

Glagol Compiler Manifest: Principles for a Visible Typed Pipeline

Sanjin Gumbarevic
hermeticum_lab@protonmail.com

Make the tree visible.

Glagol (ᐅᐅᐅᐅᐅᐅᐅ) is the first compiler for Slovo. Its job is not merely to accept parenthesized text. Its job is to preserve Slovo's structure through parsing, checking, diagnostics, lowering, and executable output.

Principles

- Slovo specifies first; Glagol implements second.
- Parse trees, ASTs, typed ASTs, and LLVM IR are separate compiler stages.
- LLVM emission starts from checked representation, not raw source forms.
- User-source errors produce diagnostics, not compiler panics.
- Formatter output, diagnostics, examples, tests, and docs are part of the language contract.
- Native execution is a compiler target, but stable ABI promises require an explicit future contract.
- Benchmarks are local evidence unless a release explicitly defines a broader performance methodology.

Support Rule

A source feature is supported only when Glagol can:

1. parse it
2. lower it
3. type-check it
4. format it when formatting applies
5. diagnose invalid forms structurally
6. emit valid LLVM or reject it before backend emission
7. cover it with automated tests and release documentation

Partial recognition is not support.

Beta Rule

The first real beta compiler release is `1.0.0-beta`. It requires a matching Slovo beta contract, conformance suite, stable diagnostics schema, stable formatter behavior, stable package/manifest behavior, standard-library compatibility rules, and release reviews with no blocking manifest drift.