The Contract Class

Your contract's logic and state (storage) is defined by themain class , in which:

1. The attributes define thecontract's state
2. Theinitialization methods
3. define how to initialize the contract's state
4. The public methods act as the contract's interface with the rest of the network

Defining the Contract

The contract is just another class, with its own attributes and methods. Todifferentiate it from other internal classes simply decorate it using theNEAR Bindgen decorator/macro .

- JavaScript
- Rust

contract-ts/src/contract.ts loading ... See full example on GitHubcontract-rs/src/lib.rs loading ... See full example on GitHub Under the hood, theNEAR Bindgen decorator/macro traverses the class, generating the necessary code to:

1. Transform the code into a valid NEAR contract.
2. Expose public methods, so they can be called externally.
3. Serialize objects for internal storage and communication with external actors.

The State

Each account has its own state (storage), whichonly they can modify buanyone can see .

The state is defined and modified through themain class' attributes .

Contractspay for their storage by locking part of their balance. Currently it costs~1 Ⓝ to store100kb

Key-Value Storage The contract actually uses akey-value storage to persist values. This however is abstracted from you by the SDK throughserialization . Prefer SDK Collections When defining attributes,always preferSDK collections over native ones, since they are optimized forserialization .

Initializing the State

There are two ways to initialize the account's state, and they can co-exist:

1. Aninitialization method
2. that receives the attributes needed for the state
3. Adefault state
4. , which will be used untilinit
5. is invoked, or a method writes into the state

Initialization Method

To define an initialization method simply decorate it with theinitialization macro .

The method will now be able to define the initial state's values, raising an error if invoked whilethe state is already initialized .

- JavaScript
- Rust

contract-ts/src/contract.ts loading ... See full example on GitHubinfo To make the initialization mandatory use@NearBindgen({requireInit: true}) caution In JavaScript youmust always define adefault state contract-rs/src/lib.rs loading ... See full example on GitHubinfo To make the initialization mandatory use#[derive(PanicOnDefault)] in the contract's structure

Default State

Contracts can define adefault state to use if no initialize method is called. This is, if any method is invoked before aninit happens, the contract will use the default values.

Once any method writes into the state, the state will be considered initialized.

- JavaScript
- Rust

contract-ts/src/contract.ts loading ... See full example on GitHub In JavaScript, the default state is defined by the initialization parameters in the class definition.

caution In Javascript youmust always assign values toall the class' parameters . This ensures they get correctly deserialized to their intended type. contract-rs/src/lib.rs loading ... See full example on GitHub

Interface

All thepublic methods are exposed to the network as the contract's interface.

- JavaScript
- Rust

@ NearBindgen ({ }) class

Contract

{

@ initialize ({

...

}) init ({

...

})

{

/ *public init method* /

}

@ view ({ }) get_message ({

...

})

{

/ *public view method* /

}

@ call ({ }) add_message ({

...

})

{

/ *public call method* /

}

private

internal_search (

...

)

{

/ *private internal method* /

}

@ call ({ privateFunction :

true }) set_owner ({

...

})

{

/ *public, panics when caller is not the contract's account* /

} }

[near_bindgen]

impl

Contract

{

[init]

pub

fn

init (

...

)

->

Self

{

/ *public init method* /

} pub

fn

get_message (& self ,

...

)

{

/ *public view method* /

} pub

fn

add_message (& mut

self ,

...

)

{

/ *public call method* /

} fn

internal_search (& self ,

...

)

{

/ *private internal method* /

}

[private]

pub

fn

set_owner (& mut

self ,

...

)

{

/ *public, panics when caller is not the contract's account* /

} }

Public Methods

Public methods can be categorized in three types:init methods,view methods, andcall methods.

- Init Methods
- : They define how to initialize the state of the contract.
- View Methods
- : Donot mutate
- the statenor call
- other contracts. They can
- be called for free by everyone,without needing
- a NEAR account.
- Call Methods
- : They can mutate the state and performactions
- such
- as calling other contracts.

caution By defaultview methods have200TGas to execute, to increase this you can simply invoke them ascall methods.
danger By defaultinit methods are public, make sure todecorate them asprivate , orbatch call the initialization on deploy

Private Methods

Sometimes you will want some methods to remain public, but only be callable by the contract's account. Such is the case for example ofcross-contract callbacks .

For this, you can use theprivate macro/decorator.

- JavaScript
- Rust

@ call ({ privateFunction :

true }) callback (

...

) { // this method can only be called by the contract's account }

[private]

pub

fn

callback (& mut

self ,

...

) { // this method can only be called by the contract's account }

Payable Methods

By defaultall methods panic if a userattaches money while calling them. To enable a method to receive money use the payable decorator.

- JavaScript
- Rust

@ call ({ payableFunction :

true }) deposit_and_stake (

...

) { // this method can receive money from the user }

[payable]

pub

fn

deposit_and_stake (& mut

self ,

...

) { // this method can receive money from the user }

Input & Return Types

The contract can receive and return anynative type , including complex structures. However, since contracts communicate through their interfaceusing JSON :

- Always prefernative types
- overSDK Collections
- in the input & return types.
- Replaceu64
- /u128
- forstrings
- (U64
- /U128
- in the Rust SDK). Edit this page Last updatedonApr 12, 2024 byDamian Parrino Was this page helpful? Yes No Need some help?Chat with us or check ourDev Resources ! Twitter Telegram Discord Zulip

Previous Modules, Types & Structs Next Environment