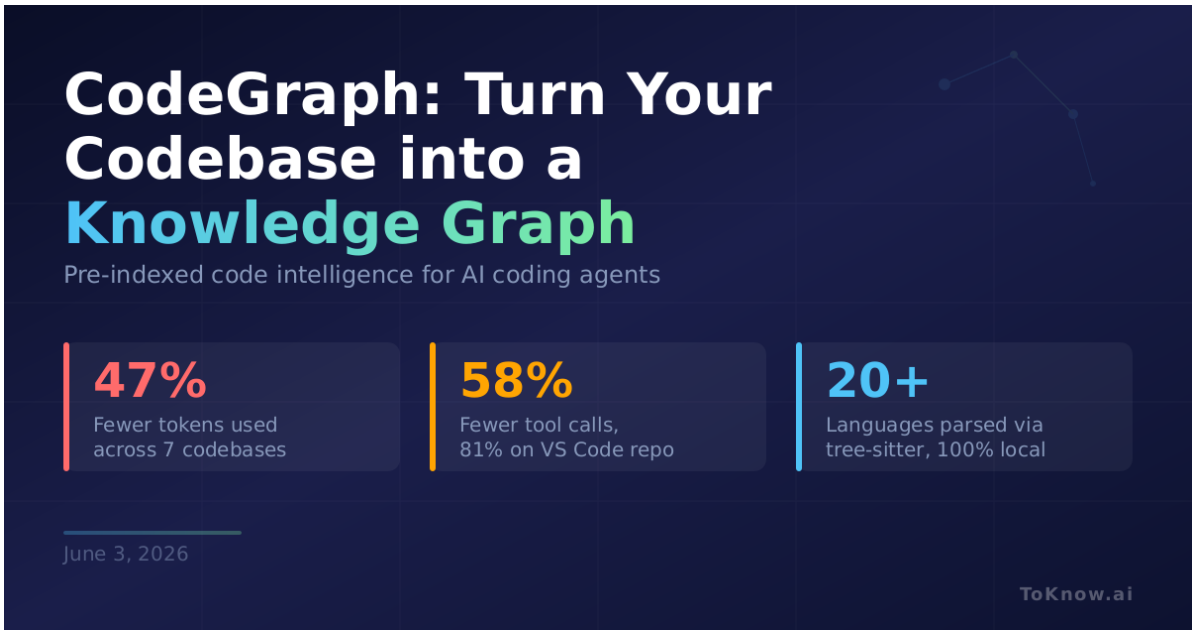
CodeGraph: Turn Your Codebase into a Knowledge Graph So AI Agents Stop Wasting Tokens

Kabui, Charles

2026-06-03

Read at ToKnow.ai



Every time you start a new AI coding session, the agent re-reads your project from scratch, burning tokens on file discovery. CodeGraph fixes this by pre-indexing your repository into a

local SQLite-backed knowledge graph that agents query through an MCP server. It uses tree-sitter to parse 20+ languages, extracts symbol relationships, call graphs, and dependency edges, then serves them on demand. Benchmarked across seven open-source codebases including VS Code (~10k files) and Django (~3k files), CodeGraph cut token usage by 47%, tool calls by 58%, and wall-clock time by 22% on average. On VS Code specifically, tool calls dropped 81%. It plugs into Claude Code, Cursor, Codex, Gemini CLI, OpenCode, and Kiro with a single `codegraph install` command. A file watcher keeps the graph current as you code, and everything runs locally with no external API calls.

Tokens cost money. At Opus 4.8 pricing ($5/$25 per million input/output tokens), a developer exploring a large codebase burns real dollars on context loading. CodeGraph replaces those exploratory file reads with instant graph queries: a structural question like "what depends on this module?" that would require reading 20+ files gets answered in a fraction of the tokens. It also handles cross-language boundaries that trip up static analysis, bridging Swift/Objective-C calls, React Native bridges, TurboModules, and Expo modules.

This fits a broader pattern: AI coding agents are maturing past raw model improvements into tooling that makes each token more productive. Dynamic workflows scale agents horizontally. CodeGraph makes each individual agent session cheaper and faster by front-loading structural understanding.

Sources:

- CodeGraph GitHub Repository
- CodeGraph Documentation
- CodeGraph npm Package
