TLDR

- This post provides feedback on the developer experience using Stylus
- Introduces a new developer toolkit repository that provides easy to integrated Stylus building blocks
- Share information on the gas cost benchmarks that were performed

Hi guys, we're a blockchain development services & research company - LimeChain. We conducted research and benchmarking on Arbitrum Stylus, as detailed in our benchmark repository linked below. While exploring its capabilities, we encountered several hurdles that impacted the developer experience (DevEx). In this brief, we are sharing those findings with the hope of addressing them in the future in order to maximise the developer adoption of Stylus.

- Benchmark - stylus-benchmark
- Stylus toolkit - stylus-toolkit

Disclaimer

The following notes represent the subjective

perception of the web3 developers (infrastructure + dApp devs) at LimeChain who spend time developing Stylus programs.

Feedback

1. The development lifecycle is broken between the native rust tooling and cargo stylus

. One example of this is the lack of compile

command on the cargo stylus

CLI. For some developers, it may be non-intuitive to use the native cargo

CLI tool for compilation and then use another CLI (cargo stylus

) for checking the binary and deploying it. Additional friction is introduced due to the fact that there needs to be an explicit (different from the default) compilation target and compression configuration for Stylus programs. This showcases that there is a need for a single "build, no config necessary" command.

1. The lack of constructor support (at least it is not evident that it is supported) increases the code complexity. An ERC20 contract that normally accepts symbol

, name

and decimals

as constructor arguments introduces the usage of generics in Stylus contracts. This increases the size (words) and complexity of the code that needs to be written compared to Solidity.

1. Stylus programs require developers to set up the project for both binary and library and that is non-intuitive. Binaries are required for exploring the ABI

of the program, whereas a library is required for the deployment process. This makes the development environment and configuration more complex and surely adds confusion for devs on how they need to configure their project.

1. The current development stack is tailored for the development of programs with a single entrypoint. This is not the case for the development of EVM-based contracts. As developers with extensive EVM experience, we noticed additional friction when setting up the project where we had to compile and deploy multiple Stylus programs. This is seen in the benchmark repository where a user needs to explicitly comment/uncomment and make available only a single entrypoint.
2. Although there is a local Arbitrum Nova node with Stylus support that can be started using 1-2 CMD commands (kudos to that), we believe that an integrated local EVM+ module is necessary. Ideally, there should be a robust and user-friendly setup for local development that simulates both EVM and WASM in order to provide the best DevEx.
3. Even compact and WASM-optimised crates (f.e ed25519-compact

) have to be explicitly configured and stripped down to fit into the size requirements of Stylus programs. We believe that a dedicated effort must be put in order to provide a set of primitives (cryptographic functions, utilities such as MIPs etc) that are optimised to deliver the best performance as Stylus programs.

We believe that Stylus has huge potential to move the needle in terms of developer adoption in the web3 space and for that

reason, we've spent time developing with it. Although Stylus is still in its early stages of development we see its great potential.