Claude Cheatsheet

ANTHROPIC · 2026 EDITION

Everything you need to master Claude models, Claude Code, prompting, and MCP integrations.

- Opus 4.6 — Most Capable
- Sonnet 4.6 — Best Value
- Haiku 4.5 — Fastest
- Claude Code — Agentic Dev

MARCH 2026

MODEL LINEUP

Claude Opus 4.6 — 200K ctx

`claude-opus-4-6`

Extended Thinking · Best Reasoning · Vision · Tool Use

Claude Sonnet 4.6 — 200K ctx

`claude-sonnet-4-6`

Speed + Quality · Vision · Tool Use · Code Default

Claude Haiku 4.5 — 200K ctx

`claude-haiku-4-5-20251001`

Fastest · Tool Use · Low Cost · High Volume

KEY NUMBERS

200K	8K	∞
TOKEN WINDOW	MAX OUTPUT	MCP SERVERS

CORE CAPABILITIES

- Vision & image analysis
- Multi-step tool use
- Extended thinking mode
- Code gen & execution
- PDF & doc parsing
- MCP integrations
- Web search (w/ tools)
- Agentic sub-agents

EXTENDED THINKING

Enable with **thinking: {type:"enabled", budget_tokens: N}**. Claude reasons step-by-step before responding. Use for math, logic, long planning tasks. Opus 4.6 supports up to **64K thinking tokens**. Billed at normal input rates.

```
// API: enable extended thinking
"thinking": {
  "type": "enabled",
  "budget_tokens": 10000
}
```

CLAUDE CODE — POWER TIPS

CLAUDE.MD

Drop a `CLAUDE.md` in any directory. Claude reads it automatically for persistent context, coding standards, and project rules every session.

SUB-AGENTS / PARALLELISM

Use the `Agent` tool to spin up parallel sub-agents. Run research + coding + tests simultaneously — massive time savings on big tasks.

HOOKS

Configure hooks in `settings.json` to auto-run linters, formatters, or scripts before/after tool calls. Fully automated workflows.

MCP SERVERS

Add any MCP server via `claude mcp add` CLI or `settings.json`. Gives Claude new superpowers: Figma, GitHub, Slack, databases, and more.

! SHELL SHORTCUT

Type `! command` in the Claude prompt to run a shell command inline. Output flows directly into the conversation context.

FAST MODE

Toggle `/fast` to enable faster streaming output with Opus 4.6. Same model, optimized for speed. Great for quick iterations.

PROJECT MEMORY

Use the auto-memory system at `~/.claude/projects/` to persist knowledge across sessions. Memories survive context resets.

KEYBOARD SHORTCUTS

↑	Navigate previous commands
`Ctrl+C`	Cancel current operation
`Ctrl+Z`	Undo last file edit
`Ctrl+R`	Search command history
`Shift+Tab`	Auto-accept all diffs (be careful!)
`Esc`	Exit current mode / cancel
`/help`	Show all slash commands
`/clear`	Clear conversation context
`/compact`	Summarize & compress context
`/review-pr`	Review GitHub pull request
`/mcp`	List connected MCP server status
`/cost`	Show token usage & cost for session
`/doctor`	Diagnose environment issues
`/model`	Switch model mid-session
`/permissions`	View & manage tool permissions

CLI FLAGS

`claude -p "..."`	One-shot non-interactive mode
`claude -c`	Continue last conversation
`--resume <id>`	Resume a specific past session

FIGMA MCP — SETUP GUIDE

Model Context Protocol

1 GET FIGMA API TOKEN

Figma → Settings → Account → Personal Access Tokens → Generate new token. Copy it — shown once only.

2 ADD MCP VIA CLI

Run in terminal:

```
# Add Figma MCP server globally
claude mcp add figma \
  -- npx -y figma-developer-mcp \
  --figma-api-key figd_XXXX --stdio
```

3 OR EDIT SETTINGS.JSON DIRECTLY

File: `~/.claude/settings.json`

```
// ~/.claude/settings.json
{
  "mcpServers": {
    "figma": {
      "type": "stdio",
      "command": "npx",
      "args": ["-y",
        "figma-developer-mcp",
        "--figma-api-key", "figd_XXXX",
        "--stdio"]
    }
  }
}
```

4 WHAT YOU CAN DO

Ask Claude to: inspect Figma components, extract design tokens, read layouts, generate code from designs, and sync styles — all in the CLI.

5 VERIFY CONNECTION

Start Claude Code and ask: *"List my Figma files"* or *"Get the design tokens from [Figma URL]"*

LATEST FEATURES (2025–2026)

Web Search Tool
Real-time web browsing via tool use — live data, no cutoff

Computer Use
Control a real desktop: click, type, screenshot, navigate apps

Parallel Agents
Spawn multiple sub-agents working simultaneously on subtasks

MCP Ecosystem
100s of MCP servers: GitHub, Slack, Linear, Figma, Postgres, and more

Streaming Tool Use
Real-time streaming during long tool calls and agentic loops

Prompt Caching
Cache large system prompts — up to 90% cost reduction on repeat calls

Batch API
Process 10k+ requests async with 50% cost discount via Message Batches

Admin API
Enterprise controls: user mgmt, audit logs, SSO, usage policies

`--model <id>` Force a specific model

✍️ PROMPTING BEST PRACTICES

🎯 BE SPECIFIC

Include file paths, function names, error messages. The more context, the better the result. Vague → vague.

📋 XML TAGS FOR STRUCTURE

Use `<context>`, `<instructions>`, `<example>` tags to clearly separate parts of your prompt.

🔄 ITERATE, DON'T RESTART

Build on responses. "Now make it faster", "Add error handling", "Refactor this function" — conversation context is your superpower.

🔑 API QUICK REFERENCE

```
# Install SDK
npm install @anthropic-ai/sdk

// Minimal API call
import Anthropic from '@anthropic-ai/sdk';
const client = new Anthropic();
const msg = await client.messages.create({
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  messages: [{
    role: "user",
    content: "Hello!"
  }]
});
```

`ANTHROPIC_API_KEY`	Set env var — SDK auto-detects
`max_tokens`	Required — set to 8192 for long output
`system`	Top-level param (not in messages array)
`stream: true`	Enable streaming responses
`tools: [...]`	Define JSON Schema tool definitions