Creating Accounts

You might want to create an account from a contract for many reasons. One example: You want toprogressively onboard users, hiding the whole concept of NEAR from them at the beginning, and automatically create accounts for them (these could be sub-accounts of your main contract, such asuser123.some-cool-game.near).

Since an account with no balance is almost unusable, you probably want to combine this with the token transfer fromthe last page . You will also need to give the account an access key. Here's a way do it:

NearPromise . new ("subaccount.example.near") . createAccount ( ) . addFullAccessKey (near . signerAccountPk ( )) . transfer (BigInt (250_000_000_000_000_000_000_000)) ;

// 2.5e23yN, 0.25N In the context of a full contract:

import

{

NearPromise , near }

from

"near-sdk-js" ;

@ NearBindgen ({ }) export

class

Contract

{ @ call ({

privateFunction :

true

}) createSubaccount ({ prefix })

{ const subaccountId =

{ prefix } . { near . currentAccountId ( ) } ; return

NearPromise . new (subaccount_id) . createAccount ( ) . addFullAccessKey (near . signerAccountPk ( )) . transfer (BigInt (250_000_000_000_000_000_000_000)) ;

// 2.5e23yN, 0.25N } } Things to note:

- addFullAccessKey
- – This example passes in the public key of the human or app that signed the original transaction that resulted in this function call (signerAccountPk
-). You could also useaddAccessKey
- to add a Function Call access key that only permits the account to make calls to a predefined set of contract functions.
- { privateFunction: true }
- – if you have a function that spends your contract's funds, you probably want to protect it in some way. This example does so with a perhaps-too-simple{ privateFunction: true }
- decorator parameter. Edit this page Last updatedonJan 20, 2023 byDennis Was this page helpful? Yes No

Previous Sending Native Tokens Next Deploying Contracts