# Cursor Modes

A cheat sheet. *Six modes, one tradeoff map — pick the right tool for the job.*

## 01 Tab

*AI autocomplete as you type.*

- Writing boilerplate or finishing a thought.
- Unlimited on all paid plans (including Teams).
- Hit Tab again to jump to the next predicted edit.

`Tab · Esc · Cmd+→ (one word)`

**Watch out:** you still own every line. Review before merge.

## 02 Ask

*Read-only Q&A about your codebase.*

- Before touching unfamiliar code.
- Blast radius: what breaks if I change X?
- Surfaces existing utilities and design intent.

`Shift+Tab → Ask`

**Watch out:** can't edit files. Switch modes when you're ready to write.

## 03 Agent

*Autonomous: reads, edits, runs, iterates.*

- Outcome is clearly defined.
- Change spans multiple files.
- Give it a spec: failing test, types, behavior.

`Shift+Tab → Agent`

**Watch out:** say what NOT to touch. Checkpoint any wrong turn.

## 04 Inline Edit

*Surgical edits at the cursor. (Cmd+K)*

- You can point at what needs to change.
- Fits in one sentence, one location.
- Doesn't cross files — use Agent if it does.

`Select → Cmd+K → describe → accept`

**Watch out:** Cmd+K on the wrong selection wastes time.

## 05 Plan

*Research → questions → editable plan → execute.*

- Requirements are ambiguous.
- Change is cross-cutting (auth, logging, layers).
- You want to review the approach before code.

`Shift+Tab → Plan`

**Watch out:** when in doubt, Plan first. Save plans to workspace.

## 06 Debug

*Purpose-built error diagnosis.*

- Tests go red or terminal throws.
- Trigger it from the failure output itself.
- More focused than Ask or Agent for this job.

`"Debug with AI" on red output`

**Watch out:** underused. Reach for it before opening Ask.

## Which mode, quickly

*A rough picker — when the signal is obvious, trust it.*

| | |
|---|---|
| **If you don't understand the code yet** | *reach for* **Ask** |
| **If you can point at it and say it in one sentence** | *reach for* **Inline Edit (Cmd+K)** |
| **If the outcome is clear and it spans files** | *reach for* **Agent** |
| **If requirements are ambiguous or cross-cutting** | *reach for* **Plan** |
| **If tests are red or the terminal threw an error** | *reach for* **Debug** |
| **If you need 20+ files of context at once** | *reach for* **Agent + Max Mode** |

## About Max Mode

*It's a context-window dial, not a quality dial.*

**Use it for** refactors across 20+ files, monorepo-wide search, cross-service work.

**Skip it for** single-file changes, anything under 10 files, targeted bug fixes.

**Context is a budget.** Point the agent at what it needs — `@file` · `@folder` · `@symbol` · `@terminal` · `@past chats` · `@branch` — and start a fresh chat when the conversation anchors on wrong assumptions.