



블록체인과 스마트 컨트랙트 기초 실습

2024. 01. 26

최해웅 리버밴스(주)

haeungchoi.libervance@gmail.com , haeung@gist.ac.kr

CONTENTS


01. 블록체인 소개

02. 스마트 컨트랙트, dApp 소개

03. Token 소개

04. ERC20 토큰 작성 및 배포 실습 (1)

05. NFT 작성 및 배포 실습 (2)



01. 블록체인 소개

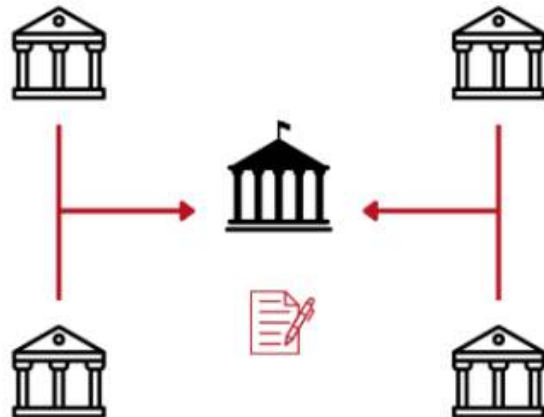
01. 블록체인 소개

블록체인이란?

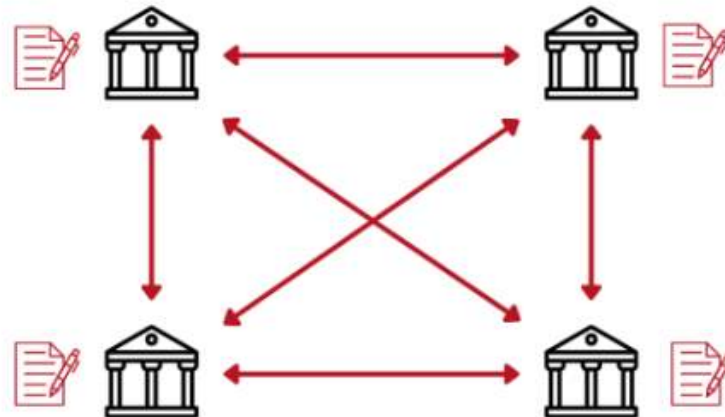
분산원장 (Distributed Ledger)

- 원장 (또는 장부)을 중앙 관리 주체 없이 peer-to-peer (P2P) 로 공유 및 관리
- 합의 알고리즘을 사용해 피어들이 가진 장부의 내용이 다를 때 진위를 합의할 수 있음

Centralised Ledger



Decentralised Ledger

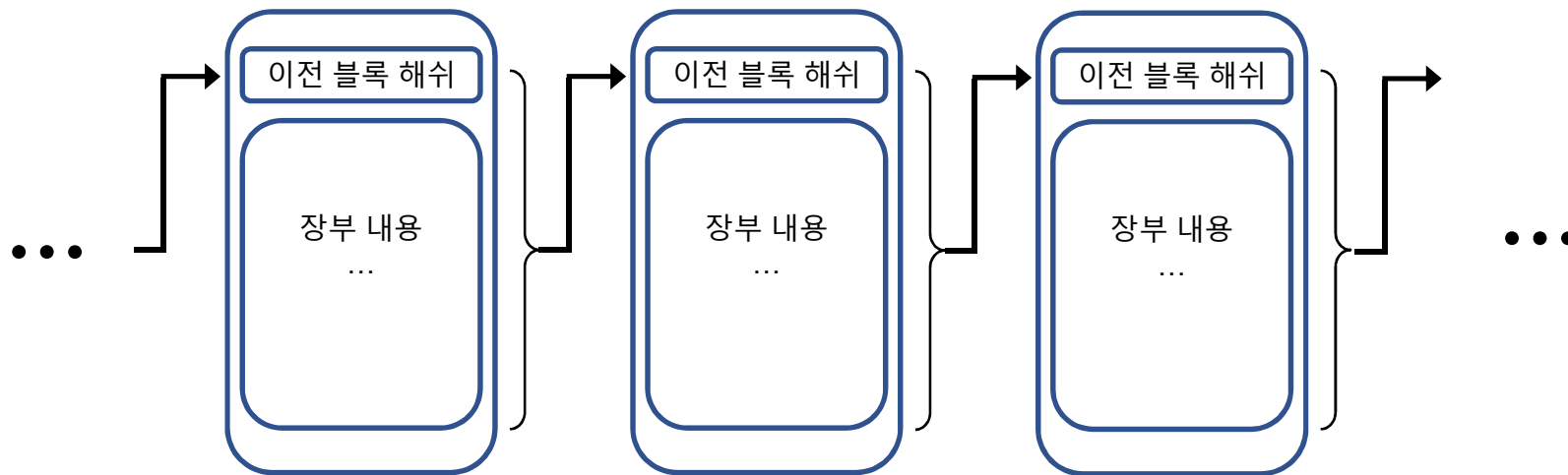


01. 블록체인 소개

블록체인이란?

블록체인

- 분산원장 기술의 일종
- 원장을 블록이라는 작은 단위로 나누어 관리
- 이전 블록과 다음 블록을 암호학적으로 연결하여 블록의 내용을 조작하지 못하도록 함



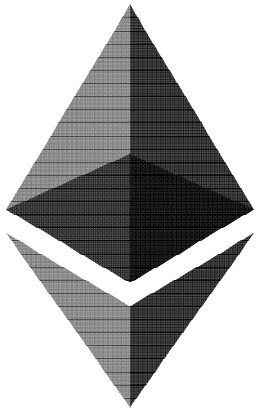
01. 블록체인 소개

블록체인의 플랫폼 예시



비트코인 (Bitcoin)

- 최초의 상업적 블록체인
- 비트코인이라는 암호화폐를 발행 (채굴) 및 전송 가능
- 스마트 컨트랙트 기능 없음



이더리움 (Ethereum)

- 스마트 컨트랙트를 지원하는 2세대 블록체인
- Binance 체인 등 다른 많은 블록체인들이 이더리움 기술에 기반함
- <https://ethereum.org/ko/>

01. 블록체인 소개

블록체인의 장단점

블록체인의 장점

- 탈중앙성/검열저항성: 중앙 관리 주체에 의한 검열이 없음
- 투명성: 장부의 모든 입출력 내용이 P2P 공유되고, 누구나 그 내용을 확인 및 검증 가능
- 자기부인 방지: 한 번 블록체인에 올라간 내용은 수정 및 조작이 불가능함
- 보안성: 서비스 주체가 분산되어 있어 DDoS 공격 등에 강인함

블록체인의 단점

- 느린 거래 확정 속도: 분산 노드들 간의 합의가 이루어지는데 시간이 오래 걸림
- 낮은 효율성: 장부를 분산 관리하면 중앙 관리에 비해 관리비용 (처리 수수료, 스토리지 등) 효율 감소



02. 스마트 컨트랙트, dapp 소개

02. 스마트 컨트랙트 소개

스마트 컨트랙트란?

분산원장에 어떤 내용을 쓸 것인가?

- “A가 B에게 1만원을 보냄” 등의 내용을 쓰면 거래장부 역할을 함
- 특정 기능을 수행하는 프로그램을 쓰면 계약서 (contract) 역할을 함

스마트 컨트랙트 예시

- “A가 1만원을 B에게 보내면, B는 그 중 5000원을 C에게 전달한다” 라는 내용의 계약
- 이 계약을 프로그래밍 언어로 작성하여 블록체인 상에서 실행 가능
- 블록체인에 올라간 내용은 조작이 불가능하고 모두에게 공개되어 있음
- 즉, 계약 내용이 투명하게 공개되어 있고, 그 내용 그대로 실행되는 것이 보장되어 있음
- 신뢰할 수 없는 상대방에게 계약내용 이행을 강제할 수 있음

02. 스마트 컨트랙트 소개

Dapp (distributed application; 탈중앙 애플리케이션)

Dapp

- 스마트 컨트랙트 (프로그램) 구동 가능
- 거래내용을 저장할 수 있는 데이터베이스 역할 가능
- 블록체인을 기반으로 동작하는 다양한 애플리케이션 가능

다양한 Dapp 예시: <https://dappradar.com/rankings>

- 쇼핑, 결제
- DeFi: 다양한 암호화폐 (코인, 토큰) 교환, 거래, 예치, 대출
- 게임
- NFT 아트, 수집품
- 투표/거버넌스 (DAO; decentralized autonomous organization)
- 소셜: steemit 등
- 승부예측, 추천



03. 토큰 (Token) 소개

03. Token 소개

토큰 (Token) 이란?

토큰: 스마트 컨트랙트로 작성된, 교환 가능한 가상 자산

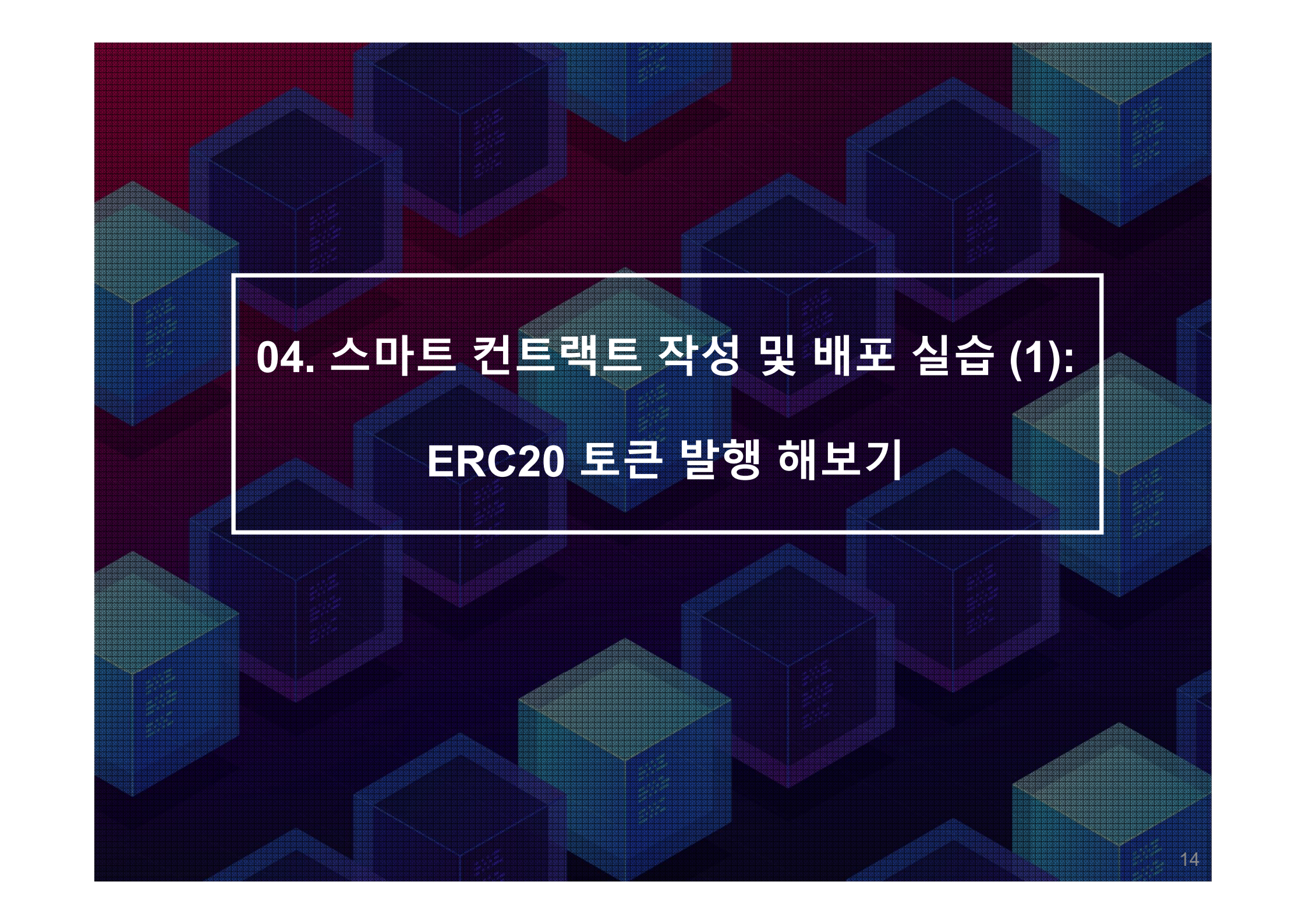
- 대부분 블록체인에는 기본 결제 용도로 사용되는 암호 화폐 (코인) 존재
- 토큰이란, 블록체인 스마트 컨트랙트로 작성된 또다른 종류의 암호 화폐임
 - 특정 dapp 을 이용하기 위해 필요한 이용권 개념인 유틸리티 토큰
 - 투표권/지분 개념의 거버넌스 토큰
 - 그 외 다양한 용도의 토큰
- 토큰의 기능 및 형식에 대한 표준으로 ERC20, ERC721, ERC1155 등이 있음
- 코인/토큰의 명확한 구분 없이 혼용되기도 함
- 수백가지 이상의 다양한 토큰 존재 <https://coinmarketcap.com/tokens/>

03. Token 소개

Fungible Token V.S. Non-Fungible Token (NFT)



- Fungible token: 한 토큰이 다른 토큰과 구별될 수 없으며 동일한 가치를 지님
 - 주로 화폐 역할을 함
 - 주요 기능: 전송, 잔고 확인, 토큰 발행 (mint) 및 소각 (burn)
 - 대표적인 표준 형식: ERC20
- Non-fungible token: 토큰은 다른 토큰과 구별할 수 있는 고유한 ID를 지님
 - 디지털 수집품, 게임 아이템, 증명서 등의 역할
 - 주요 기능: 토큰ID, 소유자 확인, 전송, 토큰 발행 (mint) 및 소각 (burn)
 - 대표적인 표준 형식: ERC721, ERC1155

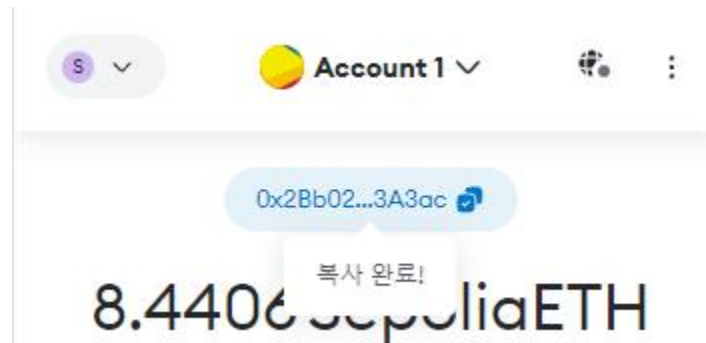


04. 스마트 컨트랙트 작성 및 배포 실습 (1): ERC20 토큰 발행 해보기

04. 스마트 컨트랙트 작성 및 배포 실습

실습 전 준비 사항

1. 크롬 브라우저, 인터넷 연결
2. 메타마스크 (MetaMask) 확장 프로그램 설치 (<https://metamask.io/>)
3. 메타마스크 새 지갑 만들기 or 지갑 불러오기, 비밀번호 백업
 - 주의: 비밀번호 노출
4. 메타마스크 설정
 - 테스트 네트워크 보기 ON
 - 네트워크 전환 (Sepolia)
5. 이더리움 계정 주소 취합
 - Google 설문 사용 (<https://forms.gle/TorPmmP2zGrdyeW88>)



04. 스마트 컨트랙트 작성 및 배포 실습

실습 0: 개발환경 준비

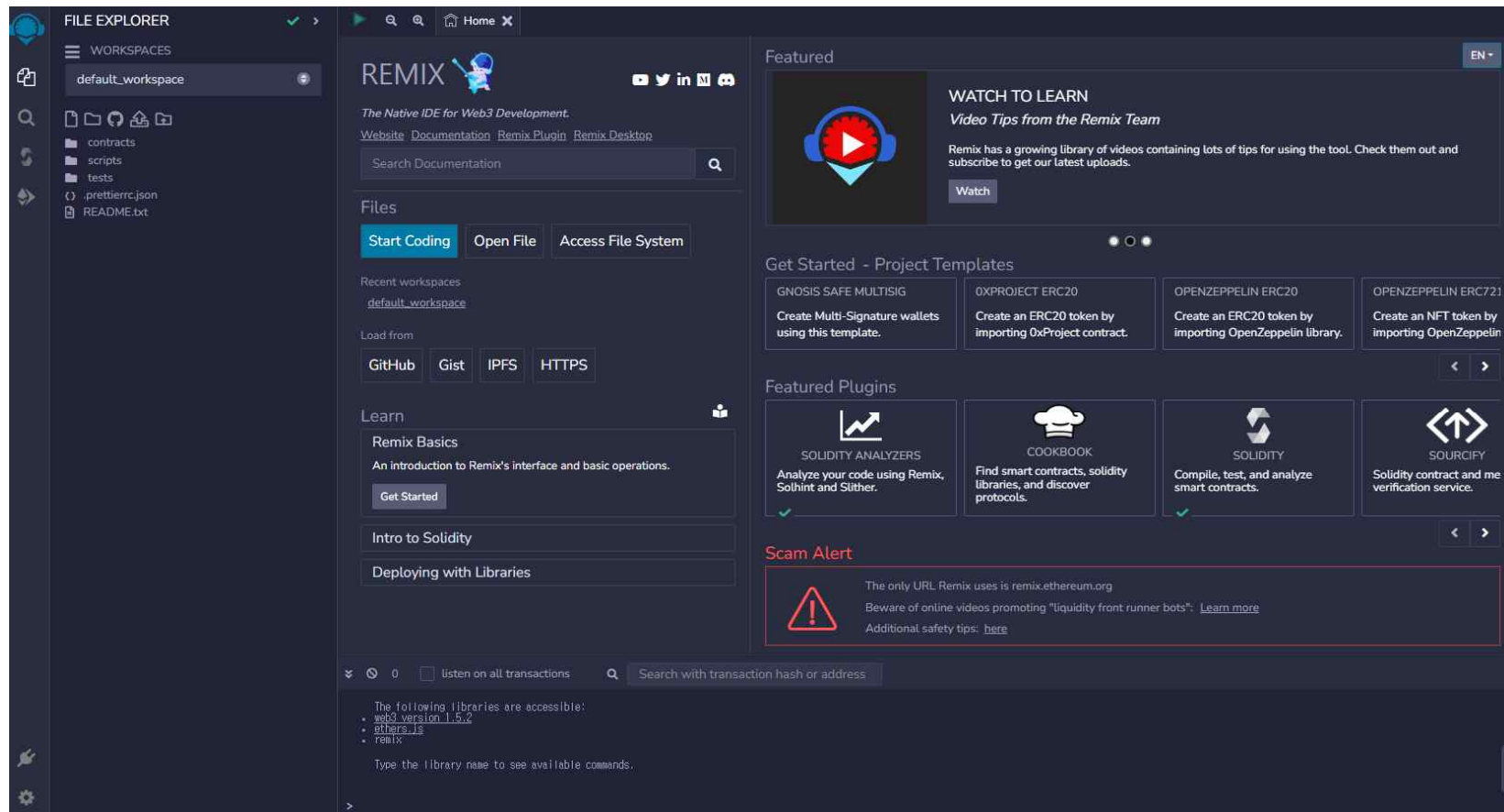
이더리움 기반 스마트컨트랙트 개발환경 준비

- Remix (<https://remix.ethereum.org/>)
 - GUI, 웹브라우저 기반 IDE
 - 간편하고 쉬움
 - 간단한 기능을 빠르게 구현하고 테스트 하기 적합
- Truffle (<https://trufflesuite.com/docs/truffle/>)
 - Node.js (javascript) 기반 IDE 패키지
 - 복잡한 기능 구현 및 대규모 프로젝트에 적합
 - 프론트엔드와 통합된 dapp 개발 가능
 - 다양한 예제
- Hardhat (<https://hardhat.org/docs>)
 - Truffle 과 비슷한 IDE
 - Truffle 기능의 대부분 이용가능하면서 접근성 및 난이도가 쉬움

04. 스마트 컨트랙트 작성 및 배포 실습

실습 0: 개발환경 준비

Remix IDE 살펴보기 (<https://remix.ethereum.org/>)



04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

실습 개요

1. ERC20 소스코드 살펴보기
 - Solidity 문법 기초
 - ERC20 토큰 주요 기능
2. ERC20 컨트랙트 작성 및 배포
 - 컨트랙트 소스코드 작성 및 컴파일
 - 가상네트워크 (VM) 에 배포, 컨트랙트와 상호작용 (transfer, mint, balanceOf)
3. 심화 기능 추가하여 소스코드 수정
 - 특정 사용자만 추가 발행할 수 있도록 제한
4. 공개 네트워크 (Sepolia) 에 배포 및 상호작용

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

ERC20 토큰 소스코드 불러오기

(<https://github.com/OpenZeppelin/openzeppelin-contracts/>)

Home의 load from 메뉴에서 GitHub 클릭하여 다음 url 붙여넣기:

- ERC20.sol

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol>

- IERC20.sol

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol>

- IERC20Metadata.sol

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/extensions/IERC20Metadata.sol>

- Context.sol

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Context.sol>

- draft-IERC6093.sol (IERC20Errors)

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/interfaces/draft-IERC6093.sol>

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

Solidity 기본 문법 (Solidity 버전, import, contract 선언)

```
1 // SPDX-License-Identifier: MIT
2 // OpenZeppelin Contracts (last updated v5.0.0) (token/ERC20/ERC20.sol)
3
4 pragma solidity ^0.8.20; Solidity 버전
5
6 import {IERC20} from "./IERC20.sol";
7 import {IERC20Metadata} from "./extensions/IERC20Metadata.sol";
8 import {Context} from "../../utils/Context.sol";
9 import {IERC20Errors} from "../../interfaces/draft-IERC6093.sol";
10
11 abstract contract ERC20 is Context, IERC20, IERC20Metadata, IERC20Errors {
12     키워드: contract 스마트 컨트랙트 이름
13     mapping(address account => uint256) private _balances;
14     mapping(address account => mapping(address spender => uint256)) private _allowances;
15     uint256 private _totalSupply;
16     string private _name;
17     string private _symbol;
18 }
```

Import

상속 (inherit) 선언

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

Solidity 기본 문법 (변수 선언)

```
abstract contract ERC20 is Context, IERC20, IERC20Metadata, IERC20Errors {  
    mapping(address account => uint256) private _balances;  
    mapping(address account => mapping(address spender => uint256)) private _allowances;  
  
    uint256 private _totalSupply;  
  
    string private _name;  
    string private _symbol;  
}
```

타입

변수 이름

가시성 (visibility)

변수 선언 문법

- 타입: uint, string, address, mapping (dictionary 타입과 유사) 등
- 가시성: public, private

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

Solidity 기본 문법 (함수 정의)

```

                                생성자
constructor(string memory name_, string memory symbol_) {
    _name = name_;
    _symbol = symbol_;
}

                                view 또는 pure 함수 표시
function name() public view virtual returns (string memory) {
    return _name;
}

                                가시성 (visibility)                                리턴 타입
                                함수 이름
function transfer(address to, uint256 value) public virtual returns (bool) {
    address owner = _msgSender();
    _transfer(owner, to, value);
    return true;
}

function _mint(address account, uint256 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidReceiver(address(0));
    }
    _update(address(0), account, value);
}

```

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

Solidity 기본 문법 (함수 정의)

- 가시성
 - 접근제어자 (access controller) 역할
 - public: 어디서나 호출 가능 (컨트랙트 내부 또는 외부)
 - external: 컨트랙트 외부에서만 호출 가능
 - internal: 컨트랙트 및 컨트랙트를 상속한 다른 컨트랙트에서 호출 가능
 - private: 컨트랙트 내의 다른 함수만 호출 가능
- view 함수와 pure 함수
 - view 함수: 데이터 수정 없이 보기만 하는 함수
 - pure 함수: 데이터 읽기/쓰기 모두 없는 함수 (ex: 숫자 두개 곱셈함수 등)

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

ERC20 토큰 주요 기능 살펴보기 (토큰 기본 정보)

```
function name() public view virtual returns (string memory) {  
    return _name;  
}  
function symbol() public view virtual returns (string memory) {  
    return _symbol;  
}  
function decimals() public view virtual returns (uint8) {  
    return 18;  
}  
function totalSupply() public view virtual returns (uint256) {  
    return _totalSupply;  
}
```

- 이름
- 심볼 (이름 약자)
- decimals (자릿수 단위)
- totalSupply

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

ERC20 토큰 주요 기능 살펴보기 (balanceOf, transfer)

```
function balanceOf(address account) public view virtual returns (uint256) {  
    return _balances[account];  
}  
  
function transfer(address to, uint256 value) public virtual returns (bool) {  
    address owner = _msgSender();  
    _transfer(owner, to, value);  
    return true;  
}
```

- balanceOf(account)
 - account 계좌가 가진 잔고 반환
- transfer(to, value)
 - msgSender 계좌에서 to 계좌로 value 만큼 토큰 전송

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

ERC20 토큰 주요 기능 살펴보기 (transferFrom, approve)

```
function transferFrom(address from, address to, uint256 value) public virtual returns (bool) {
    address spender = _msgSender();
    _spendAllowance(from, spender, value);
    _transfer(from, to, value);
    return true;
}

function approve(address spender, uint256 value) public virtual returns (bool) {
    address owner = _msgSender();
    _approve(owner, spender, value);
    return true;
}
```

- transferFrom (from, to, value)
 - from 계좌에서 to 계좌로 value 만큼 토큰 전송
 - 토큰 소유자가 미리 approve 해놓은 사람만 실행 가능
- approve(spender, value)
 - spender 가 본인의 토큰을 최대 value 만큼 사용할 수 있도록 허용

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 코드 작성

Alert

×

This contract may be abstract, it may not implement an abstract parent's methods completely or it may not invoke an inherited contract's constructor correctly.

OK

```
10
11 abstract contract ERC20 is Context, IERC20, IERC20Metadata, IERC20Errors {
12
13     mapping(address account => uint256) private _balances;
14     mapping(address account => mapping(address spender => uint256)) private _allowances;
15     uint256 private _totalSupply;
16     string private _name;
17     string private _symbol;
```

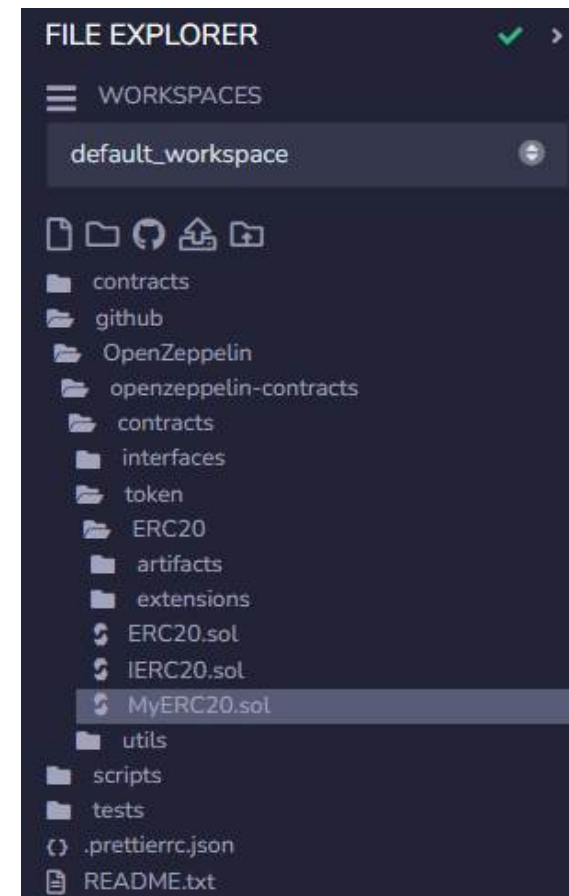
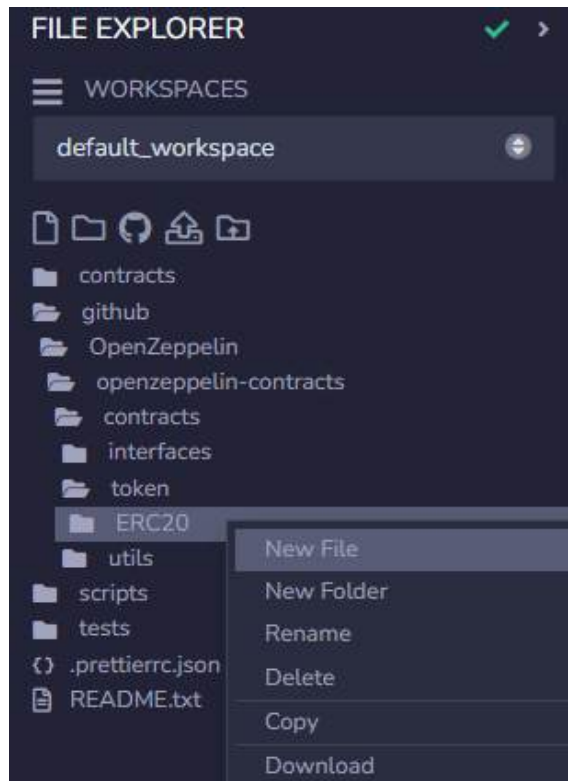
- ERC20 은 Abstract 컨트랙트 이므로 배포 불가능
- 토큰 발행 관련 기능 미구현

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 코드 작성

- 현재 경로에 “MyERC20.sol” 파일 생성



04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 코드 작성

- “MyERC20.sol” 파일 작성

```
// SPDX-License-Identifier: MIT
// MyERC20 version 0.0.1

pragma solidity ^0.8.20;
import {ERC20} from “./ERC20.sol”;

contract MyERC20 is ERC20 {
    uint256 public _initialSupply = 1000000 * (10**18);
    constructor(string memory name_, string memory symbol_) ERC20(name_, symbol_) {
        _mint(_msgSender(), _initialSupply);
    }

    function mint(address to, uint256 value) public {
        _mint(to, value);
    }
}
```


04. 스마트 컨트랙트 작성 및 배포 실습

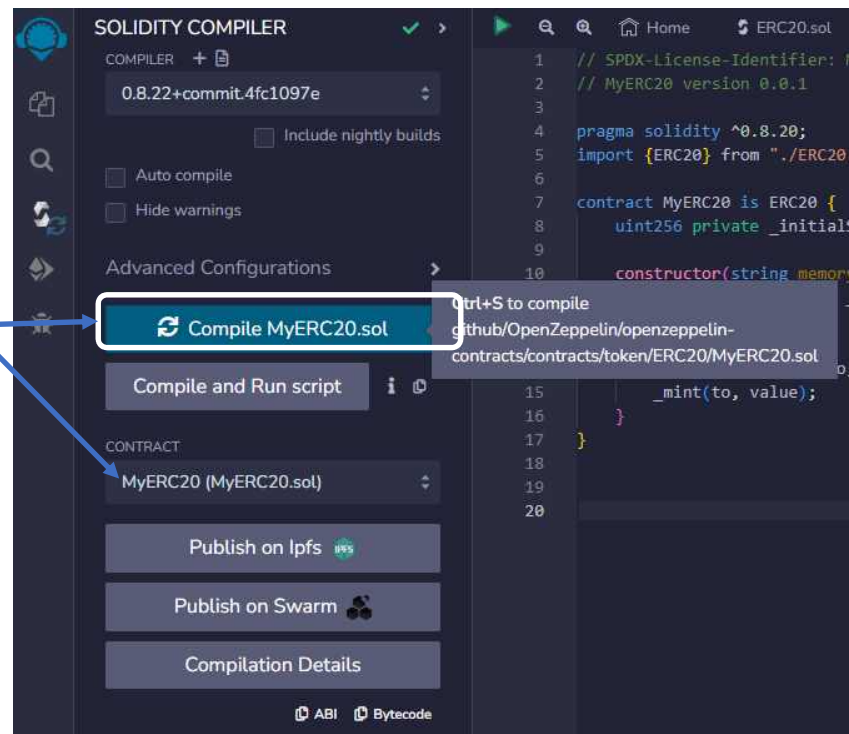
실습 1: ERC20 토큰

MyERC20 컴파일

- “MyERC20.sol” 파일 컴파일

1. 컴파일 할 소스 선택

2. 컴파일 버튼 클릭



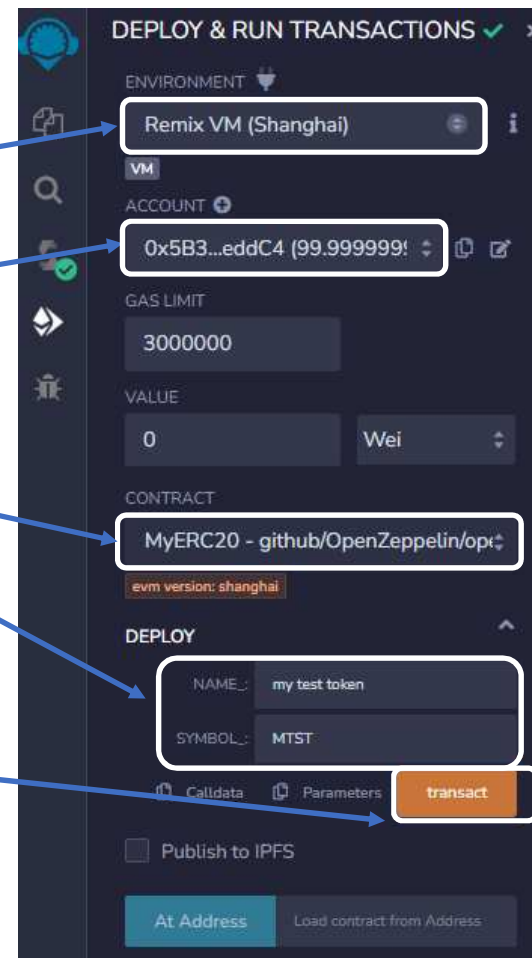
04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 배포 (Remix VM)

- Remix VM (가상 블록체인) 에 배포

1. 배포할 네트워크 선택
2. 배포자 계정 선택
3. 배포할 컨트랙트 선택
4. 생성자 입력값 (이름, 심볼) 입력
5. 배포 실행



04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 배포 (Remix VM)

- 배포 결과 (console)

The screenshot displays the Remix IDE console with a dark theme. At the top, there's a search bar with the text "Search with transaction hash or address". Below it, a green checkmark icon indicates a successful transaction. The console output shows the following details:

- [vm] from:** 0x5B3...eddc4 **to:** MyERC20.(constructor) **value:** 0 wei **data:** 0x608...00000 **logs:** 1 **hash:** 0xc0d...33bfl
- status:** 0x1 Transaction mined and execution succeed
- transaction hash:** 0xc0dc3e1d9770904a974c7e06275717e63965a7c73781b2fb05c2d72909733bfl
- block hash:** 0xf8793f93d5af4c9a198b0a6b95ba8b13352427f7bf9b13c88e2e87f52dc8417c
- block number:** 8
- contract address:** 0x358AA13c52544ECCEf680ADD0f801012ADAD5eE3
- from:** 0x5B38Da6a701c568545dCfcB03FcB875156beddC4
- to:** MyERC20.(constructor)
- gas:** gas
- transaction cost:** 1015773 gas

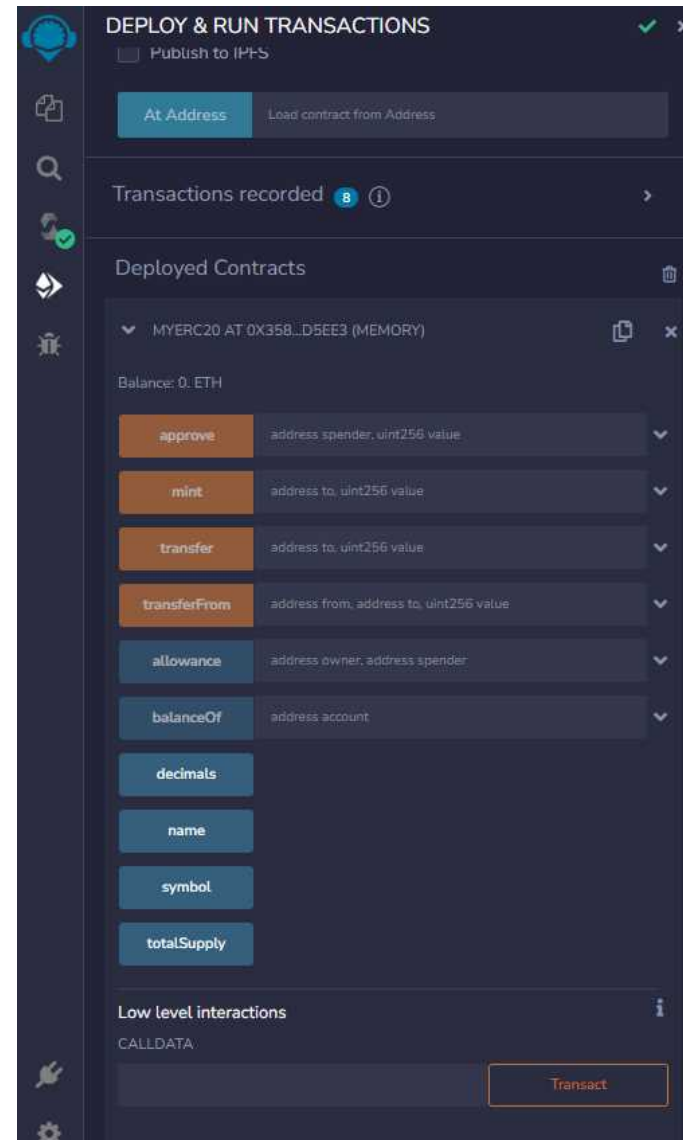
A "Debug" button is visible in the top right corner of the console area.

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 컨트랙트와 상호작용

- 배포된 MyERC20 컨트랙트의 함수 호출 가능
 - ✓ name, symbol 호출하여 확인
 - ✓ totalSupply, balanceOf 호출하여 확인
 - ✓ transfer, mint 작동하는지 확인



04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 코드 수정 (ownerble)

- 현재 코드는 아무나 신규발행 가능. 최초 배포자만 추가 발행 가능하도록 코드 수정

```
// SPDX-License-Identifier: MIT
// MyERC20 version 0.0.2

pragma solidity ^0.8.20;
import {ERC20} from "./ERC20.sol";

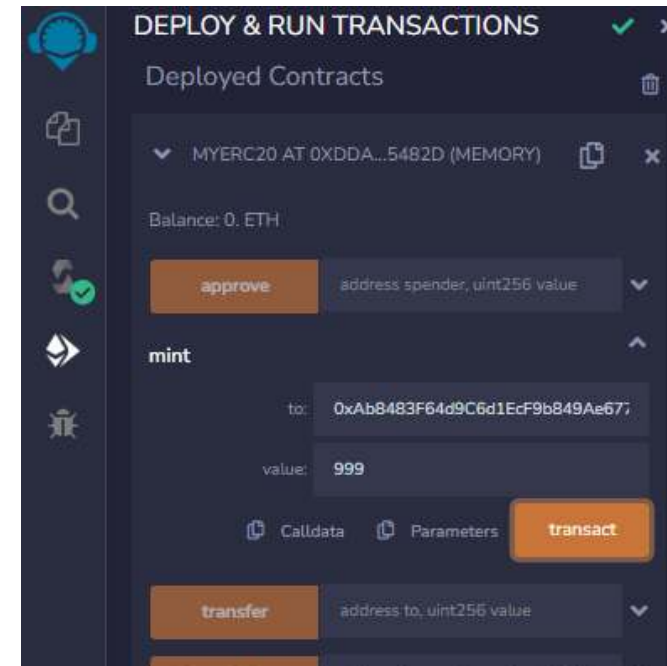
contract MyERC20 is ERC20 {
    uint256 public _initialSupply = 1000000 * (10**18);
    address public _owner;
    constructor(string memory name_, string memory symbol_) ERC20(name_, symbol_) {
        _mint(_msgSender(), _initialSupply);
        _owner = _msgSender();
    }
    function mint(address to, uint256 value) public {
        require(_owner == _msgSender(), "caller is not owner");
        _mint(to, value);
    }
}
```

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

MyERC20 코드 수정 (ownerble)

- 수정된 컨트랙트를 다시 컴파일 후 재배포
- 배포자와 다른 ACCOUNT 로 변경 후 mint 시도
- 거래가 무효 (revert) 됨



```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03Fc8875f56beddC4 to: MyERC20.symbol() data: 0x95d...89b41
```

```
transact to MyERC20.mint pending ...
```

```
transact to MyERC20.mint errored: Error occured: revert.
```

```
revert
```

```
The transaction has been reverted to the initial state.  
Reason provided by the contract: "caller is not owner".  
Debug the transaction to get more information.
```

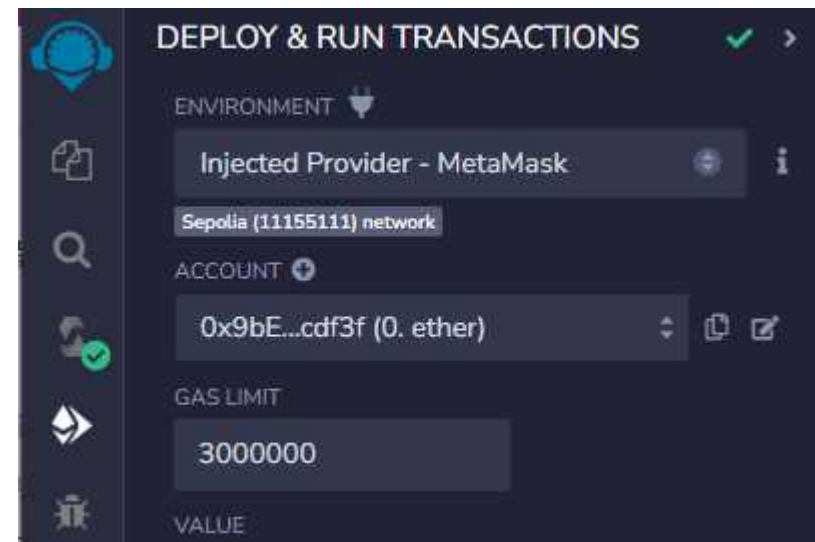
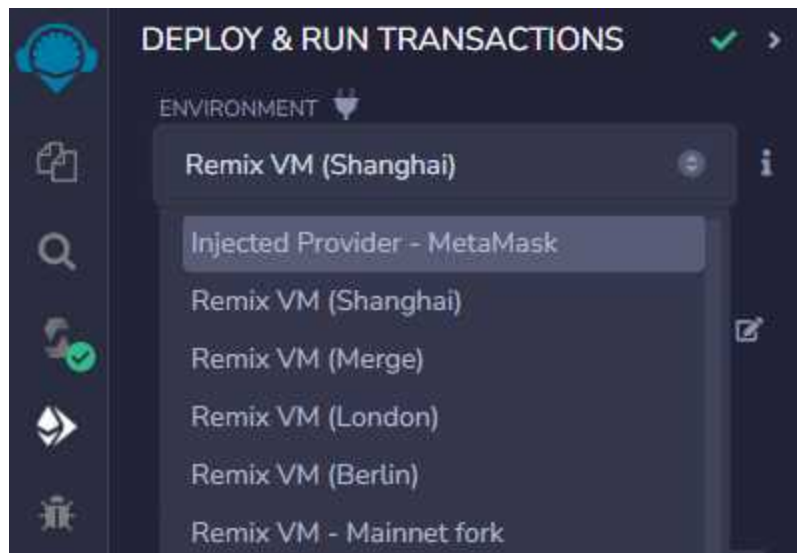
```
[vm] from: 0xAb8...35cb2 to: MyERC20.mint(address,uint256) 0xdda...5482d value: 0 wei data: 0x40c...003e7  
logs: 0 hash: 0x703...bbda1
```

04. 스마트 컨트랙트 작성 및 배포 실습

실습 1: ERC20 토큰

공개 네트워크에 배포하기

- 배포 화면의 ENVIRONMENT 드롭다운에서 “Injected Provider – MetaMask” 선택
- MetaMask 연결 허용
- MetaMask 앱 우상단에서 네트워크 선택 > Sepolia
- 재배포 (GAS LIMIT 1000000 미만 정도로 설정)



04. 스마트 컨트랙트 작성 및 배포 실습


실습 1: ERC20 토큰

Metamask 상에서 잔고 확인

- console 에서 만들어진 contract address 복사
- MetaMask 앱의 토큰 탭 > 토큰 가져오기 메뉴에서
토큰 계약 주소란에 붙여넣기
- ✓ 토큰 잔고 확인
- ✓ mint 호출 후 잔고 재확인

```
[block:4736875 txIndex:9] from: 0x9be...cdf3f to: MyERC20.(constructor) value: 0
logs: 1 hash: 0xc35...c137f
status      0x1 Transaction mined and execution succeed
transaction hash 0xf63144fa2d8a529bbc5ef138410b09269fc025dc9f5695f37b659883d08e7
block hash  0xc358761a2b044295523edaad67e5906f108f63f81b76ce416b268fc4822c
block number 4736875
contract address 0xcc57e8c23b4062d4a602c7dd6720513b616758d5
from 0x9bed56216b3e9477b62ea5ad5ec8fd3a0cdcdf3f
```





05. 스마트 컨트랙트 작성 및 배포 실습 (2): NFT 발행 해보기

05. 스마트 컨트랙트 작성 및 배포 실습 (2)

실습 2: NFT 발행

실습 개요

1. NFT 마켓플레이스 (OpenSea) 살펴보기
 - NFT 종류 확인 (예술품, 게임, 수집품 등)
 - NFT의 주요 요소 확인 (미디어, 수량제한, 트레잇 등)
2. 테스트넷 OpenSea 페이지를 사용하여 NFT 발행 하기
 - 생성(create) 메뉴의 NFT 발행 (mint an NFT) 도구 활용
 - 새 컬렉션 생성
 - 컬렉션에 새 NFT를 발행하여 추가
 - 발행된 NFT 확인
 - Sepolia 네트워크 활용

05. 스마트 컨트랙트 작성 및 배포 실습 (2)

실습 2: NFT 발행

오픈씨 (OpenSea) 둘러보기: <https://opensea.io/>

The screenshot displays the OpenSea marketplace interface. At the top, there's a navigation bar with 'OpenSea', 'Drops', 'Stats', and 'Create' buttons. A search bar is located next to the user's wallet balance (0 ETH, 0 WETH). Below the navigation bar, there are tabs for different categories: 'All', 'Art', 'Gaming', 'Memberships', 'PFPs', 'Photography', and 'Music'. The main content area features a grid of featured NFT collections, each with a representative image, name, and floor price. Below this grid, there are two tables showing trending and top collections. The 'Trending' table lists 'Seekers of Paragon' and 'Together'. The 'Top' table lists 'Sofa Maker' and 'ANOMALY A.I. by Star Im'.

Rank	Collection	Floor Price	Volume
1	Seekers of Paragon	0.04 ETH	35 ETH
2	Together	< 0.01 ETH	8 ETH

Rank	Collection	Floor Price	Volume
6	Sofa Maker	0.16 ETH	49 ETH
7	ANOMALY A.I. by Star Im	0.07 ETH	17 ETH

05. 스마트 컨트랙트 작성 및 배포 실습 (2)

실습 2: NFT 발행

Mint an NFT (좌상단 create 메뉴)

1. 새 컬렉션 만들기
 - 로고 이미지
 - 컬렉션 이름, 약자
 - 배포할 네트워크 선택

First, you'll need to create a collection for your NFT

You'll need to deploy an ERC-1155 contract on the blockchain to create a collection for your NFT. [What is a contract?](#)

Logo image ⓘ



Drag and drop or click to upload

You may change this after deploying your contract.

Recommended size: 350 x 350. File types: JPG, PNG, SVG, or GIF

Contract name ⓘ

My Collection Name

Token symbol ⓘ

MCN

Blockchain ⓘ



Goerli

Most popular

Estimated cost to
deploy contract:
US\$0.00



Mumbai

Cheaper

Estimated cost to
deploy contract:
US\$0.00



Sepolia

[Change](#)

Estimated cost to
deploy contract:

05. 스마트 컨트랙트 작성 및 배포 실습

실습 2: NFT 발행

Mint an NFT

2. 새 NFT 정보 작성

- 이미지
(또는 동영상)
- 이름
- 공급량
- 설명
- 외부 링크
- 특성 (trait)

NFT 생성
아이템이 발행된 후에는 해당 정보를 변경할 수 없습니다.

가고 싶은 데로 가.

컬렉션 *
명인 내방술 Your own words
Sepolia - ERC-1155
모든 컬렉션에 해당되는 것은 아닙니다. 자세히 알아보기

이름 *
Word No, 1

공급량 *
1

설명
Go where you like to go.

외부 링크
<https://collection.io/item/123>

특성
특성은 아이템의 속성을 설명합니다. 컬렉션 페이지 내에 필터로 표시되며 아이템 페이지에도 나열됩니다.

Language : Korean ✎ ✕

Font : Bazzi ✎ ✕

생성

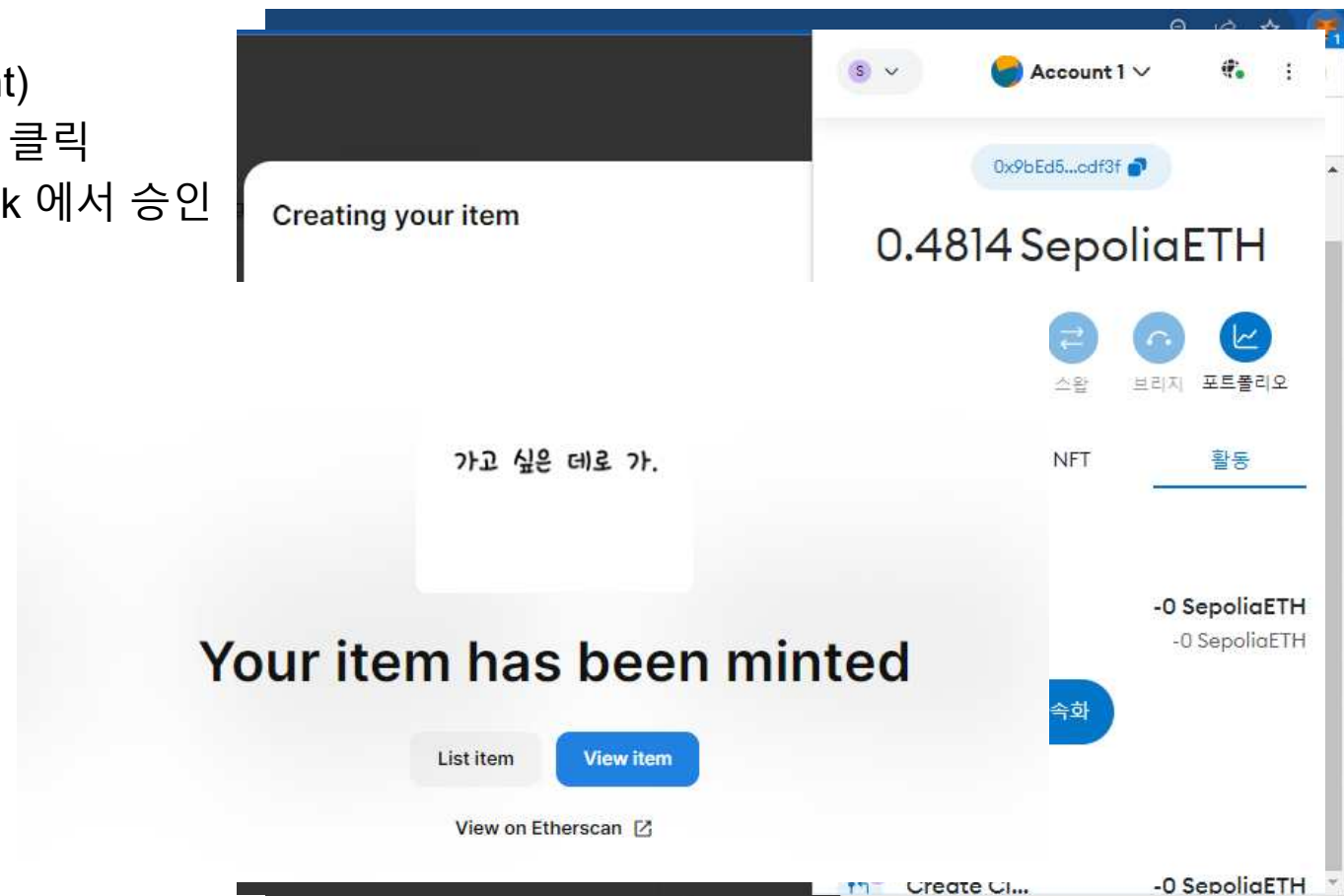
05. 스마트 컨트랙트 작성 및 배포 실습

실습 2: NFT 발행

Mint an NFT

3. NFT 발행 (mint)

- 생성버튼 클릭
- MetaMask 에서 승인



05. 스마트 컨트랙트 작성 및 배포 실습

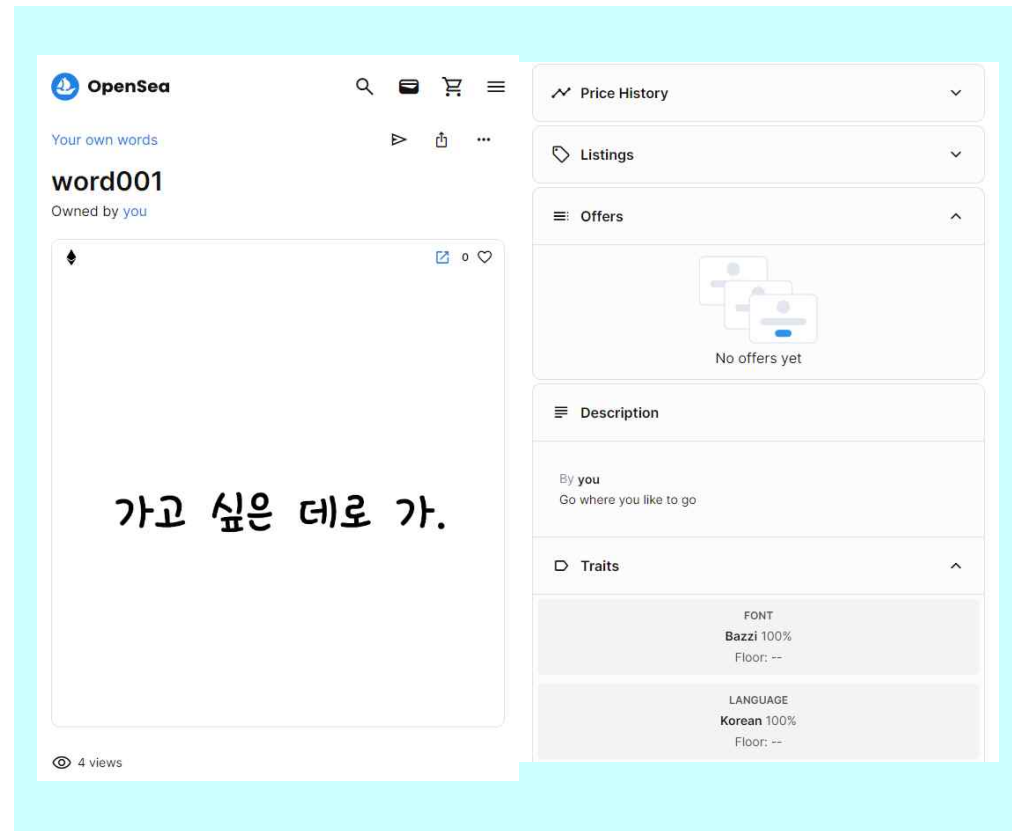
실습 2: NFT 발행

Mint an NFT

4. 발행된 NFT 확인

- 예제:

<https://testnets.opensea.io/assets/sepolia/0x4148a8166bdb71c2d75f746f8f14e7d9cb737fbf/1>



See also

추가 참고 자료:

- ❑ Crypto Zombies (<https://cryptozombies.io/ko/course>)
 - 스마트 컨트랙트 튜토리얼
 - Solidity 소스코드 작성, 테스트, 배포, dapp 프론트엔드 기초 등
- ❑ Etherscan (<https://etherscan.io/>)
 - 이더리움의 모든 정보를 볼 수 있는 블록 익스플로러
 - 테스트넷 전용 사이트 제공 (<https://sepolia.etherscan.io/>)
- ❑ Infura (<https://www.infura.io/>)
 - 다양한 블록체인에 안정적으로 접속할 수 있는 엔드포인트 제공
 - Sepolia 네트워크 테스트 이더 제공 (<https://www.infura.io/faucet/sepolia>)
- ❑ Sepolia PoW Faucet (<https://sepolia-faucet.pk910.de/>)
 - Sepolia 네트워크 테스트 이더 제공 (회원가입 없음)
 - 스팸 방지를 위해 약간의 작업 증명 필요



Thank You