

Hands-on Session: Build Polygon ENS

Polygon 네트워크에서 도메인 서비스 만들어보기

2022-05-14 SOPT Web3 Basic Study Saturday

Roadmap: Build a toy project

- 지난 4주 동안 Ethereum 생태계를 중심으로 블록체인의 기초에 대해 살펴보았습니다.
- 학습한 키워드: ERC20 / ERC721 (NFT) / Layer 1 & Layer 2 / Interchain & Dex (Bridge)
- 5월 3주차 (5월 14일)부터 앱잼 팀빌딩이 시작되기 직전인 6월 25일까지 하나의 작은 Web3/Crypto 관련 토이 프로젝트를 진행할 예정입니다. (3인에 1팀이 어떨까요?)
- 5월 21일, 6월 4일, 11일, 18일, 25일 총 5번이 있습니다. 6월 25일 (토) 까지 진행합니다.
- 우리가 지금까지 학습했던 내용을 바탕으로 주제를 선정해서 토이 프로젝트를 만드는 활동을 앞으로 한 달 동안 진행할 예정입니다.
- 그리고, 여러분이 선정한 주제 가운데에서, 제가 학습하면 좋을 키워드를 뽑아 특강 형태로 진행할 예정입니다.
- 오늘 스터디가 끝나기 전까지 본인이 하고 싶은 주제를 한 번 생각해보세요.

어떤 (Ethereum) 프로젝트를 진행할 수 있을까요?

NFT 몬스터 기반 간단한 게임 만들기



<https://laufeyson-game.vercel.app/>
<https://pokemonnft.vercel.app/>

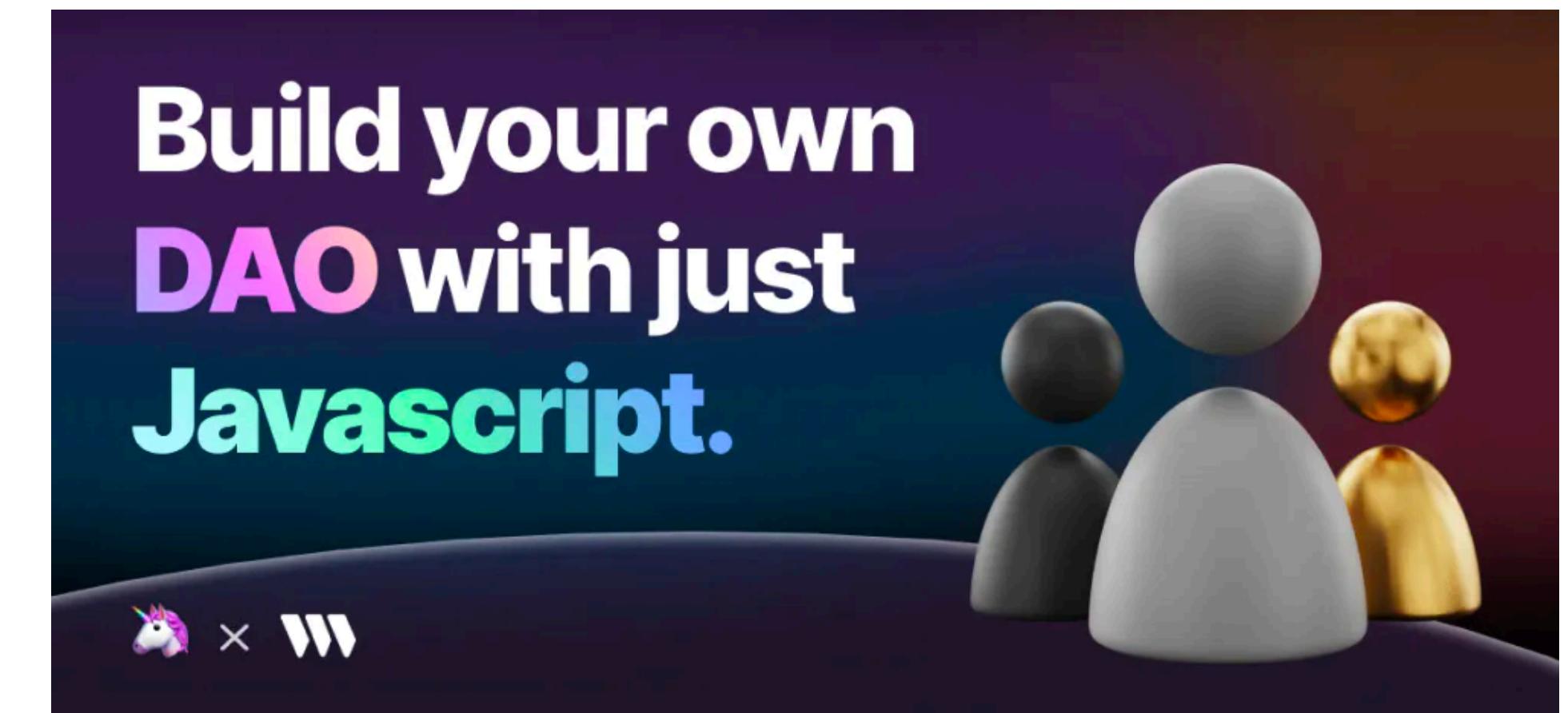
Social Token / NFT 민팅 사이트 만들기

NFT Marketplace 사이트 만들기

Ethereum을 백엔드 서버로 이용하는 나무위키

zk: 익명으로 투표할 수 있는 앱 만들기

나만의 (익명) DAO 조직 만들기



<https://toebeansdao.netlify.app/>

DeFi: 복권, 로또 사이트 만들기

DeFi: 날씨 및 주가 예측 (토토) 사이트 만들기

가장 싼 NFT 가격을 긁어서 보여주는 API 서버

zk: 익명으로 송금할 수 있는 Mixer 만들기

dApps

Challenges	Tokens / Winner	Winners
 Build a DEX	50,000	5
 Build an NFT Marketplace	50,000	5
 Build an NFT Minting UI	10,000	5
 Build a DAO Launcher	30,000	5

Tooling

Challenges	Tokens / Winner	Winners
 Combine Provider Libraries	30,000	5
 Staking + Governance UI	30,000	5
 Running Archway CLI in Local Env	50,000	5
 Load Generator Script	40,000	5
 Rewards Tracking Dashboard	40,000	5
 Build a Wallet	50,000	5

Content

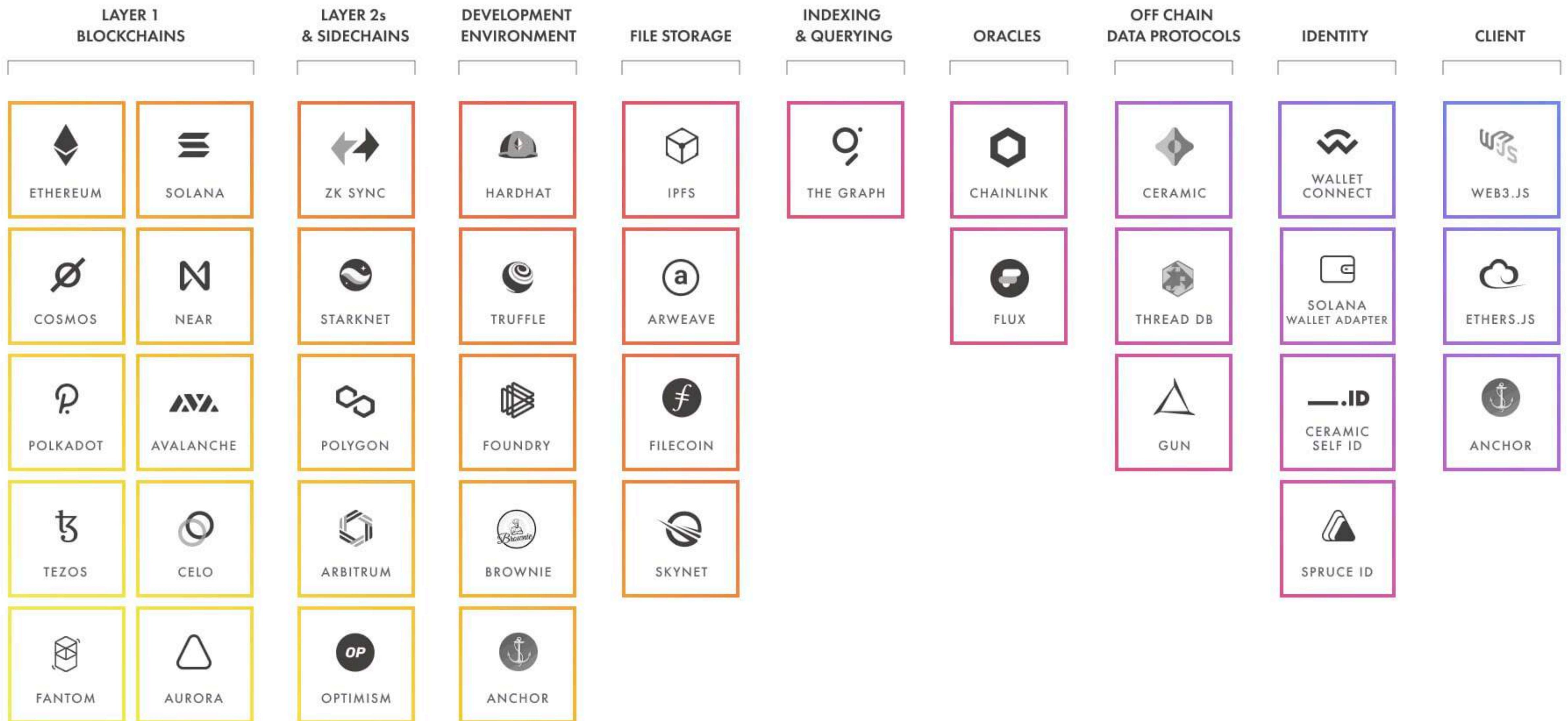
Challenges	Tokens / Winner	Winners
 Create Validator Content	5,000	30
 Create dApp Development Content	5,000	30
 Create Community Content	5,000	30

Grant Program!

Build to Earn

[https://gitcoin.co/explorer?
network=mainnet&idx_status=open&applicants=ALL&order_by=-web3_created](https://gitcoin.co/explorer?network=mainnet&idx_status=open&applicants=ALL&order_by=-web3_created)
<https://open.harmony.one/300m-on-bounties-grants-daos/apply-for-grants-or-dao>
<https://philabs.notion.site/Testnet-Challenges-9449524d909c4831ac303ed485194b16>

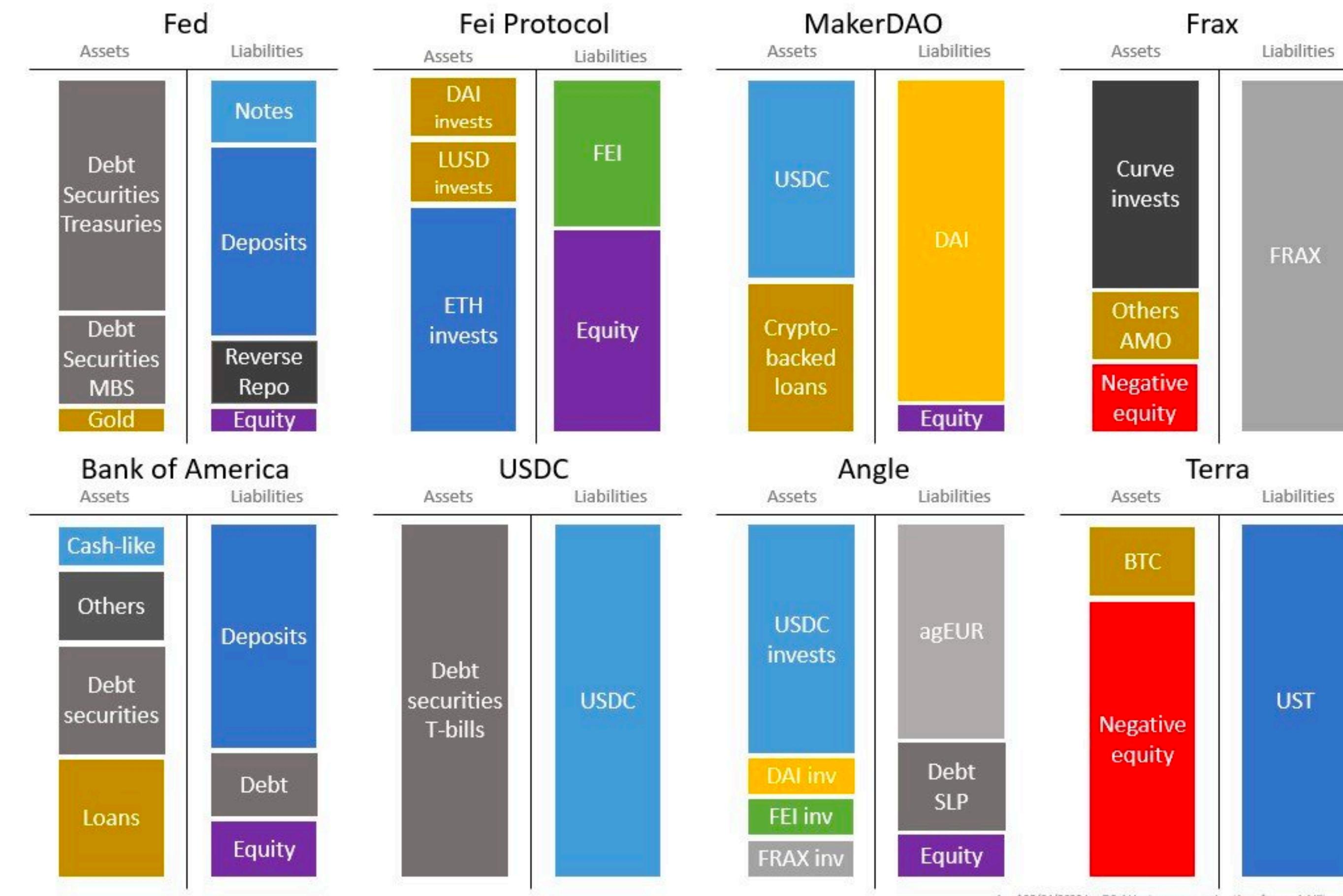
The web3 stack



Edge & Node

Before started: What was happened to Terra?

100% 코드로 이루어진 금융 생태계가 실패하다



```
mkdir cool-domains  
cd cool-domains  
npm init -y  
npm install --save-dev hardhat
```

bash

```
npx hardhat  
npm install --save-dev @nomiclabs/hardhat-waffle ethereum-waffle chai @nomiclabs/hardhat-  
ethers ethers // 어짜피 자동으로 됨 yes? 나올 때 엔터만 해주세요.  
npm install @openzeppelin/contracts
```

shell

```
// 잘 깔렸는지 확인  
npx hardhat compile  
npx hardhat test
```

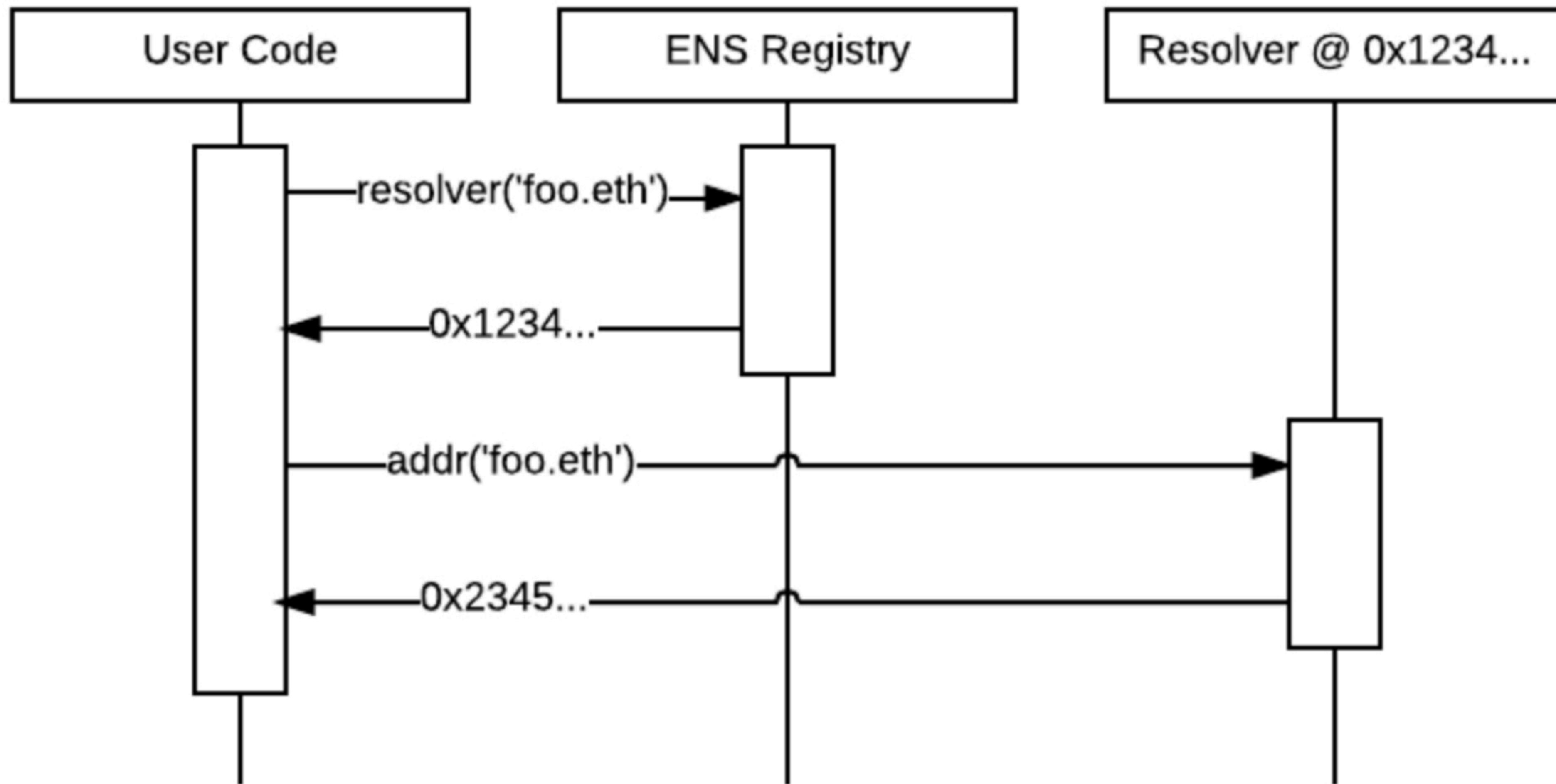


```
$ npx hardhat test  
Compiling 1 file with 0.7.3  
Compilation finished successfully
```

Greeter

```
Deploying a Greeter with greeting: Hello, world!  
Changing greeting from 'Hello, world!' to 'Hola, mundo!'  
✓ Should return the new greeting once it's changed (803ms)
```

```
1 passing (805ms)
```



◆ Domains.sol ×

contracts > ◆ Domains.sol

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.10;
4
5 import "hardhat/console.sol";
6
7 contract Domains {
8     constructor() {
9         console.log("THIS IS MY DOMAINS CONTRACT. NICE.");
10    }
11 }
```

Smart Contract는 다른 언어에서의 Class와 유사한 개념을 갖는다.

이더리움의 경우, 하나의 Contract에 따른 하나의 Instance만 갖는 싱글톤 객체이다.

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure under 'COOL-DOMAINS'. It includes 'contracts' (with 'Domains.sol'), 'node_modules', 'scripts' (with 'run.js' selected), 'test', '.gitignore', 'hardhat.config.js', 'package-lock.json', 'package.json', and 'README.md'.
- Run Tab**: Shows the file 'run.js' with the following code:

```
scripts > JS run.js > ...
1 const main = async () => {
2     const domainContractFactory = await hre.ethers.getContractFactory('Domains');
3     const domainContract = await domainContractFactory.deploy();
4     await domainContract.deployed();
5     console.log("Contract deployed to:", domainContract.address);
6 };
7
8 const runMain = async () => {
9     try {
10         await main();
11         process.exit(0);
12     } catch (error) {
13         console.log(error);
14         process.exit(1);
15     }
16 };
17
18 runMain();
```

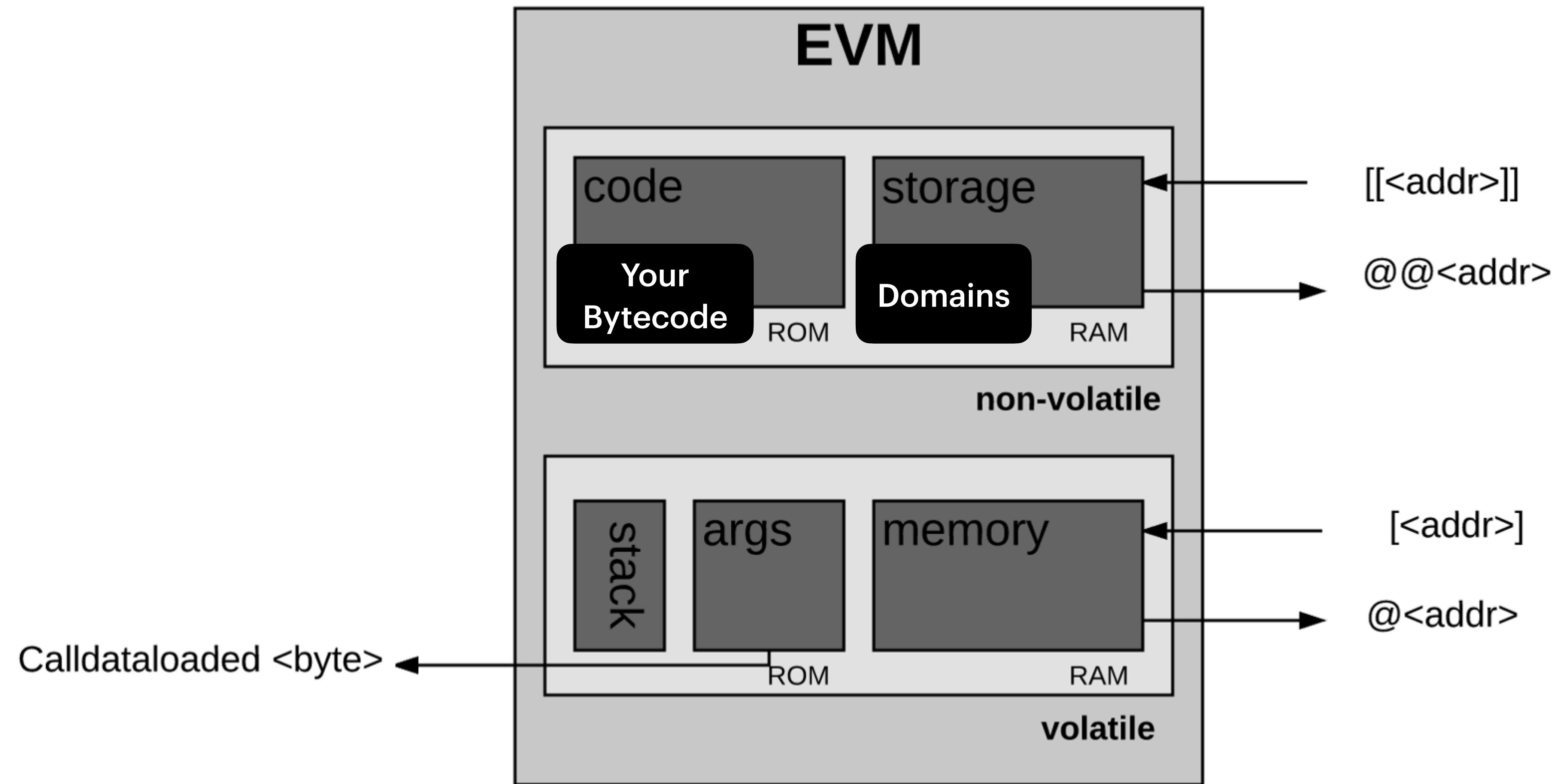
배포를 위한 스크립트를 작성한다.

중요한 것은, Smart Contract를 getContractFactory에서 가져와야 하고 이름을 명시해야 한다는 것이다.

Deploy를 시작하고, 블록체인에 mine 될 때까지 기다리도록 deployed()로 await 한다.

contracts > ◆ Domains.sol

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.10;
4
5 import "hardhat/console.sol";
6
7 contract Domains {
8     // A "mapping" data type to store their names
9     mapping(string => address) public domains; // state variable: 영구히 블록체인에 storage 타입으로 저장되는 타입
10                                // domains는 도메인의 소유자를 의미함 ex) sigridjin.eth의 소유자는 진형의 지갑이다.
11     constructor() {
12         console.log("THIS IS MY DOMAIN CONTRACT. NICE.");
13     }
14
15     // A register function that adds their names to our mapping
16     function register(string calldata name) public { // calldata: final 키워드를 붙인 파라미터와 동일
17         domains[name] = msg.sender;
18         console.log("%s has registered a domain!", msg.sender);
19     }
20
21     // This will give us the domain owners' address
22     function getAddress(string calldata name) public view returns (address) {
23         return domains[name];
24     }
25 }
```



scripts > JS run.js > ...

```
1 const main = async () => {                                // config에 명시된 지갑 주소를 가져온다. 로컬이라면, 랜덤으로 만든다.
2     const [owner, randomPerson] = await hre.ethers.getSigners();
3     const domainContractFactory = await hre.ethers.getContractFactory('Domains');
4     const domainContract = await domainContractFactory.deploy();
5     await domainContract.deployed();
6     console.log("Contract deployed to:", domainContract.address);
7     console.log("Contract deployed by:", owner.address);
8
9     const txn = await domainContract.register("doom");
10    await txn.wait();
11
12    const domainOwner = await domainContract.getAddress("doom");
13    console.log("Owner of domain:", domainOwner);
14 }
15
16 const runMain = async () => {
17     try {
18         await main();
19         process.exit(0);
20     } catch (error) {
21         console.log(error);
22         process.exit(1);
23     }
24 };
25
26 runMain();
```

.Domains.sol

JS hardhat.config.js ●

JS run.js

JS hardhat.config.js > [?] <unknown> > 🔑 solidity

```
12
13 // You need to export an object to set up your config
14 // Go to https://hardhat.org/config/ to learn more
15
16 /**
17 * @type import('hardhat/config').HardhatUserConfig
18 */
19 module.exports = {
20   solidity: "0.8.10",
21 };
22
```

... more info go to <https://hardhat.org/config/> or run `hardhat help` for ...
✖ ~ /Documents/github/cool-domains └ npx hardhat run scripts/run.js

Compiled 2 Solidity files successfully

THIS IS MY DOMAIN CONTRACT. NICE.

Contract deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3

Contract deployed by: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266 has registered a domain!

Owner of domain: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

contracts > ◆ Domains.sol

```
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.10;
3
4 import "hardhat/console.sol";
5
6 contract Domains {
7     mapping(string => address) public domains; // domains는 도메인의 소유자를 의미함 ex) sigridjin.eth의 소유자는 진형의 지갑이다.
8
9     // Checkout our new mapping! This will store values
10    mapping(string => string) public records;           // records는 domain을 어디로 연결할지를 알려주는 문자열 값이다.
11
12 constructor() {
13     console.log("Yo yo, I am a contract and I am smart");
14 }
15
16 function register(string calldata name) public {
17     // Check that the name is unregistered (explained in notes)
18     require(domains[name] == address(0));
19     domains[name] = msg.sender;
20     console.log("%s has registered a domain!", msg.sender);
21 }
22
23
24 function getAddress(string calldata name) public view returns (address) {
25     // Check that the owner is the transaction sender
26     return domains[name];
27 }
28
29 function setRecord(string calldata name, string calldata record) public {
30     // Check that the owner is the transaction sender
31     require(domains[name] == msg.sender);
32     records[name] = record;
33 }
34
35 function getRecord(string calldata name) public view returns(string memory) {
36     return records[name];
37 }
38 }
```

Domains.sol

hardhat.config.js

run.js

X

scripts > run.js > ...

```
1  const main = async () => {
2    // The first return is the deployer, the second is a random account
3    const [owner, randomPerson] = await hre.ethers.getSigners();
4    const domainContractFactory = await hre.ethers.getContractFactory('Domains');
5    const domainContract = await domainContractFactory.deploy();
6    await domainContract.deployed();
7    console.log("Contract deployed to:", domainContract.address);
8    console.log("Contract deployed by:", owner.address);
9
10   let txn = await domainContract.register("doom");
11   await txn.wait();
12
13   const domainAddress = await domainContract.getAddress("doom");
14   console.log("Owner of domain doom:", domainAddress);
15
16   // Trying to set a record that doesn't belong to me!
17   txn = await domainContract.connect(randomPerson).setRecord("doom", "Haha my domain now!");
18   await txn.wait(); // 작동을 하지 않을 것이다. 주인이 아닌 randomPerson이 setRecord를 호출했기 때문이다.
19 }
20
21 const runMain = async () => {
22   try {
23     await main();
24     process.exit(0);
25   } catch (error) {
26     console.log(error);
27     process.exit(1);
28   }
29 }
30
31 runMain();
```

```
~/Documents/github/cool-domains npx hardhat run scripts/run.js
Compiled 1 Solidity file successfully
Yo yo, I am a contract and I am smart
Contract deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3
Contract deployed by: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266 has registered a domain!
Owner of domain doom: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Error: Transaction reverted without a reason string
    at Domains.setRecord (contracts/Domains.sol:31)
    at HardhatNode._mineBlockWithPendingTxs (/Users/sigridjin.eth/Documents/github/cool-domains/node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1773:23)
    at HardhatNode.mineBlock (/Users/sigridjin.eth/Documents/github/cool-domains/node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:466:16)
    at EthModule._sendTransactionAndReturnHash (/Users/sigridjin.eth/Documents/github/cool-domains/node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1504:18)
    at HardhatNetworkProvider.request (/Users/sigridjin.eth/Documents/github/cool-domains/node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:118:18)
```

contracts > ◆ Domains.sol

```

1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.10;
3
4 // Don't forget to add this import
5 import { StringUtils } from "./libraries(StringUtils.sol";
6 import "hardhat/console.sol";
7
8 contract Domains {
9     // Here's our domain TLD!
10    string public tld; // 뒤에 .ninja 라고 붙일 TLD를 설정한다.
11
12    mapping(string => address) public domains;
13    mapping(string => string) public records;
14
15    // We make the contract "payable" by adding this to the constructor
16    constructor(string memory _tld) payable {
17        tld = _tld; // payable 추가하면 컨트랙트가 MATIC을 받을 수 있다.
18        console.log("%s name service deployed", _tld);
19    }
20
21    // This function will give us the price of a domain based on length
22    function price(string calldata name) public pure returns(uint) { // pure 함수는 순수함수이므로 값을 수정할 수 없다. 가격을 예측해줄 뿐이다.
23        uint len = StringUtils.strlen(name);
24        require(len > 0);
25        if (len == 3) {
26            return 5 * 10**17; // 5 MATIC = 5 000 000 000 000 000 (18 decimals). We're going with 0.5 Matic cause the faucets don't give
27        } else if (len == 4) {
28            return 3 * 10**17; // To charge smaller amounts, reduce the decimals. This is 0.3
29        } else {
30            return 1 * 10**17; // 도메인의 길이가 짧을 수록 더 많은 MATIC을 지불해야 하도록 구현했다.
31        }
32    }
33
34    function register(string calldata name) public payable{
35        require(domains[name] == address(0));
36
37        uint _price = price(name);
38
39        // Check if enough Matic was paid in the transaction
40        require(msg.value >= _price, "Not enough Matic paid");
41
42        domains[name] = msg.sender;
43        console.log("%s has registered a domain!", msg.sender);
44    }
45
46    // Other functions unchanged
47 }
```

scripts > JS run.js > [ej] runMain

```
1  const main = async () => {
2    const domainContractFactory = await hre.ethers.getContractFactory('Domains');
3    // We pass in "ninja" to the constructor when deploying
4    const domainContract = await domainContractFactory.deploy("ninja");
5    await domainContract.deployed();
6
7    console.log("Contract deployed to:", domainContract.address);
8
9    // We're passing in a second variable - value. This is the moneyyyyyyyyyy
10   let txn = await domainContract.register("mortal", {value: hre.ethers.utils.parseEther('0.1')});
11   await txn.wait(); // parseEther: 0.1 MATIC을 보낸다. (Ether라고 하지만 이건 이더리움 기준으로 되어 있어서 그런 것이다)
12
13  const address = await domainContract.getAddress("mortal");
14  console.log("Owner of domain mortal:", address);
15
16  const balance = await hre.ethers.provider.getBalance(domainContract.address);
17  console.log("Contract balance:", hre.ethers.utils.formatEther(balance));
18 }
19
20 const runMain = async () => {
21   try {
22     await main();
23     process.exit(0);
24   } catch (error) {
25     console.log(error);
26     process.exit(1);
27   }
28 };
29
30 runMain();
```

2



jin.sigrid

Sigrid Naming Service V3

jin.sigrid

Owned by **you** 2 favorites

Price History

All Time

No item activity yet

Description

Created by **you**
A domain on the Ninja name service

About Sigrid Naming Service V3

Details

Listings

Offers

◆ Domains.sol 1

JS hardhat.config.js

JS run.js

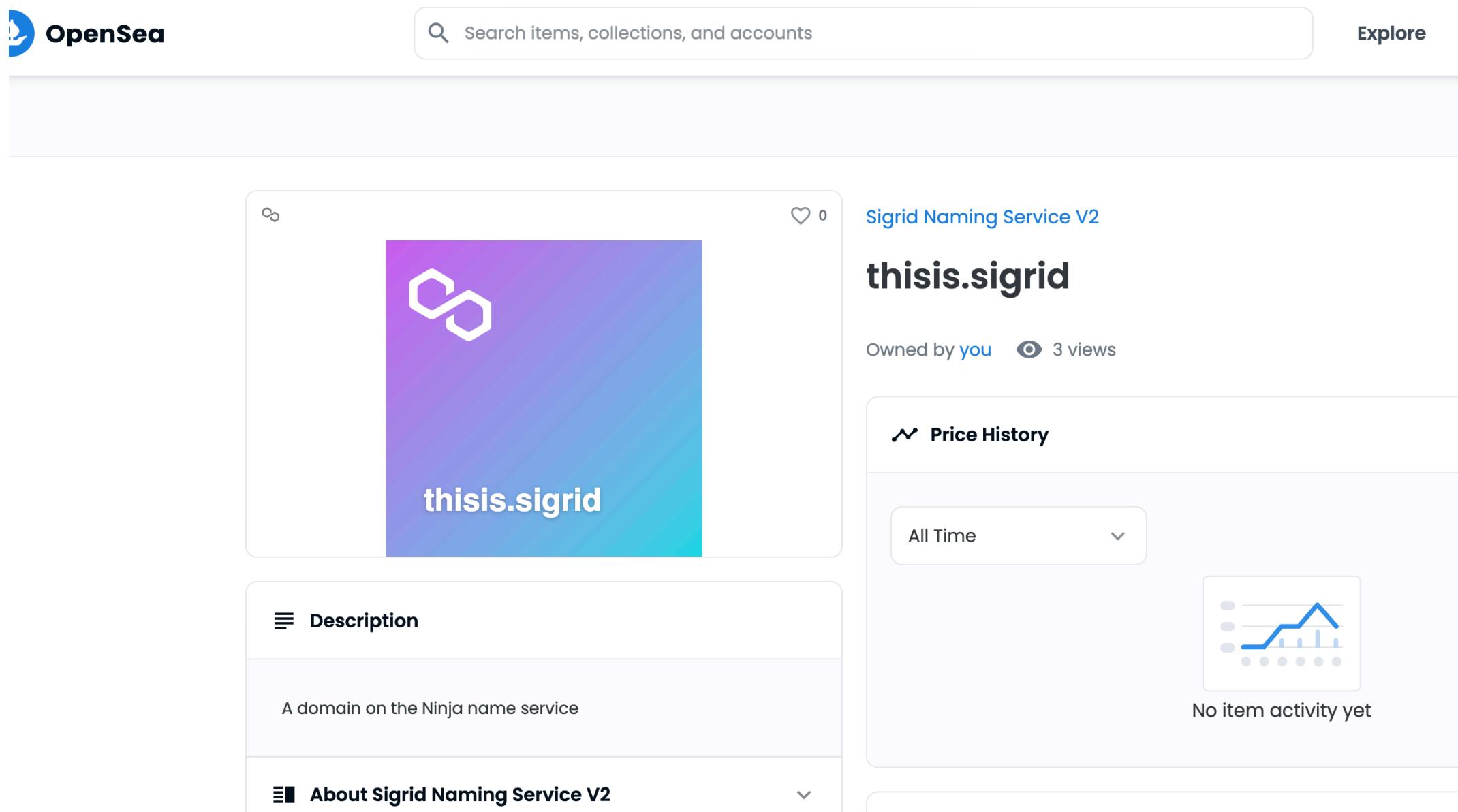
◆ StringUtils.sol ×

contracts > libraries > ◆ StringUtils.sol

```
1 // StringUtils.sol
2 // SPDX-License-Identifier: MIT
3 // Source:
4 // https://github.com/ensdomains/ens-contracts/blob/master/contracts/ethregistrar\(StringUtils.sol
5 pragma solidity >=0.8.4;
6
7 library StringUtils {
8     /**
9      * @dev Returns the length of a given string
10     *
11     * @param s The string to measure the length of
12     * @return The length of the input string
13     */
14     function strlen(string memory s) internal pure returns (uint) {
15         uint len;
16         uint i = 0;
17         uint bytelength = bytes(s).length;
18         for(len = 0; i < bytelength; len++) {
19             bytes1 b = bytes(s)[i];
20             if(b < 0x80) {
21                 i += 1;
22             } else if (b < 0xE0) {
23                 i += 2;
24             } else if (b < 0xF0) {
25                 i += 3;
26             } else if (b < 0xF8) {
27                 i += 4;
28             } else if (b < 0xFC) {
29                 i += 5;
30             } else {
31                 i += 6;
32             }
33         }
34         return len;
35     }
36 }
```

<https://gist.github.com/AlmostEfficient/669ac250214f30347097a1aeedcdfa12>

```
contract balance: 0.1
~/Documents/github/cool-domains ➔ npx hardhat run scripts/run.js
ninja name service deployed
Contract deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3
0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266 has registered a domain!
Owner of domain mortal: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Contract balance: 0.1
```



- Use a contract from OpenZeppelin to easily mint ERC721 tokens
- Create SVGs for our NFT and use on chain storage
- Setup token metadata (the data the NFT will hold)
- Mint it!

```

// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.10;

// We first import some OpenZeppelin Contracts.
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

import {StringUtils} from "./libraries(StringUtils.sol";
// We import another help function
import {Base64} from "./libraries/Base64.sol";

import "hardhat/console.sol";

// We inherit the contract we imported. This means we'll have access
// to the inherited contract's methods.
contract Domains is ERC721URIStorage {
    // Magic given to us by OpenZeppelin to help us keep track of tokenIds.
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    string public tld;

    // We'll be storing our NFT images on chain as SVGs
    string svgPartOne = '<svg xmlns="http://www.w3.org/2000/svg" width="270" height="270" fill="none">';
    string svgPartTwo = '</text></svg>';

    mapping(string => address) public domains;
    mapping(string => string) public records;

    constructor(string memory _tld) payable ERC721("Ninja Name Service", "NNS") {
        tld = _tld;
        console.log("%s name service deployed", _tld);
    }

    function register(string calldata name) public payable {
        require(domains[name] == address(0));

        uint256 _price = price(name);
        require(msg.value >= _price, "Not enough Matic paid");

        // Combine the name passed into the function with the TLD
        string memory _name = string(abi.encodePacked(name, ".", tld));
        // Create the SVG (image) for the NFT with the name
        string memory finalSvg = string(abi.encodePacked(svgPartOne, _name, svgPartTwo));
        uint256 newRecordId = _tokenIds.current();
        uint256 length = StringUtils.strlen(name);
        string memory strLen = Strings.toString(length);

        console.log("Registering %s.%s on the contract with tokenID %d", name, tld, newRecordId);

        // Create the JSON metadata of our NFT. We do this by combining strings and encoding as base64
        string memory json = Base64.encode(
            abi.encodePacked(
                '{"name": "",',
                '_name',
                '", "description": "A domain on the Ninja name service", "image": "data:image/svg+xml;base64,',
                Base64.encode(bytes(finalSvg)),
                '", "length": "' ,
                strLen,
                '"}'
            )
        );
    }
}

```

```

uint256 newRecordId = _tokenIds.current();
uint256 length = StringUtils.strlen(name);
string memory strLen = Strings.toString(length);

console.log("Registering %s.%s on the contract with tokenID %d", name, tld, newRecordId);

// Create the JSON metadata of our NFT. We do this by combining strings and encoding as base64
string memory json = Base64.encode(
    abi.encodePacked(
        '{"name": "",',
        '_name',
        '", "description": "A domain on the Ninja name service", "image": "data:image/svg+xml;base64,',
        Base64.encode(bytes(finalSvg)),
        '", "length": "' ,
        strLen,
        '"}'
    )
);

string memory finalTokenUri = string( abi.encodePacked("data:application/json;base64," , json));

console.log("\n-----");
console.log("Final tokenURI", finalTokenUri);
console.log("-----\n");

_safeMint(msg.sender, newRecordId);
_setTokenURI(newRecordId, finalTokenUri);
domains[name] = msg.sender;

_tokenIds.increment();
}

function getAddress(string calldata name) public view returns (address) {
    // Check that the owner is the transaction sender
    return domains[name];
}

function setRecord(string calldata name, string calldata record) public {
    // Check that the owner is the transaction sender
    require(domains[name] == msg.sender);
    records[name] = record;
}

function getRecord(string calldata name) public view returns(string memory) {
    return records[name];
}

```

<https://gist.github.com/farzaa/f13f5d9bda13af68cc96b54851345832>

<https://gist.github.com/jyptthemiracle/475d743e39f2c14a5549520579f1e4a2>

UzQyTVRaakxTNHpPRFF0TgPZMk5TMHV0VGswTFRFdU5ERTRMUzQyTURndE1pNHhPRGQyTFRrdU16RmpMUzR3TVRNdExqYzN0UzR4T0RVdE1TNDFNemd1TlRjeUxUSXVNakE0WVRRdU1qVwd0QzR5TlNBd0lEQWdNU0F4TGpZeU5TMHhMa1U1Tld3M0xqZzR0QzAwTgPVNVl5NDJ0am0d0TgPNNE550XhMa1F5TmkwdU5Ua2dNaTR4T1RjdExqVTVjekV1TlRJNUxqSXd0Q0F5TgPFNU55NDFPV3czTgprNE5DQTBMa1U1WVRRdU5USwd0QzQxTw1Bd0lEQWdNU0F4TGpVNE9TQXhMall4Tm1NdU16ZzBMa1kyTlM0MU9UUwdNUzQwTVRndU5qQTRJRE1lTVRnM2RqWVNRE15YkRZdU9EVXR0QzR3TmpWmxUwVNRE15Wxk0d01UTXRManMzTlMwdU1UzzFMVEV1TlRNNExtNDF0ekl0Twk0eU1EaGh0QzR5TlNBMEqxSTFJREFnTUNBd0xURXV0akkxTFRFdU5UazFURFF4TGpRMU5pQX10QzQxT1dNdExqWTJPQzB1TXpnM0xURXv0REkyTFM0MU9TMHlMakU1TnkwdU5UbHpmVEV1TlRJNUxqSXd0QzB5TgPFNU55NDFPV3d0TVRRdU9EWtBJRGd1TmpVMVlUUXVNalVnTkM0eU5TQxdJREFnTUMweExqWx10U0F4TGpVNU5XTXRmak00Tnk0Mk55MHV0VGcxSURFdU5ETTBmuZQxTnpJZ01pNHlNRGgyTVRjdU5EUxhZeTB1TURFekxqYzN0UzR4T0RVZ01TNDFNemd1TlRjeU1ESXVNakE0WVRRdU1qVwd0QzR5TlNBd0lEQWdNQ0F4TGpZeU5TQXhMa1U1Tld3eE5DNDR0a1FnT0M0Mk5UVmpMa1kyT0M0ek9EY2dNUzQwTwpZdU5Ua2dNaTR4T1RjdU5UbHpuNUzQxTwprdExqSXd0Q0F5TgPFNU55MHV0VGxzTVRBdU1EZ3hMVFV1T1RBeE1EWXVPRFV0TkM0d05qVwdNVEF1TURneExUVXVPVF4Wxk0Mk5qZ3RMak00TnlBeExqUX10aTB1TlRrZ01pNHhPVGN0TgPVNNW6RV0VEk1TgPjd05DQX1MakU1Tnk0MU9XzdNMamc0TkNBMEqxVTvZVFF1TlRjZ05DNDfNaUF3SURBZ01TQXhMa1U0T1NBeExqWxh0bU11TxpnMExqWtJ0UzQxT1RRZ01TNDBNVGd1TmpBNE1ESXVNvgczGprdU16RxhZeTR3TVRNdU56YzFMUzR4T0RVZ01TNDFNemd0TgPVM01pQXlMakl3T0dFMEqxSTFJRF1TwpVZ01DQxdJRE0TVM0Mk1qVwdNUzQxT1Rwc0xUY3VPRGcwSURRdU56SXhZeTB1TmpZNEqxTTR0eTB4TGpReU5pNDfpUzB5TgPFNU55NDFPWE10TVM0MU1qa3RMakl3TkMweUxqRTV0eTB1TlRsc0xUY3VRGcwTFRDdU5UbGh0QzQxTw1BMEqxVx1JREFnTUNBeExURXv0VGc1TFRFdU5qRTJZeTB1TxpnMuXTNDJ0a1V0TgPVNU5DMHhMa1F4T0MwdU5qQTRMVE11TVRnM2RpMDJMakF6Tw13dE5pNDROU0EwTgpbMk5YWTJmakF6Tw1NdExqQXhNeTqzTnpVdU1UzzFjREV1TlRNNEqxVTNNaUF5TgPjd09HRTBmakkxSURRdU1qVwdNQ0F3SURBZ01TNDJNa1VnTVM0MU9UVnNNVFF1T0RzMe1EZ3V0a1UxWxk0Mk5qZ3VNemczSURFdU5ESTJMa1U1SURJdU1UazNMa1U1Y3pFdU5USTVMUzR5TURRZ01pNHhPVGN0TgPVNWJERTBMapcyTkMwNExqWtFOV011TmpVm0xTNhpPVFFnTVM0eU1EURMamsxSURFdU5UzzVmveV1TmpFmmN5NDfpVFF0TVM0ME1Uz3V0akE1TFRJdU1UzzNw1UxTgVek9HTXvnREV6Tfm0M056VxrMakU0TlMweExqVxpQzB1TlRjeUxUSXVNakE0WVRRdU1qVwd0QzR5TlNBd0lEQWdNQzB4TgPzeU5TMHhMa1U1Tld3dE1UUXVPVGt6TFRndU56ZzJlaUlnWm1sc2JEMglJmlptWmlJdlBqeGtaV1p6UGp4c2FXNwxZWEpIY21Ga2FXVnVkQ0JwWkQwaVFpSwd1REU5SwpBaU1Ia3hQu0l3SwLCNE1qMg1NamN3Sw1CNU1qMg1NamN3Sw1CbmNtRmthV1Z1ZEZwdFYUnpQU0oxYzJWeVuZQmhZM1ZQYmxWelpTSStQSE4wYjNBZ2MzUnZjQzFqYjJ4dmNqMg1Jmk5pTldWbFpTSXZQanh6ZEc5d0lH0W1abk5sZEQwaU1TSwdjM1J2Y0MxmIyeHzjajBpSxpcalpEZGx0Q0lNyZNSdmNDMXZjR0ZqYvhSNVBTSXVPVGtpTHo00EwyebibVzoY2tkeV1XUnBaVzUwUGp3dlpHVm1jejQ4ZedwNGRDQjRQU016Twk0Mu1pQjVQU015TxpFaUlHwnZiblF0YzjsNlpUMG1NamNpSUdacGJHdzlJaU5tWm1ZaU1HwnBiSFJsY2owaWRYsNlQ05CS1NJZ1pt0XVkbzFtWvcxcGJIazljbEJzZfhNZ1NtRnJZWEowwVNCVFlXNXpMRVJsYw1GV2RTQ1RZvzV6TEU1dmRHOGdRMj1zYjNjZ1JXMXZhbWtzUVhCd2JhvWdRMj1zYjNjZ1JXMXZhbWtzYzJGdwN5MxpaWEpwWmlJZ1pt0XVkQzEzWldsmbFIUTlJbUp2YkdRaVbtMXZjb1joYkM1dWFNXFZVhd2ZEdwNGRENdhMM04yWho0PSIsImx1bmd0aCI6IjYifQ==

Owner of domain mortal: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Contract balance: 0.1

```

const main = async () => {
    const domainContractFactory = await hre.ethers.getContractFactory('Domains');
    const domainContract = await domainContractFactory.deploy("ninja");
    await domainContract.deployed();

    console.log("Contract deployed to:", domainContract.address);

    // CHANGE THIS DOMAIN TO SOMETHING ELSE! I don't want to see OpenSea full of bananas lol
    let txn = await domainContract.register("banana", {value: hre.ethers.utils.parseEther('0.1')});
    await txn.wait();
    console.log("Minted domain banana.ninja");

    txn = await domainContract.setRecord("banana", "Am I a banana or a ninja??");
    await txn.wait();
    console.log("Set record for banana.ninja");

    const address = await domainContract.getAddress("banana");
    console.log("Owner of domain banana:", address);

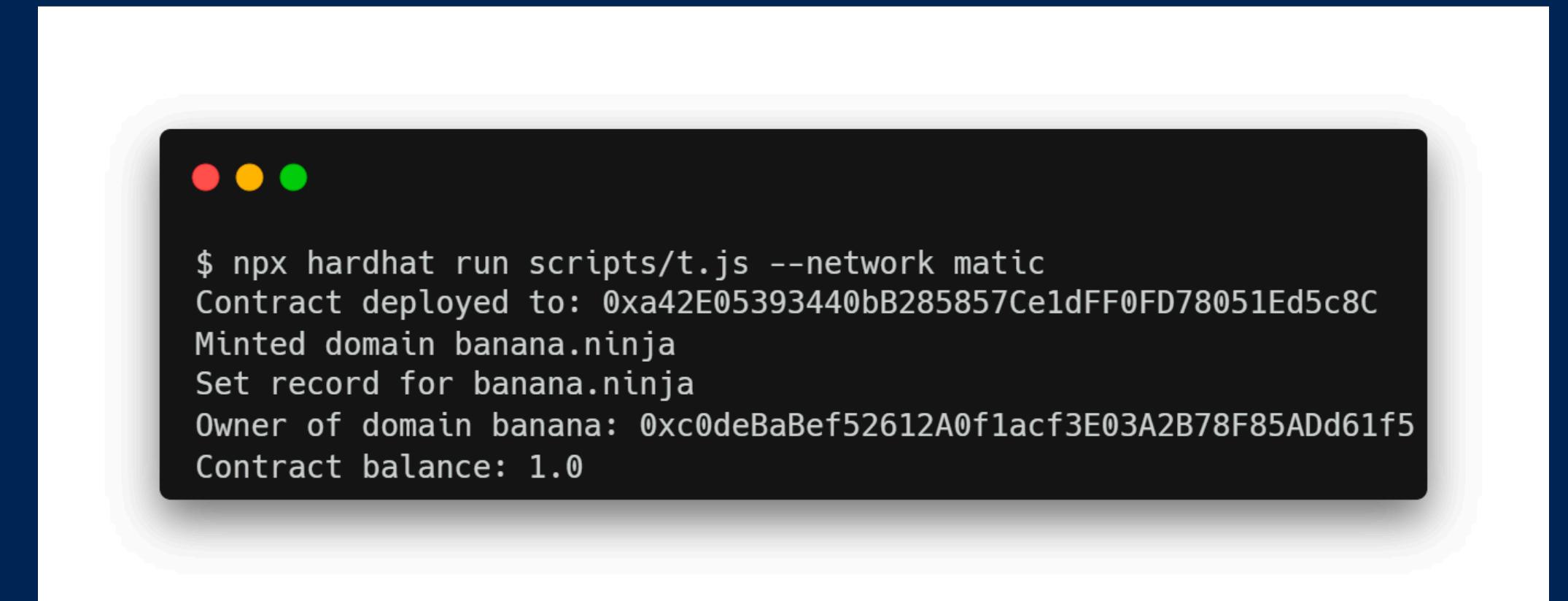
    const balance = await hre.ethers.provider.getBalance(domainContract.address);
    console.log("Contract balance:", hre.ethers.utils.formatEther(balance));
}

const runMain = async () => {
    try {
        await main();
        process.exit(0);
    } catch (error) {
        console.log(error);
        process.exit(1);
    }
};

runMain();

```

<https://docs.allthatnode.com/>



```

$ npx hardhat run scripts/t.js --network matic
Contract deployed to: 0xa42E05393440bB285857Ce1dFF0FD78051Ed5c8C
Minted domain banana.ninja
Set record for banana.ninja
Owner of domain banana: 0xc0deBaBef52612A0f1acf3E03A2B78F85ADd61f5
Contract balance: 1.0

```

Adverb Name Service

The most modified API on the blockchain!



Wallet: 0x39d9...9359

domain

.ly

whats ur ninja power?

Mint

Recently minted domains!

surprising.ly

smart.ly
eye bee S M R T



built with @_buildspace

<https://github.com/cmdeLLinger/polygon-naming-service>