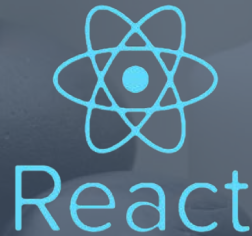


BLOCKCHINESE YOUR SPA

OR

INTEGRATE YOUR SINGLE PAGE APPLICATION WITH BLOCKCHAIN



WHAT's this is about?

1

BLOCKCHAINS OVERVIEW

2

DECENTRALIZED APPLICATIONS VS CLASSIC BEs

3

JS and WEB 3.0

4

CHALLENGES AND ISSUES

5

ANGULAR APPLICATION EXAMPLE

SAY CRYPTO CURRENCY

ONE MORE TIME

BLOCKCHAIN TYPES

PERMISSION-LESS



Ethereum



Tron



EOS

PERMISSIONED



Hyperledger fabric



Ripple

Blockchain applications

Decentralized applications (DApp)

Smart contracts

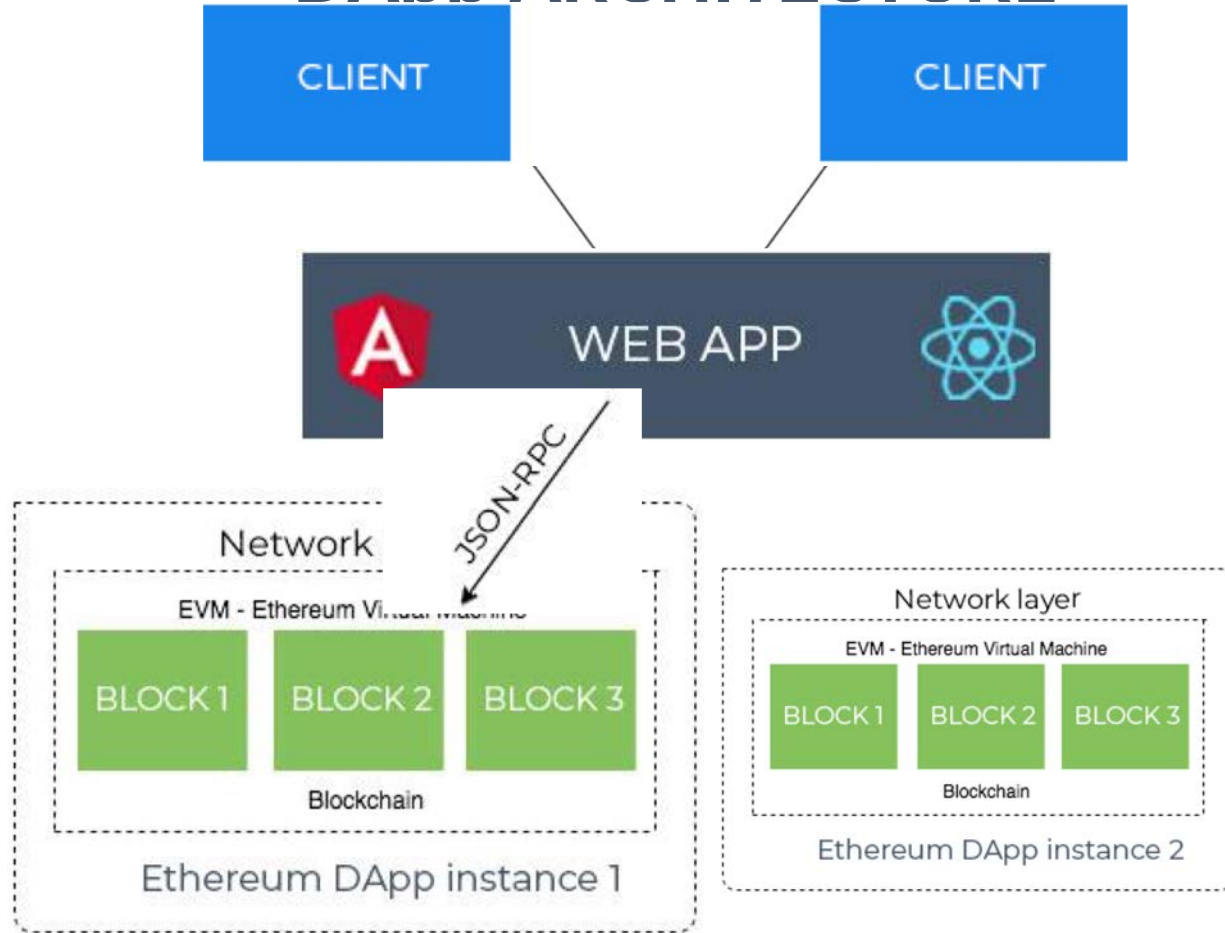
EVM and Solidity language

They are like backends, services

Global and irreversible



DApp ARCHITECTURE



The context

Ethereum angular superheroes

All | Marvel | DC | JavaScript



The context

Ethereum angular superheroes



Hulk

When Bruce Banner changes into the Hulk, he becomes an unstoppable beast of near unlimited strength, power, and destruction. The Hulk's strength is probably the greatest in the Marvel universe, with many foes falling to his thunderous attacks. The Hulk is also able to leap great distances traveling for miles before bounding upwards again. For his size, the Hulk is incredibly fast and can run great distances at extreme speeds. He generally travels by jumping as described above though. The hulk is also highly resistant to damage, being near impervious to most forms of damage. Very little has been known to faze the Hulk, except those of the same power level as the Hulk such as The Thing, Thor, Abomination, and others. Even when the Hulk is damaged, he heals quickly, and his endurance makes him an untiring creature capable of immense destruction. The Hulk is truly a marvel, both in his capability to defeat any enemy that would get in his way and as a being capable of destroying much that mankind has worked so hard to create.

[Back to list](#)

[Add a review](#)

Score

Review text

The context

Ethereum angular superheroes

Add a superhero

Superhero name

Hero universe

Superhero avatar url

Superhero description

Add hero

Back to list

TRADITIONAL BACKEND VS BLOCKCHAIN

	TRADITIONAL	DECENTRALIZED
ARCHITECTURE	REST, GraphQL	JSON-RPC, gRPC
PERFORMANCE	Fast	Slow
AUTHENTICATION	Flexible	SSH key pairs based
APPLICATIONS	Services based	Smart contracts
ADOPTION	Well known	Requires onboarding

JSON-RPC

Protocol **agnostic**

Has only **one method** to send data
(POST on HTTP).

JSON to encode requests and responses
between client and server.



JSON-RPC REQUESTS

curl

```
curl -XPOST -d '{"method": "getSuperHeroes",  
"params": [ "marvel" ], "id": 123 }'  
'https://superheroes.com/rpc'
```

JavaScript

```
fetch("https://superheroes.com/rpc", {  
  method: "POST",  
  body: JSON.stringify({ method: "getSuperHeroes",  
    params: [ "marvel" ], id: 123 }  
}));
```

Contracts ABI

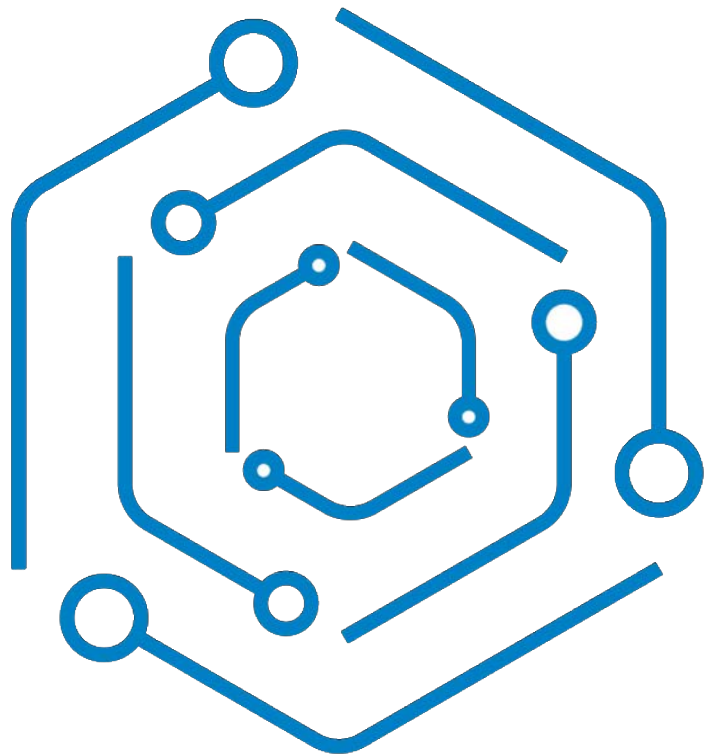
Application Binary Interface

Contract specification in JSON format

Provides list of contract methods and events signatures.

Provides all the information for constructing JSON-RPC requests

Natively supportable by JavaScript



```
{  
  "contractName": "Superheroes",
```

```
  "abi": [  
    {"name": "superheroes"...},  
    {"name": "NewReview"...},  
    {"name": "NewSuperhero"...},  
    {"name": "review"...},  
    {"name": "addSuperhero"...},  
    {"name": "getSuperHeroes"...},  
    {"name": "getHero"...},  
    {"name": "getSuperheroReviews"...}  
  ],
```

```
  "bytecode": "0x608060405234801561001057600080fd5b506118bc80...",
```

```
  "deployedBytecode": "0x608060405260043610610077576000357c01...",
```

```
  "sourceMap": "59:2121:1:-;:::8:9:-1:5:2;:::30:1:27:20:12:5:2...",
```

```
  "source": "pragma solidity ^0.5.0;\npragma experimental ABI..."
```

```
  "networks": {
```

```
    "3": {  
      "events": {},  
      "links": {},
```

```
      "address": "0xFB3716816d93168788d306fCEab06636CA44682D",
```

```
      "transactionHash": "0x6add8e5116b481a815c82e72c88056b7d8de535b742cae4e6060be2accfd4d7e"
```

```
    }  
  }  
}
```

```
{
  "contractName": "Superheroes",
  "abi": [
    {
      "constant": true,
      "inputs": [],
      "name": "getSuperHeroes",
      "outputs": [
        {
          "components": [
            {
              "name": "id",
              "type": "uint256"
            },
            {
              "name": "name",
              "type": "string"
            }
          ],
          "name": "result",
          "type": "tuple[]"
        }
      ],
      "signature": "0x8261cf2b"
    }
  ]
}
```


Ethereum JSON-RPC REQUESTS

Read operations, no parameters. Calling getSuperHeroes

```
fetch("https://localhost:7545", {  
  method: "POST",  
  body: JSON.stringify({  
    params: [{  
      data: "0x8261cf2b",  
      to: "0xFB3716816d93168788d306fCEab06636CA44682D"},  
      "latest"  
    ],  
    jsonrpc: "2.0",  
    method: "eth_call" id: 1  
  })  
});
```

```
{
  "contractName": "Superheroes",
  "abi": [
    {
      "constant": true,
      "inputs": [
        {
          "name": "id",
          "type": "uint256"
        }
      ],
      "name": "getHero",
      "outputs": [
        {
          "components": [
            {
              "name": "id",
              "type": "uint256"
            },
            {
              "name": "name",
              "type": "string"
            }
          ]
        }
      ]
    },
    {
      "signature": "0x21d80111"
    }
  ]
}
```

Ethereum JSON-RPC REQUESTS

Read with parameters. E.g. `getHero(10)`

```
fetch("https://localhost:7545", {
  method: "POST",
  body: JSON.stringify({
    params: [{
      data: "0x21d8011100000000000000000000000000000000000000000000000000000000.....16",
      to: "0xFB3716816d93168788d306fCEab06636CA44682D"},
      "latest"
    ],
    jsonrpc: "2.0",
    method: "eth_call", id: 1
  })
});
```

The call result

Read with parameters. E.g. `getHero(10)`

[illegible]

Decode hex result

Hexadecimal string to ASCII example

```
hexToAscii(hexString) => {  
  const hex = hexString.toString();  
  let str = "";  
  for (let n = 0; n < hex.length; n += 2) {  
    str += String.fromCharCode(parseInt(hex.substr(n, 2), 16));  
  }  
  return str;  
}
```

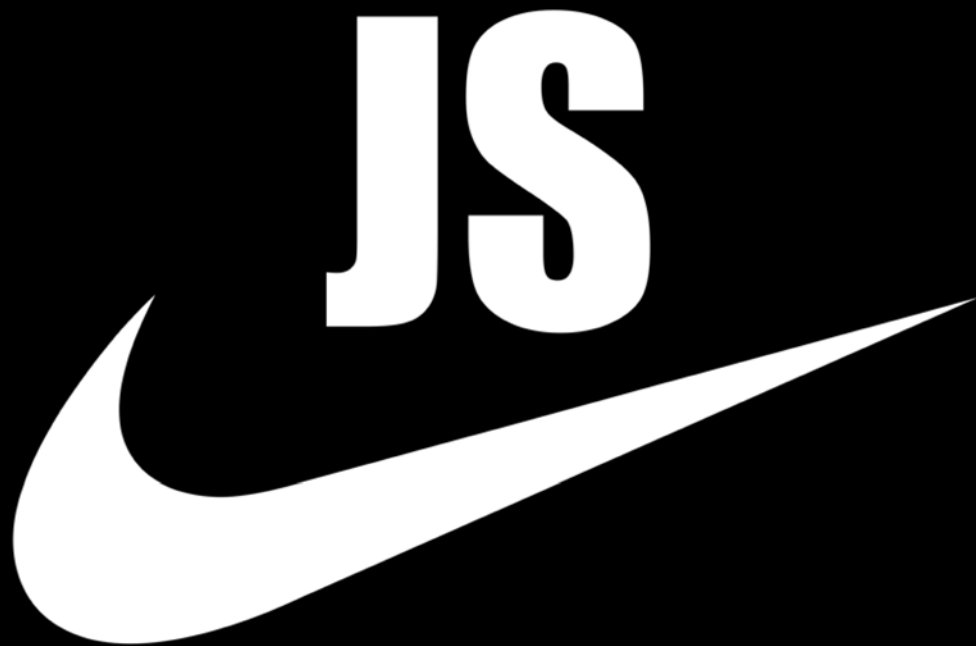
```
hexToAscii('53757065726d616e'); // Superman
```

Decode hex result

Hexadecimal number to decimal.

```
hexadecimalNumberToDecimal(hexNumber) => {  
  const hexStringNumber = hexNumber.toString(16);  
  return parseInt(hexString, 10);  
}
```

```
hexadecimalNumberToDecimal(16); // 10
```



JUST LIBRARY IT

Blockchain JS libraries



Ethers.js



TRON**LINK**

WEB3.JS

Decode RPC calls results.

Encode parameters.

Make RPC requests.

Send transaction. Basically write operation.

Listen to events.

Get Ethereum user accounts.

Get Ethereum details and information.



Ethereum JSON-RPC REQUESTS

Decoding the result with Web3JS

```
import Web3 from 'web3';  
import superheroesABI from './build/contracts/Superheroes.json';  
const web3 = new Web3();  
  
const functionABI = superheroesABI.abi.find((abiItem) => {  
  return abiItem.name === route.data['getHero'];  
});  
  
const decoded = web3.eth.abi.decodeParameters(functionABI.outputs,  
result);
```

Getting callable ABI with WEB3

```
import Web3 from 'web3';
```

```
import superheroesABI from './build/contracts/Superheroes.json';
```

```
const web3 = new Web3();
```

```
const myContract: any = new web3.eth.Contract(  
  superheroesABI,  
  'http://localhost:7445/'  
);
```

Calling contract functions

Read operations

```
myContract.methods.getHeroes().call()
```

Read with parameters

```
myContract.methods.getHeroe(10).call();
```

Calling contract functions

Write operations, transactions

```
myContract.methods
```

```
.addHero(  
  'Super man',  
  'http://test.com/avatar',  
  'DC',  
  'Some description here'  
)
```

```
.send({  
  from: web3.eth.defaultAccount  
});
```

Wallets and browser extensions



METAMASK



FABRIC CHROME EXTENSION

HYPERLEDGER
FABRIC



TRON**LINK**

MetaMask extension

Injects Web3JS instance into global variable.

Provides list of user accounts.

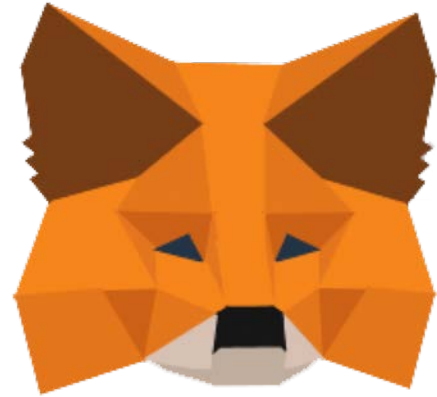
Creates, stores SSH key pairs.

Signs transactions with SSH keys.

Chrome, Firefox and Opera.

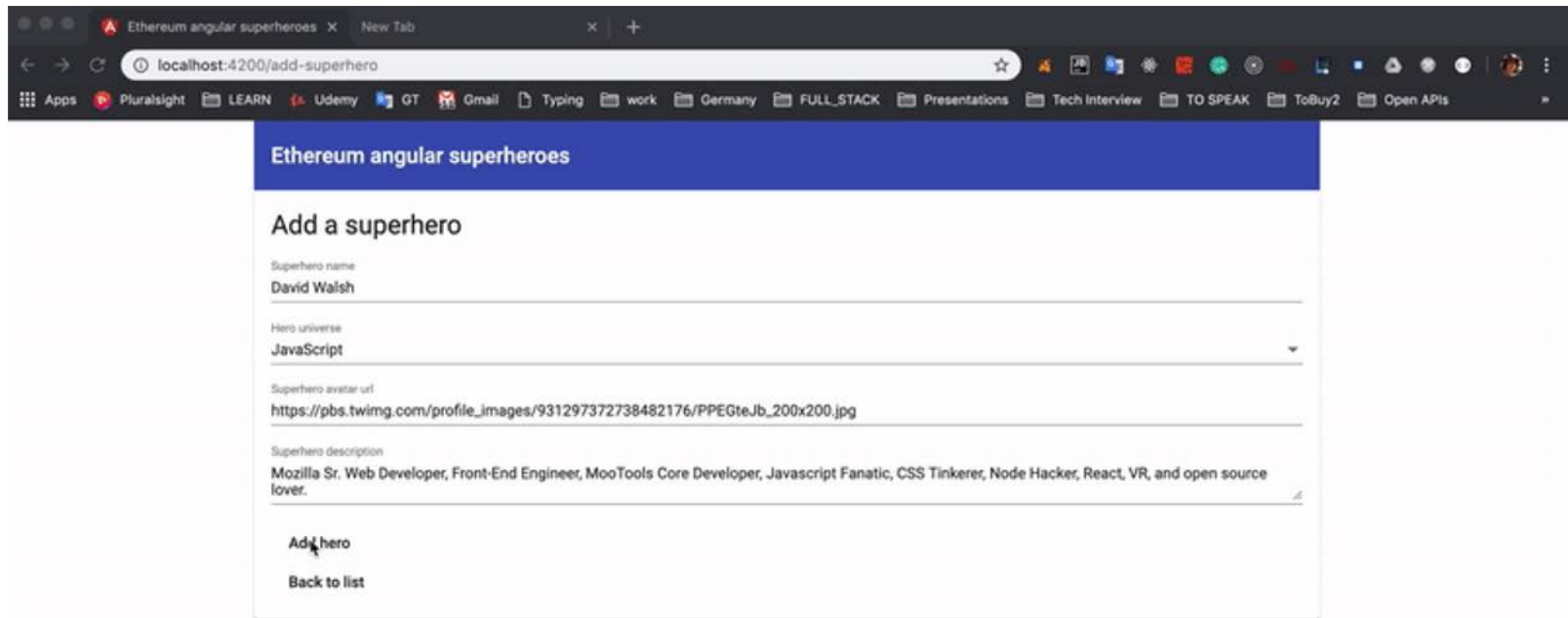
Listens to Ethereum events.

Stores crypto tokens.



METAMASK

MetaMask sign and send transaction



The screenshot shows a web browser window with the address bar displaying `localhost:4200/add-superhero`. The browser's tab bar shows "Ethereum angular superheroes" and "New Tab". The browser's bookmark bar contains various links like "Apps", "Pluralsight", "LEARN", "Udemy", "GT", "Gmail", "Typing", "work", "Germany", "FULL_STACK", "Presentations", "Tech Interview", "TO SPEAK", "ToBuy2", and "Open APIs".

The web application has a blue header with the text "Ethereum angular superheroes". Below the header is a form titled "Add a superhero". The form contains the following fields:

- Superhero name:** A text input field containing "David Walsh".
- Hero universe:** A dropdown menu with "JavaScript" selected.
- Superhero avatar url:** A text input field containing "https://pbs.twimg.com/profile_images/931297372738482176/PPEGteJb_200x200.jpg".
- Superhero description:** A text area containing "Mozilla Sr, Web Developer, Front-End Engineer, MooTools Core Developer, Javascript Fanatic, CSS Tinkerer, Node Hacker, React, VR, and open source lover."

At the bottom of the form are two buttons: "Add hero" and "Back to list".

Odd and unusual stuff :(

Solidity returns arrays instead of objects.

```
// [1, 'Superman', 'https://test.com/avatar', 'DC', 'Has many superpowers']  
const [id, name, avatar, category, description] = RPCData;
```

Ethereum doesn't support float numbers.

```
// const value = 3.14;  
const factor = 100;  
const factorValue = 3.14 * factor; // => 314;
```

Odd and unusual stuff 2 :(

Solidity BigNumber.

The maximum safe integer in JavaScript ($2^{53} - 1$).

The maximum uint256 value ($2^{256} - 1$).

Encoding/decoding data.

Blockchains are slow. Average transaction time is 20 sec.

Subscribing to ethereum events

```
const socket = new WebSocket( 'wss://localhost:7545',);
socket.addEventListener('open', (event) => {
  const request = {
    id: 3,
    method: 'eth_subscribe',
    params: ['logs', {
      topics: [functionABI.signature],
      address: ABI.networks.address
    }]
  };
  socket.send(JSON.stringify(request));
});
```

// On hero created event

Listening to ethereum WSS events

```
socket.addListener('message', (event) => {  
  const data = JSON.parse(event.data);  
  if (data.params && data.params.result) {  
    const web3 = new Web3();  
    const decoded = web3.eth.abi.decodeLog(  
      functionABI.inputs,  
      data.params.result.data,  
      data.params.result.topics);  
    const [author, mark, text] = decoded;  
  }  
});
```

Events with WEB3JS

```
const web3 = new Web3();  
const myContract: any = new web3.eth.Contract(  
  superheroesABI,  
  'http://localhost:7445/'  
);  
  
myContract.events.NewHero()  
  .on('data', (event) => {  
    const [id, name, avatar, category, description] = event.returnValues  
    .on('error', console.error);
```


Conclusion

Ethereum and DApps overview.

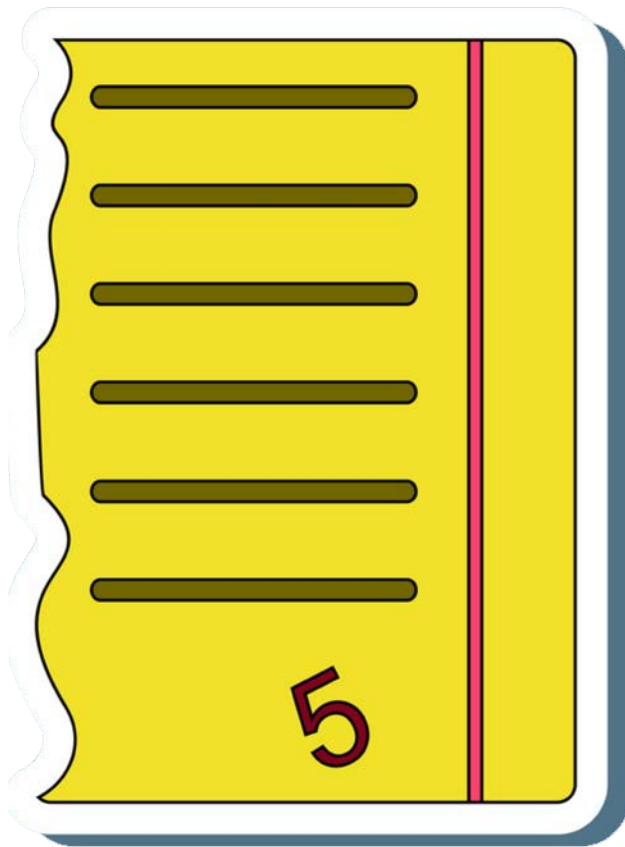
Make requests to Ethereum with vanilla JS.

Encoding request parameters.

Decoding response results.

Subscribing and listening to WSS Events.

Working with WEB3JS.



Thanks for attention

Example angular-ethereum project:

<https://bit.ly/2Lw6zLO>

Learning resources:

<https://cryptozombies.io/>

<http://www.dappuniversity.com/>

<https://academy.ivanontech.com/>

Contact me:

<https://www.linkedin.com/in/noadm/>

<https://github.com/DmitriyNoa>

dmytro.zharkov@gmail.com

