

# Count on NEAR

Our counter example is a friendly decentralized app that stores a number and exposes methods to increment, decrement, and reset it.

## Obtaining the Counter Example

You have two options to start the Counter Example.

1. You can use the app through GitHub 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/counters>

## 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. `src` JavaScript
4. `lib.rs` Rust

```
├── sandbox-ts # sandbox testing | ├── src | | ├── main.ava.ts | ├── ava.config.cjs | └── package.json
├── src # contract's code | ├── contract.ts | ├── package.json # package manager | ├── README.md |
├── tsconfig.json # test script | ├── sandbox-ts # sandbox testing | ├── src | | ├── main.ava.ts | |
├── ava.config.cjs | ├── package.json | ├── src # contract's code | ├── lib.rs | ├── build.sh # build script |
├── Cargo.toml # package manager | ├── README.md | └── rust-toolchain.toml
```

## Frontend

The counter example includes a frontend interface designed to interact seamlessly with an existing smart contract that has been deployed. This interface allows users to increase or decrease the counter as needed.

### 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, use the + and - buttons to increase and decrease the counter. Then, use the Gameboy buttons to reset it and make the counter blink an eye!

Frontend of the Counter

### Understanding the Frontend

The frontend is composed by a single HTML file (`/index.html`). This file defines the components displayed in the screen.

The website's logic lives in `/index.js`, which communicates with the contract through `/near-wallet.js`. You will notice in `/index.js` the following code:

- `near` JavaScript

`frontend/index.js` loading ... [See full example on GitHub](#) It indicates our app, when it starts, to check if the user is already logged in and execute either `signedInFlow()` or `signedOutFlow()`.

## Smart Contract

The contract presents 4 methods: `get_num`, `increment`, `decrement`, and `reset`. The method `get_num` retrieves the current value, and the rest modify it.

- `src` JavaScript
- `lib.rs` Rust

contract-ts/src/contract.ts loading ... [See full example on GitHub](#) contract-rs/src/lib.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 ./test.sh tip The integration 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/counter.wasm

## Optional - create an account

near create-account --useFaucet

## Deploy the contract

cd contract-rs ./build.sh near deploy ./target/wasm32-unknown-unknown/release/counter.wasm tip To interact with your contract from the [frontend](#), simply replace the variable CONTRACT\_NAME in the index.js file.

## CLI: Interacting with the Contract

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

## Get the current number of the counter

near view counter.near-examples.testnet get\_num

## Increment the counter

## Replace with your account ID

near call counter.near-examples.testnet increment --accountId

## Decrement the counter

## Replace with your account ID

near call counter.near-examples.testnet decrement --accountId

# Reset the counter to zero

## Replace with your account ID

`near call counter.near-examples.testnet reset --accountId tip` If you're using your own account, replace `counter.near-examples.testnet` with `youraccountId`.

## Moving Forward

A nice way to learn is by trying to expand the contract. Modify it by adding a parameter `toIncrement` and `decrement`, so the user can choose by how much to change the value. For this, you will need to use knowledge from the [anatomy](#) and [storage](#) sections. [Edit this page](#) Last updated on Feb 21, 2024 by Sam Snowman(赵正中) Was this page helpful? Yes No

[Previous](#) [Home](#) [Next](#) [Guest Book](#)