Public Method Types

Methods can be called externally by using thepub identifier within the#[near_bindgen] macro which will expose the method in the compiled WASM bytecode.

It is important to only mark methods that should be called externally as public. If you need a contract to call itself, you can mark the function as public but add the#[private] annotation so that it will panic if called from anything but the contract itself.

A basic usage of this would look like the following:

[near_bindgen]

impl

MyContractStructure

{ pub

fn

some_method (& mut

self)

{ // .. method logic here } } Where this would exposesome_method from the WASM binary and allow it to be called externally.

Expand to see generated code

[cfg(target_arch =

"wasm32")]

[no_mangle]

pub

extern

"C"

fn

some_method ( )

{ near_sdk :: env :: setup_panic_hook ( ) ; if

near_sdk :: env :: attached_deposit ( )

!=

0

{ near_sdk :: env :: panic ("Method some_method doesn\'t accept deposit" . as_bytes ( )) ; } let

mut contract :

MyContractStructure

=

near_sdk :: env :: state_read ( ) . unwrap_or_default ( ) ; contract . some_method ( ) ; near_sdk :: env :: state_write (& contract) ; }

Exposing trait implementations

Functions can also be exposed through trait implementations. This can be useful if implementing a shared interface or standard for a contract. This code generation is handled very similarly to basicpub functions, but the#[near_bindgen] macro only needs to be attached to the trait implementation, not the trait itself:

pub

trait

MyTrait

{ fn

trait_method (& mut

self) ; }

[near_bindgen]

impl

MyTrait

for

MyContractStructure

{ fn

trait_method (& mut

self)

{ // .. method logic here } } In this example, the generated code will be the same as the previous example, except with a different method name.

Expand to see generated code

[cfg(target_arch =

"wasm32")]

[no_mangle]

pub

extern

"C"

fn

trait_method ( )

{ near_sdk :: env :: setup_panic_hook ( ) ; if

near_sdk :: env :: attached_deposit ( )

!=

0

{ near_sdk :: env :: panic ("Method trait_method doesn\'t accept deposit" . as_bytes ( )) ; } let

mut contract :

MyContractStructure

=

near_sdk :: env :: state_read ( ) . unwrap_or_default ( ) ; contract . trait_method ( ) ; near_sdk :: env :: state_write (& contract) ; } Edit this page Last updatedonAug 24, 2022 byDamián Parrino Was this page helpful? Yes No

Previous Collections Nesting Next Contract Mutability