It would be interesting to discuss what languages do we expect/want/recommend people to use for writing smart contracts for Ethereum 2.0. Honestly, I think I'm just poking around in the dark, but maybe that's another reason to put this out and gather thoughts.

Taking note of what @gcolvin said in the EVM performance topic, I think this could be divided into three scenarios:

1. Ethereum won't switch to eWASM

– at least not anytime soon, even after the completion of Serenity. Quite a pessimistic timeline, but it could happen.

1. The switch to eWASM is successful during Serenity

, the only place where the EVM comes up is the legacy EVM shard, but EVM-to-eWASM pathways exists and most people/dapps use the eWASM shards anyway.

1. The use of the EVM and eWASM continues in parallel

.

Scenario #1

- How would the current (EVM-era) smart contract language landscape change

by the fact that Ethereum is now in Serenity?

- Would there be a need for a new language

?

Scenario #2

Here I suspect the options would be the following:

- Any of the traditional languages

that compile to WASM will probably get an eWASM compiler as well. Last time I checked C and C++ had full WASM support, Rust, (Go?) and some others had experimental support, and the long-term goal was to create compilers to as many languages as possible. While this opens up a lot of possibilities, I guess some WASM-supporting languages will be much better for writing Serenity smart contracts than others

: * if for nothing else, because of the idiosyncrasies of eWASM compared to WASM

;

- because of the (non-)existence of compilers

and their quality/developer base (see the current state of WASM compilers);

- because of the idiosyncrasies of the languages

. This could be the heaviest one. * Within this, it could be that different languages will be better at different things, e.g. X will be more gas-efficient, Y will yield safer contracts, etc. Which languages will be suitable for what purposes?

If this can be ascertained in advance, we could make better recommendations in time and avoid the spread of problematic and unsafe languages.

- Within this, it could be that different languages will be better at different things, e.g. X will be more gas-efficient, Y will yield safer contracts, etc. Which languages will be suitable for what purposes?

If this can be ascertained in advance, we could make better recommendations in time and avoid the spread of problematic and unsafe languages.

- if for nothing else, because of the idiosyncrasies of eWASM compared to WASM

;

- because of the (non-)existence of compilers

and their quality/developer base (see the current state of WASM compilers);

- because of the idiosyncrasies of the languages

. This could be the heaviest one. * Within this, it could be that different languages will be better at different things, e.g. X will be more gas-efficient, Y will yield safer contracts, etc. Which languages will be suitable for what purposes?

If this can be ascertained in advance, we could make better recommendations in time and avoid the spread of problematic and unsafe languages.

- Within this, it could be that different languages will be better at different things, e.g. X will be more gas-efficient, Y will yield safer contracts, etc. Which languages will be suitable for what purposes?

If this can be ascertained in advance, we could make better recommendations in time and avoid the spread of problematic and unsafe languages.

- If I understand it correctly, he current languages used in Ethereum

will (also) only be usable in Serenity if they get a compiler. * I heard that one of the goals of the yevm project is to be able to compile yul

to eWASM. * Does this mean that, provided that yevm succeeds, Solidity will also be able to compile to eWASM through yul?

- At Devcon, @axic said that Vyper could

compile to yul (so that + yevm could = eWASM-compatibility), although I haven't heard about such effort from Vyper devs. The Vyper repo gives the impression it's only for the EVM.

- Does this mean that, provided that yevm succeeds, Solidity will also be able to compile to eWASM through yul?

- At Devcon, @axic said that Vyper could

compile to yul (so that + yevm could = eWASM-compatibility), although I haven't heard about such effort from Vyper devs. The Vyper repo gives the impression it's only for the EVM.

- Will either Solidity

or Vyper

have any non-yul ways of compiling to eWASM, or does this hinge solely on yul and/or yevm?

- I heard that one of the goals of the yevm project is to be able to compile yul

to eWASM. * Does this mean that, provided that yevm succeeds, Solidity will also be able to compile to eWASM through yul?

- At Devcon, @axic said that Vyper could

compile to yul (so that + yevm could = eWASM-compatibility), although I haven't heard about such effort from Vyper devs. The Vyper repo gives the impression it's only for the EVM.

- Does this mean that, provided that yevm succeeds, Solidity will also be able to compile to eWASM through yul?

- At Devcon, @axic said that Vyper could

compile to yul (so that + yevm could = eWASM-compatibility), although I haven't heard about such effort from Vyper devs. The Vyper repo gives the impression it's only for the EVM.

- Will either Solidity

or Vyper

have any non-yul ways of compiling to eWASM, or does this hinge solely on yul and/or yevm?

- I don't know how probable this is, but could it be that eWASM will demand a new smart contract language

/ a new custom language will perform better than all others?

Here, it would also be interesting to look at how traditional languages and smart contract-specific languages compare.

Scenario #3

Obviously, this would be some kind of a mix of what comes up in scenarios #1

and #2

.

If you have recommendations, answers, additions, please post them!

Also, if I got something wrong (I’m quite certain I did), feel free correct me!