猿でもわかるEthereum入門

@syrohei

Ethereumはイーサリアム（英: Ethereum）は、イーサリアム・プロジェクト[1]により開発が進められている、分散型アプリケーション (DApps) やスマート・コントラクトを構築するためのプラットフォームの名称、及び関連するオープンソース・ソフトウェア・プロジェクトの総称である。@wikipedia

????? ʕ•ᴥ•ʔ ??????

What’s your Money ?

お金とはなんですか？

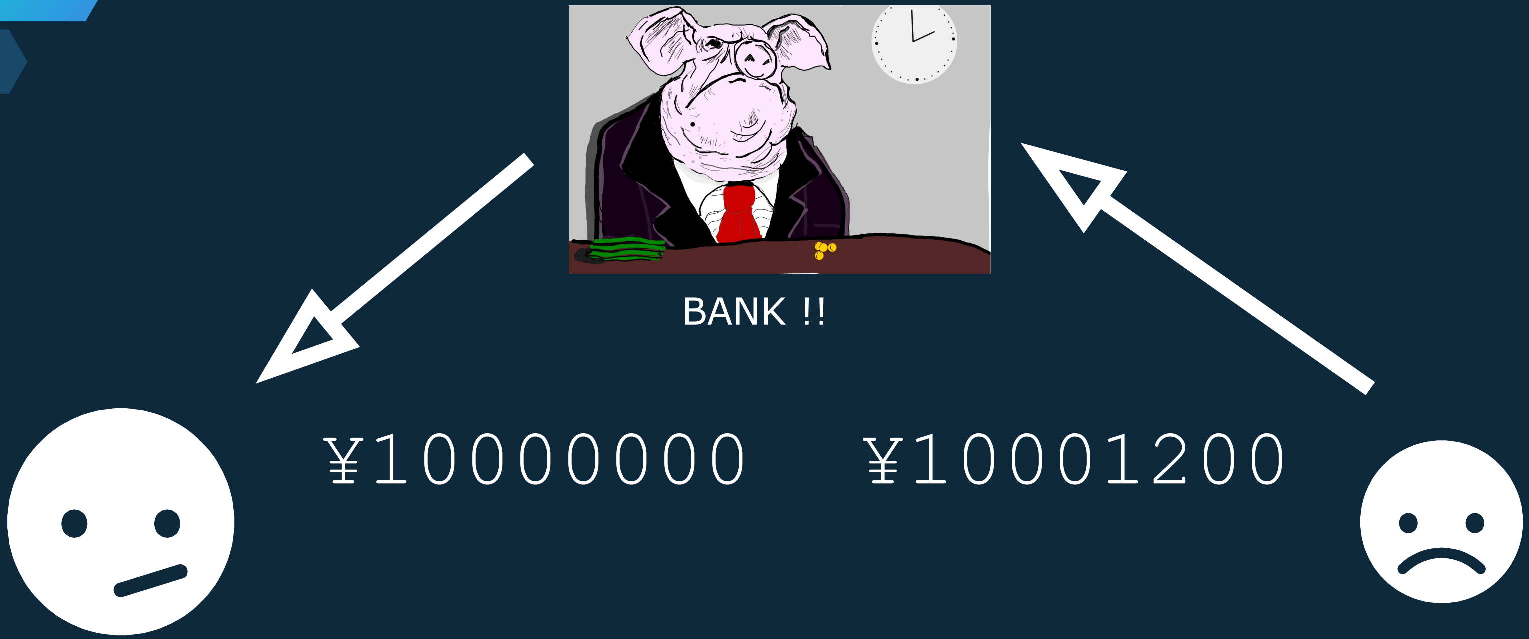
送りたい

¥10000000

信用がない

BANK !!

信用が一応ある



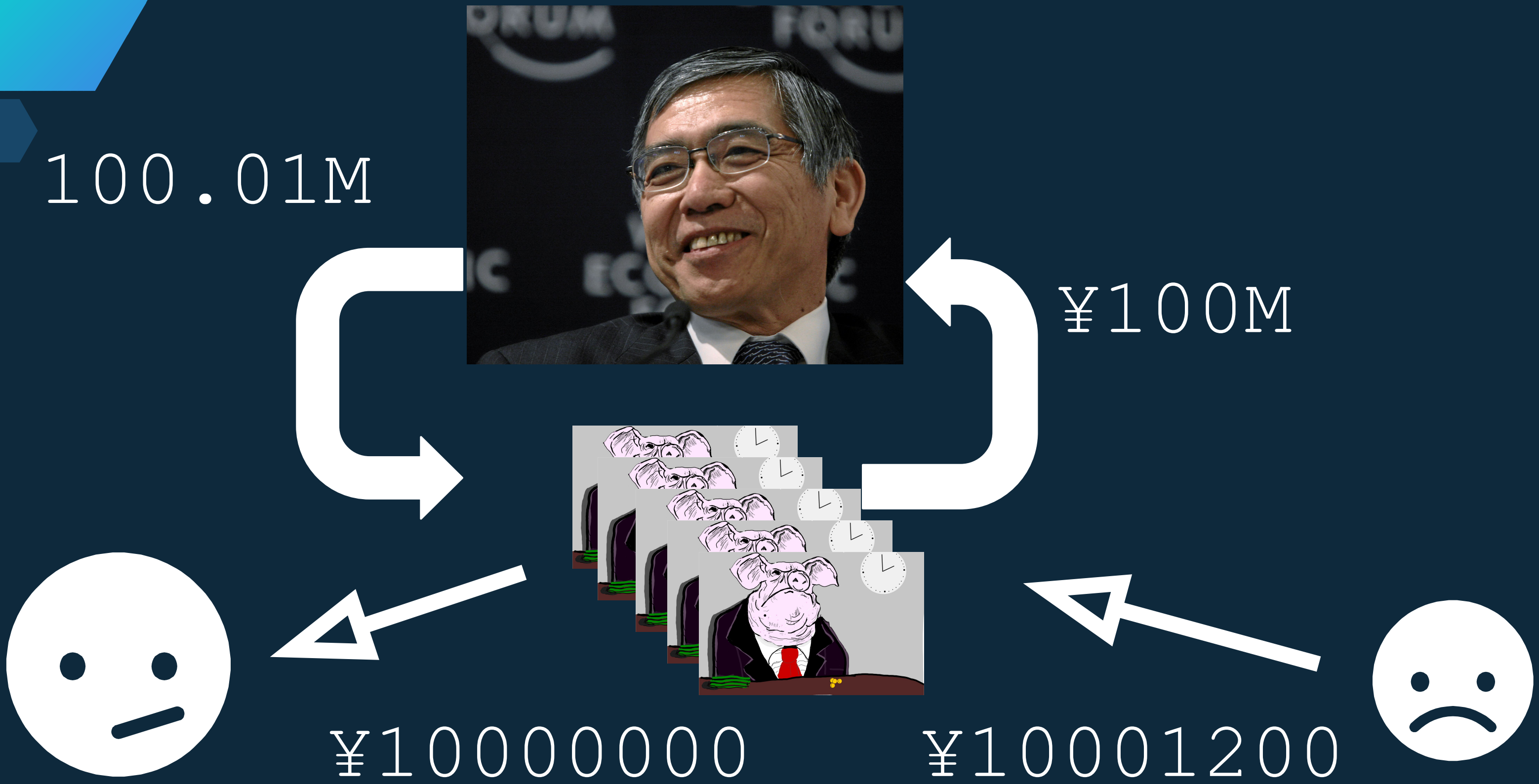
BANK !!

¥10000000 ¥10001200

Hi,
IJIGEN NO-
IJIGEN NO-

信用が
ある KURODA !!

画像提供: https://www.flickr.com/photos/15237218@N00/4318431982
CC BY-NC-SA 2.0
Taken by World Economic Forum.



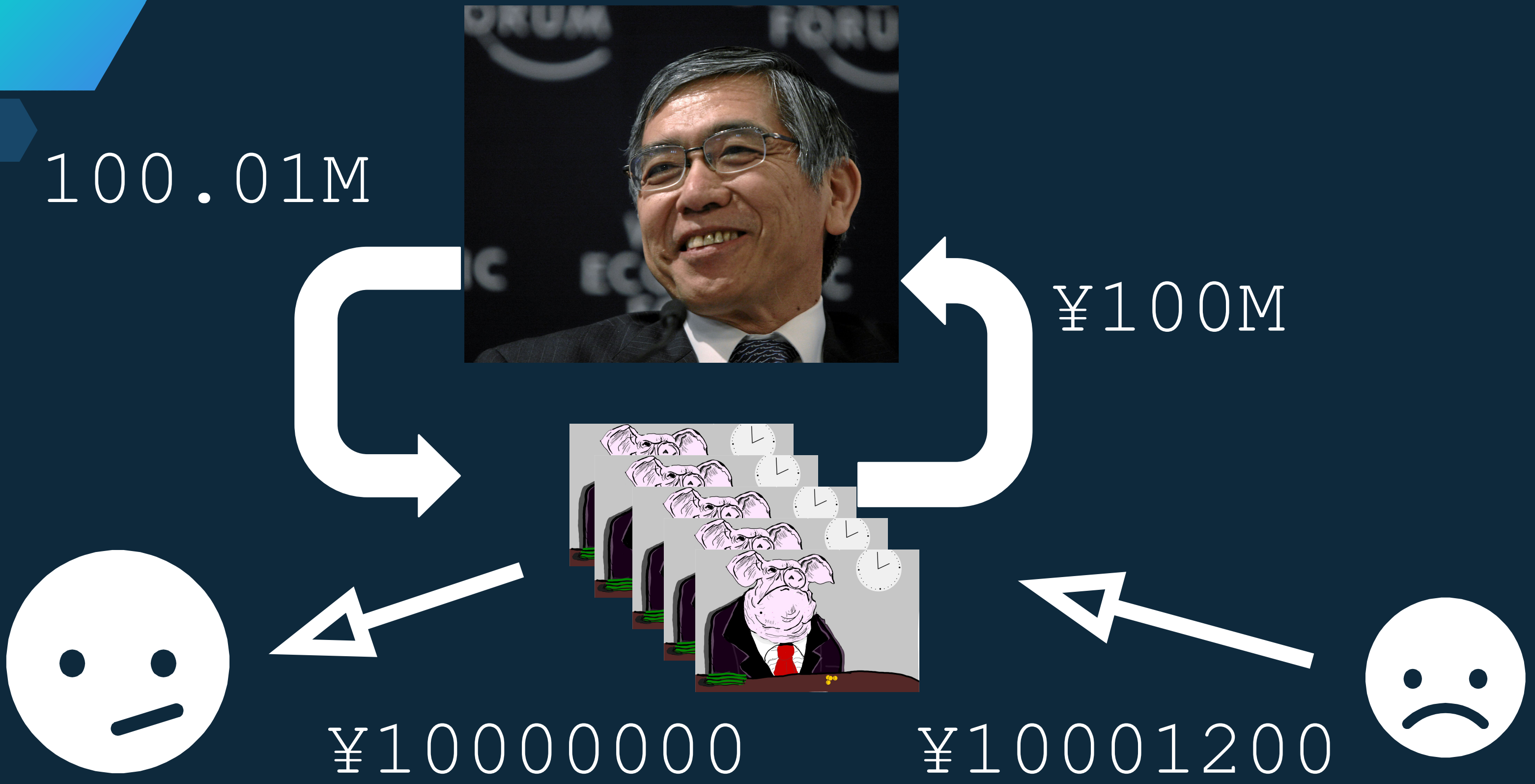
100.01M

¥100M

¥10000000

¥10001200

How to design Money ?

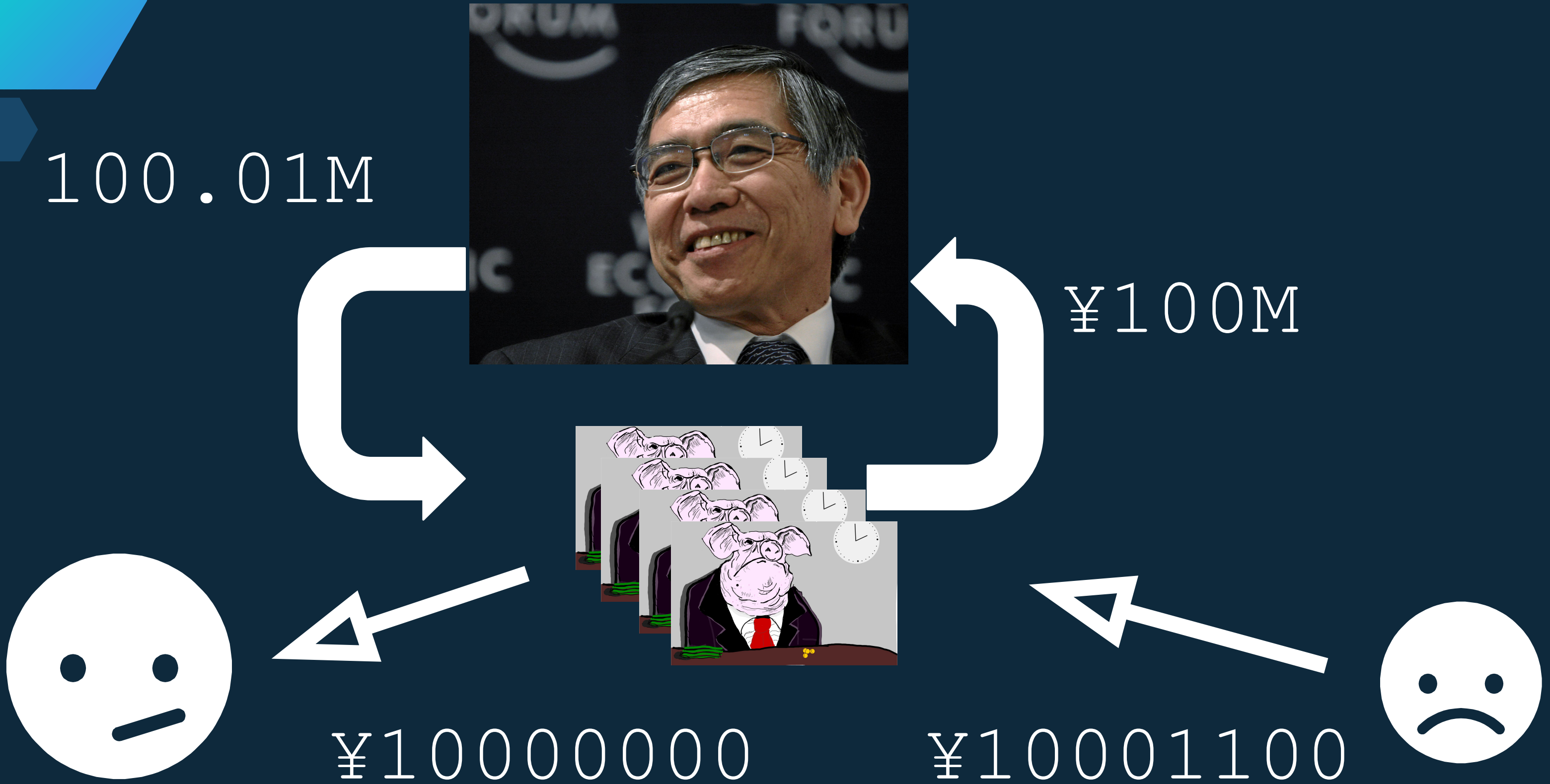


100.01M

¥100M

¥10000000

¥10001200

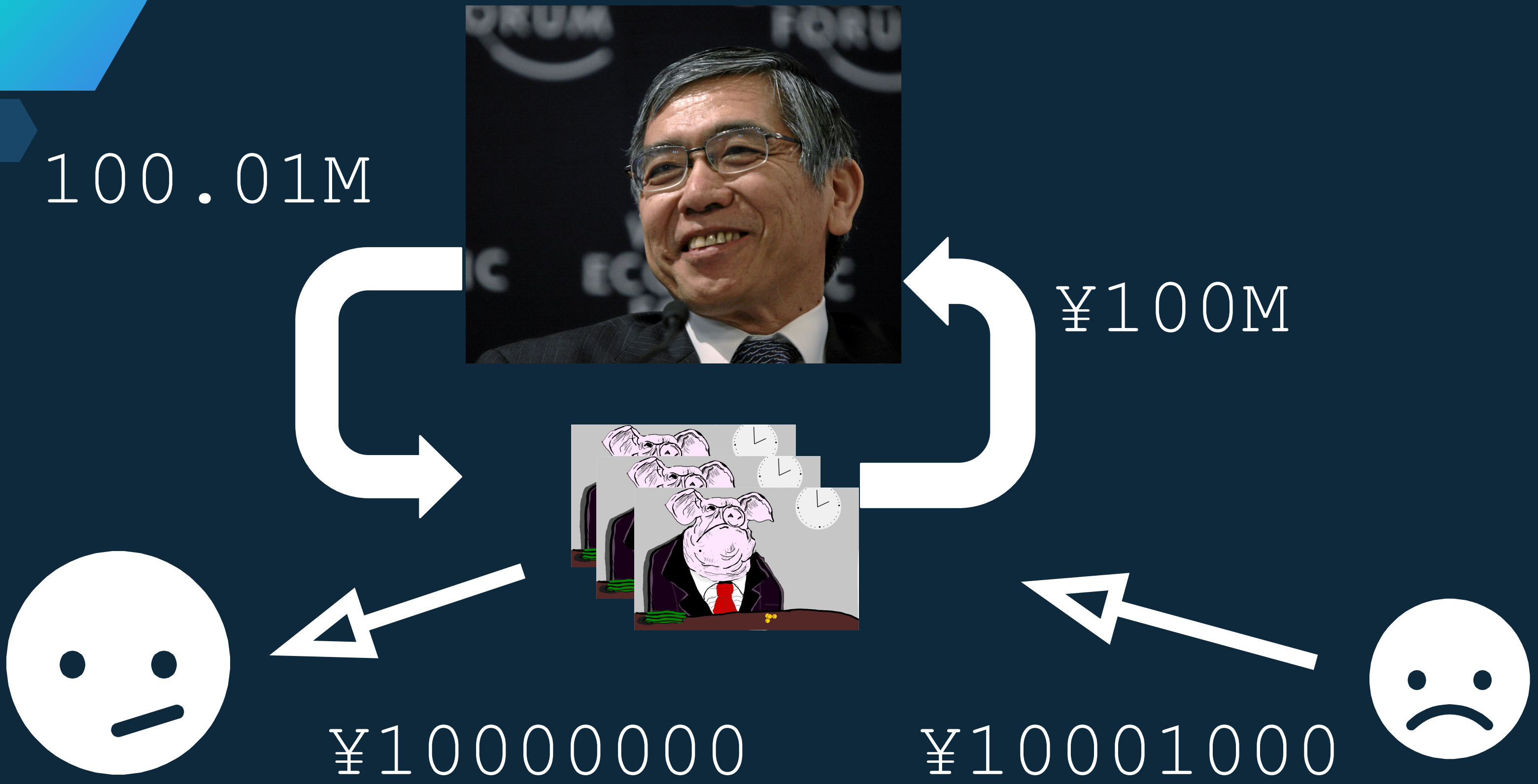


100.01M

¥100M

¥10000000

¥10001100

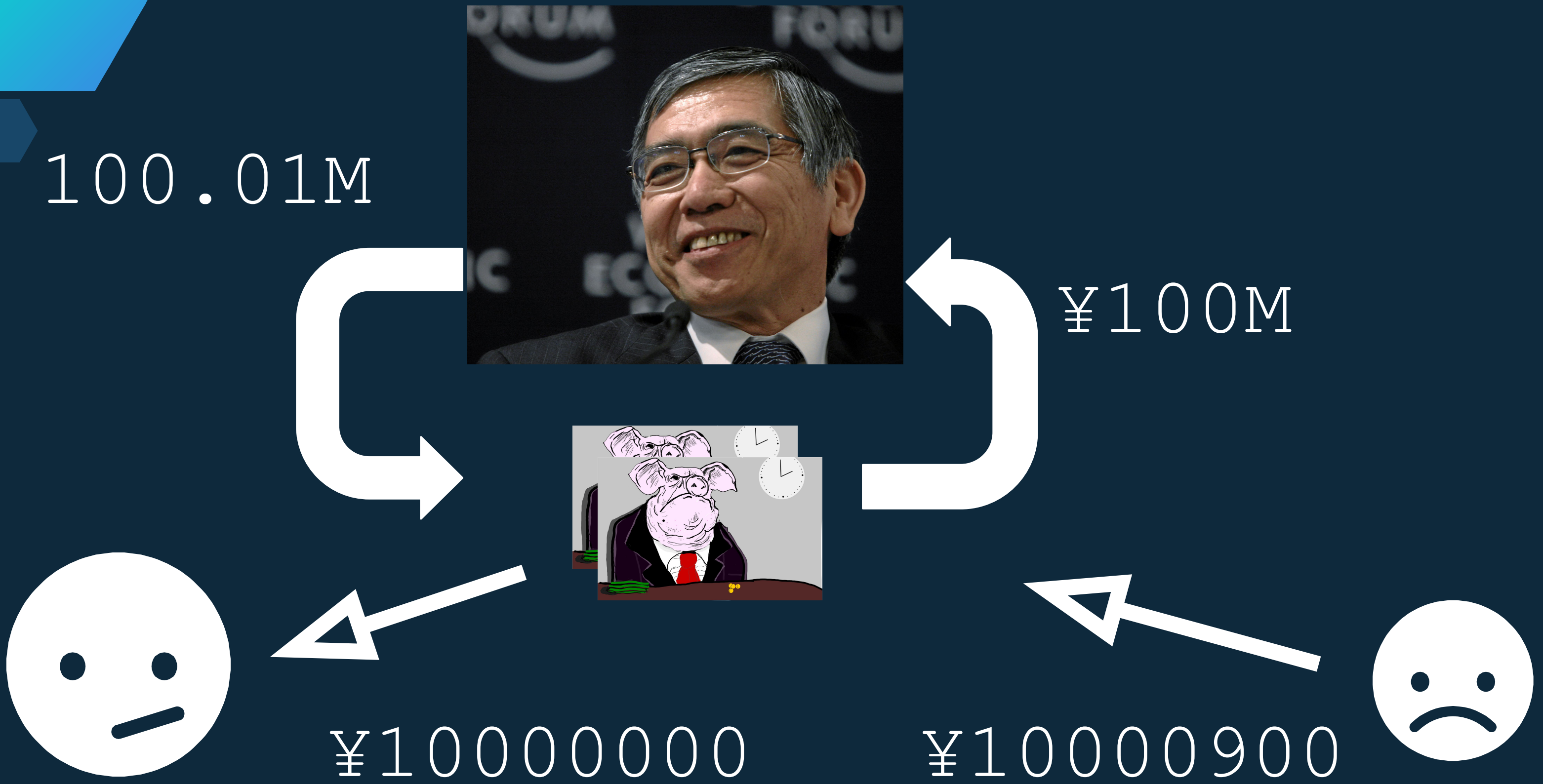


100.01M

¥100M

¥10000000

¥10001000

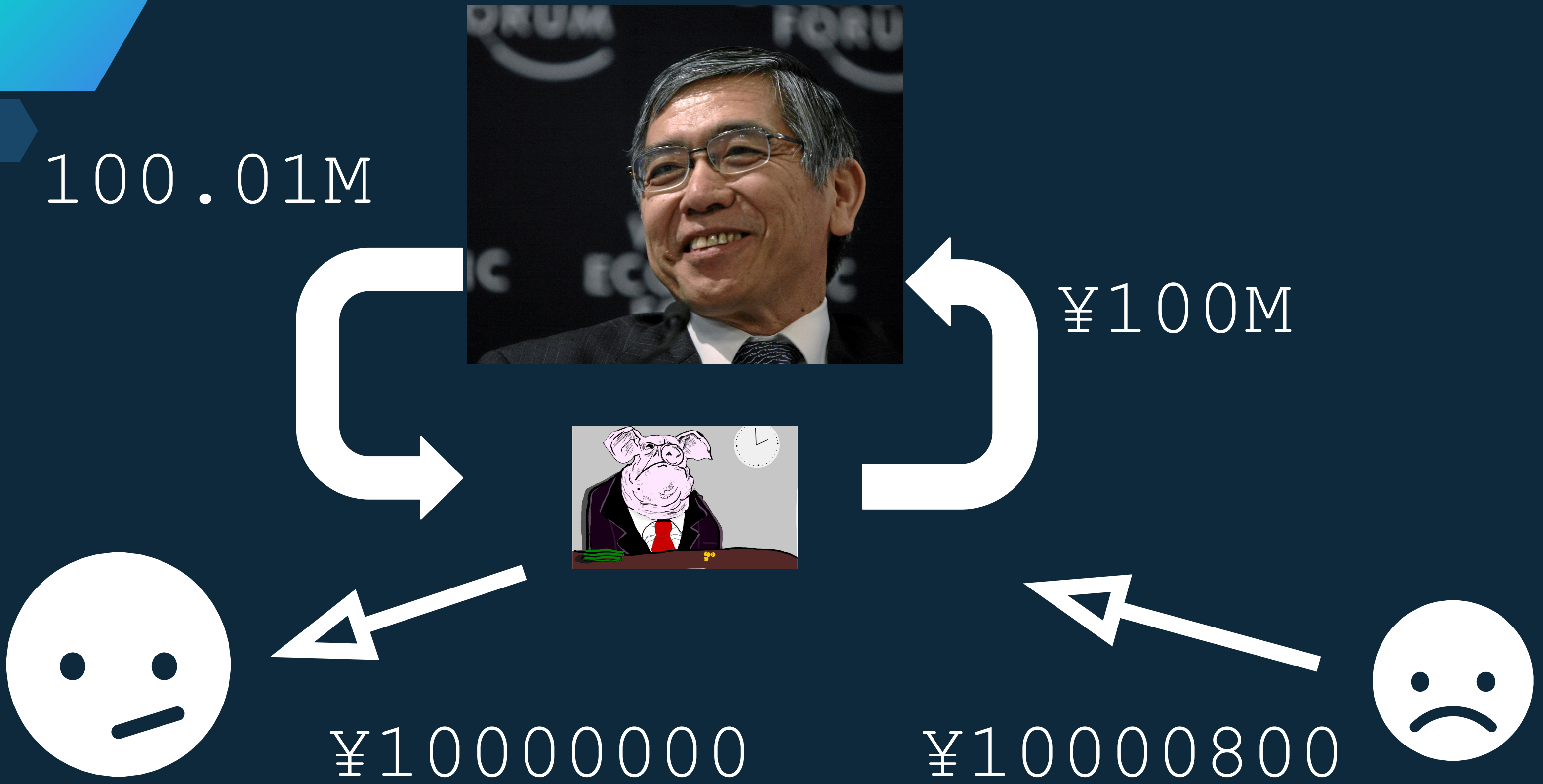


100.01M

¥100M

¥10000000

¥10000900

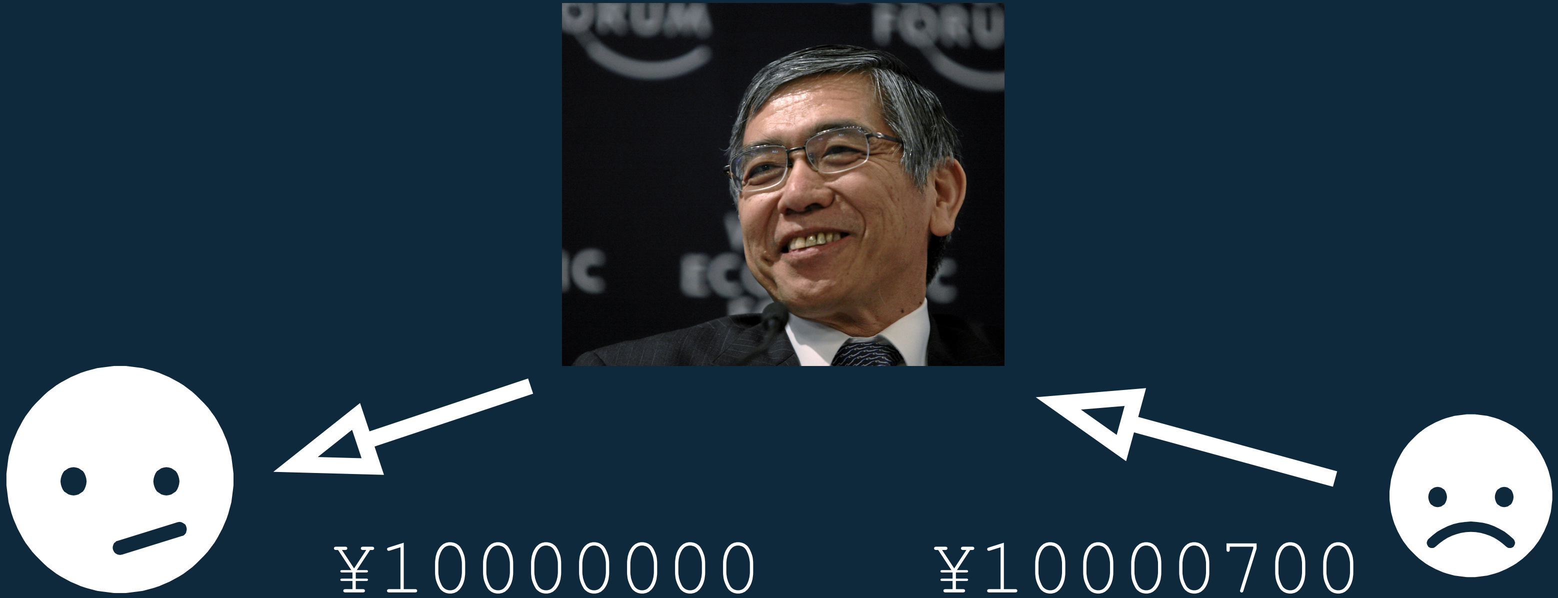


100.01M

¥100M

¥10000000

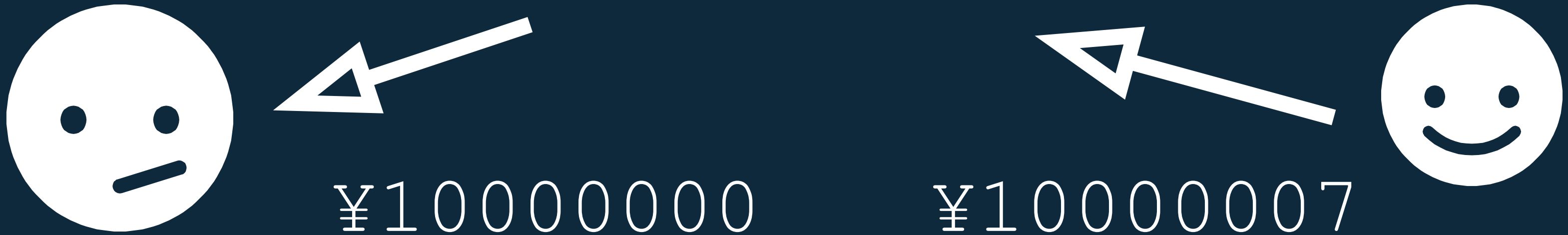
¥10000800



¥10000000

¥10000700

bitcoin



¥10000000

¥10000007

bitcoin

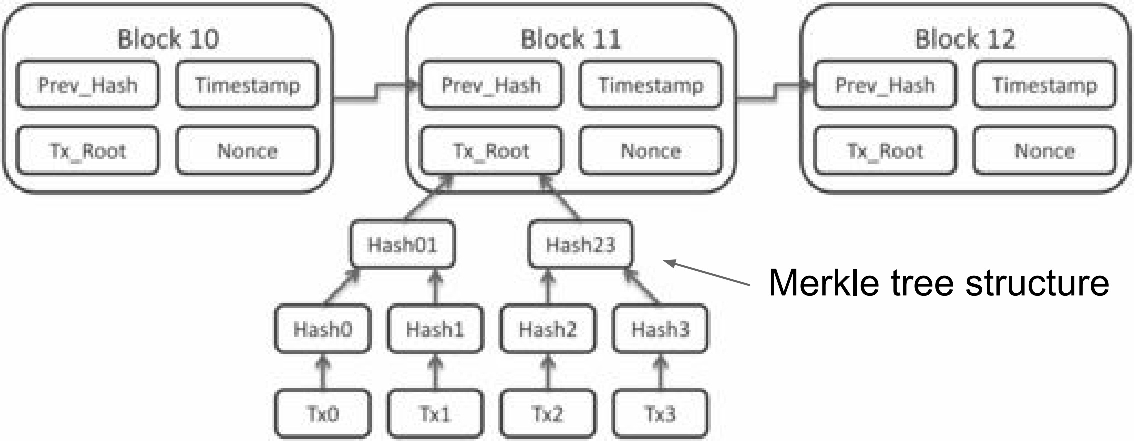
litecoin

namecoin

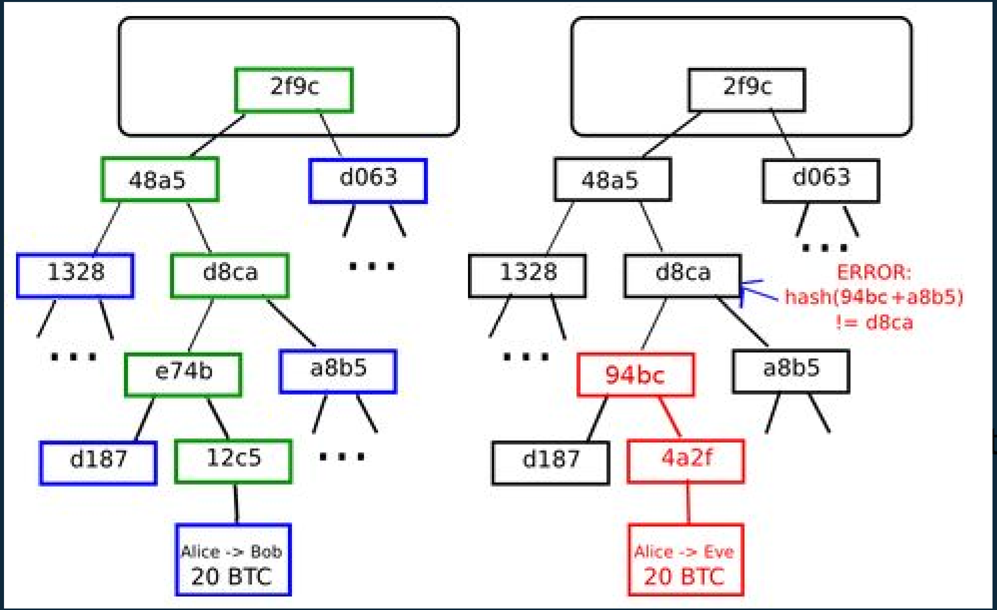
Alt coin series are more than 300 !!

What is BlockChain

BlockChain : Data Structure that Linked previous block_Id (hash) and includes transactions.



What is merkle tree



tructure

Bitcoin’s Problem

Bitcoin’s Problem

Scalability

Approx 6~7 tx/sec

Visa payments specs over 10000 tx/sec affome

Centralized Consensus

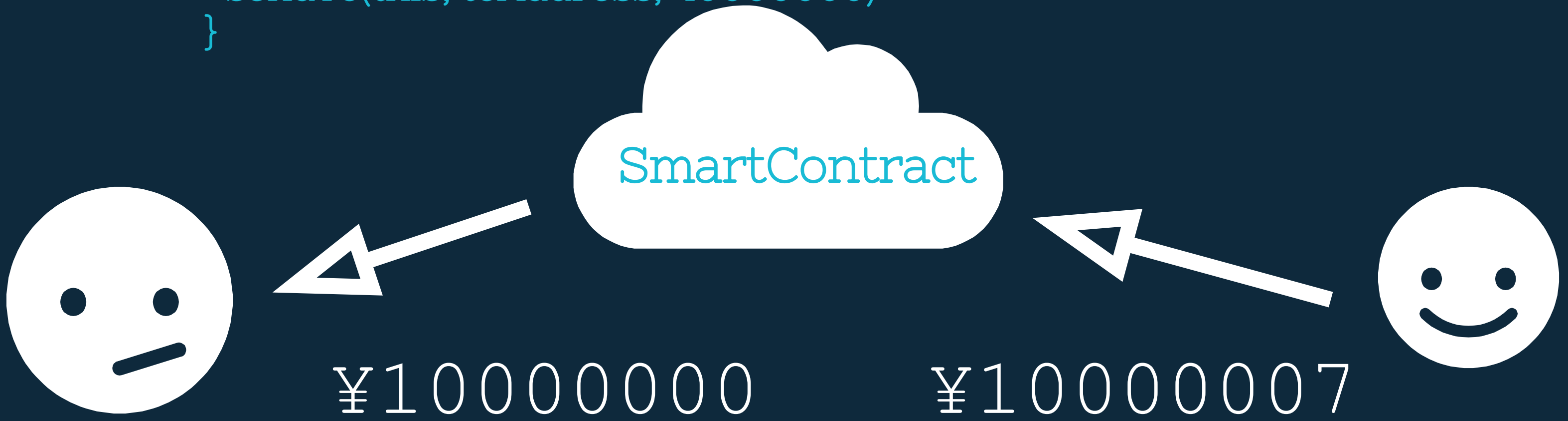
Miner has been centralized and low efficiency.

Bitcoin mining chips made by China and Taiwan more than 90%

Solution : **We know design approach Smart Contract application and plugged High scale Economics Consensus Algorithm**

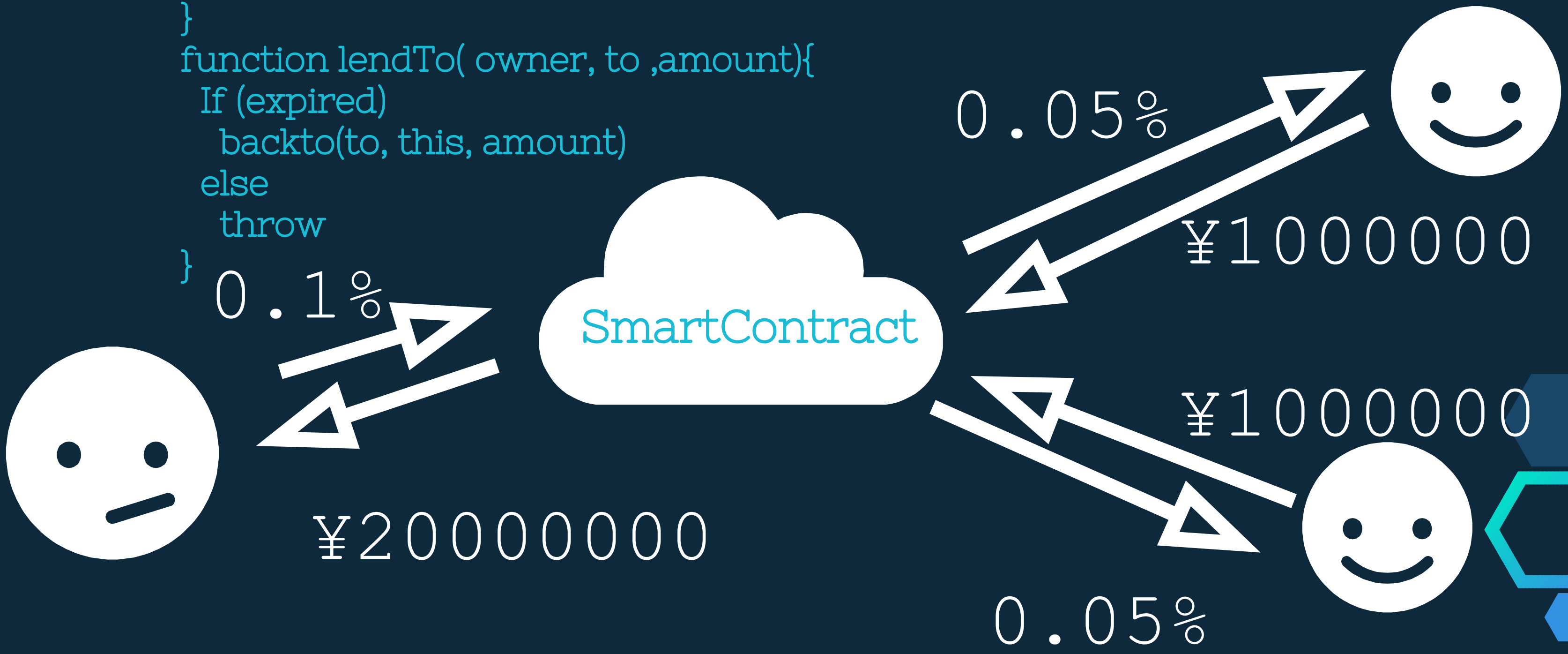
SmartContract (Extends Bitcoin Script) like a Application

```
If ( owner_moneys >= 10000007) {
  Balance -= msg.sender
  sendTo(this, toAddress, 10000000)
}
```



Bank as a Service

```
Import backto(_to, _this, _amount)
If ( owner_moneys >= 20000000) {
  Balance -= msg.sender
  lendTo(this, toAddress,  20000000)
}
function lendTo( owner, to ,amount){
  If (expired)
    backto(to, this, amount)
  else
    throw
}
```



0.1%

¥20000000

SmartContract

0.05%

¥1000000

¥1000000

0.05%

ethereum
HOMESTEAD RELEASE
BLOCKCHAIN APP PLATFORM

What is Ethereum ?

Open Source Software Development

Smart Contract Application Platform

The World Consensus Application Build tools

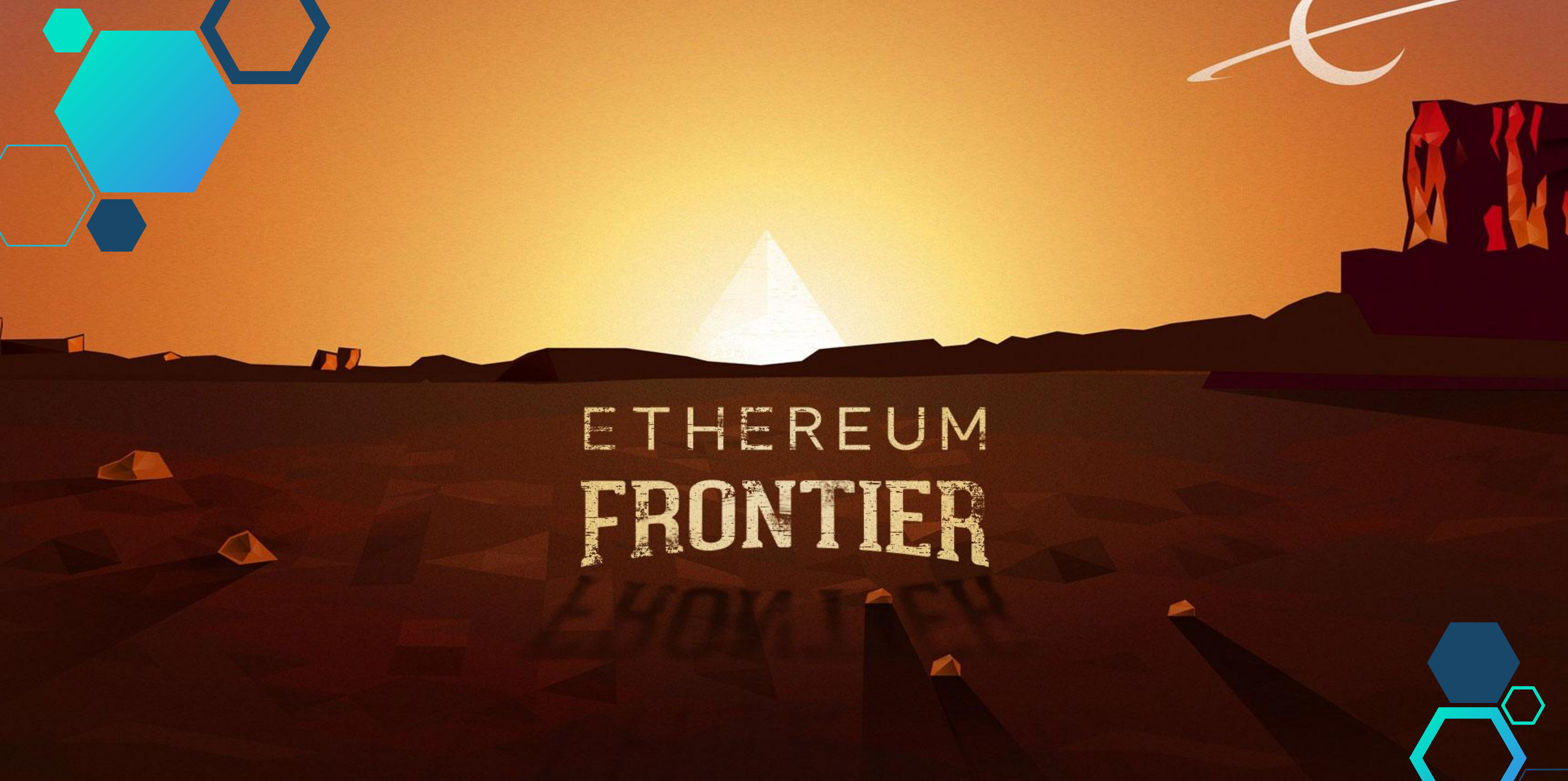
P2P Software consensus Architecture (BlockChain)

The Next WEB (Web3) Software Design approach

Proof of Work Economics Consensus Design and Gas Price model

Feature FROZEN

v0.9.36 Released 7 July 2015



v1.0.0 Released 29 July 2015

HOMESTEAD

ethereum.org

v1.3.4 Released 2 Feb 2016

World Etherum Nodes over 7577 !!

Anyone can Build up Smart Contract Application.

This is a Server-Less application architecture. If you’d like to start service and create your application you can build quickly without Server (it like a PaaS)

Decentralized Application Software Stack

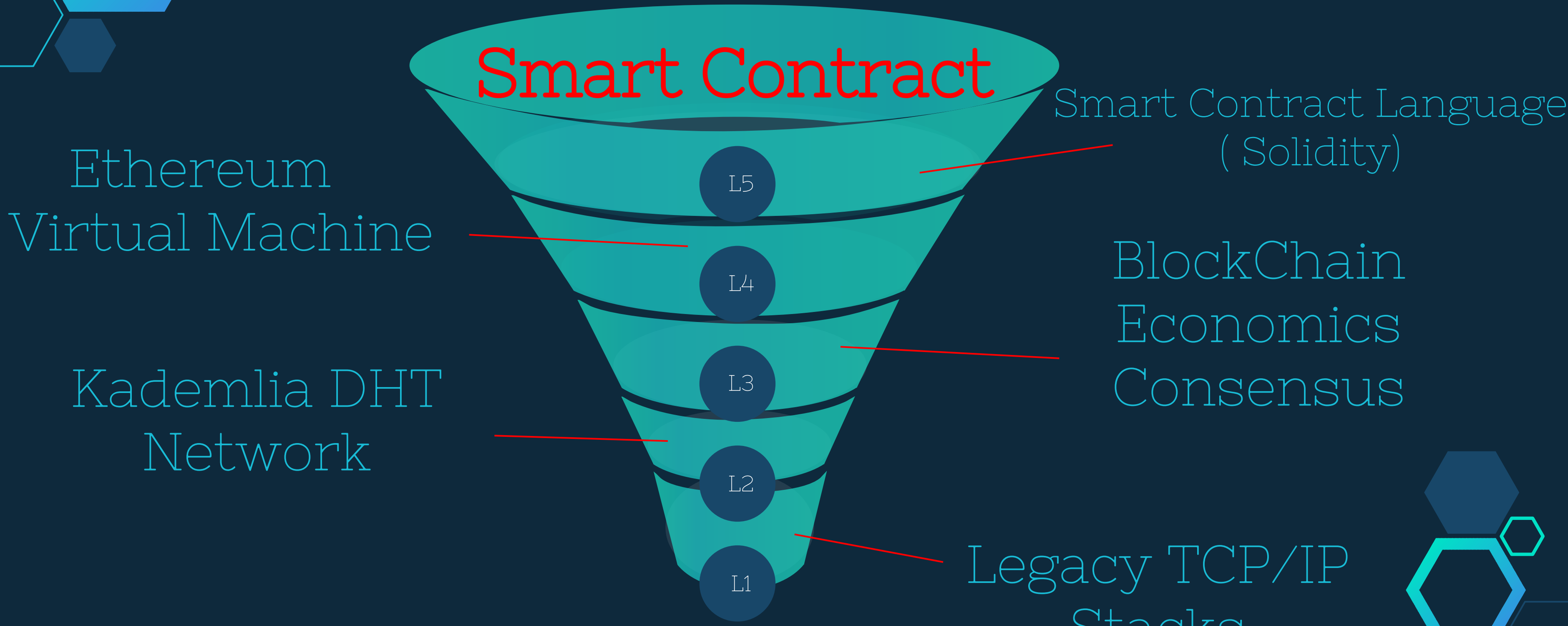
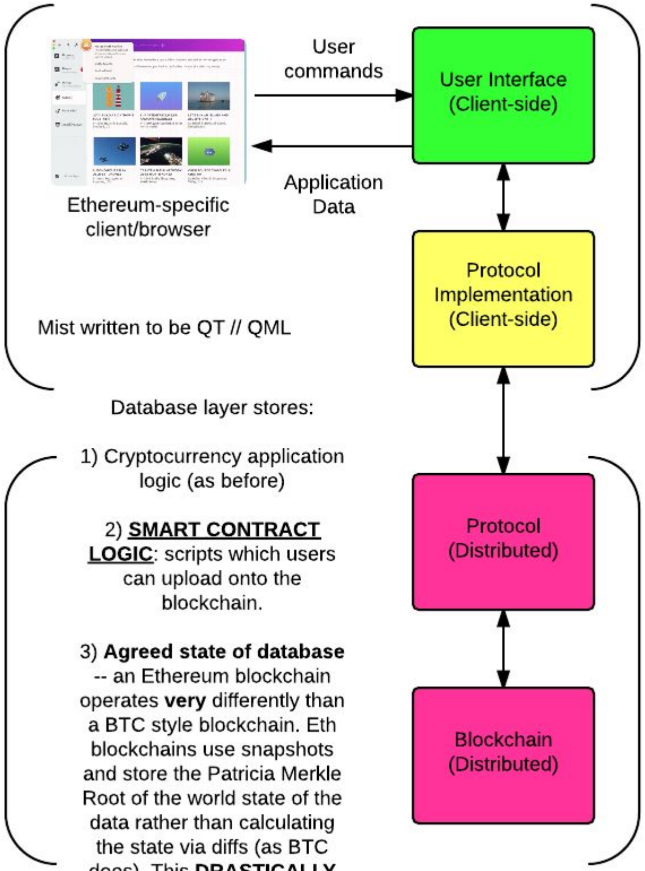


Diagram featured by Yusaku Senga

ETHEREUM DATABASE ARCHITECTURE: As before, but drastically improved world state calculation.

CLIENT SIDE

DISTRIBUTED



User commands

Application Data

User Interface (Client-side)

Protocol Implementation (Client-side)

Ethereum-specific client/browser

Mist written to be QT // QML

Protocol (Distributed)

Blockchain (Distributed)

Database layer stores:

1) Cryptocurrency application logic (as before)

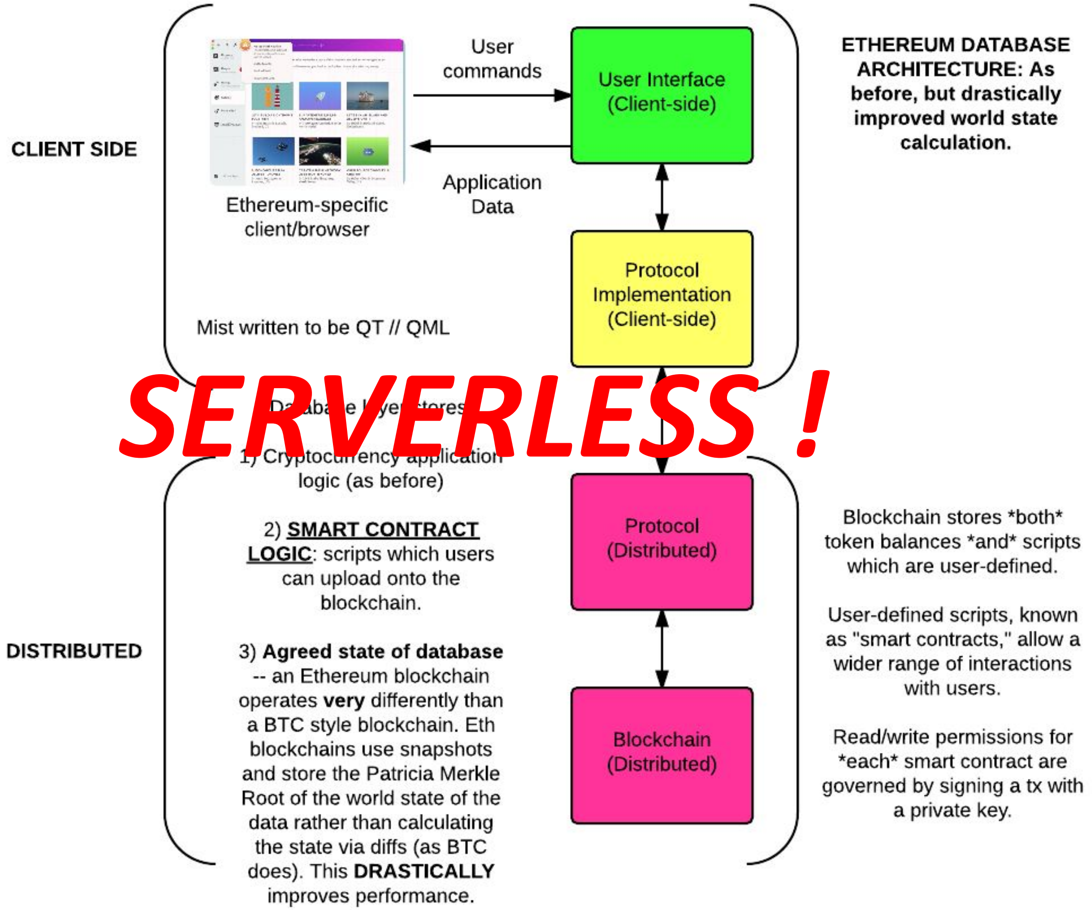
2) **<u>SMART CONTRACT LOGIC</u>**: scripts which users can upload onto the blockchain.

3) **Agreed state of database** -- an Ethereum blockchain operates **very** differently than a BTC style blockchain. Eth blockchains use snapshots and store the Patricia Merkle Root of the world state of the data rather than calculating the state via diffs (as BTC does). This **DRASTICALLY** improves performance.

Blockchain stores *both* token balances *and* scripts which are user-defined.

User-defined scripts, known as "smart contracts," allow a wider range of interactions with users.

Read/write permissions for *each* smart contract are governed by signing a tx with a private key.



ETHEREUM DATABASE ARCHITECTURE: As before, but drastically improved world state calculation.

CLIENT SIDE
Ethereum-specific client/browser
User commands
Application Data
User Interface (Client-side)
Protocol Implementation (Client-side)
Mist written to be QT // QML
DISTRIBUTED
Protocol (Distributed)
Blockchain (Distributed)

Database layer stores:

1) Cryptocurrency application logic (as before)

2) **SMART CONTRACT LOGIC**: scripts which users can upload onto the blockchain.

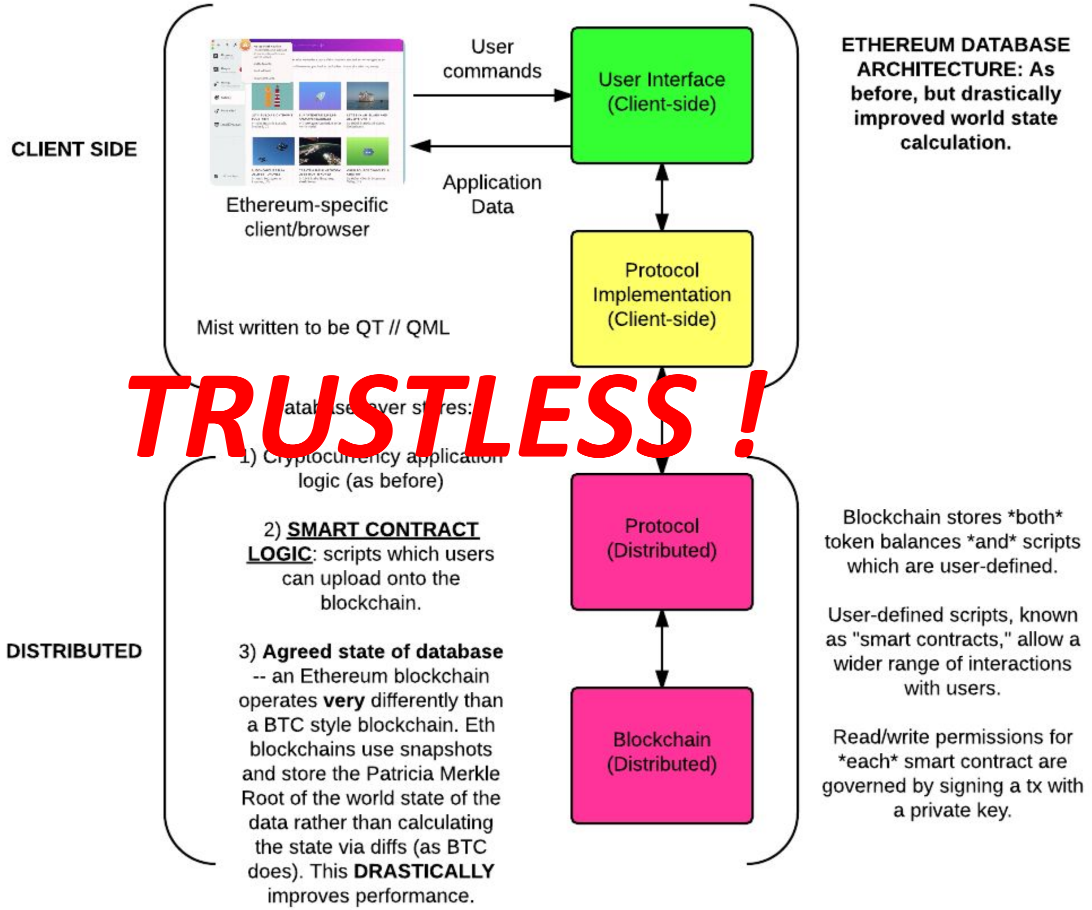
3) **Agreed state of database** -- an Ethereum blockchain operates **very** differently than a BTC style blockchain. Eth blockchains use snapshots and store the Patricia Merkle Root of the world state of the data rather than calculating the state via diffs (as BTC does). This **DRASTICALLY** improves performance.

Blockchain stores *both* token balances *and* scripts which are user-defined.

User-defined scripts, known as "smart contracts," allow a wider range of interactions with users.

Read/write permissions for *each* smart contract are governed by signing a tx with a private key.

SERVERLESS !



TRUSTLESS !

ETHEREUM DATABASE ARCHITECTURE: As before, but drastically improved world state calculation.

CLIENT SIDE
User commands
Application Data
User Interface (Client-side)
Protocol Implementation (Client-side)
Ethereum-specific client/browser
DISTRIBUTED
Protocol (Distributed)
Blockchain (Distributed)

Mist written to be QT // QML

Database server stores:

1) Cryptocurrency application logic (as before)

2) **SMART CONTRACT LOGIC**: scripts which users can upload onto the blockchain.

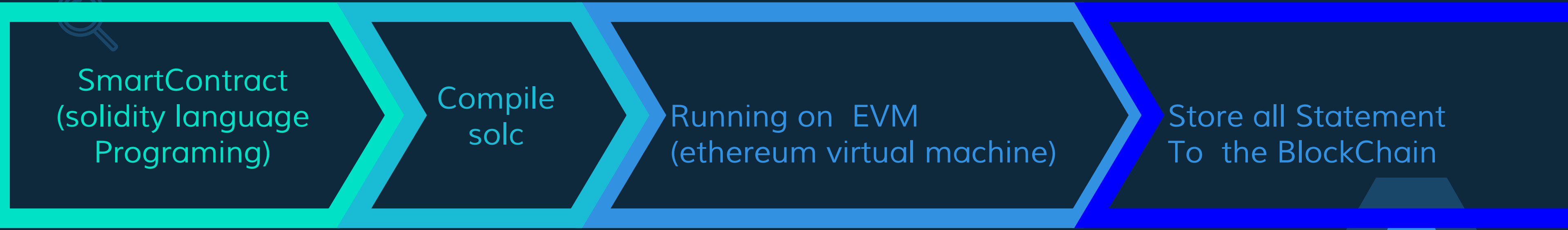
3) **Agreed state of database** -- an Ethereum blockchain operates **very** differently than a BTC style blockchain. Eth blockchains use snapshots and store the Patricia Merkle Root of the world state of the data rather than calculating the state via diffs (as BTC does). This **DRASTICALLY** improves performance.

Blockchain stores *both* token balances *and* scripts which are user-defined.

User-defined scripts, known as "smart contracts," allow a wider range of interactions with users.

Read/write permissions for *each* smart contract are governed by signing a tx with a private key.

How it works



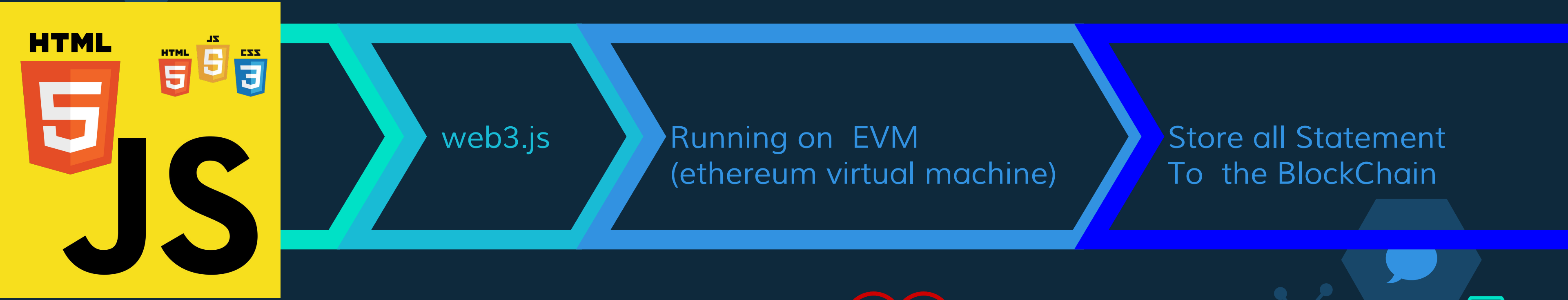
SmartContract (solidity language Programing)

Compile solc

Running on EVM (ethereum virtual machine)

Store all Statement To the BlockChain

How it works



web3.js

Running on EVM
(ethereum virtual machine)

Store all Statement
To the BlockChain

Integrate to IPFS

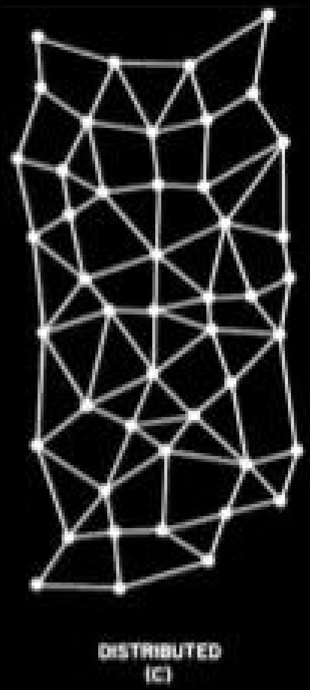
IPFS

InterPlanetary File System

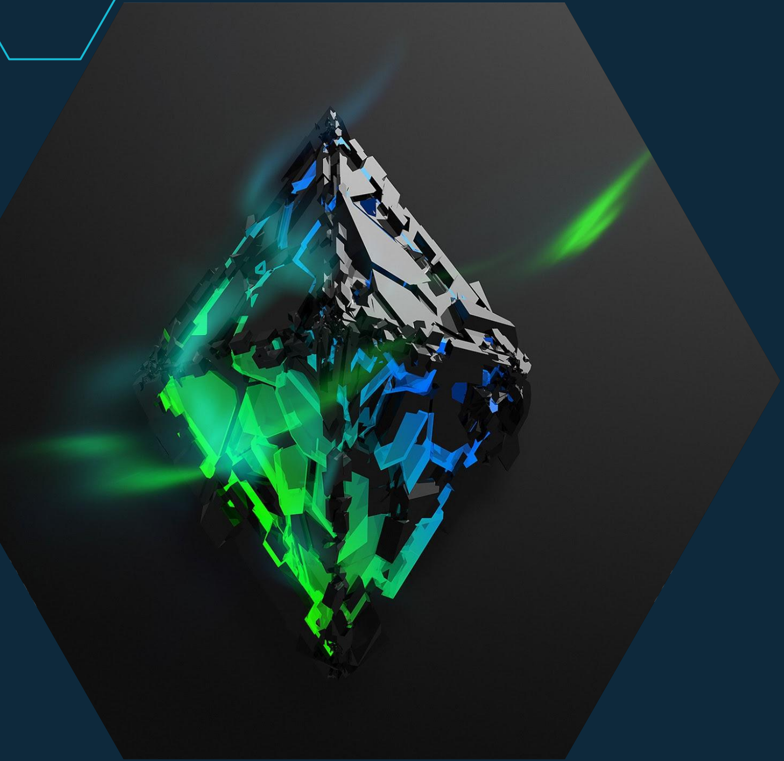
Sivachandran

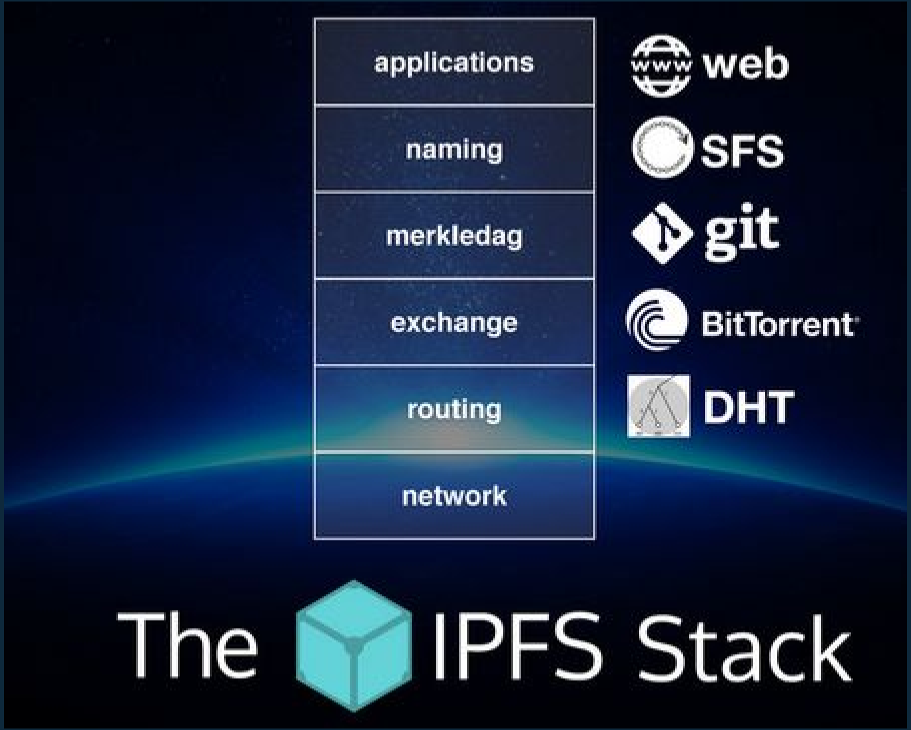
IPFS: Topology

- Fully distributed network
- Node
  - No server/client
  - Acts both server and client
  - Connected with every other node
- Web original design

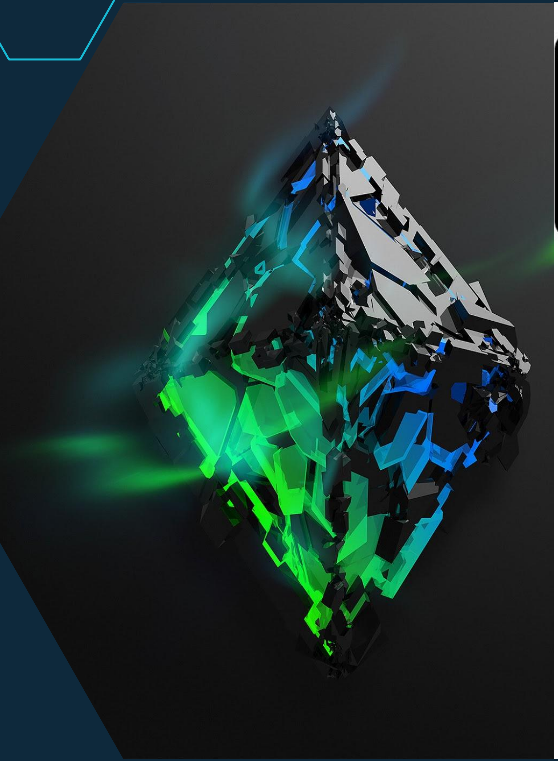


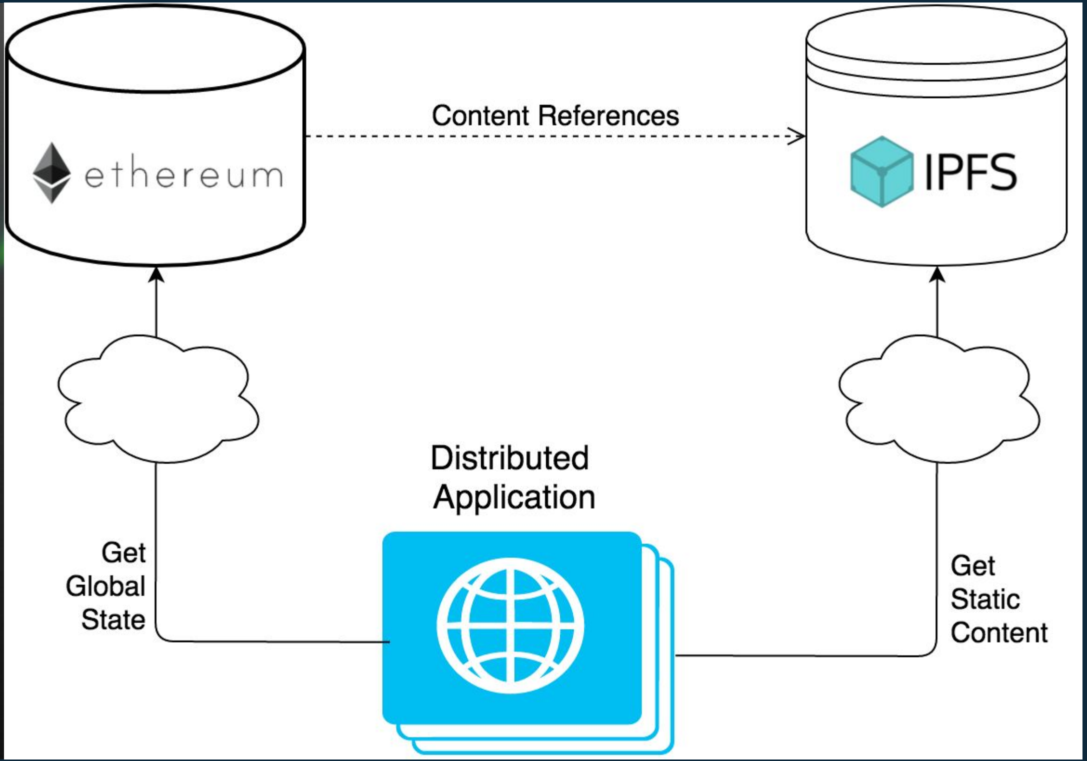
Tools : IPFS
P2P file system





Integrate Etherum and IPFS





Smart Contract Application Deploy process.

Private net

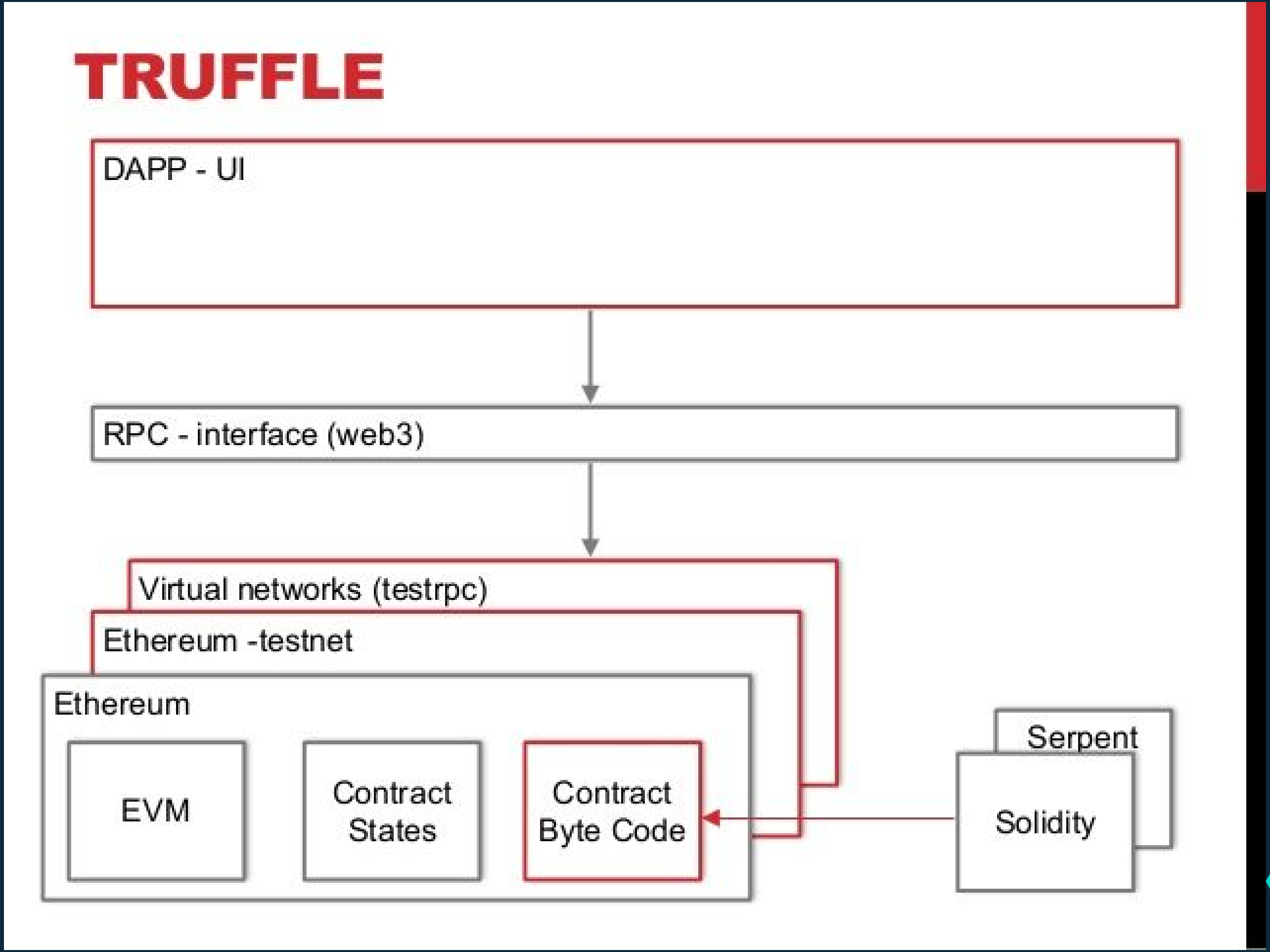
Test net

Live net

Tools : TRUFFLE

A Developing tools for Smart Contract and Dapps. Supported Solidity Smart Contract Language. And it works additional component react.js, meteor.js and it can be useful first steps of Dapps tutorials.

```
$ truffle init
$ truffle create:MetaCoin
$ truffle migrate
$ truffle test
$ truffle serve
```



TRUFFLE

DAPP - UI

RPC - interface (web3)

Virtual networks (testrpc)

Ethereum -testnet

Ethereum

EVM

Contract States

Contract Byte Code

Serpent

Solidity

DEMo

DEMo

```
$ git clone  https://github.com/syrohei/truffle-tutorial
```

On Going project on Ethereum:
Singular DTV

They finished Crawd Sale 2nd oct.

Sold out in 15min !
about $7.5M

There are source code is

https://github.com/ConsenSys/singulardtv-contracts

Thanks!

Any questions?

You can find me at:

@syrohei

syrohei@gmail.com