Donation

Our donation example enables to forward NEAR Tokens to an account while keeping track of it. It is one of the simplest examples on making a contract handle tranfers.

Frontend of the Donation App

Obtaining the Donation Example

You have two options to start the Donation Example.

1. You can use the app throughGithub Codespaces
2. , which will open a web-based interactive environment.
3. Clone the repository locally and use it from your computer.

Codespaces Clone locally https://github.com/near-examples/donation-examples.git

Structure of the Example

The example is divided in two main components:

1. The smart contract, available in two flavors: rust and javascript
2. The frontend, that interacts with an already deployed contract.
3. JavaScript
4. Rust

┌── sandbox-ts # sandbox testing | ├── src | | └── main.ava.ts | ├── ava.config.cjs | └── package.json ├── src # contract's code | ├── contract.ts | ├── model.ts | └── utils.ts ├── package.json # package manager ├── README.md └── tsconfig.json # test script ┌── tests # workspaces testing | ├── workspaces.rs ├── src # contract's code | ├── donation.rs | └── lib.rs ├── Cargo.toml # package manager ├── README.md ├── rust-toolchain.toml

Frontend

The donation example includes a frontend that interacts with an already deployed smart contract, allowing user to donate NEAR tokens to a faucet service.

Running the Frontend

To start the frontend you will need to install the dependencies and start the server.

cd frontend yarn yarn start Go ahead and login with your NEAR account. If you don't have one, you will be able to create one in the moment. Once logged in, input the amount of NEAR you want to donate and press the donate button. You will be redirected to the NEAR Wallet to confirm the transaction. After confirming it, the donation will be listed in the "Latest Donations".

Understanding the Frontend

The frontend is composed by a single HTML file (/index.html), while the logic lives in/index.js , which communicates with the contract through/near-interface.js .

- index.js
- near-interface.js
- near-wallet.js

frontend/index.js loading ... See full example on GitHub frontend/near-interface.js loading ... See full example on GitHub frontend/near-wallet.js loading ... See full example on GitHub An interesting aspect of the donation example is that it showcases how to retrieve a result after being redirected to the NEAR wallet to accept a transaction.

Smart Contract

The contract exposes methods to donate tokens (donate), and methods to retrieve the recorded donations (e.g.get_donation_by_number).

- JavaScript
- Rust

contract-ts/src/contract.ts loading ... See full example on GitHub contract-rs/src/donation.rs loading ... See full example on GitHub

Testing the Contract

The contract readily includes a set of unit and sandbox testing to validate its functionality. To execute the tests, run the following commands:

- JavaScript
- Rust

cd contract-ts yarn yarn test cd contract-rs cargo test tip Theintegration tests use a sandbox to create NEAR users and simulate interactions with the contract.

Deploying the Contract to the NEAR network

In order to deploy the contract you will need to create a NEAR account .

- JavaScript
- Rust

Optional - create an account

near create-account --useFaucet

Deploy the contract

cd contract-ts yarn build near deploy ./build/donation.wasm

Optional - create an account

near create-account --useFaucet

Deploy the contract

cd contract-rs cargo near build near deploy ./target/wasm32-unknown-unknown/release/donation.wasm tip To interact with your contract from thefrontend , simply replace the variableCONTRACT_NAME in theindex.js file.

CLI: Interacting with the Contract

To interact with the contract through the console, you can use the following commands

Get donations

Optional arguments for pagination

near view donation.near-examples.testnet get_donations --args='{"from_index": 0,"limit": 10}'

Get beneficiary

near view donation.near-examples.testnet get_beneficiary

Get number of donors

near view donation.near-examples.testnet number_of_donors

Get donation for an account

Require accountId

near view donation.near-examples.testnet get_donation_for_account --args='{"account_id"}'

Donate to the contract

Replace with your account ID

Require deposit

near call donation.near-examples.testnet donate --accountId--deposit 0.1 tip If you're using your own account, replacedonation.near-examples.testnet with youraccountId .

Moving Forward

A nice way to learn is by trying to expand a contract. Modify the donation example so it accumulates the tokens in the contract instead of sending it immediately. Then, make a method that only thebeneficiary can call to retrieve the tokens. Edit this page Last updatedonMar 13, 2024 bydj8yf0µl Was this page helpful? Yes No

Previous Guest Book Next Cross Contract Call