Server-less **dApp** on **Mobile** using **Blockchain**

MOBILEDAY [·] elefos

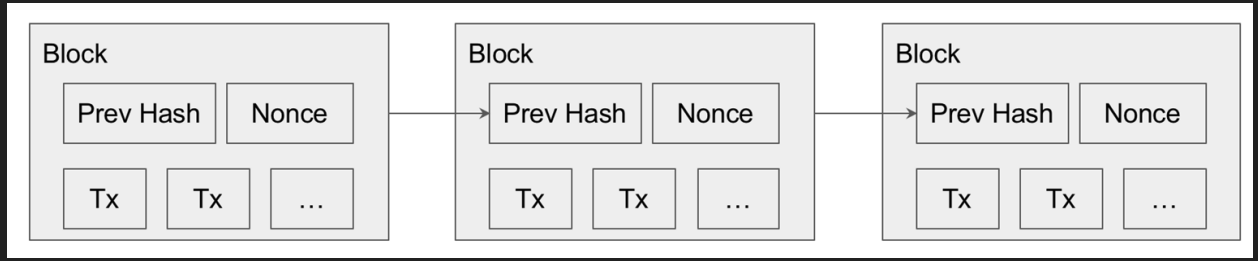
LE YEN THANH

Cofounder, CTO startup Denchlabs
Founder BusMap - Xe buýt thành phố
Website: lythanh.xyz
Email: abc@lythanh.xyz

MOBILEDAY elefos

What is Blockchain?

1) Blockchain is a concept
2) A blockchain, is a growing list of records, called blocks, which are linked using cryptography
3) Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data



MOBILEDAY [·] elefos

Why blockchain?

Trustless

Decentralized

Immutable

MOBILEDAY [·] elefos

What is a dApp?

DApp is an abbreviation for Decentralized Application.

DApps are new paradigm for building apps where a back end centralized server is replaced by a decentralized peer to peer network, i.e. a Blockchain

What is a smart contract?

A smart contract is a computer code running on top of a blockchain containing a set of rules under which the parties to that smart contract agree to interact with each other.

Smart contract on Ethereum – Written in Solidity

```
pragma solidity ^0.4.15;

contract MyBitcoin {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function MyBitcoin(uint256 initialSupply) {
        balanceOf[msg.sender] = initialSupply;              // Give the creator all initial tokens
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) {
        require(balanceOf[msg.sender] >= _value);           // Check if the sender has enough
        require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
        balanceOf[msg.sender] -= _value;                    // Subtract from the sender
        balanceOf[_to] += _value;                           // Add the same to the recipient
    }
}
```

MOBILEDAY elefos

Smart contract on EOS – Written in C++

```
1  #include <eosiolib/eosio.hpp>
2  #include <eosiolib/print.hpp>
3  using namespace eosio;
4
5  class hello : public eosio::contract {
6    public:
7        using contract::contract;
8
9        [[eosio::action]]
10       void hi( account_name user ) {
11          print( "Hello, ", name{user} );
12       }
13 };
14
15 EOSIO_ABI( hello, (hi) )
```

MOBILEDAY elefos

Smart contract on Aelf – Written in C#

```
using AElf.Sdk.CSharp;
using AElf.Sdk.CSharp.Types;
using Google.Protobuf;
using Google.Protobuf.WellKnownTypes;
using CSharpSmartContract = AElf.Sdk.CSharp.CSharpSmartContract;
using Api = AElf.Sdk.CSharp.Api;

namespace AElf.Contracts.Examples
{
    public class SimpleTokenContract : CSharpSmartContract
    {
        [SmartContractFieldData("${this}.Balances", DataAccessMode.AccountSpecific)]
        public Map Balances = new Map("Balances");

        public Map TransactionStartTimes = new Map("TransactionStartTimes");
        public Map TransactionEndTimes = new Map("TransactionEndTimes");

        public async Task<object> InitializeAsync(Hash account, ulong qty)
        {
            await Balances.SetValueAsync(account, qty.ToBytes());
            return null;
        }

        public override async Task InvokeAsync()
        {
            var tx = Api.GetTransaction();
```

MOBILEDAY elefos

Smart contract on Zilliqa – Written in Scilla

```
library HelloWorld

let one_msg =
  fun (msg : Message) =>
  let nil_msg = Nil {Message} in
  Cons {Message} msg nil_msg

let not_owner_code = Int32 1
let set_hello_code = Int32 2

(***************************************************)
(*             The contract definition             *)
(***************************************************)

contract HelloWorld
(owner: ByStr20)

field welcome_msg : String = ""

transition setHello (msg : String)
  is_owner = builtin eq owner _sender;
  match is_owner with
  | False =>
    msg = {_tag : "Main"; _recipient : _sender; _amount : Uint128 0; code : not_owner_code};
    msgs = one_msg msg;
    send msgs
  | True =>
    welcome_msg := msg;
    msg = {_tag : "Main"; _recipient : _sender; _amount : Uint128 0; code : set_hello_code};
    msgs = one_msg msg;
    send msgs
  end
end
```

MOBILEDAY elefos

4 layers of a Blockchain system

1) Layer 0: Network layer
2) Layer 1: Core, Consensus Algorithm, Virtual Machine
3) Layer 2: Off-chain layer for scalability
4) Layer 3: Decentralized Application (dApp)

What do we need to learn Blockchain?

Use Blockchain for Transaction	Create dApp with smart contracts logic on Blockchain	Build your own Blockchain
⇒ Use API to connect to Blockchain system to create transaction, don't need to know how it works	⇒ Learn how to write smart contracts and how to setup a system which can interact with the Blockchain directly	⇒ Build a Blockchain platform to manipulate all the transactions and logic in your system
(building an app without Backend) - Backend development - Web development - Mobile development	**(build an app with Backend and DB)** - Smart contract implementation - Backend + DevOps - Cryptography	**(building an OS)** - Cryptography (advanced) - Math: Algebra, discrete math, group theory, number theory,… - Algorithm: Game theory, Optimization,… - Network

Blockchain Engineer

MOBILEDAY elefos

Remember: Smart contract may contain bugs => Can be hacked!

```
254
255   function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
256     uint cnt = _receivers.length;
257     uint256 amount = uint256(cnt) * _value;
258     require(cnt > 0 && cnt <= 20);
259     require(_value > 0 && balances[msg.sender] >= amount);
260
261     balances[msg.sender] = balances[msg.sender].sub(amount);
262     for (uint i = 0; i < cnt; i++) {
263         balances[_receivers[i]] = balances[_receivers[i]].add(_value);
264         Transfer(msg.sender, _receivers[i], _value);
265     }
266     return true;
267   }
268 }
269
```

Start building a dApp