Context & Prompts

A cheat sheet. *Modes are the vehicle — context and prompts are the fuel.*

01 CICE

The default. Your North on most tasks.

Context where you're working — anchor file, feature area.

Intent the change you want made, in one sentence.

Constraints what not to touch. Happy paths, untested files.

Examples one concrete data shape or output format. Most skipped, most valuable.

```
> Working on @checkoutController.ts. Add error handling for API failures.  Don't touch the happy path. Errors shaped { error, code }; timeouts retry once.
```

Note: skip @app/billing/* and you'll get a fake data format that doesn't match your API.

02 RISEN

For multi-step agent work with real guardrails.

Role the persona the agent plays (backend engineer, UX writer, ...).

Instructions the high-level task, in one or two lines.

Steps the sequence to follow. Audit first, then refactor, etc.

End goal the success state. What does done look like?

Narrowing what's off-limits. Most important field. No touching tests, etc.

```
> Senior backend on a Node API. Refactor payment service to remove direct DB calls.  Steps: Audit → extract repo layer. End: clean separation. Narrowing: don't change  public API or tests.
```

Note: useful for small tasks. Like CICE instead if it's one-and-done.

03 RTF

Quick tasks where output format matters.

Role who the model is pretending to be. TypeScript dev, SRE, etc.

Task one concrete thing to produce.

Format the exact shape of the output. "a single function", "numbered steps", "a 3-row table".

```
> Role: TypeScript dev. Task: write a debounced async callback. Format: single  exported function with a 250ms comment.
```

Note: keep a small list from turning into a 200-line response.

04 Chain-of-thought

For debugging or when you don't know what to fix.

The trigger ask the model to think first, act second.

Use for bugs you can't localize, tricky logic, unfamiliar code paths.

The phrase "Think step by step." Works surprisingly well.

In Cursor Plan mode is the same idea with a UI around it.

```
> Before any changes, walk me through how the session token gets validated.  Think step by step.
```

Note: asking for an explanation first often surfaces the bug before a line is written.

Anti-patterns, and their fixes

If one of these sounds familiar, it's the prompt, not the model.

"Clean up the code"	vague — name what to rename, what to leave alone
"Make it faster"	targets the wrong bottleneck. Name the slow function or path
"Add tests"	wrong framework or wrong scope. Specify framework + which functions
"Fix the bug"	doesn't know which bug. Include the error and stack trace
"Use a factory pattern"	implements it even if wrong — describe the problem, not the solution

Start fresh when…

If you're working back to re-explain something, that's the moment.

- you've moved to a new, unrelated task
- Agent keeps hallucinating file paths
- the conversation anchored on a wrong assumption
- you're near the context-window cap

One task per prompt. Describe the problem, not the solution — let the model propose how. Stuck? Ask the model to `write the prompt for you` (meta-prompting).