Milestones - Noir Language Development

NOTE: This is a work-in-progress and will be continuously updated as we release newer versions of Noir and re-prioritize features. This is intended to be an explanatory document for external developers interested in using Noir.

This should help describe where Noir is today, where Noir will be tomorrow, and a general idea of the expected progression of the language.

Quick intro to Noir

Noir is a domain specific language for creating and verifying proofs. Its design choices are influenced heavily by Rust.

Noir is designed for accessible and flexible development of provable programs. It abstracts away the need for developers to handcraft low-level circuits and aims at supporting different proving backends for different development needs.

Noir compiles to an intermediate language, which itself can be compiled to an arithmetic circuit or a rank-1 constraint system. This allows one to decouple the programming language completely from the backend, so Noir can support arbitrary backends.

Noir 'off-the-shelf' runs using Aztec's TurboPlonk backend (although an UltraPlonk integration is imminent).

Noir Language – End of April 2023

- Dynamic arrays
- UltraPlonk integration (CLI)
- Faster proving speeds
- Comes with optimized implementations of
- Keccak256 in 18,000 constraints
- ECDSA verification in 36,000 constraints
- Keccak256 in 18,000 constraints
- ECDSA verification in 36,000 constraints
- Benchmarks for UP will come in Q2 2023
- Faster proving speeds
- Comes with optimized implementations of
- Keccak256 in 18,000 constraints
- ECDSA verification in 36,000 constraints
- Keccak256 in 18,000 constraints
- ECDSA verification in 36,000 constraints
- Benchmarks for UP will come in Q2 2023

Noir Tooling – During Q2 2023

- Noir.js
- Embed Noir programs in browser / JavaScript environments
- Embed Noir programs in browser / JavaScript environments
- Language Server Protocol support
- Write Noir code with in-browser VSCode
- Write Noir code with in-browser VSCode
- Linter

- Lightning fast debugging
- Lightning fast debugging
- Package Registry
- One-stop shop for all Noir package needs
- Reference: Rust’s crates
- One-stop shop for all Noir package needs
- Reference: Rust’s crates
- Package Manager
- Installing Noir packages as simple as nargo install package
- Installing Noir packages as simple as nargo install package
- Linguist support
- Automatic syntax highlighting on GitHub
- Requires >200 unique :user/:repo

on GitHub; Write some Noir code!

- Automatic syntax highlighting on GitHub
- Requires >200 unique :user/:repo

on GitHub; Write some Noir code!

- Benchmarks for UltraPlonk

Noir Tooling – During Q3 2023

- Debugger
- Step through Noir programs
- Potential blocker: Noir doesn’t currently give debug symbol info
- Step through Noir programs
- Potential blocker: Noir doesn’t currently give debug symbol info
- Integration with 3rd party external development frameworks
- E.g. Foundry, Hardhat
- Request for feedback – let us know which frameworks you want! Leave comments here
- E.g. Foundry, Hardhat
- Request for feedback – let us know which frameworks you want! Leave comments here
- Resource Guesstimation
- Quick resource feedback (e.g. constraint count, proving time, memory needs)
- In Nargo / VSCode plugin / standalone app
- Quick resource feedback (e.g. constraint count, proving time, memory needs)
- In Nargo / VSCode plugin / standalone app
- Code Visualization
- Visualizing Noir source code and ACIR
- Reference: Scaffold-eth, Speed Run Ethereum

- Visualizing Noir source code and ACIR
- Reference: Scaffold-eth, Speed Run Ethereum
- More to come! Suggest what you would like to see!

Noir Language v1.0 – During Q3 2023

- Main Objective:

It just works!

- Programmable Noir smart contracts with:
- Contract scope
- Unconstrained functions
- Import syntax
- Public/private variables
- Public/private functions
- Nested function calls
- Sending to L1
- Consuming from L1
- Emit data
- BigInt library integration
- [Possibly] Traits
- Contract scope
- Unconstrained functions
- Import syntax
- Public/private variables
- Public/private functions
- Nested function calls
- Sending to L1
- Consuming from L1
- Emit data
- BigInt library integration
- [Possibly] Traits