Variables

In Solidity, there are 3 types of variables: local, state, and global. Local variables are not stored on the blockchain, while state variables are (and incur a much higher cost as a result). This is true of Arbitrum Stylus Rust smart contracts as well, although how they're defined is quite different.

In Rust,local variables are just ordinary variables you assign withlet orlet mut statements. Local variables are far cheaper than state variables, even on the EVM, however, Stylus local variables are more than 100x cheaper to allocate in memory than their Solidity equivalents.

Unlike Solidity, Rust was not built inherently with the blockchain in mind. It is a general purpose programming language. We therefore define specificstorage types to explicitly denote values intended to be stored permanently as part of the contract's state.State variables cost the same to store as their Solidity equivalents.

Global variables in Solidity, such asmsg.sender andblock.timestamp , are available as function calls pulled in from thestylus_sdk with their Rust equivalents beingmsg::sender() andblock::timestamp() , respectively. These variables provide information about the blockchain or the active transaction.

Learn more

- Rust Docs - Variables and Mutability
- Stylus SDK Rust Docs - Storage
- Stylus SDK Guide - Storage
- Solidity docs - state variables
- Solidity docs - global variables

src/lib.rs

note This code has yet to be audited. Please use at your own risk. // Only run this as a WASM if the export-abi feature is not set.

![cfg_attr(not(any(feature =

"export-abi" , test)), no_main)] extern

crate

alloc ;

use

stylus_sdk :: alloy_primitives :: { U16 ,

U256 } ; use

stylus_sdk :: prelude :: * ; use

stylus_sdk :: storage :: { StorageAddress ,

StorageBool ,

StorageU256 } ; use

stylus_sdk :: { block , console , msg } ;

[storage]

[entrypoint]

pub

struct

Contract

{ initialized :

StorageBool , owner :

StorageAddress , max_supply :

StorageU256 , }

[public]

impl

Contract

{ // State variables are initialized in aninit function. pub

fn

init (& mut

self)

->

Result < ( ) ,

Vec < u8

{ // We check if contract has been initialized before. // We return if so, we initialize if not. let initialized =

self . initialized . get ( ) ; if initialized { return

Ok (( )) ; } self . initialized . set (true) ;

// We set the contract owner to the caller, // which we get from the global msg module self . owner . set (msg :: sender ( )) ; self . max_supply . set (U256 :: from (10_000)) ;

Ok (( )) }

pub

fn

do_something ( )

->

Result < ( ) ,

Vec < u8

{ // Local variables are not saved to the blockchain // 16-bit Rust integer let _i =

456_u16 ; // 16-bit int inferred from U16 Alloy primitive let _j =

U16 :: from (123) ;

// Here are some global variables let _timestamp =

block :: timestamp ( ) ; let _amount =

msg :: value ( ) ;

console! ("Local variables: {_i}, {_j}") ; console! ("Global variables: {_timestamp}, {_amount}") ;

Ok (( )) } }

Cargo.toml

[package] name

=

"stylus_variable_example" version

=

"0.1.7" edition

=

"2021" license

=

"MIT OR Apache-2.0" keywords

=

["arbitrum" ,

"ethereum" ,

"stylus" ,

"alloy"]

[dependencies] alloy-primitives

=

"=0.7.6" alloy-sol-types

=

"=0.7.6" mini-alloc

=

"0.4.2" stylus-sdk

=

"0.6.0" hex

=

"0.4.3"

[dev-dependencies] tokio

=

{

version

=

"1.12.0" ,

features

=

["full"]

} ethers

=

"2.0" eyre

=

"0.6.8"

[features] export-abi

=

["stylus-sdk/export-abi"]

[lib] crate-type

=

["lib" ,

"cdylib"]

[profile.release] codegen-units

=

1 strip

=

true lto

=

true panic

=

"abort" opt-level

=

"s" Edit this page Previous Primitive Data Types Next Constants