Smart contracts w/ ReasonML

Implementing a counter from scratch

Matej Šima

@maht0rz | t.me/maht0rz | t.me/stove_labs | https://bit.ly/2uCFzz1

--help

- What is Tezos and why should I care?
- Tezos smart contracts ecosystem (as of April 2019)
- Overview of a Liquidity & Liquidity contracts
- Counter contract architecture
- Implementation and deployment of our contract (ReasonML)
- Interacting with our contract and increasing the counter
- Extra resources & links

What is Tezos and why should I care?

- White paper published in 2014 by Arthur Breitman
- ICO in 2017 raising over 200 million USD
- Mainnet launched in late 2018
- Implemented in OCaml

- Self amendable on the protocol level
- Proof of Stake consensus algorithm with a community built around block baking (400+ bakers)

- Active foundation support & ICO funds distribution back to the community & developers
- Native tokens called Tezzies (XTZ) (+140% from March to April 2019)

Tezos smart contracts ecosystem (April 2019)

- Natively implemented in Michelson language
  - Stack based
  - Assembly-like
  - Not the easiest to start with
  - Built with formal verification in mind (nice type system, functional)
- Languages built on top of Michelson
  - Liquidity
  - Morley
  - LIGO
  - Fi
  - …
- Ethereum has Solidity / Viper / ...

Overview of a Liquidity & Liquidity contracts

- Language itself is developed by OCamlPro (in France)
- ReasonML support was integrated recently
  - https://github.com/OCamlPro/liquidity/issues/173
- Custom literals and keywords (tz, address, contract, ...), compared to standard OCaml/ReasonML
- Contracts have storage & entry points
- There's a CLI for compilation, deployment & testing
- Deployment of a contracts happens by injecting a new operation into the blockchain

Counter contract architecture

- Each contract has its own on-chain storage
    - Our contract will store only an integer, as a counter
- Each contract has entry points
    - Our contract, will have a single entry point, to increment its counter (storage)
- Each entry point, should return a list of operations and a new storage value
    - We will not apply any operations, but we will only increment the storage value / counter
- We will allow a parametrized increment via our entry point, so that you can increment by more than just +1

Implementation and deployment of our contract

- Using Granary as an environment for contract & dapp development
- Start by setting up granary
    - $ git clone https://github.com/stove-labs/granary
    - $ source ./env/sandboxnet.sh
    - $ make start (starts a sandboxed tezos node)
    - $ make client (starts a CLI with liquidity preinstalled)
    - $ init-client (populates the sandboxed blockchain with test accounts)
- Local sandboxed Tezos node with exposed RPC is running
- Tezos-client alongside with Liquidity CLI is available as well
- Block explorer available at http://localhost:8000

Extra resources and links

- https://tzscan.io
- https://stove-labs.com
- https://tqgroup.io
- https://tezoscommons.org
- https://t.me/tezosplatform
- https://tezos.stackexchange.com