worktree agents
<code>/exit</code>	Quit (or Ctrl+D)

## 3 Keyboard Shortcuts

Shortcut	Action
Shift+Tab	Cycle permission modes
Esc × 2	Rewind (undo)
Ctrl+C	Interrupt
Ctrl+R	Search command history
Ctrl+L	Clear screen (keeps context)
Ctrl+B	Background tasks
Alt+T	Toggle thinking on/off
Tab	Autocomplete
Shift+Enter	New line
Ctrl+D	Exit

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

## 4 File References

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

## 5 Features Méconnues (But Official!)

Feature	Since	What It Does
<b>Tasks API</b>	v2.1.16	Persistent task lists with dependencies
<b>Background Agents</b>	v2.0.60	Sub-agents work while you code
<b>Agent Teams</b>	v2.1.32	Multi-agent coordination (TeamCreate/SendMessage)
<b>Auto-Memories</b>	v2.1.32	Automatic cross-session context capture
<b>Session Forking</b>	v2.1.19	Rewind + create parallel timeline
<b>LSP Tool</b>	v2.0.74	Code intelligence (go-to-def, refs)
<b>Rules Templates</b>	v3.27.4	4 ready-to-use <code>.claude/rules/</code> templates (arch/quality/test/perf)
<b><code>/review-plan</code></b>	v3.27.4	Structured plan review across 4 axes before coding
<b><code>/simplify</code></b>	v2.1.63	Architecture-level over-engineering detection + auto-fix
<b><code>/batch</code></b>	v2.1.63	Spawns 5–30 agents in isolated work-trees, each opens a PR

**Pro tip:** These aren't "secrets"—they're in the [CHANGELOG](#). Read it!

## 6 Permission Modes

Mode	Editing	Execution
Default	Asks	Asks
Auto-accept	Auto	Asks
Plan Mode	None	None

**Shift+Tab** to switch modes

## 7 Memory & Settings (2 levels)

Level	Path	Scope	Git
Project	<code>.claude/</code>	Team	Yes
Personal	<code>~/ .claude/</code>	You (all projects)	No

**Priority:** Project overrides Personal

### 7.1 `.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
├── hooks/             # Event scripts
├── rules/             # Auto-loaded rules (v3.26+)
│   ├── architecture-review.md # ← templates ready in examples/rules/
│   ├── code-quality-review.md
│   ├── test-review.md
│   └── performance-review.md
└── skills/            # Knowledge modules
```

## 8 Context Management (CRITICAL)

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

✔ 0-50%	⚠ 50-70%	◆ 70-90%	🔴 90%+
---------	----------	----------	--------

**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>

## 8.1 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 &lt;id&gt;</code>	Resume specific session

## 9 Plan Mode & Thinking

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

**Opus 4.6:** Thinking is ON by default at max budget. Keywords like “ultrathink” are cosmetic only.

Control	Action	Persistence
<b>Alt+T</b>	Toggle thinking on/off	Session
<b>/config</b>	Enable/disable globally	Permanent
<code>effort</code> <b>param</b>	API only: low/medium/high/max	Per-request

**Cost tip:** For simple tasks, Alt+T to disable thinking → faster & cheaper.

## 10 Typical Workflow

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

# 11 Quick Prompting Formula

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

## Example:

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

## 12 MCP Servers

Server	Purpose
<b>Serena</b>	Indexation + session memory + symbol search
<b>grepai</b>	Semantic search + call graph analysis
<b>Context7</b>	Library documentation
<b>Sequential</b>	Structured reasoning
<b>Playwright</b>	Browser automation
<b>Postgres</b>	Database queries
<b>doobidoo</b>	Semantic memory + Knowledge Graph

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

Check status: `/mcp`

## 13 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 &lt;id&gt;</code>	Resume specific session
<code>--teleport</code>	Teleport session from web
<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>--mcp-debug</code>	Debug MCP servers
<code>--allowedTools "Edit,Read"</code>	Whitelist tools
<code>--debug</code>	Debug output

# 14 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

## 15 Cost Optimization

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

# 16 Quick Decision Tree

Simple task	→ Just ask Claude
Complex task	→ Tasks API 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	→ Use Opus (thinking on by default)

## 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
- 6 Commit frequently after each completed task
- 7 Know what's sent — prompts, files, MCP results → Anthropic

# 17 Common Issues Quick Fix

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

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

---

**Author:** Florian BRUNIAUX | [Méthode Aristote](#) | Written with Claude  
Version 3.29.2 | March 2026