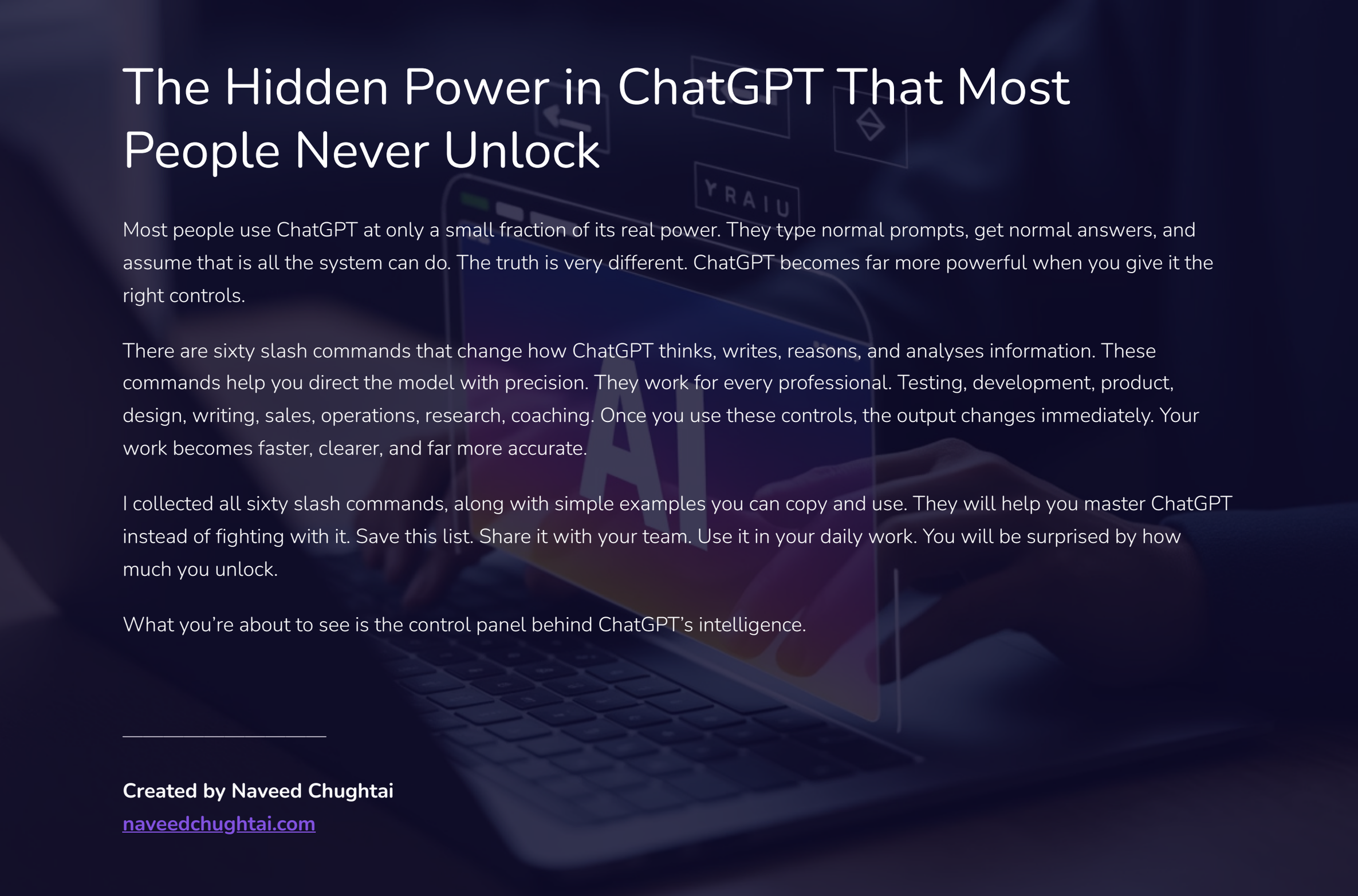
The Hidden Power in ChatGPT That Most People Never Unlock



Most people use ChatGPT at only a small fraction of its real power. They type normal prompts, get normal answers, and assume that is all the system can do. The truth is very different. ChatGPT becomes far more powerful when you give it the right controls.

There are sixty slash commands that change how ChatGPT thinks, writes, reasons, and analyses information. These commands help you direct the model with precision. They work for every professional. Testing, development, product, design, writing, sales, operations, research, coaching. Once you use these controls, the output changes immediately. Your work becomes faster, clearer, and far more accurate.

I collected all sixty slash commands, along with simple examples you can copy and use. They will help you master ChatGPT instead of fighting with it. Save this list. Share it with your team. Use it in your daily work. You will be surprised by how much you unlock.

What you're about to see is the control panel behind ChatGPT's intelligence.

Created by Naveed Chughtai
naveedchughtai.com

Communication Control Commands

These commands control how ChatGPT communicates with you. They adjust length, complexity, and style to match your exact needs.

/briefly – short, sharp answers

Example: "/briefly Explain regression testing."

/jargon – expert language

Example: "/jargon Describe SQL indexing."

/audience – tailor to a specific reader

Example: "/audience Explain cloud testing to a product owner."

/eli5 – explain simply

Example: "/eli5 What is a bug report."

/tldl – summarize long content

Example: "/tldl Summarize this 15 page document."

Structure and Format Commands

These commands transform information into specific formats. They help you organize content exactly how you need it.

/step_by_step – break a process down

Example: "/step_by_step Show me how to design a test case."

/checklist – convert into action

Example: "/checklist Create a smoke test checklist."

/exec_summary – high level summary

Example: "/exec_summary Summarize the QA status."

/format_as – generate structured formats

Example: "/format_as Table Compare manual testing and automation."

/schema – create data models

Example: "/schema Design a bug report schema."

/outline – create structured outlines

Example: "/outline Build a test strategy outline."

/mindmap – generate mind map structure

Example: "/mindmap Create a mind map for login testing."

/roadmap – produce a roadmap

Example: "/roadmap Create a 90 day QA improvement roadmap."

Perspective and Role Commands

These commands change the perspective ChatGPT uses. They let you see problems from different angles and professional viewpoints.

/act_as – respond in a defined persona

Example: "/act_as You are a QA lead. Review this test plan."

/dev_mode – answer like an engineer

Example: "/dev_mode Explain why this API fails."

/pm_mode – answer like a PM

Example: "/pm_mode Break this into milestones."

/multi_perspective – multiple viewpoints

Example: "/multi_perspective Explain flaky tests from QA, Dev, and DevOps views."

/parallel_lenses – view through different angles

Example: "/parallel_lenses Analyse this defect from UX, security, and performance."

Analysis and Comparison Commands

These commands are powerful for debugging, RCA, and decision-making.

These commands help you analyze information deeply and compare options side by side.

/swot – strengths, weaknesses, opportunities, threats

Example: "/swot Analyse our automation framework."

/compare – side by side comparison

Example: "/compare Cypress vs Playwright."

/risk_analysis – find risks and mitigation

Example: "/risk_analysis Analyse release risks."

/bias_check – find blind spots

Example: "/bias_check Review this analysis for bias."

/pitfalls – list mistakes

Example: "/pitfalls What are common mistakes in test automation."

/validate – check correctness

Example: "/validate Verify if these test cases cover all paths."

Reasoning and Thinking Commands

These commands control how ChatGPT thinks through problems. They activate deeper reasoning and more careful analysis.

01

/reflective_mode – self evaluate

Example: "/reflective_mode Review this test plan and improve it."

02

/deliberate_thinking – slower, clearer reasoning

Example: "/deliberate_thinking Analyse this intermittent bug."

03

/no_autopilot – avoid generic responses

Example: "/no_autopilot Give specific QA improvements."

04

/eval_self – critique its own answer

Example: "/eval_self Improve this bug summary."

05

/first_principles – break to fundamentals

Example: "/first_principles Explain SDLC from zero."

06

/chain_of_thought – show reasoning

Example: "/chain_of_thought Explain how you reached this conclusion."

Pro tip: Combine reasoning commands with analysis commands for the most powerful results. Try "/deliberate_thinking /chain_of_thought Analyse this complex bug."

Content Control Commands

These commands give you precise control over content structure, tone, and boundaries.

/tone – change the writing style

Example: "/tone Rewrite this in a confident tone."

/context_stack – remember layered context

Example: "/context_stack Remember these modules for future test cases."

/begin_with – control the start

Example: "/begin_with Start with the main issue."

/end_with – control the ending

Example: "/end_with End with recommendations."

/role_task_format – strict structure

Example: "/role_task_format Role: QA Lead. Task: Improve regression plan. Format: Checklist."

/rewrite_as – rewrite in a specific style

Example: "/rewrite_as Rewrite this as a formal explanation."

/guardrail – limit scope

Example: "/guardrail Stay within testing only."

/constraints – set strict limits

Example: "/constraints Only answer using QA examples."

/expand – add more depth

Example: "/expand Add more login edge cases."

/shrink – shorten without losing meaning

Example: "/shrink Shorten this test plan."

/organize – clean structure

Example: "/organize Organize these test ideas."

/refactor – clean up content or code

Example: "/refactor Improve this automation script."

Practical Work Commands

These commands help you get real work done. They generate actionable outputs you can use immediately.

/examples – provide examples

Example: "/examples Give 3 negative test cases for login."

/sources – list references

Example: "/sources Give sources for performance testing."

/prioritize – rank by impact

Example: "/prioritize Rank these test cases."

/code_translate – convert code

Example: "/code_translate Convert this Python script to JS."

/test_cases – generate tests

Example: "/test_cases Write test cases for the login() function."

/timebox – break work into time blocks

Example: "/timebox Split this QA task into 20 minute blocks."

/faq – build Q and A

Example: "/faq Create an FAQ for new testers."

/next_actions – identify next steps

Example: "/next_actions Give next 3 steps for this issue."

Creative and Problem-Solving Commands

These commands unlock creative thinking and help you solve problems in new ways.

/brainstorm – generate ideas

Example: "/brainstorm Give 10 test ideas for search."

/debug – help find issues

Example: "/debug Help me find why this API returns a null payload"

/simulate – role play or simulate

Example: "/simulate A user entering invalid data."

/story_mode – human readable narrative

Example: "/story_mode Explain how this bug happens."

/socratic – guided questioning

Example: "/socratic Help me think through this flaky test."

/improve_logic – strengthen reasoning

Example: "/improve_logic Fix this weak root cause analysis."

/prompt_tune – refine the prompt

Example: "/prompt_tune Improve this test scenario request."

Advanced Utility Commands

These final commands provide specialized utilities for specific tasks. They round out your complete command toolkit.

/metrics_mode – add KPIs and data

Example: "/metrics_mode Give QA success metrics."

/translate – translate into another language

Example: "/translate Convert these test cases into Urdu."

/reverse_prompt – guess the prompt

Example: "/reverse_prompt Guess the prompt for this test report."

> **Master these sixty commands and you will transform how you work with ChatGPT.** Each command unlocks a different capability. Combine them for even more power. Your prompts will become precise tools instead of vague requests. Your results will become exactly what you need instead of close approximations.

Save this list. Reference it daily. Share it with colleagues. The difference between basic ChatGPT use and expert use is knowing these controls exist and using them deliberately. You now have the complete toolkit. Use it.