Payable Methods

We can allow methods to accept a NEAR token transfer together with the function call. This is done so that contracts can define a fee in tokens that needs to be paid when they are used. By default the methods are not payable and they will throw an error if someone will attempt to attach tokens to them during the invocation. This is done for safety reasons, in case someone accidentally transfers tokens during the function call.

To declare a method as payable, use the({ payableFunction: true }) decorator parameter within the[NearBindgen decorated contract class]() as follows:

@ call ({

payableFunction :

true

}) my_method ({ })

{ // ... } This will allow themy_method function to be called with NEAR Tokens attached to the call, transferring tokens to the contract.

Example:

@ NearBindgen ({ }) export

class

Contract

{ @ call ({

payableFunction :

true

}) take_my_money ({ })

{ near . log ("Thanks!") ; }

@ call ({ }) do_not_take_my_money ({ })

{ near . log ("No thanks!") ; } } is equivalent to:

@ NearBindgen ({ }) export

class

Contract

{ @ call ({ }) take_my_money ({ })

{ near . log ("Thanks!") ; }

@ call ({ }) do_not_take_my_money ({ })

{ if

(near . attachedDeposit ( )

BigInt (0))

{ throw

new

Error ("Method do_not_take_my_money doesn't accept deposit") ; } near . log ("No thanks!") ; } }[Edit this page]() Last updatedonJan 20, 2023 byDennis Was this page helpful? Yes No

[Previous Private Methods]() [Next Callbacks]()