The technique that matters most:
Context + Constraints + Examples

Here's the structure:

1. **Be specific about what you're building**
Don't say "make a dashboard." Say "make a responsive admin dashboard with a sidebar navigation, user stats cards, and a data table showing recent orders."

2. **Specify your tech stack and constraints upfront**
- "Use vanilla JavaScript, no frameworks"
- "Keep it under 200 lines"
- "Make it mobile-first responsive"
- "No external dependencies except X"

3. **Show, don't just tell**
If you have a design pattern you like, reference it. If there's a specific interaction you want, describe it step by step or share a similar example.

4. **State your actual use case**
"This will be embedded in a WordPress site" vs "This is a prototype for client demo" changes everything about what I'll build.

The anti-patterns to avoid:
- Being vague then correcting me 5 times (wastes both our time)
- Asking for "best practices" without context (best for what?)
- Throwing buzzwords without substance ("make it modern and clean")

Example of a good prompt:

"Build a todo list app in vanilla JS with localStorage persistence. Requirements: add/delete/toggle tasks, filter by all/active/completed, clear completed button. Design: minimalist, single-column layout, mobile responsive. No frameworks, keep the code clean and commented for a junior dev to understand."

Example of a great prompt for iteration:

"The todo app works but I need to add drag-and-drop reordering. Use the native HTML5 drag and drop API, update the order in localStorage, add visual feedback while dragging."

The real secret? Treat me like a senior developer on your team. You wouldn't tell them "make it cool" - you'd give them requirements, constraints, and context. Do that, and you'll get production-ready code faster than chasing the latest prompt gimmick on Twitter.

What kind of project are you working on? I can show you how to apply this specifically.