A gentle introduction: Stylus

ALPHA RELEASE, PUBLIC PREVIEW DOCS Stylus is currently tagged as analpha release. The code has not been audited, and shouldnot be used in production scenarios . This documentation is currently inpublic preview .

To provide feedback, click theRequest an update button at the top of this documentjoin the Arbitrum Discord , or reach out to our team directly by completingthis form . This introduction is for developers who want to build on Arbitrum using popular programming languages, like Rust. This capability is made possible by Stylus, a new way to write EVM-compatible smart contracts using your favorite programming languages.

In a nutshell:

- Stylus lets you write smart contracts in programming languages that compile down to WASM, such asRust, C, C++, and many others
- .
- Rich language support already exists for Rust: you can use the Stylus SDK and CLI tool tostart building today.
- Stylus smart contracts benefit fromArbitrum's EVM equivalence
- thanks to a second, coequal WASM virtual machine.
- Solidity contracts and Stylus contracts are fully interoperable. In Solidity, you can call a Rust program and vice versa.
- Stylus contracts are overan order of magnitude faster with significantly lower gas fees
- due to the superior efficiency of WASM programs.
- Memory is 100-500x cheaper
- when using Stylus, unlocking new use cases now that consuming RAM is viable on the blockchain.

What's Stylus?

Stylus is an upgrade to Arbitrum Nitro, the tech stack powering Arbitrum One, Arbitrum Nova, and Arbitrum Orbit chains. This upgrade adds a second, coequal virtual machine to the EVM, where EVM contracts continue to behave exactly as they would in Ethereum. We call this paradigmEVM+ sinceeverything is entirely additive.

This second virtual machine executes WebAssembly (WASM) rather than EVM bytecode. WASM is a modern binary format popularized by its use in major web standards, browsers, and companies to speed up computation. Built to be fast, portable, and human-readable, WASM also has sandboxed execution environments for security and simplicity. Working with WASM is nothing new for Arbitrum chains. Ever since theNitro upgrade , WASM has been a fundamental component of Arbitrum's fully functioning fraud proofs.

With a WASM VM, any programming language that can compile down to WASM are within the bounds of Stylus. While many popular programming languages can be compiled into WASM, some compilers are more suitable for smart contract development than others, like with Rust, C, and C++. Other languages, such as Go, Sway, Move, and Cairo, can be supported as well. Languages that include their own runtimes, like Python and Javascript, are harder for Stylus to support, although not impossible. Third-party contribution in the form of libraries for new and existing languages is welcomed!

Compared to using Solidity, WASM programs are much more efficient. There are many reasons for this, including the decades of compiler development for Rust and C. WASM also has a faster runtime than the EVM, resulting in faster execution. Generally, a10x improvement has been seen for contracts using WASM languages compared to contracts using Solidity.

How is this possible?

Stylus is only possible because of Arbitrum Nitro's unique fraud-proving technology. When there's a dispute on an Arbitrum network, Nitro replays the execution of the chainin WASM. Honest Arbitrum validators will then bisect what is being disputed until a single invalid step is identified and checked on-chain via a“one-step proof.”

This means Nitro is able to deterministically provearbitrary WASM .

If WASM can be proven in general, then the correctness ofany program that compiles down to WASM can be proven. This is the key technological leap that makes Stylus possible.

For a detailed overview of Nitro's technical architecture, see thedocumentation or theNitro whitepaper .

Why does this matter?

Stylus innovates on many levels, with the key ones described here:

One chain, many languages

There are estimated to be roughly 20k Solidity developers, compared to 3 million Rust developers or 12 million C developers [1]. Developers are now free to use their preferred programming language, all interoperable on any Arbitrum chain with Stylus. Scaling to the next billion users requires onboarding the next million developers.

EVM+

Stylus brings the best of both worlds. Developers still get all of the benefits of the EVM, including the ecosystem and liquidity, while getting efficiency improvements and access to existing libraries in Rust, C, and C++. All without changing anything about how the EVM works. EVM equivalence is no longer the ceiling, it's the floor.

Advanced cryptography is now inexpensive

Use cases not practical in the EVM are now possible in Stylus. Computation is over 10x improved. Memory is over 100x improved. Cryptography libraries can now be deployed as custom precompiles, permissionlessly. The greatest breeding ground for blockchain innovation has arrived.

Opt-in reentrancy

Stylus doesn't just improve on cost and speed. WASM programs are also safer. Reentrancy is a common vulnerability that developers can only attempt to mitigate in Solidity. Stylus provides cheap reentrancy detection, and using the Rust SDK, reentrancy is disabled by default, unless intentionally overridden.

Fully interoperable

Solidity programs and WASM programs are completely composable. If working in Solidity, a developer can call a Rust program or rely on another dependency in a different language. If working in Rust, all Solidity functionalities are accessible out of the box.

How does it work?

There are four main steps for bringing a Stylus program to life — coding, compilation, execution, and proving.

Coding

In principle, developers can write smart contracts in any programming language that can be compiled into WASM. In practice, some high-level languages generate far more performant WASMs than others.

Initially, there will be support for Rust, C, and C++. However, the levels of support will differ at first. Rust has rich language support from day one, with an open-source SDK that makes writing smart contracts in Rust as easy as possible. C and C++ are supported off the bat too, which will enable deploying existing contracts in those languages on-chain with minimal modifications.

The Stylus SDK for Rust contains the smart contract development framework and language features most developers will need to use in Stylus. The SDK also makes it possible to perform all of the EVM-specific functionalities that smart contract developers are used to. Check out theRust SDK Guide and theCrate Docs .

Compilation

Stylus programs are compiled twice. Once from a high-level language (such as Rust, C, or C++) to WASM, and then once more, in a process calledactivation , from WASM to a node's native machine code (such as ARM or x86).

The first stage of compilation happens either using the CLI tool provided in the Stylus SDK for Rust or by using any other compiler, such as Clang for C and C++. Once compiled, the WASM is posted on-chain. While the contract's behavior is now defined, it still cannot be called until after it's activated.

Activating a Stylus program requires a new precompile,ArbWasm . This precompile produces efficient binary code tailored to a node's native assembly. During this step, a series of middlewares ensure user programs can be safely executed and deterministically fraud-proven. Instrumentation includes gas metering, depth-checking, memory charging, and more to guarantee all WASM programs are safe for the chain to execute.

Execution

Stylus programs are executed in a fork ofWasmer , the leading WebAssembly runtime, with minimal changes made to optimize their codebase for blockchain-specific use cases. Wasmer executes native code much faster than Geth executes EVM bytecode, contributing to the significant gas savings that Stylus provides.

EVM contracts continue to execute the same way that they did prior to Stylus. When a contract is called, the difference between an EVM contract and a WASM program can be seen via anEOF -inspired contract header. From there, the contract is executed using its corresponding runtime. Contracts written in Solidity and WASM languages can make cross-contract calls to each other, meaning a developer never has to consider what language the contract was written in. Everything is interoperable.

Proving

Nitro has a happy case and a sad case. Most of the time, it's in a happy case, compiling execution history to native code. In the sad case where there is a dispute between validators, Nitro compiles execution history to WASM to conduct interactive fraud proofs on Ethereum. Stylus is a natural extension to Nitro's fraud-proving technology, utilizing it to not only bisect over execution history but also any WASM programs deployed by developers.

What can I do?

The Stylus testnet is open to all. A lot of thought has been given to creating the best programming experience possible. However, the work doesn't stop here. Feedback gained from developers will help drive Stylus to the next level, improving tooling, documentation, and language features. Becoming an early adopter of Stylus is the best way to get familiar with the opportunities it presents.

Many possibilities are enabled because of Stylus, to name a few:

- Custom precompiles for alternative signature schemes, such as secp256r1
- Generative art libraries that consume a bunch of RAM
- Bringing existing games written in C++ on-chain
- Compute-heavy AI models

The most exciting part about Stylus is all of the novel use cases that haven't even been dreamt of in the EVM. This is because none of this was possible, until today.

While many developers will be drawn to the net new use cases, rebuilding existing applications in Stylus will also open the door to innovation. dApps have never been faster, cheaper, or safer.

If you're a developer interested in Stylus, visit the quickstart , join the Discord channel , and start building!

Wen mainnet?

Stylus is inalpha , which means there are a lot of planned improvements on the roadmap! A few high-level next steps for Stylus include:

- An audit of the Stylus source code
- to ensure the safety of the contracts, as well as the Stylus SDK
- Enabling Stylus for mainnet Orbit Chains
- , which can be deployed and upgraded permissionlessly
- Submitting a proposal to the Arbitrum DAO, to activate Stylus on Arbitrum testnets, and eventually Arbitrum One and Arbitrum Nova mainnet

To keep up with the latest announcements, join the Stylus Discord channel and follow the Arbitrum Developers Twitter account . Edit this page Last updatedonMar 7, 2024 Previous Public preview: What to expect Next Quickstart: Write a smart contract in Rust using Stylus