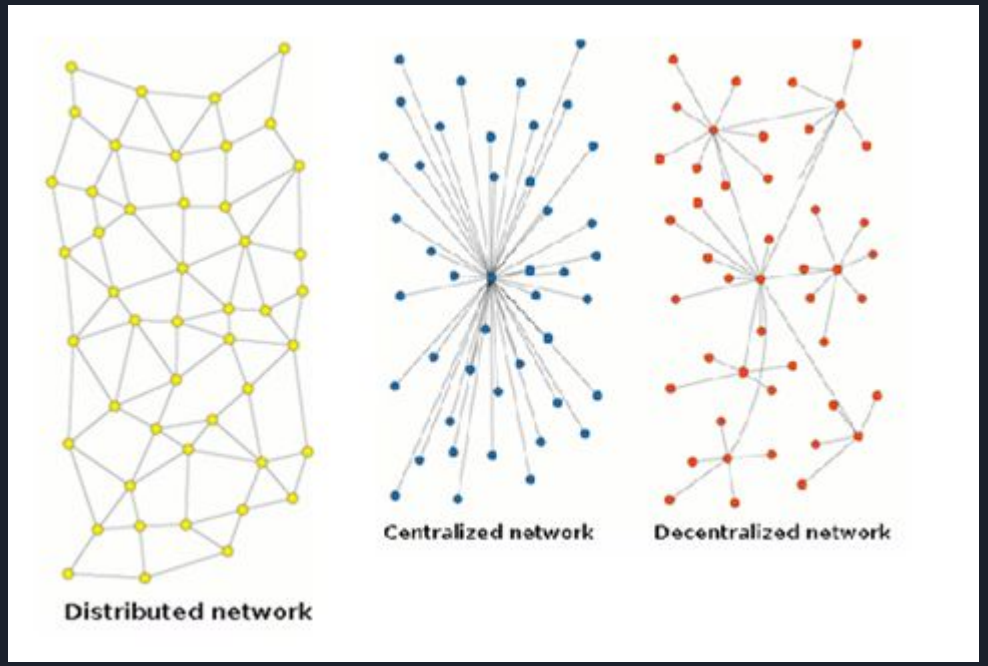
DLT - relayr

Tomasz Waszczyk
14.12.2018

Systemy zdecentralizowane tzw. Web3.0

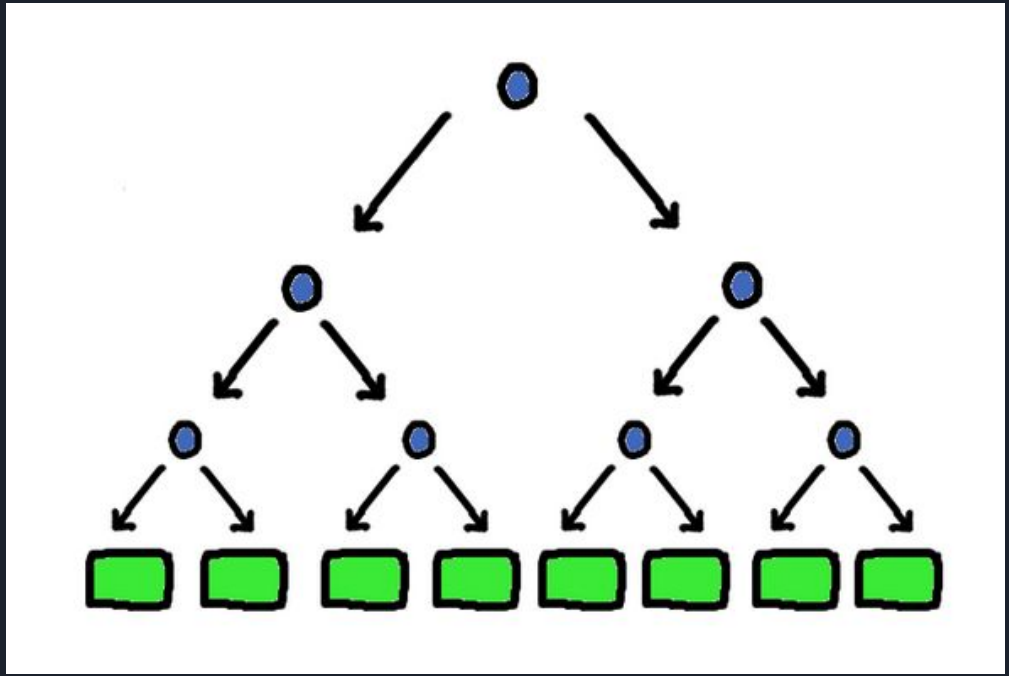


Czym jest DLT?

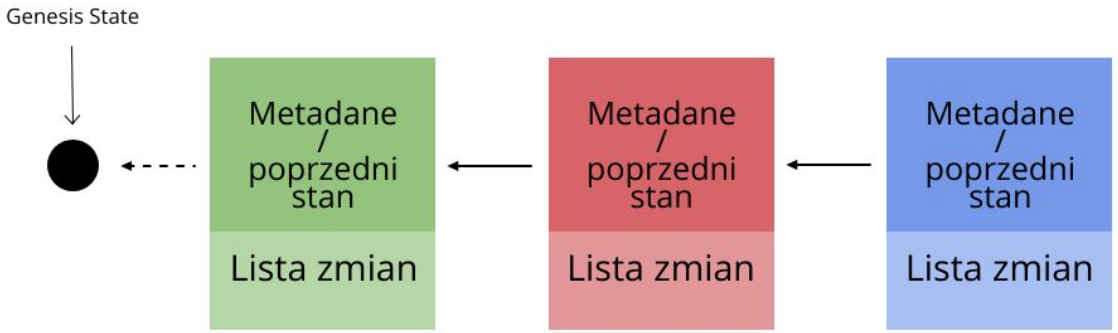
- **Rozproszona struktura danych** - organizująca "transakcję"/zdarzenia - rejestr
- **Algorytm Konsensusu -** aby zdecydować kto może tę strukturę modyfikować
- **Kryptografia -** żeby to było bezpieczne/odporne na modyfikacje (tamperproof)
- **Zachęty (Incentives**) - żeby mogło działać bez nadzoru i zaufania

Teoria - problem bizantyjskich generałów

Merkle Tree

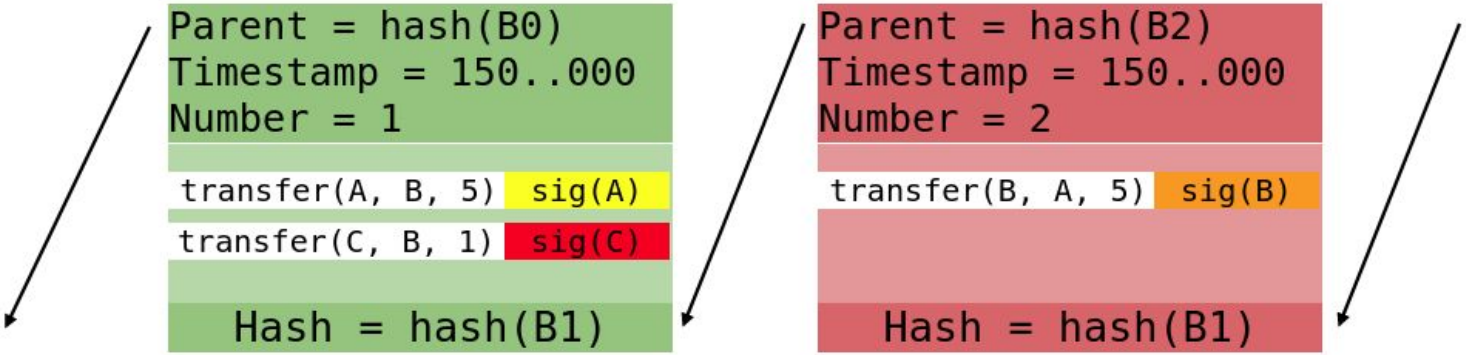


Blockchain



Niemutowalna struktura danych zawierająca wszystkie zmiany który zostały zaplikowane do danego punktu w czasie

Blockchain



Hashes - prevent tampering (e.g. KECCAK256)

Signatures - authorize the actions (e.g. ECDSA)

Element losowości czyli po co są potwierdzenia UTXO

- Rozmycie dodawania nowego bloku
- Algorytm odpowiada kiedy nowa dana NIE odpowiada na warunki zapytania
- Może się mylić

Adres Bitcoin Adresy są identyfikatorami, których używasz, aby wysłać bitcoiny do kogoś innego.

Podsumowanie	
Adres	18bXSCSXiTD3DB3XEz851VpB4ZK49rkprT
Hash 160	53506db70d87c4ab7dd75824508e80f4080c58fb

Transakcje	
Liczba transakcji	34
Razem otrzymano	393.53905561 BTC
Ostateczny stan	113.8121983 BTC

Żądanie płatności | Przycisk dotacji

Transakcje (Najpierw starsze)

Filtr

3f8c6ba12fae02c0fd6a474fd8852fc328e93332a2bebea724adf03de9ef503e — **2018-12-13 23:58:11**

Brak danych wejściowych (nowo wygenerowane monety)	18bXSCSXiTD3DB3XEz851VpB4ZK49rkprT	12.65069157 BTC
	Nie można zdekodować adresu wyjściowego	0 BTC

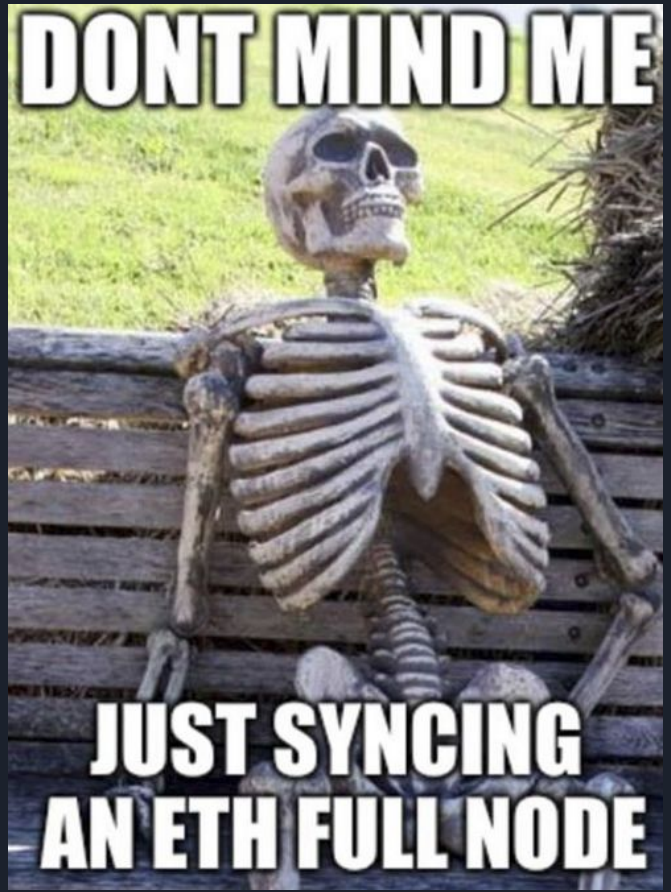
1 Potwierdzenia | 12.65069157 BTC

Czym jest Ethereum?

Ethereum to "The World Computer"

Block Time	14 seconds
Consensus	Proof of Work - ethash*
State	Arbitrary
Transactions	Turing-complete / programmable
Launched	2015
Block Reward	3.75ETH (+uncles) ~ Unlimited coins

Jak połączyć się z siecią Ethereum?



Narzędzia

- Truffle
- Ganache-cli
- Solidity
- Web3
- Parity
- Infura
- Nodejs

- MetaMask
- React/React Native/Angular
- Rust
- IPFS/swarm/Kubernetes

Truffle


SWEET TOOLS FOR SMART CONTRACTS
The Truffle Suite gets developers from idea to dapp as comfortably as possible.
contract owned {
address public owner;
constructor() public {
owner = msg.sender;
}
modifier onlyOwner {
require(msg.sender == owner); _;
}
function transferOwnership(addr
suite-header

Truffle

1. Compile
2. Deploy (migrate)

Debug jest trudny. Poziom rejestrów EVM.

Ganache-CLI

Installation

`ganache-cli` is written in Javascript and distributed as a Node package via `npm`. Make sure you have Node.js (>= v6.11.5) installed.

```
npm install -g ganache-cli
```

`ganache-cli` utilizes `ganache-core` internally, which is distributed with optional native dependencies for increased performance. If these native dependencies fail to install on your system `ganache-cli` will automatically fallback to `ganache-core`'s pre-bundled JavaScript build.

Having problems? Be sure to check out the FAQ and if you're still having issues and you're sure its a problem with `ganache-cli` please open an issue.

Solidity => EWASM

ERC20

ERC223

Solidity

- Gas
- DAO

Remix

```
// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// ----------------------------------------------------------------------------
pragma solidity ^0.4.17;

contract ERC20 {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);
    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}

contract contractB {
  address tracker_0x_address = 0xd26114cd6EE289AccF82350c8d8487fedB8A0C07; // ContractA Address
  mapping ( address => uint256 ) public balances;

  function deposit(uint tokens) public {

    // add the deposited tokens into existing balance
    balances[msg.sender]+= tokens;

    // transfer the tokens from the sender to this contract
    ERC20(tracker_0x_address).transferFrom(msg.sender, address(this), tokens);
  }

  function returnTokens() public {
    balances[msg.sender] = 0;
    ERC20(tracker_0x_address).transfer(msg.sender, balances[msg.sender]);
  }

}
```

Web3.js

Installation

Node.js

```
npm install web3
```

Yarn

```
yarn add web3
```

Web3.js

Usage

Use the `web3` object directly from the global namespace:

```
console.log(web3); // {eth: .., shh: ...} // It's here!
```

Set a provider (`HttpProvider`):

```
if (typeof web3 !== 'undefined') {
  web3 = new Web3(web3.currentProvider);
} else {
  // Set the provider you want from Web3.providers
  web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
}
```

Parity - light/full client

parity

Geth

Geth

Felix Lange edited this page on Dec 21, 2017 · 17 revisions

`geth` is the the command line interface for running a full ethereum node implemented in Go. It is the main deliverable of the Frontier Release

Capabilities

By installing and running `geth`, you can take part in the ethereum frontier live network and

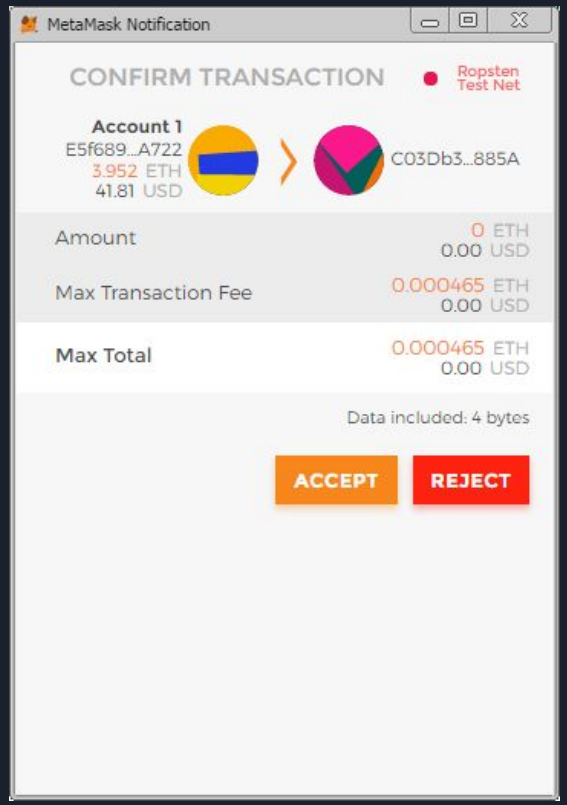
- mine real ether
- transfer funds between addresses
- create contracts and send transactions
- explore block history
- and much much more

Infura

YOUR ACCESS TO THE ETHEREUM NETWORK

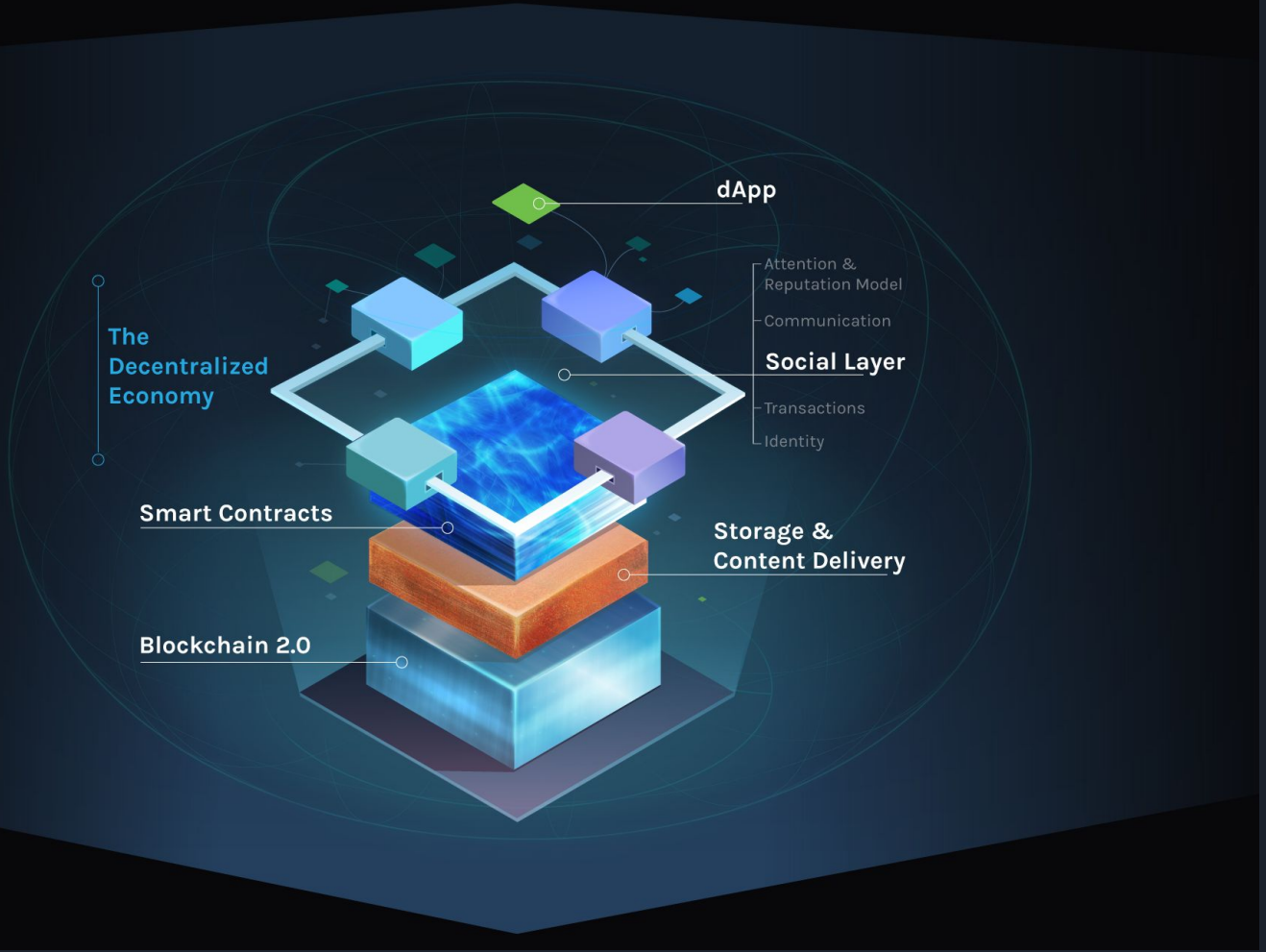
Our easy to use API and developer tools provide secure, reliable, and scalable access to Ethereum and IPFS. We provide the infrastructure for your decentralized applications so you can focus on the features.

MetaMask

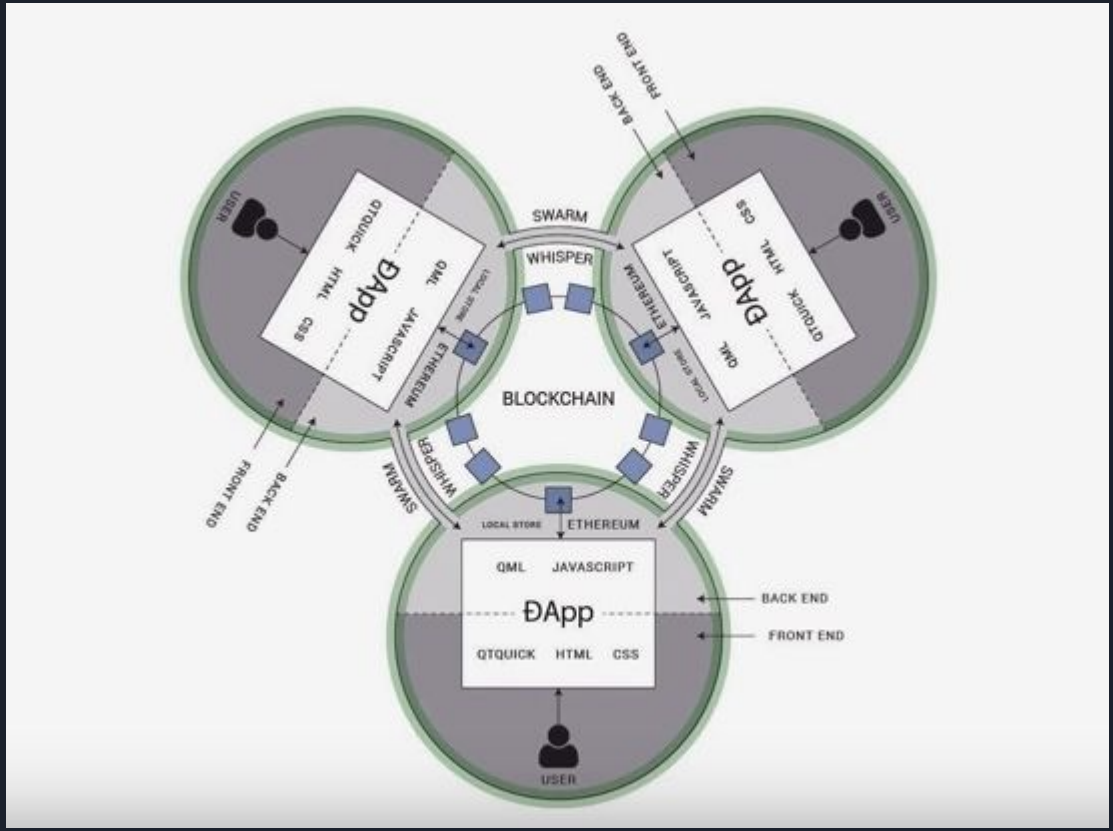

MetaMask Notification
CONFIRM TRANSACTION
Ropsten Test Net
Account 1
E5f689...A722
3.952 ETH
41.81 USD
C03Db3...885A
Amount
0 ETH
0.00 USD
Max Transaction Fee
0.000465 ETH
0.00 USD
Max Total
0.000465 ETH
0.00 USD
Data included: 4 bytes
ACCEPT
REJECT

Deploy smart contract’u

1. Dostęp do węzła Ethereum
2. Napisanie kodu w Solidity
3. Przetestowanie
4. Odblokowanie (unlock) adresu Ethereum z węzła
5. $truffle compile
6. $truffle migrate


dApp
Attention &
Reputation Model
Communication
Social Layer
Transactions
Identity
The
Decentralized
Economy
Smart Contracts
Storage &
Content Delivery
Blockchain 2.0

Dapp


USER
QTQUICK
HTML
CSS
DApp
QML
JAVASCRIPT
LOCAL STORE
ETHEREUM
SWARM
WHISPER
BLOCKCHAIN
BACK END
FRONT END
LOCAL STORE
ETHEREUM
QML
JAVASCRIPT
DApp
QTQUICK
HTML
CSS
USER
BACK END
FRONT END
SWARM
WHISPER
WHISPER
SWARM
LOCAL STORE
ETHEREUM
QML
JAVASCRIPT
ÐApp
QTQUICK
HTML
CSS
BACK END
FRONT END
USER

Jak programistycznie komunikować się z blockchainem?

Wiemy już w jaki sposób połączyć się z siecią uruchamiając węzeł.

Mamy dwa sposoby, żeby odczytywać blockchain lub wysyłać transakcje.

- Możemy użyć kodu węzła w postaci biblioteki, (na razie mało popularne; jedynie na mobile)
- Lub skorzystać z wystawionego interfejsu JSON-RPC

Komunikacja

- RPC
- Web3.js
- MetaMask
- nodejs

RPC

```
# Uruchamiamy parity, które domyslnie wystawia JSON-RPC over HTTP na porcie :8545
$ parity --jsonrpc-apis eth

# Do wykonywania requestow polecam narzędzie httpie (https://httpie.org/)
$ http localhost:8545 jsonrpc=2.0 id=1 method=eth_blockNumber params:='[]'

# 10/ Większość liczb (U256) będzie zakodowana jako hex
HTTP/1.1 200 OK
Content-Type: application/json
Date: Tue, 03 Apr 2018 07:42:34 GMT
Transfer-Encoding: chunked

{
   "id": 1,
   "jsonrpc": "2.0",
   "result": "0x6622a9"
}
```

```
# 11/ Balans (ETH) konta
$ http localhost:8545 jsonrpc=2.0 id=1 method=eth_getBalance
params:='["0x00a329c0648769a73afac7f9381e08fb43dbea72"]'
HTTP/1.1 200 OK
Content-Type: application/json
Date: Tue, 03 Apr 2018 07:46:48 GMT
Transfer-Encoding: chunked

{
   "id": 1,
   "jsonrpc": "2.0",
   "result": "0x456b9629f0a15c00"
}
```

web3.js

```
// Importujemy klasę web3
const Web3 = require('web3')

run()

async function run () {
 // Tworzymy nową instancję i podajemy adres nodea
 const web3 = new Web3('http://localhost:8545')
 // Pobieramy block
 const block = await web3.eth.getBlock('latest');

 console.log(block)

 // Dane w otrzymanym obiekcie są już przetworzone, nie musimy parsować heksów.
 const { number, hash, transactions, timestamp } = block
 console.log(`
Block number: ${number}
Block hash: ${hash}
Block date: ${new Date(timestamp * 1000)}
Number of transactions: ${transactions.length}
`)
```

Web3.js - transakcje

```
const Web3 = require('web3')
run()
async function run () {
 const web3 = new Web3('http://localhost:8545')
 // 3/ Wykorzystujemy listę kont, żeby wybrać nadawcę i odbiorcę
 const accounts = await web3.eth.getAccounts();
 const from = accounts[0];
 const to = from;
 // Przesyłamy dodatkowo jakąś wartość w ETH
 const value = 50000;
 try {
   // 4/ wysylamy transakcje podając tylko te pola, na których nam zależy.
   // Reszta (gas,gasPrice,nonce) zostanie dobrana automatycznie.
   const txHash = await web3.eth.sendTransaction({
     from, to, value
   })
   // W przypadku zaakceptowania transakcji przez użytkownika dostajemy jej hash.
   console.log(`Transaction sent with hash: ${txHash}`)
 } catch (e) {
   // Obsługujemy potencjalny błąd (lub odrzucenie przez użytkownika)
   console.error(`Could not send transaction: ${e}`)
```

MetaMask

```
window.addEventListener('load', () => {
   // sprawdzamy czy nasza przeglądarka wspiera Web3
   if (window.web3) {
     // Tworzymy nową instancję globalnie (nasza wersja)
     window.web3 = new Web3(web3.currentProvider);
     // I uruchamiamy aplikację.
     start()
   } else {
     // W przeciwnym wypadku odpalamy monit o instalację MetaMask.
     alert('Non-Ethereum browser detected. You should consider trying MetaMask!');
   }
 })
```

EIP-1102

Nowy standard

EIP-1102 opisuje nowy standard komunikacji pomiędzy MetaMask (czy innymi przeglądarkami, jak np. Mist) dla Dapp.
Dapps będzie musiał poprosić o udostępnienie danych użytkownika.

```
if (window.ethereum) {
   window.web3 = new Web3(ethereum);
   try {
     // Poproś o szczegółowe dane usera (jak konta)
     await ethereum.enable();
```

MetaMask cd.

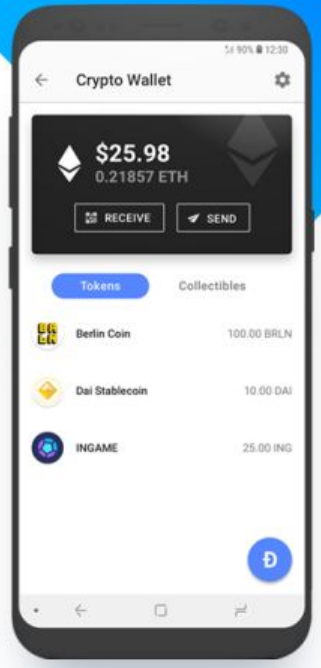
```
function start () {
   // Pobieramy najnowszy blok
   web3.eth.getBlock('latest', (err, block) => {
     console.log('block', err, block);
     setTimeout(start, 2500);
     if (err) {
       return
     }

     // 2/ I zapisujemy numer i hash do drzewa DOM.
     document.querySelector('.last-block').textContent =
       `#${block.number} (${block.hash.substr(0, 6)}..${block.hash.substr(-4)})`;
   });
 }
```

Co dalej?

- Skalowalność ETH - Ethereum 2.0
- Hard Fork Casper
- EOS

Adopcja



Opera browser for Android with built-in crypto wallet

The first major browser to integrate a crypto wallet, enabling seamless access to the emerging web of tomorrow (Web 3).

POBIERZ Z Google Play

Media release

Boerse Stuttgart
DIE PRIVATANLEGERBÖRSE

Stuttgart, Germany, 12 December 2018

solarisBank supports Boerse Stuttgart Group in the development of the trading system for its crypto trading venue

Technology platform with banking license becomes partner for technology and banking services

As a technology and banking partner, solarisBank supports Boerse Stuttgart Group in creating an end-to-end infrastructure for digital assets. Together with solarisBank, Boerse Stuttgart Group is developing the trading system for its crypto trading venue, which is set to launch in the first half of 2019. In addition, solarisBank will provide the required crypto trading banking services to Boerse Stuttgart Group.

The web has come to dominate pretty much every aspect of our lives and has brought plenty of problems too, including: a revolutionary change in how we communicate via social media, the enormous spread of false information and hate speech, online political manipulation, threats to net neutrality, the mishandling and abuse of our private data and the rise of monstrously-large, enormously-powerful, the obliteration of entire industries, basically-unregulated internet-tech companies like Facebook and Amazon – and the über rich who profit from them.

To nie jest silver bullet

Slajd specjalnie dla Grzegorza - 51%

DevCon4

What is Substrate?

Gdzie jestesmy na osi czasu historii?

Smart contracts = EWASM (Ethereum WebAssembly)

Rozwój Frontendu

Linki

1. https://remix.ethereum.org
2. https://wiki.parity.io/JSONRPC
3. https://github.com/trufflesuite/ganache-cli
4. https://github.com/ethereum/eth2.0-specs
5. https://economicgraph.linkedin.com/research/linkedin-2018-emerging-jobs-report

Dziękuję za uwagę.

Pytania? Uwagi?