You Asked

summarize this article pls

https://pub.towardsai.net/4-techniques-to-optimize-ai-coding-efficiency-759c7273d8e0

Gemini Fast

This article provides a guide on how to shift from using AI as a simple "code generator" to using it as a sophisticated "architectural partner." It highlights that while AI can write code quickly, the real efficiency gains come from how you structure the interaction and the codebase itself.

The **four key techniques** discussed are:

1. The "Modular-First" Architecture

Efficiency is often lost when AI is asked to handle large, monolithic files because the model's "context window" becomes cluttered with irrelevant information.

- **The Technique:** Break your project into small, single-responsibility modules (files).
- **Why it works:** When you provide the AI with a small, 50-line file, it has a much higher accuracy rate than when navigating a 500-line file. It also reduces the "hallucination" of variables or functions that don't exist in that specific scope.

2. Context Stitching (The "Skeleton" Method)

Instead of feeding the AI your entire codebase, you should provide a "skeleton" of the project.

- **The Technique:** Share the file structure and the headers/interfaces of related modules, but not the full implementation details of every file.
- **Why it works:** This gives the AI enough context to understand how the current task fits into the larger system without hitting token limits or confusing the model with unnecessary logic.

3. Iterative Prompting with "Pseudo-Code" Drafts

Jumping straight to a final solution often leads to bugs.

- **The Technique:** Ask the AI to write the logic in **pseudo-code or comments first**, then review that logic before asking it to generate the actual Python or JavaScript code.

- **Why it works:** This allows you to catch logical flaws at the "blueprint" stage. It’s much faster to correct a bullet point in a plan than to debug 100 lines of syntactically correct but logically broken code.

4. Automated Feedback Loops (Test-Driven Development)

The most efficient way to use AI is to let the computer, not the human, do the initial "complaining."

- **The Technique:** Provide the AI with the error logs from your terminal or the output of a failed unit test.
- **Why it works:** AI is exceptionally good at "fixing" specific errors when given the exact trace. By feeding it the error message directly, you skip the step of having to explain what went wrong, allowing the AI to self-correct based on real-world feedback.

Key Takeaway

The article concludes that **AI coding efficiency is a management task.** You aren't just a "coder" anymore; you are a "manager" of an AI agent. By organizing your code modularly and providing precise, scoped context, you minimize the AI's mistakes and maximize its output quality.