Hands-on Session: Build Polygon ENS

Polygon 네트워크에서 도메인 서비스 만들어보기

2022-06-22 Smiling Leo Web3 Study

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

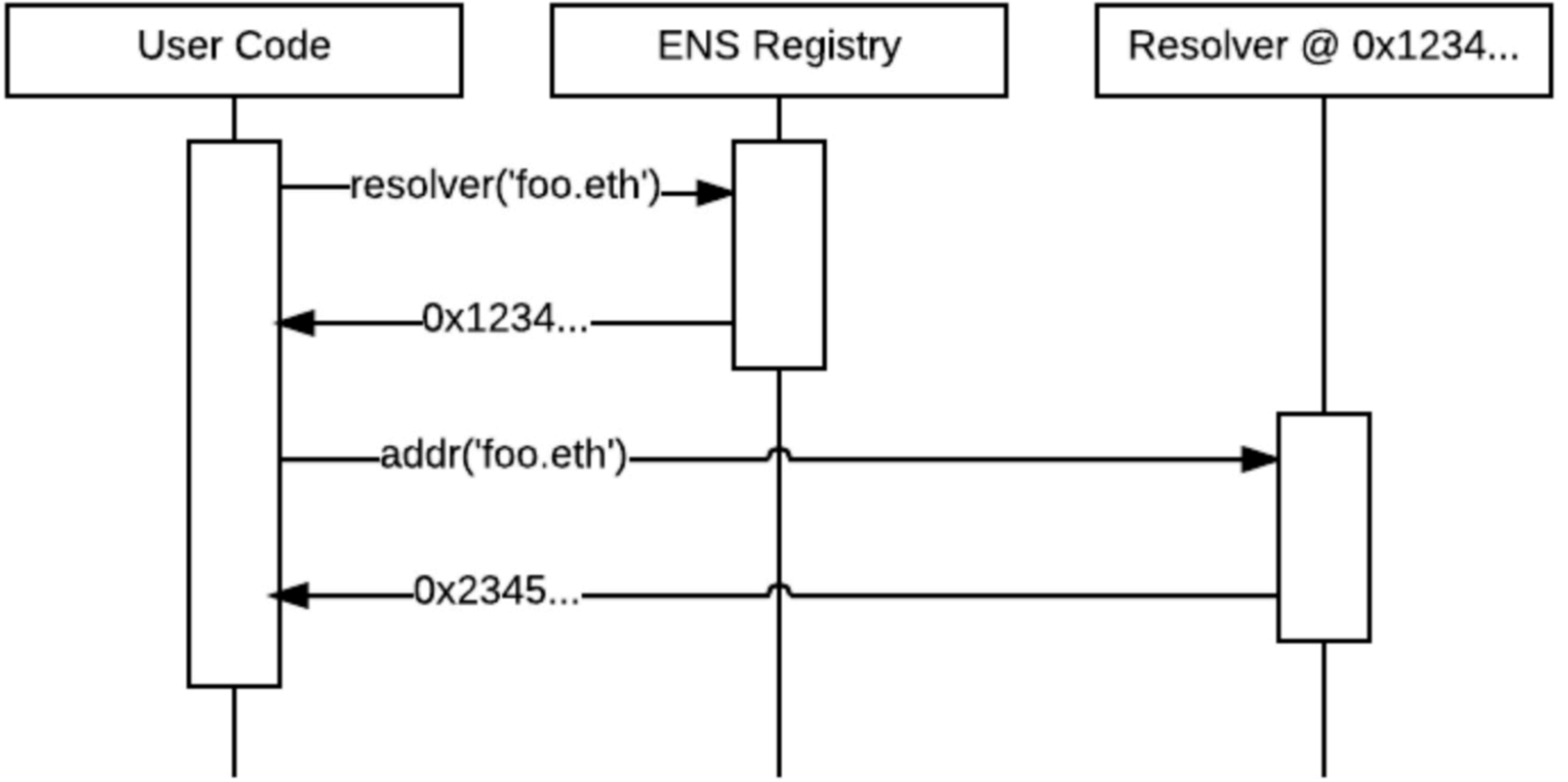
bash

```shell
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

```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)



https://www.mk.co.kr/news/economy/view/2019/08/658067/

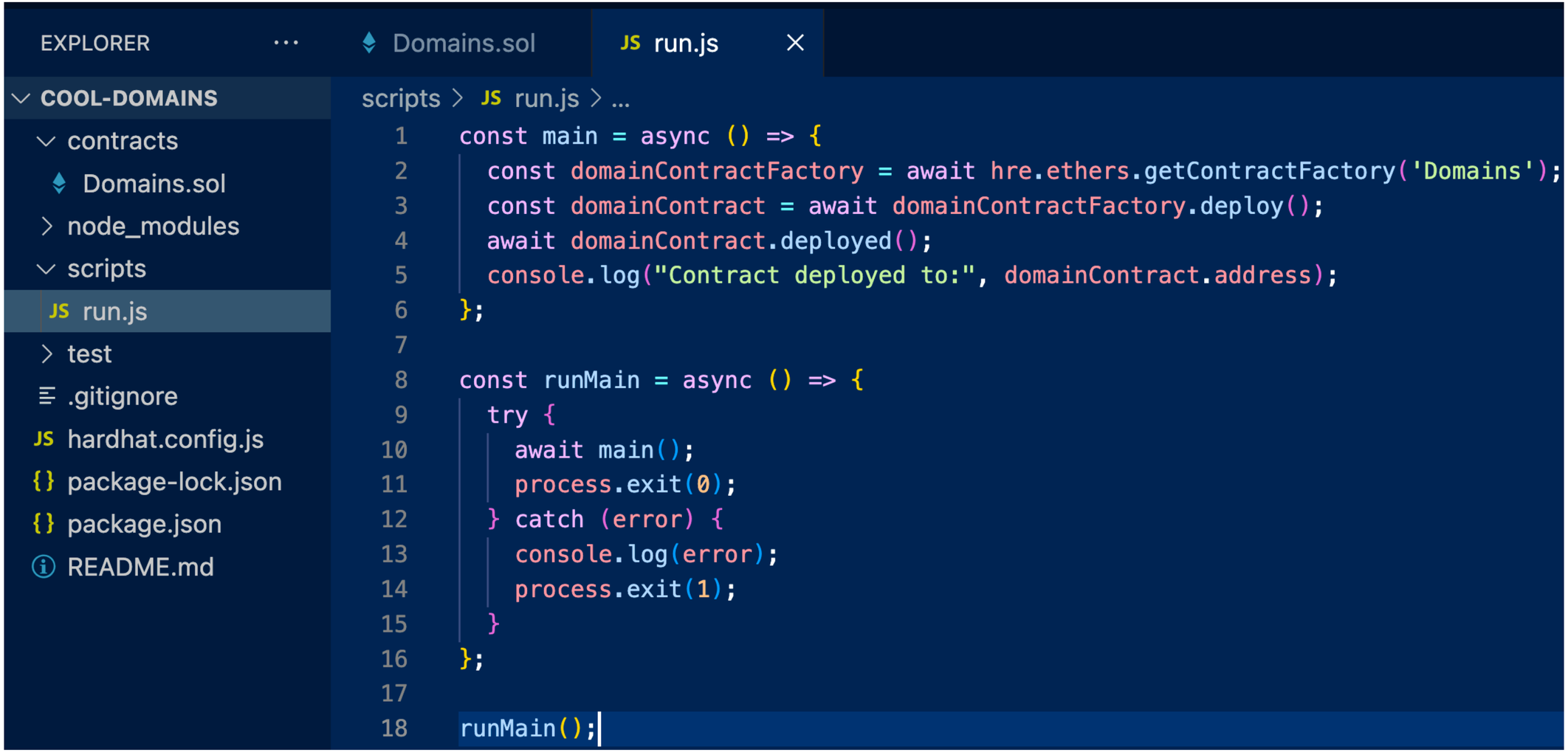
Domains.sol

contracts > Domains.sol

```sol
// SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.8.10;

import "hardhat/console.sol";

contract Domains {
  constructor() {
    console.log("THIS IS MY DOMAINS CONTRACT. NICE.");
  }
}
```

Smart Contract는 다른 언어에서의 Class와 유사한 개념을 갖는다.

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



배포를 위한 스크립트를 작성한다.

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

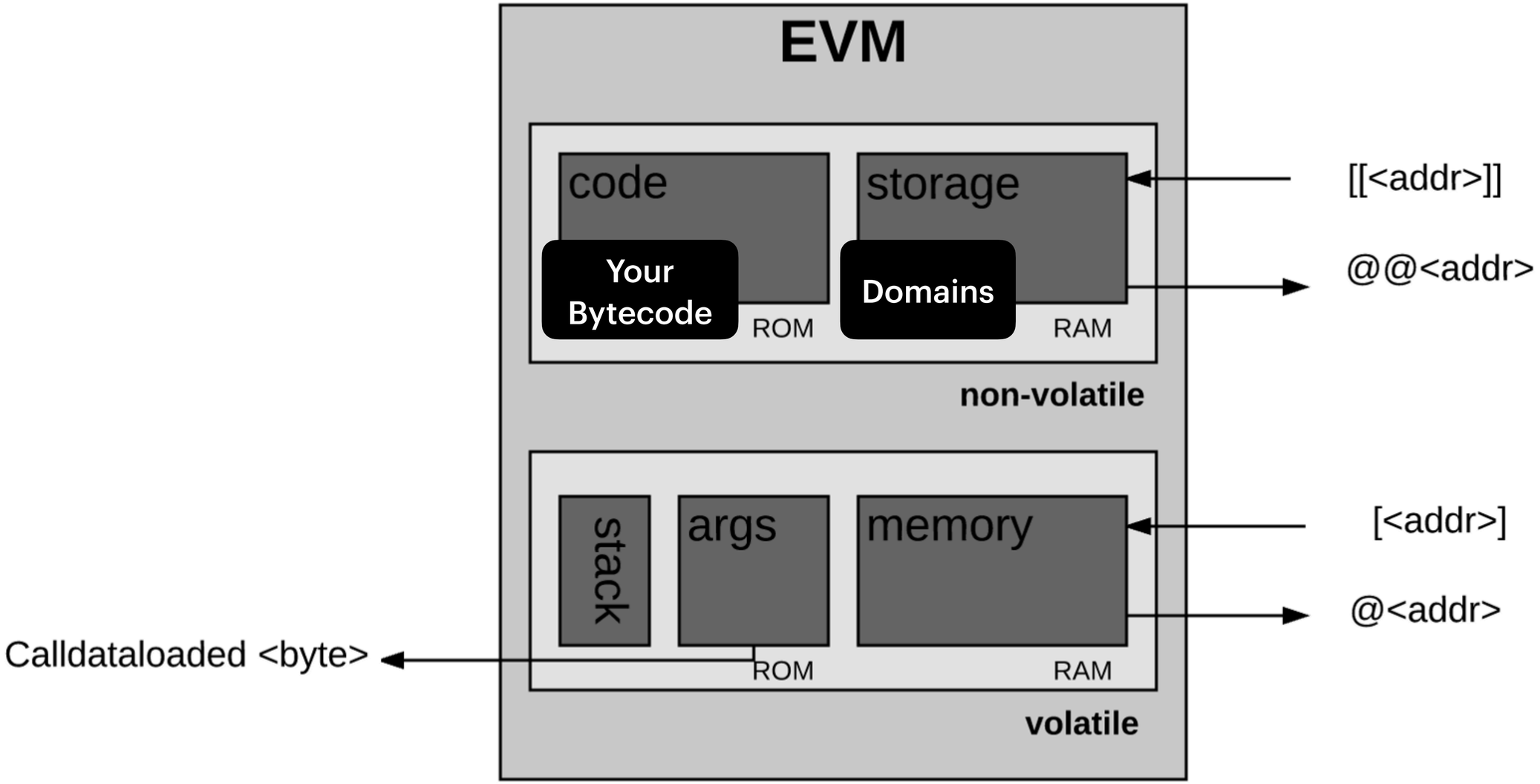
Deploy를 시작하고, 블록체인에 mine 될 때까지 기다리도록 deployed()로 await 한다.

Domains.sol | run.js

contracts > Domains.sol

```sol
// SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.8.10;

import "hardhat/console.sol";

contract Domains {
  // A "mapping" data type to store their names
  mapping(string => address) public domains; // state variable: 영구히 블록체인에 storage 타입으로 저장되는 타입
                                             // domains는 도메인의 소유자를 의미함 ex) sigridjin.eth의 소유자는 진형의 지갑이다.
  constructor() {
    console.log("THIS IS MY DOMAIN CONTRACT. NICE.");
  }

  // A register function that adds their names to our mapping
  function register(string calldata name) public {  // calldata: final 키워드를 붙인 파라미터와 동일
    domains[name] = msg.sender;
    console.log("%s has registered a domain!", msg.sender);
  }

  // This will give us the domain owners' address
  function getAddress(string calldata name) public view returns (address) {
    return domains[name];
  }
}
```

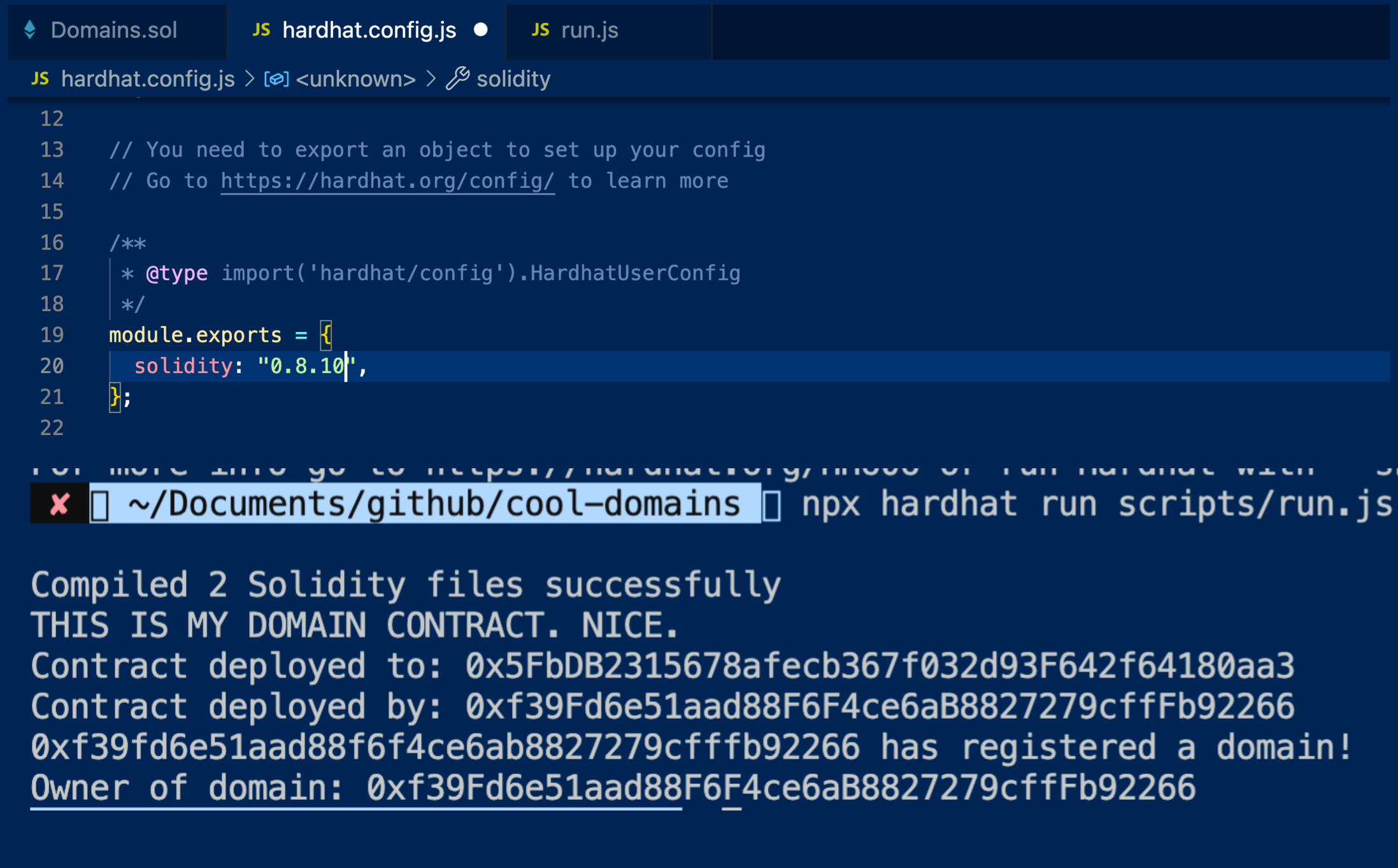
https://gist.github.com/jypthemiracle/475d743e39f2c14a5549520579f1e4a2


EVM
code
storage
Your Bytecode
Domains
ROM
RAM
non-volatile
stack
args
memory
ROM
RAM
volatile
[[<addr>]]
@@<addr>
[<addr>]
@<addr>
Calldataloaded <byte>

Domains.sol | run.js

scripts > run.js > ...

```
const main = async () => {                    // config에 명시된 지갑 주소를 가져온다. 로컬이라면, 랜덤으로 만든다.
  const [owner, randomPerson] = await hre.ethers.getSigners();
  const domainContractFactory = await hre.ethers.getContractFactory('Domains');
  const domainContract = await domainContractFactory.deploy();
  await domainContract.deployed();
  console.log("Contract deployed to:", domainContract.address);
  console.log("Contract deployed by:", owner.address);

  const txn = await domainContract.register("doom");
  await txn.wait();

  const domainOwner = await domainContract.getAddress("doom");
  console.log("Owner of domain:", domainOwner);
}

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

runMain();
```


Domains.sol
hardhat.config.js
run.js
hardhat.config.js > <unknown> > solidity
// You need to export an object to set up your config
// Go to https://hardhat.org/config/ to learn more
/**
* @type import('hardhat/config').HardhatUserConfig
*/
module.exports = {
solidity: "0.8.10",
};
~/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

Domains.sol | hardhat.config.js | run.js

contracts > Domains.sol

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

import "hardhat/console.sol";

contract Domains {
    mapping(string => address) public domains; // domains는 도메인의 소유자를 의미함 ex) sigridjin.eth의 소유자는 진형의 지갑이다.

    // Checkout our new mapping! This will store values
    mapping(string => string) public records;

    constructor() {
        console.log("Yo yo, I am a contract and I am smart");
    }

    function register(string calldata name) public {
        // Check that the name is unregistered (explained in notes)
        require(domains[name] == address(0));
        domains[name] = msg.sender;
        console.log("%s has registered a domain!", msg.sender);
    }


    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];
    }
}
```

// records는 domain을 어디로 연결할지를 알려주는 담는 문자열 값이다.
ex) sigridjin.eth는 0xABCDE.. 로 연결하거나, IP 주소로 연결할 수도 있다.

Domains.sol | hardhat.config.js | run.js

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-netw
ork/provider/modules/eth.ts:1504:18)
    at HardhatNetworkProvider.request (/Users/sigridjin.eth/Documents/github/cool-domains/node_modules/hardhat/src/internal/hardhat-network/provi
der/provider.ts:118:18)
```

Domains.sol 1 | hardhat.config.js | run.js

contracts > Domains.sol

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

// Don't forget to add this import
import { StringUtils } from "./libraries/StringUtils.sol";
import "hardhat/console.sol";

contract Domains {
  // Here's our domain TLD!
  string public tld;  // 뒤에 .ninja 라고 붙일 TLD를 설정한다.

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

  // We make the contract "payable" by adding this to the constructor
  constructor(string memory _tld) payable {
    tld = _tld;                    // payable 추가하면 컨트랙트가 MATIC을 받을 수 있다.
    console.log("%s name service deployed", _tld);
  }

  // This function will give us the price of a domain based on length
  function price(string calldata name) public pure returns(uint) {  // pure 함수는 순수함수이므로 값을 수정할 수 없다. 가격을 예측해줄 뿐이다.
    uint len = StringUtils.strlen(name);
    require(len > 0);
    if (len == 3) {
      return 5 * 10**17; // 5 MATIC = 5 000 000 000 000 000 000 (18 decimals). We're going with 0.5 Matic cause the faucets don't give
    } else if (len == 4) {
      return 3 * 10**17; // To charge smaller amounts, reduce the decimals. This is 0.3
    } else {
      return 1 * 10**17;     // 도메인의 길이가 짧을 수록 더 많은 MATIC을 지불해야 하도록 구현했다.
    }
  }

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

    uint _price = price(name);

    // Check if enough Matic was paid in the transaction
    require(msg.value >= _price, "Not enough Matic paid");

    domains[name] = msg.sender;
    console.log("%s has registered a domain!", msg.sender);
  }
  // Other functions unchanged
}
```

https://gist.github.com/jypthemiracle/475d743e39f2c14a5549520579f1e4a2

Domains.sol 1 | hardhat.config.js | run.js

scripts > run.js > 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

Description

Created by you

A domain on the Ninja name service

About Sigrid Naming Service V3

Details

Sigrid Naming Service V3

jin.sigrid

Owned by you 2 favorites

Price History

All Time

No item activity yet

Listings

Offers

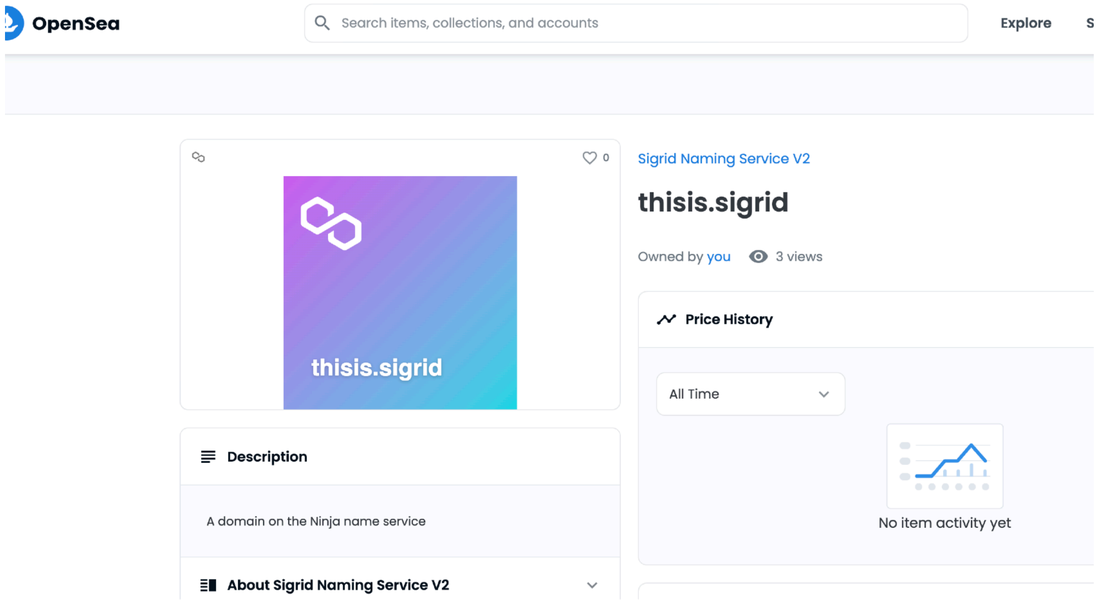
Domains.sol 1 | hardhat.config.js | run.js | StringUtils.sol

contracts > libraries > StringUtils.sol

```sol
// StringUtils.sol
// SPDX-License-Identifier: MIT
// Source:
// https://github.com/ensdomains/ens-contracts/blob/master/contracts/ethregistrar/StringUtils.sol
pragma solidity >=0.8.4;

library StringUtils {
    /**
     * @dev Returns the length of a given string
     *
     * @param s The string to measure the length of
     * @return The length of the input string
     */
    function strlen(string memory s) internal pure returns (uint) {
        uint len;
        uint i = 0;
        uint bytelength = bytes(s).length;
        for(len = 0; i < bytelength; len++) {
            bytes1 b = bytes(s)[i];
            if(b < 0x80) {
                i += 1;
            } else if (b < 0xE0) {
                i += 2;
            } else if (b < 0xF0) {
                i += 3;
            } else if (b < 0xF8) {
                i += 4;
            } else if (b < 0xFC) {
                i += 5;
            } else {
                i += 6;
            }
        }
        return len;
    }
}
```

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

```
~/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!

```solidity
// 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": "',
        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/jypthemiracle/475d743e39f2c14a5549520579f1e4a2

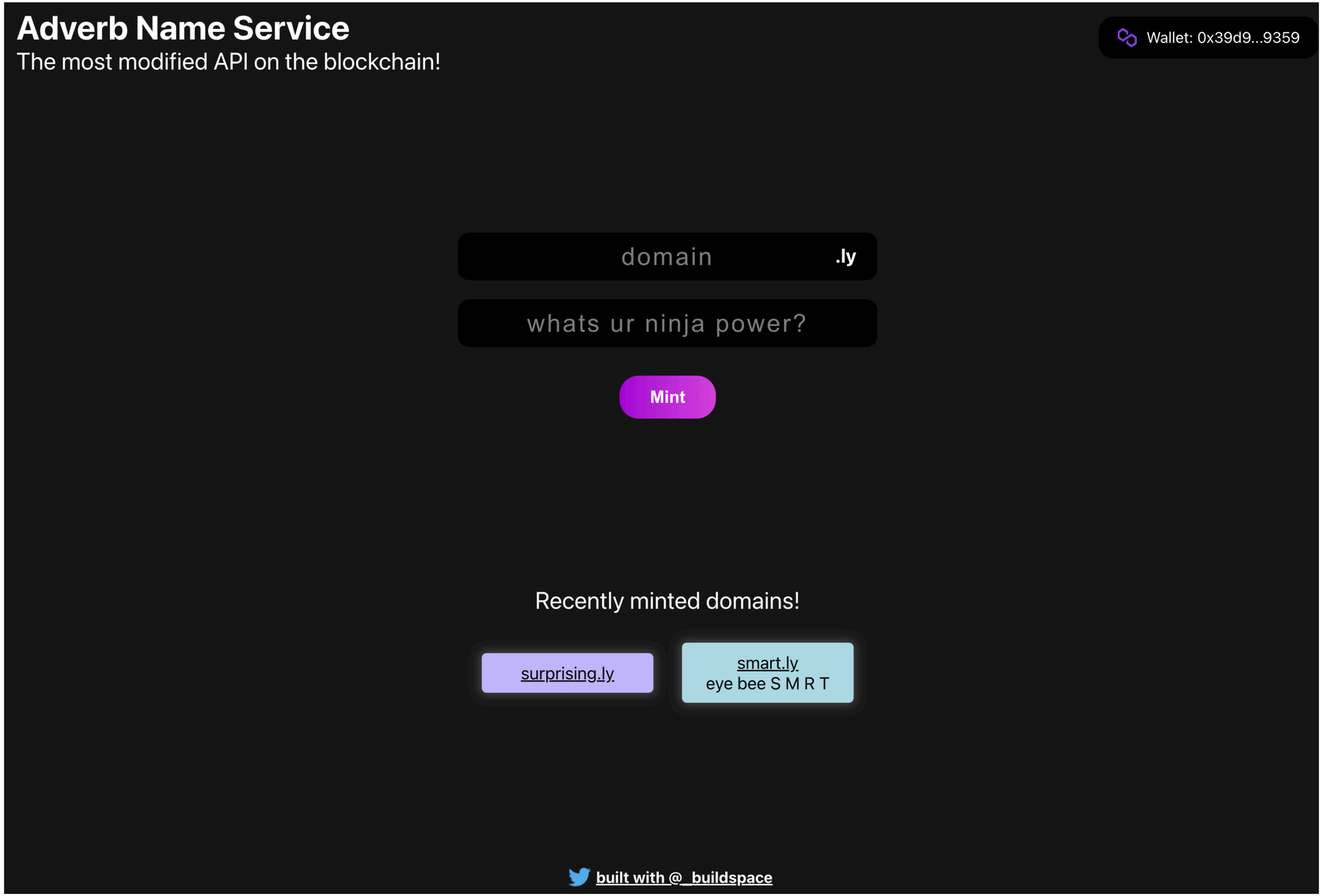
```
UzQyTVRaakxTNHpPRFF0TGpZMk5TMHVOVGswTFRFdU5ERTRMUzQyTURndE1pNHhPRGQyTFRrdU16RmpMUzR3TVRNdExqYzNOUzR4T0RVdE1TNDFNemd1TlRjeUxUSXVNakE0WVRRdU1qVWdOQzR5TlNBd0lEQWdNU0F4TGpZeU5TMHhMalU1Tld3M0xqZzRO
QzAwTGpVNVl5NDJOamd0TGpNNE55QXhMalF5TmkwdU5Ua2dNaTR4T1RjdExqVTVjekV1TlRJNUxqSXdOQ0F5TGpFNU55NDFPV3czTGpnNE5DQTBMalU1WVRRdU5USWdOQzQxTWlBd0lEQWdNU0F4TGpVNE9TQXhMall4Tm1NdU16ZzBMalkyTlM0MU9UUWdN
UzQwTVRndU5qQTRJREl1TVRnM2RqWXVNRE15YkRZdU9EVXROQzR3TmpWMkxUWXVNRE15WXk0d01UTXRMamMzTlMwdU1UZzFMVEV1TlRNNExTNDF0ekl0TWk0eU1EaGhOQzR5TlNBMExqSTFJREFnTUNBd0xURXVOakkxTFRFdU5UazFURFF4TGpRMU5pQXlO
QzQxT1dNdExqWTJPQzB1TXpnM0xURXVOREkyTFM0MU9TMHlMakU1TnkwdU5UbHpMVEV1TlRJNUxqSXdOQzB5TGpFNU55NDFPV3d0TVRRdU9EWTBJRGd1TmpVMVlUUXVNalVnTkM0eU5TQXdJREFnTUMweExqWXlOU0F4TGpVNU5XTXRMak00Tnk0Mk55MHVO
VGcxSURFdU5ETTBMUzQxTnpJZ01pNHlNRGgyTVRjdU5EUXhZeTB1TURFekxqYzNOUzR4T0RVZ01TNDFNemd1TlRjeUlESXVNakE0WVRRdU1qVWdOQzR5TlNBd0lEQWdNQ0F4TGpZeU5TQXhMalU1Tld3eE5DNDROalFnT0M0Mk5UVmpMalkyT0M0ek9EY2dN
UzQwTWpZdU5Ua2dNaTR4T1RjdU5UbHpNUzQxTWprdExqSXdOQ0F5TGpFNU55MHVOVGxzTVRBdU1EZ3hMVFV1T1RBeElEWXVPRFV0TkM0d05qVWdNVEF1TURneExUVXVPVEF4WXk0Mk5qZ3RMak00TnlBeExqUXlOaTB1TlRrZ01pNHhPVGN0TGpVNWN6RXVO
VEk1TGpJd05DQXlMakU1Tnk0MU9XdzNMamc0TkNBMExqVTVZVFF1TlRJZ05DNDFNaUF3SURBZ01TQXhMalU0T1NBeExqWXhObU11TXpnMExqWTJOUzQxT1RRZ01TNDBNVGd1TmpBNElESXVNVGczZGprdU16RXhZeTR3TVRNdU56YzFMUzR4T0RVZ01TNDFN
emd0TGpVM01pQXlMakl3T0dFMExqSTFJRFF1TWpVZ01DQXdJREV0TVM0Mk1qVWdNUzQxT1RWc0xUY3VPRGcwSURRdU56SXhZeTB1TmpZNExqTTROeTB4TGpReU5pNDFPUzB5TGpFNU55NDFPWE10TVM0MU1qa3RMakl3TkMweUxqRTVOeTB1TlRsc0xUY3VP
RGcwTFRRdU5UbGhOQzQxTWlBMExqVXlJREFnTUNBeExURXVOVGc1TFRFdU5qRTJZeTB1TXpnMUxTNDJOalV0TGpVNU5DMHhMalF4T0MwdU5qQTRMVEl1TVRnM2RpMDJMakF6TW13dE5pNDROU0EwTGpBMk5YWTJMakF6TW1NdExqQXhNeTQzTnpVdU1UZzFJ
REV1TlRNNExqVTNNaUF5TGpJd09HRTBMakkxSURRdU1qVWdNQ0F3SURBZ01TNDJNalVnTVM0MU9UVnNNVFF1T0RZMElEZ3VOalUxWXk0Mk5qZ3VNemczSURFdU5ESTJMalU1SURJdU1UazNMalU1Y3pFdU5USTVMUzR5TURRZ01pNHhPVGN0TGpVNWJERTBM
amcyTkMwNExqWTFOV011TmpVM0xTNHpPVFFnTVM0eU1EUXRMamsxSURFdU5UZzVMVEV1TmpFMmN5NDFPVFF0TVM0ME1UZ3VOakE1TFRJdU1UZzNWalUxTGpVek9HTXVNREV6TFM0M056VXRMakU0TlMweExqVXpPQzB1TlRjeUxUSXVNakE0WVRRdU1qVWdO
QzR5TlNBd0lEQWdNQzB4TGpZeU5TMHhMalU1Tld3dE1UUXVPVGt6TFRndU56ZzJlaUlnWm1sc2JEMGlJMlptWmlJdlBqeGtaV1p6UGp4c2FXNWxZWEpIY21Ga2FXVnVkQ0JwWkQwaU1TSWdjM1J2Y0MxamIyeHZjajBpSXpCalpEZGxOQ0lnYzNSdmNDMXZj
R0ZqYVhSNVBTSXVPVGtpTHo0OEwyeHBibVZoY2tkeVlXUnBaVzUwUGp3dlpHVm1jejQ4ZEdWNGRDQjRQU0l6TWk0MUlpQjVQU0l5TXpFaUlHWnZiblF0YzJsNlpUMGlNamNpSUdacGJHdzlJaU5tWm1ZaUlHWnBiSFJsY2owaWRYSnNLQ05CS1NJZ1ptOXVk
QzFtWVcxcGJIazlJbEJzZFhNZ1NtRnJZWEowWVNCVFlXNXpMRVJsYW1GV2RTQlRZVzV6TEU1dmRHOGdRMjlzYjNJZ1JXMXZhbWtzUVhCd2JHVWdRMjlzYjNJZ1JXMXZhbWtzYzJGdWN5MXpaWEpwWmlJZ1ptOXVkQzEzWldsbmFIUTlJbUp2YkdRaVBtMXZj
blJoYkM1dWFXNXFZVHd2ZEdWNGRENDhMM04yWno0PSIsImxlbmd0aCI6IjYifQ==
```

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
```



https://github.com/cmdellinger/polygon-naming-service

DNS project로 배우는 NFT 민팅 방법 (Frontend)

브랜치 설명 및 구현 요구사항

- step1 : Contract
  - 등록한 모든 도메인을 `string[]` 으로 반환하는 솔리디티 메소드를 만든다.
- step2 : frontend
  - 메타마스크 지갑이 연결되어 있는 지 체크하는 메소드를 만든다.
- step3 : frontend
  - 메타마스크 지갑을 연결하는 메소드를 만든다.
- step4 : frontend
  - 메타마스크 지갑이 연결되었을 때, 도메인 NFT를 민팅하는 메소드를 구현한다.
- step5 : frontend
  - Rinkeby Network에 연결되지 않았을 때, 연결하는 메소드를 구현한다.
- step6 : frontend
  - 도메인 NFT를 민팅하는 트랜잭션이 성공적으로 실행되었을 때, record를 설정하는 트랜잭션을 실행한다.
  - 민팅이 완료되면, 자동으로 새로고침한다. 새로 민팅한 NFT가 페이지에 뜬다.
- step7 : frontend
  - 도메인 NFT를 민팅한 주인이라면, record를 변경할 수 있다.

참고할 문서

- Metamask Documentation
- AllThatNode Tutorials
- ethers.js
- Ethereum Wiki
- Buildspace

https://github.com/Web3-Study-with-Sigrid-Jin/SOPT-web3-study/blob/main/week_6/lecture.md

지난 3주차 과제를 다시 보기: ERC721 Mint 프론트엔드 구성하기

- 과제 요구사항
  - 1단계: 이번 2주차에 올렸던 컨트랙트 요구사항을 구현하되, IPFS로 메타데이터를 업로드하도록 리팩토링한다.
  - 2단계: 현재 브라우저에 메타마스크 지갑이 설치되어 있는지 확인할 수 있다.
  - 3단계: 메타마스크 지갑이 설치되어 있다면, 웹 사이트에 메타마스크 지갑을 연결할 수 있다.
  - 4단계: Rinkeby Network에 연결되어 있지 않다면, 해당 네트워크로 연결 요청하는 메시지를 표시할 수 있다.
  - 5단계: `mint` 버튼을 클릭하면, 컨트랙트에서 연결된 지갑으로 이전 NFT ID 번호에서 값 1이 증가한 NFT를 발급받을 수 있다.
  - 6단계: fleek.co 에서 완성된 정적 페이지를 배포하고, 발급한 NFT Collection을 OpenSea Testnet에서 확인한다.
- 다음 저장소를 클론해서 프론트엔드 작업을 시작해주세요.
  - https://github.com/jypthemiracle/buildspace-nft-course-starter
- Tip) Provider는 Wallet 또는 ethers.js, web3.js와 같이 Ethereum 블록체인과 통신할 때 사용하는 API Node를 의미함
- Tip) Signer는 현재 import한 private key의 계정들 여러 개를 의미함. 이를 이용해 내가 “서명” 함으로서 내가 트랜잭션을 보낼 수 있음.