cupertino v1.2.0: lands the right Apple doc 9 times in 10

Released 2026-05-21.

What you actually get

If you've ever asked an AI coding assistant about a Swift API and watched it confidently invent a method that doesn't exist (or point you at a deprecated 2017 version), cupertino is what I built for that.

This release: AI assistants using cupertino now land the right Apple documentation page on the first try **9 times out of 10**. Was 5 out of 10 in v1.1.0.

A few concrete cases:

- Ask about `URLSession` async/await, get the modern API. Not the completion-handler page from 2017.
- Ask "what replaced `NSURLConnection`", go straight to `URLSession`.
- Ask about `@Observable`, don't get it fused with the older `ObservableObject`.
- Compare Combine and AsyncSequence, get both pages, not a hallucinated synthesis.

352,712 indexed Apple documentation pages, zero garbage rows on the merged corpus. Runs locally, no cloud, no subscription, free and open source.

Upgrade is one command:

```
brew upgrade cupertino && cupertino setup
```

The rest of this post is for the technically curious.

The measurement

Rank-1 accuracy on a 50-query canonical-lookup corpus (Swift stdlib types, Foundation, SwiftUI, UIKit, AppKit, Combine, framework roots) went from **52% in v1.1.0** to **92% in v1.2.0**.
Cross-validated on two more corpora, zero regressions across 110 paired queries, statistically significant at McNemar p ? 1e-5 on the largest corpus.

Corpus	v1.1.0	v1.2.0
Canonical lookup (50 queries)	26/50 rank-1	46/50 rank-1
Canonical lookup V2 (30 queries, no overlap with the first set)	19/30 rank-1	28/30 rank-1
Deprecation pairs (30 modern/legacy triples like `URLSession` vs `NSURLConnection`)	27/30 prefers modern	30/30 prefers modern

Three independent samples, same direction, similar effect size. Not a fluke.

Methodology lives at `docs/design/search-quality-eval.md`; per-query rank movements live in docs/audits. If a number is wrong in public, an audit markdown is wrong first.

How it works

cupertino is the retrieval layer of RAG, nothing else. The AI agent does the generation; cupertino just returns the right Apple page when one is asked for.

The substrate is SQLite FTS5 with field-weighted BM25, populated by an AST extraction pass at index time that pulls symbol signatures, generic constraints, platform availability, and deprecation markers into queryable columns. No embeddings, no vector database, no reranker.

Anthropic's published RAG recipe (Contextual Retrieval, September 2024) keeps a BM25 stage alongside the dense one as the non-optional floor. cupertino takes that floor and runs it without the rest, on a corpus class where that's enough: identifier-heavy, terminology-precise, Apple API docs.

What it still gets wrong

Query class	Rank-1	Status
Prose / conceptual ("what's the deal with...")	~27%	Open: candidate fix in design
Acronym / synonym ("KVO", "GCD", "CALayer")	~18%	Open: needs a new index pass
Querying by structural attribute (`actor type`, `initializer`, `Hashable conformance`)	P@5 ~0.25	Open: search path doesn't consult the symbol metadata table

These were weak in v1.1.0 too. The next release is meant to move them. If it does, the live page will say so. If it doesn't, the same page will say that, in the same row, without anyone editing copy.

This is not a feature roadmap. It is a list of things the tool does badly, published next to the things it does well.

Upgrade

```
brew upgrade cupertino
cupertino setup
```

`cupertino setup` pulls the v1.2.0 database bundle from GitHub. If you had v1.1.0 installed locally, the migrator handles the schema bumps idempotently.

Full release write-up: docs/release-writeup-v1.2.0.md. Source for every measurement: docs/audits/.