From Bad to Great Prompts

7 real examples — see how vague requests become precision-engineered prompts

🟥 PERSONA 🟩 TASK 🟪 CONTEXT 🟧 EXEMPLAR 🟩 FORMAT 🟦 CONSTRAINTS

01 Code Review Guide

① BAD PROMPT — written by someone new to AI

🟥 WEAK

```
write me something about code reviews that i can send to my
team. make it not too long and helpful thanks
```

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA
`"write me something…"`
`→ Who should Claude be?`
Define the role or expertise Claude should adopt. It shapes tone, vocabulary, and depth.

TASK
`"…something about code reviews…"`
`→ What exactly to produce?`
Name the deliverable — a guide, checklist, Slack post. Vague nouns give vague output.

CONTEXT
`"…send to my team…"`
`→ Who is this for & why now?`
Team size, skill level, current pain points. Context lets Claude tailor advice that actually fits.

EXEMPLAR
`(missing entirely)`
`→ What does good look like?`
A sample tone or before/after example calibrates style faster than any description.

FORMAT
`"…not too long…"`
`→ Concrete structure & length?`
Specify headings, bullets vs prose, word count. "Not too long" means nothing without numbers.

CONSTRAINTS
`"…helpful thanks"`
`→ What to avoid or require?`
Hard rules: no jargon, must include real examples, avoid generic advice like "be nice".

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

[PERSONA] You are a senior software engineer and engineering manager with 10+ years leading teams.

[TASK] Write a practical guide on giving constructive code review feedback for the next PR cycle.

[CONTEXT] Team is 6 mid-level .NET/C# devs — technically solid but leave vague, blunt comments.

[EXEMPLAR] "Consider extracting this into a helper method — it would clarify intent and simplify testing."

[FORMAT] 1-sentence intro, 5 named principles each with explanation + concrete example. Under 350 words.

[CONSTRAINT] No academic language. Every principle must include a real PR scenario. No bullet-point walls.

02 Bug Report

① BAD PROMPT — written by someone new to AI

■ WEAK

```
help me write a bug report for this issue we found. its about
the login not working sometimes
```

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA

`"help me write…"`

`→ No role defined`

Is Claude a QA engineer? A developer? A technical writer? Each writes bug reports very differently.

TASK

`"…a bug report…"`

`→ For which system?`

GitHub Issues, Jira, Linear, or an internal doc? Each has different conventions and required fields.

CONTEXT

`"…login not working sometimes"`

`→ What's actually known?`

Frequency, reproduction steps, environment, what was already tried. Without this, the report is useless.

EXEMPLAR

`(missing entirely)`

`→ What's the team's standard?`

A real previous bug report from the tracker shows the expected structure and level of detail.

FORMAT

`(nothing mentioned)`

`→ Which fields are required?`

Severity, steps to reproduce, expected vs actual, environment, logs — define the sections explicitly.

CONSTRAINTS

`(nothing mentioned)`

`→ Who will read this?`

Is this for a developer, a PM, or a client? Only include confirmed facts — no speculation.

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

[PERSONA] You are a QA engineer experienced with writing clear, developer-ready bug reports.

[TASK] Write a formal bug report for a login intermittency issue for our GitHub Issues tracker.

[CONTEXT] Users randomly logged out mid-session on ASP.NET Core app. 1 in 10 sessions, Chrome only, since Friday's deploy.

[EXEMPLAR] Structure: Title, Severity, Environment, Steps to Reproduce, Expected, Actual, Logs, Possible Cause.

[FORMAT] Markdown with headers. Title pattern: "[Bug] — ". 2-4 lines per section.

[CONSTRAINT] Label anything unconfirmed as 'hypothesis'. No personal names. Neutral technical language only.

03 Blog Post

① BAD PROMPT — written by someone new to AI

■ WEAK

```
write a blog post about AI tools for developers. make it interesting
```

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA

```
(no voice defined)
→ Who is writing this?
```

A practitioner's voice is very different from an analyst's. The author persona sets credibility and tone.

TASK

```
"…blog post about AI tools…"
→ What angle?
```

"AI tools for developers" is enormous. An opinion piece? A comparison? A tutorial? Pick one angle.

CONTEXT

```
"…for developers…"
→ Which developers?
```

Frontend? Backend? What level? What do they already know? What's their biggest frustration?

EXEMPLAR

```
"…make it interesting"
→ Interesting like what?
```

Link to a post with the energy you want — dry-but-insightful, opinionated, conversational. "Interesting" is subjective.

FORMAT

```
(nothing specified)
→ Structure and word count?
```

How long? TL;DR? Code snippets? Subheadings? A CTA? Blog posts need skeleton-first thinking.

CONSTRAINTS

```
(nothing)
→ What to avoid?
```

No hype language, no listicle padding, must reference real experience, avoid obscure tools.

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

[PERSONA] You are a working .NET developer with a tech blog and hands-on experience with AI coding assistants.

[TASK] Write an opinionated comparison of GitHub Copilot vs Cursor for backend developers.

[CONTEXT] Audience: mid-to-senior .NET/C# devs, skeptical of hype, value honest trade-offs over marketing.

[EXEMPLAR] Direct, slightly dry tone — like a senior dev at a team lunch. Reference workflow, not feature lists.

[FORMAT] 2-sentence intro, 3 sections (Copilot wins / Cursor wins / verdict), closing. 500-650 words. Prose only.

[CONSTRAINT] No "game-changer" or "revolutionary". One concrete scenario per section. No definitive winner.

04 API Documentation

① BAD PROMPT — written by someone new to AI

■ WEAK

```
can you document this API endpoint for me? its a POST endpoint
that creates a user
```

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA

`"can you document…"`

`→ What style of writer?`

Stripe-style docs feel different from AWS docs. Is Claude a technical writer or a developer writing for devs?

TASK

`"…document this API endpoint…"`

`→ What level of docs?`

Reference page, getting-started guide, inline comment, or OpenAPI spec block? Totally different outputs.

CONTEXT

`"…POST that creates a user"`

`→ What are the actual fields?`

Request body schema, required vs optional fields, authentication, error codes — all missing here.

EXEMPLAR

`(nothing provided)`

`→ What's the doc style?`

Paste an existing endpoint. Consistency with existing pages is critical in API docs.

FORMAT

`(nothing specified)`

`→ Markdown, HTML, OpenAPI?`

Define sections: description, request, response, error codes, code example. Specify the output format.

CONSTRAINTS

`(nothing)`

`→ What standards apply?`

REST conventions, no internal jargon, must document all error codes, show both success and failure responses.

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

```
[PERSONA] You are a technical writer who specialises in clean,
developer-friendly REST API documentation.

[TASK] Write a reference documentation page for the POST /api/users endpoint
for our public developer portal.

[CONTEXT] Fields: email, password, role (admin|member), optional displayName.
Returns 201/400/409. Bearer token required.

[EXEMPLAR] Structure: Description paragraph → Request table → Response JSON
blocks → curl code sample.

[FORMAT] Markdown. Table: Name | Type | Required | Description. Two JSON
examples: 201 success and 400 error. One curl snippet.

[CONSTRAINT] No tutorial narrative. Present tense only. No internal
implementation details. Field descriptions: one sentence max.
```

05 Job Ad

① BAD PROMPT — written by someone new to AI

■ WEAK

```
write a job posting for a senior developer we need to hire.
make it sound good
```

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA

`"write a job posting…"`

`→ Who is doing the hiring?`

An HR generalist writes differently than an engineering manager. The hiring voice affects candidate trust.

TASK

`"…for a senior developer…"`

`→ What stack? What scope?`

Senior means different things at different companies. Define the role scope, team, and reporting line.

CONTEXT

`"…we need to hire…"`

`→ Why now? What's the team like?`

Company stage, team size, tech stack, remote/hybrid policy — what makes this role worth applying to?

EXEMPLAR

`"…make it sound good"`

`→ Good like whom?`

Link to a job post you admire. Stripe's posts sound very different from Google's — 'good' is not a style.

FORMAT

`(nothing specified)`

`→ Which sections are required?`

About company, about role, responsibilities, requirements, nice-to-haves, what we offer. Define the order.

CONSTRAINTS

`(nothing)`

`→ Tone, bias, length limits?`

No corporate clichés, gender-neutral language, max 5 bullets per section, include salary range.

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

[PERSONA] You are an engineering manager writing a job posting that attracts strong senior developers — not HR copy.

[TASK] Write a job posting for a Senior Backend Engineer (.NET/C#) for LinkedIn and our careers page.

[CONTEXT] 40-person B2B SaaS, Series A. 8 backend engineers. Azure-hosted microservices. Fully remote, Europe timezone.

[EXEMPLAR] Honest, direct energy of a Basecamp or Linear post — no hype, no fake urgency. Show the actual day.

[FORMAT] Company intro (2 sentences), What you'll do (5), Looking for (5), Nice to have (3), We offer (4). Under 400 words.

[CONSTRAINT] No "rock star", "ninja", "fast-paced". Gender-neutral. Include salary range (80k-100k EUR). Be honest about what's hard.

06 Code Refactoring

① BAD PROMPT — written by someone new to AI

■ WEAK `refactor this code its messy and hard to read please fix it`

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA

`(no role)`

`→ What expertise level?`

A clean-code purist refactors differently than a pragmatic architect. Persona calibrates the trade-offs made.

TASK

`"refactor this code…"`

`→ What kind of refactor?`

Rename for clarity? Extract methods? Reduce complexity? Apply a design pattern? Each is a different operation.

CONTEXT

`"…its messy and hard to read"`

`→ What's the actual smell?`

Long methods? God class? Nested conditionals? Magic numbers? Name the problem — don't just say 'messy'.

EXEMPLAR

`(nothing provided)`

`→ What's the target style?`

Show a snippet from elsewhere in the codebase that represents the style you're aiming for.

FORMAT

`(nothing)`

`→ What should output include?`

Just the refactored code? Code + inline comments? A summary of decisions made? Define the deliverable.

CONSTRAINTS

`(nothing)`

`→ What must not change?`

Public API surface, method signatures, edge-case behavior. Define what's in scope vs frozen.

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

[PERSONA] You are a senior .NET developer who follows Clean Code principles and values readability over cleverness.

[TASK] Refactor the C# method below to improve readability and reduce cyclomatic complexity.

[CONTEXT] ~80 lines, 4 levels of nested ifs, magic numbers, mixed business logic + data access in a legacy service class.

[EXEMPLAR] Target style: small private methods with intention-revealing names, early returns, named constants.

[FORMAT] Return: refactored method first, then a short numbered list of changes made and why.

[CONSTRAINT] Do not change the public method signature. No new dependencies or interfaces. Scope is this method only.

07 Developer Onboarding Plan

① BAD PROMPT — written by someone new to AI

■ WEAK

```
make an onboarding plan for our new developer starting next week
```

↓ Decompose into 6 parts ↓

② BREAK IT APART — what every good prompt needs

PERSONA
`"make an onboarding plan…"`
`→ From whose perspective?`
Engineering manager vs HR vs team lead — each prioritises different things in a week-one plan.

TASK
`"…onboarding plan…"`
`→ What timeframe? What depth?`
First day? First week? 30/60/90 days? A calendar, a checklist, a Notion doc? Be specific.

CONTEXT
`"…new developer starting next week"`
`→ What kind of developer?`
Seniority, stack, role scope? What's the team working on? Remote or in-office? What's already set up?

EXEMPLAR
`(nothing provided)`
`→ What's worked before?`
Reference a previous onboarding doc or a named framework like '30-60-90 day plan' to anchor the structure.

FORMAT
`(nothing specified)`
`→ How will this be used?`
A Notion page? Confluence doc? Daily schedule table? The delivery format shapes the whole structure.

CONSTRAINTS
`(nothing)`
`→ What's off-limits or required?`
No meetings before day 2, must include a 'first commit' milestone, max 3 meetings per day in week one.

③ THE GOOD PROMPT — rebuilt with all 6 parts

✓ STRONG

[PERSONA] You are an engineering manager who has onboarded 10+ developers and knows week one sets the tone.

[TASK] Create a structured 5-day onboarding plan for a new mid-level backend developer, as a Notion doc.

[CONTEXT] Mid-level C#/.NET dev joining a fully remote team of 8. Stack: ASP.NET Core, Azure, SQL Server, GitHub Actions.

[EXEMPLAR] 30-60-90 compressed to 5 days: Orient (day 1), Understand (days 2-3), Contribute (days 4-5).

[FORMAT] Day-by-day table: Day | Goal | Morning | Afternoon | End-of-day checkpoint. Then a Manager setup checklist (8-10 items).

[CONSTRAINT] Max 2 meetings per day. Each day ends with a concrete small win. No generic HR tasks — technical integration only.