Hello, World!

April 16, 2026 / 5 min read / 958 words

Hello World!

I'm Oscar — Principal Engineer and Founder of **M31 Labs**, a consultancy and engineering laboratory working at the frontier of what LLM-assisted humans can build. This is post zero.

> **ABOUT THIS BLOG**
>
> Part laboratory notebook, part product log, part human. All groovy.

Table of contents

How we got here

M31 Labs did not exist before February 19th, 2026. That evening I sketched the first parse-table loader for **gotreesitter**[1]. **Six days later**, on February 25th, I posted *"I ported Tree-sitter to Go"* to Hacker News[2] — because of course I did — and got exactly the feedback a six-day-old project deserves: *interesting, maybe, but too green, too "vibe-coded," too much surface area for one person.*

They weren't wrong. Six days is six days. They also weren't the whole picture.

> **Why I'm writing this at all**
>
> Not to re-litigate the thread. To leave a marker. Two months on, the artifact is real, reproducible, and on a shipping cadence. The best response to "is this serious?" is receipts.

What I took away

THREE LESSONS THE THREAD CLARIFIED

- Many folks still (in some cases, willfully) have their heads in the sand about LLM-assisted software. That's okay, and temporary.
- Discourse *about* a technology and *use* of the technology barely overlap.
- The goal is unchanged — ship software that makes or saves money. The means changed.

The stack you're reading this on

This blog isn't hosted prose. It's the full dogfood chain, Go all the way down:

gosx[3]
: A Go-native web platform. Components in `.gsx` (Go with embedded markup), server-rendered by default, interactive islands compiled to WASM bytecode. No JavaScript toolchain. m31labs.dev is a gosx app.

md++[4]
: A single-package plain-text document system with admonitions, footnotes, math, emoji, diagrams, and a real AST. Renders the post you're reading right now.

gotreesitter[1]
: Pure-Go tree-sitter runtime. 206 grammars. No CGo. Parses the md++ source before md++ renders it.

FLOWCHART DIAGRAM

```
flowchart LR
  draft[Markdown++ source] --> mdpp[mdpp parser]
  mdpp --> gts[gotreesitter runtime]
  gts --> grammars[compiled grammar registry]
  gts --> tree[parse tree]
  tree --> ast[mdpp AST]
  ast --> html[rendered HTML cache]
  html --> gosx[gosx blog route]
  gosx --> page[server-rendered page]
```

Turtles, Go turtles, all the way down.

Two months of shipping, in public

Release	Date	Highlight
v0.12.0	2026-03-28	ABI 15 reserved-word support in `grammargen`; Python & C# parity
v0.13.0	2026-03-31	`SkipTreeParse` hook — read source without paying for an AST
v0.13.3	2026-04-04	GLR large-file perf: **4+ min ~420ms** on a 147KB Go file
v0.13.4	2026-04-05	Injection-parser arena leak fixed (**~3 MB/parse 0**)
v0.14.0	2026-04-17	`grammargen`-compiled Go blob; arena overhaul; Zig upstream migration

v0.14.0, in concrete numbers

The big one is memory. On `BenchmarkSelfParseWarmReuse` (gotreesitter parsing its own gnarliest root files, Docker 4G / 4 CPU):

mode	pre-0.14.0	0.14.0	delta
cold (fresh Parser per iter)	574 MB/op	245 MB/op	**−57.3%**
warm (one Parser reused)	498 MB/op	320 MB/op	**−35.7%**
warm + GC drain	522 MB/op	328 MB/op	**−37.1%**

Cold-path bytes-per-op drops by a factor of 574245≈2.34. Two changes drove most of it:

1. **The Go grammar ships as a `grammargen`-compiled blob.**
   Our pure-Go LALR(1) + LR(1) state-splitting compiler produces the parse tables instead of tree-sitter's C ones. Side effect[†]: a long-standing dead end in the C tables — where `}` had no valid action after certain nested `switch` / `case` / `if` patterns — no longer exists. gotreesitter parsing its *own* `parser_reduce.go` now returns `HasError=false`. The old blob wrapped it in `ERROR`.
2. **Arena sizing flipped from `sourceLen × 4` to `sourceLen / 4`**, and the retention ceiling is preserved across resets. The old formula over-allocated by 10–16× on dense Go. The new one starts small and grows adaptively; warm loops stop re-reallocating the primary slab.

Plus a Zig grammar migration to the active upstream org (**28% smaller blob** — 62,948 45,316 bytes) and a retry-path fix that releases losing candidate-tree arenas eagerly instead of waiting on GC finalization. Full notes: ..

> [†] *The "side effect" is that the port is now, in a narrow and specific sense, **more correct** than the reference C implementation on one real Go file. I did not expect that to be the artifact of the week.*

What it looks like to use

```
import (
    "github.com/odvcencio/gotreesitter"
    "github.com/odvcencio/gotreesitter/grammars"
)

src  := []byte("package main\nfunc main() {}\n")
lang := grammars.GoLanguage()
tree, _ := gotreesitter.NewParser(lang).Parse(src)
```

No C toolchain. Cross-compiles to `wasip1`, `arm64`, Windows — anywhere Go goes. `go test -race` sees every line.

On AI-assisted software

> **THE QUESTION ISN'T WHETHER I USED AN LLM**
>
> It's whether the artifact holds up. gotreesitter has a 206-grammar parity harness against tree-sitter's C runtime, a benchmark suite with a CI gate, fuzz targets for parser/query/tagger, and seven outside contributors sending real PRs.Those are the things that keep a project honest. The provenance of the keystrokes is not.

Requests to "release the prompts" strike me as inane — not because the prompts are secret, but because they're genuinely not interesting. Most read like *"cmon dawggy that shouldn't go there and you know it"* followed by the model agreeing . The value isn't in the prompts; it's in knowing **when to interrupt, re-route, and call shenanigans** across many sessions for one feature, and in running the verification that tells you the output is real. If you want the recipe, read the parity harness, not the transcripts.

What's next on this blog

- [x] Post zero — this one
- [ ] v0.14.0 deep-dive: the grammargen-compiled Go blob, with diagrams
- [ ] How mdpp renders this page in one pass over a gotreesitter AST
- [ ] The gosx islands model, and why the browser is a render target not a runtime
- [ ] Release notes, build logs, and the occasional ranty rant

Thanks for reading. Back to shipping.

1. gotreesitter — pure-Go tree-sitter runtime, 206 grammars, v0.14.0 shipped 2026-04-17.
2. Show HN: I ported Tree-sitter to Go — 2026-02-25. Go read it; the critiques were sharper than I wanted them to be at the time, and the project is better for it.
3. gosx — Go-native web platform and component/runtime framework powering m31labs.dev.
4. mdpp — Markdown++ parser and renderer used for this blog's authoring pipeline.