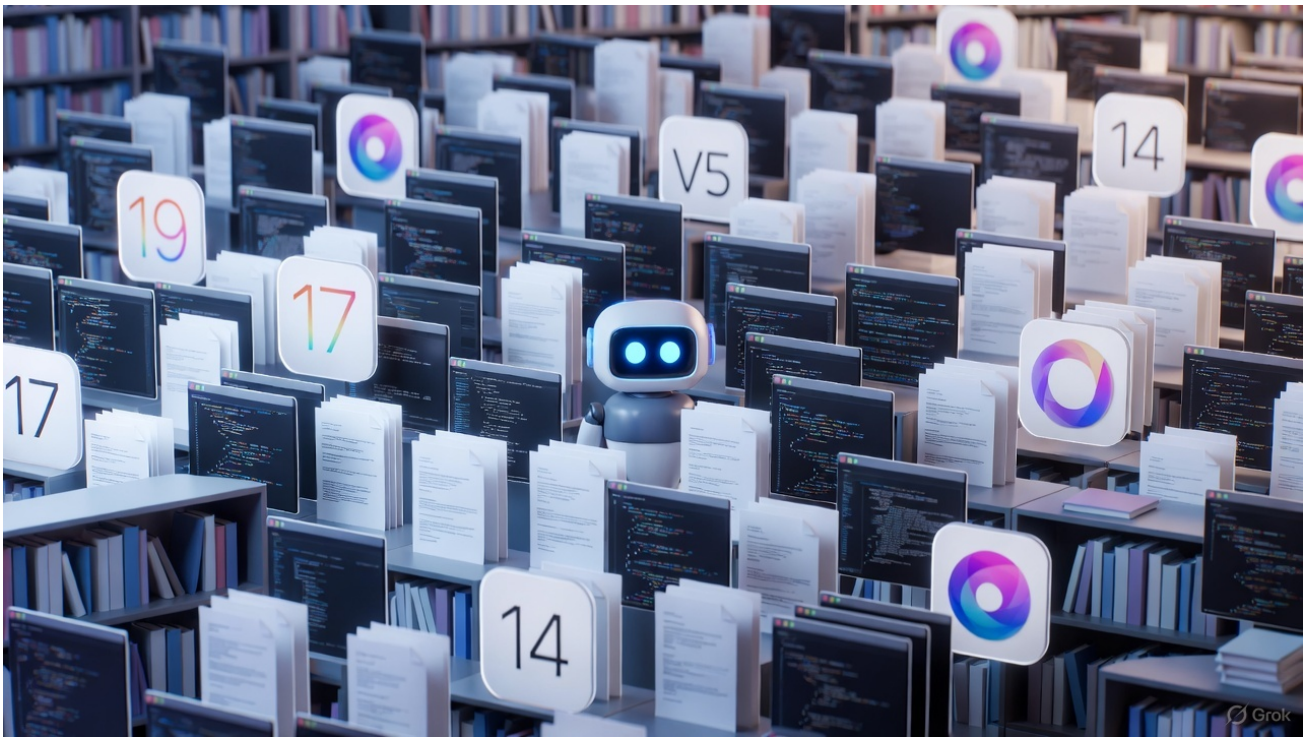

Cupertino v0.7.0: 302K Docs, OS Version Filtering, and Teaching AI to Dig Deeper

TL;DR: Finally crawled all of Apple's docs. Added OS version filtering. Made AI agents smarter about finding the right documentation.

Four days after v0.5.0, and I've shipped two more releases. At this point I'm not sure if this is momentum or obsession. Let's call it momentum.

The Numbers

Metric	v0.5.0	v0.7.0	Change
Total Documents	234,331	302,424	+29%
Frameworks	287	307	+20
Database Size	~1.9GB	~2.3GB	+21%

Three hundred thousand pages. That's every framework, every class, every method Apple has documented. The crawl ran for almost two weeks total across multiple sessions.

Top Frameworks by Document Count

Framework	Documents
Kernel	39,396
Matter	24,320

Framework	Documents
Swift	17,466
AppKit	12,443
Foundation	12,423
UIKit	11,158
Accelerate	9,114
SwiftUI	7,062

Kernel jumped from 25,000 to nearly 40,000 pages. That's IOKit, BSD interfaces, Mach APIs, driver development - the low-level stuff that's hardest to find and most valuable when you need it. Matter has 24,000+ docs for smart home development. Accelerate and SIMD documentation is fully indexed now - performance optimization guides that AI models probably didn't see much of during training.

The Feature I've Been Waiting For: OS Version Filtering

This was issue #99, and it's been on my mind since day one. Apple's documentation includes version information - which iOS/macOS version introduced each API. We now extract this.

```
cupertino search "async" --min-ios 16
```

Or in Claude:

> "Search for SwiftUI navigation APIs available in iOS 17+"

The AI agent can now filter results by platform version. No more suggestions for deprecated iOS 12 APIs when you're building for iOS 17.

How It Works

Different documentation sources need different strategies:

Source	Strategy
apple-docs	Apple's JSON API + fallbacks
sample-code	Derived from framework
apple-archive	Derived from framework
swift-evolution	Swift version mapping
swift-book/hig	Universal (all platforms)

For Apple's modern docs, there's a hidden JSON endpoint at `/tutorials/data/documentation/` that returns structured metadata including availability. For older stuff, we derive from the framework - if it's UIKit, we know the minimum versions.

The cool part? You can filter by *any* platform:

```
--min-ios 16
--min-macos 14
```

```
    --min-tvos 17
    --min-watchos 10
    --min-visionos 1
```

visionOS filtering. For when you're building spatial computing apps and need APIs that actually exist on the platform.

Teaching AI Where to Look

Here's something I discovered while using Cupertino with Claude: the AI doesn't know what it doesn't know.

When Claude searches for "CALayer animation", it gets API reference - classes, methods, properties. Good stuff. But the *conceptual* documentation - the "why" behind Core Animation - lives in Apple Archive. And Claude had no idea.

The Problem

I had added archive support in v0.2.3. Users could search with `--source apple-archive`. But AI agents don't read release notes. They read tool descriptions once, then forget. Every search was missing the deep conceptual guides.

The Solution: Teasers

Now every search result includes hints from alternate sources:

```
# Search Results for "CALayer animation"

## Results from apple-docs (5 found)

1. **CALayer** - The basic object for managing and presenting drawable
content...
2. **CABasicAnimation** - An animation that applies a single keyframe...

---

**Other sources to explore:**

- **apple-archive**: 3 results (Core Animation Programming Guide, Animation
Types...)
- **samples**: 2 results (LayerPlayer, AnimationExplorer)
- **swift-evolution**: 1 result (SE-0421: Generalize async sequence...)

Use `source` parameter to search: apple-archive, samples, hig,
swift-evolution...
```

Every search response is now a teaching moment. Claude sees there are archive guides available. It sees sample code exists. It learns *in context* how to dig deeper.

Human Interface Guidelines: Design Docs for AI

Here's something I didn't expect to matter as much as it does: HIG support.

Apple's Human Interface Guidelines aren't just for designers. When Claude is helping you build a button, a navigation pattern, or a settings screen, it should know Apple's design recommendations - not just the API.

```
cupertino search "buttons" --source hig
```

Now when you ask "what's the recommended button style for destructive actions?", Claude can actually look it up instead of guessing. Platform-specific guidance for iOS, macOS, watchOS, visionOS - all searchable.

The HIG docs are marked as "universal" for availability - they apply across all OS versions. Design principles don't deprecate like APIs do.

One Tool to Rule Them All

Speaking of simplification - I consolidated the search tools.

Before v0.7.0:

- `search_docs` - Apple documentation
- `search_hig` - Human Interface Guidelines
- `search_samples` - Sample code
- ...and more

Now:

- `search` - Everything, with a `source` parameter

```
# CLI
cupertino search "buttons" --source hig
cupertino search "networking" --source samples
cupertino search "actors" --source swift-evolution
cupertino search "everything" --source all
```

One tool, eight sources, cleaner mental model. The `source` parameter accepts:

- `apple-docs` (default) - API reference
- `samples` - Working code examples
- `hig` - Design guidelines
- `apple-archive` - Legacy conceptual guides
- `swift-evolution` - Language proposals
- `swift-org` - Swift.org documentation
- `swift-book` - The Swift Programming Language
- `packages` - Swift package docs
- `all` (searches everything)

The `all` option is surprisingly useful. When you're not sure where something lives, just search everywhere.

What I Learned

Building for AI is different than building for humans.

Humans read documentation. They explore. They remember "oh yeah, there's an archive section I should check." AI agents don't work that way. They have tool descriptions and search results. That's it.

The best way to teach an AI agent? Put the lesson in every response it sees. Not in documentation it reads once. Not in tool descriptions it might forget. In the actual output, every single time.

That's why teasers matter. That's why source-aware messaging matters. Every search result is documentation.

The Road Ahead

With 302,424 pages indexed and version filtering working, the foundation is solid. What's next:

1. **Semantic code search** - Search sample code by what it *does*, not just keywords
2. **Framework relationships** - Search UIKit, get relevant Foundation results too
3. **Smarter ranking** - Weight results by your target platform

The crawl is complete. The curation continues.

Install or Update

```
# New install
brew tap cupertinohq/tap https://codeberg.org/CupertinoHQ/homebrew-tap.git
brew install cupertinohq/tap/cupertino

# Update existing
brew upgrade cupertino

# Or one-line install
bash <(curl -sSL
https://codeberg.org/CupertinoHQ/cupertino/raw/branch/main/install.sh)
```

Then:

```
cupertino setup  # Download 2.3GB database (~30 seconds)
cupertino serve  # Start MCP server
```

Try the new features:

```
# Filter by OS version
cupertino search "SwiftUI navigation" --min-ios 17

# Search design guidelines
cupertino search "buttons destructive" --source hig

# Search legacy conceptual guides
cupertino search "Core Animation" --source apple-archive
```

```
# Search everything at once
cupertino search "async await" --source all
```

Cupertino is an Apple Documentation MCP Server. 302,424 pages of Apple documentation, searchable by AI. Source at codeberg.org/CupertinoHQ/cupertino.