Smart Contract 개발

오상문

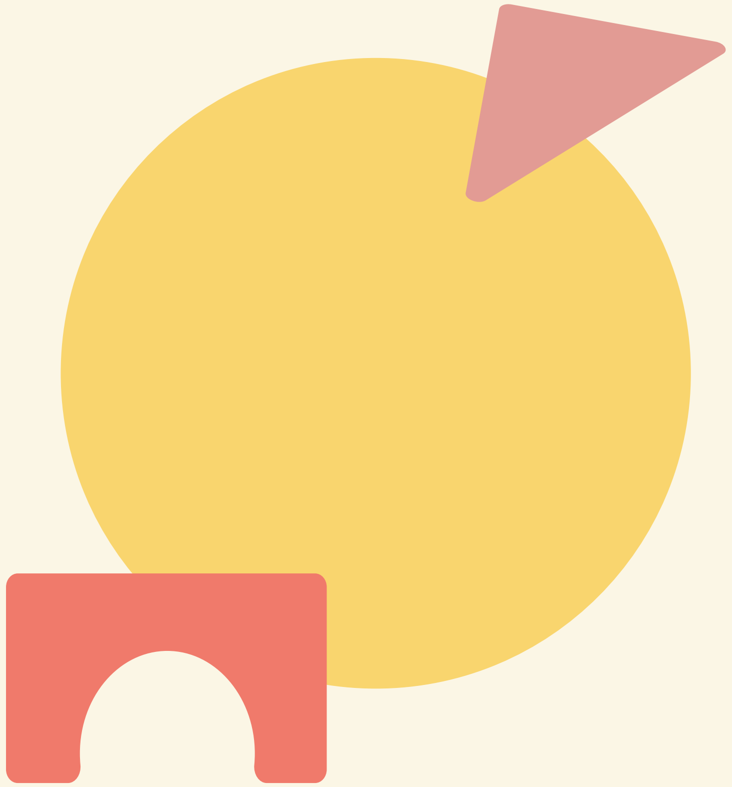
Sep. 2022

✓ Sample Project

https://github.com/3rdstage/smart-contract-dev-101-sample

✓ Presentation

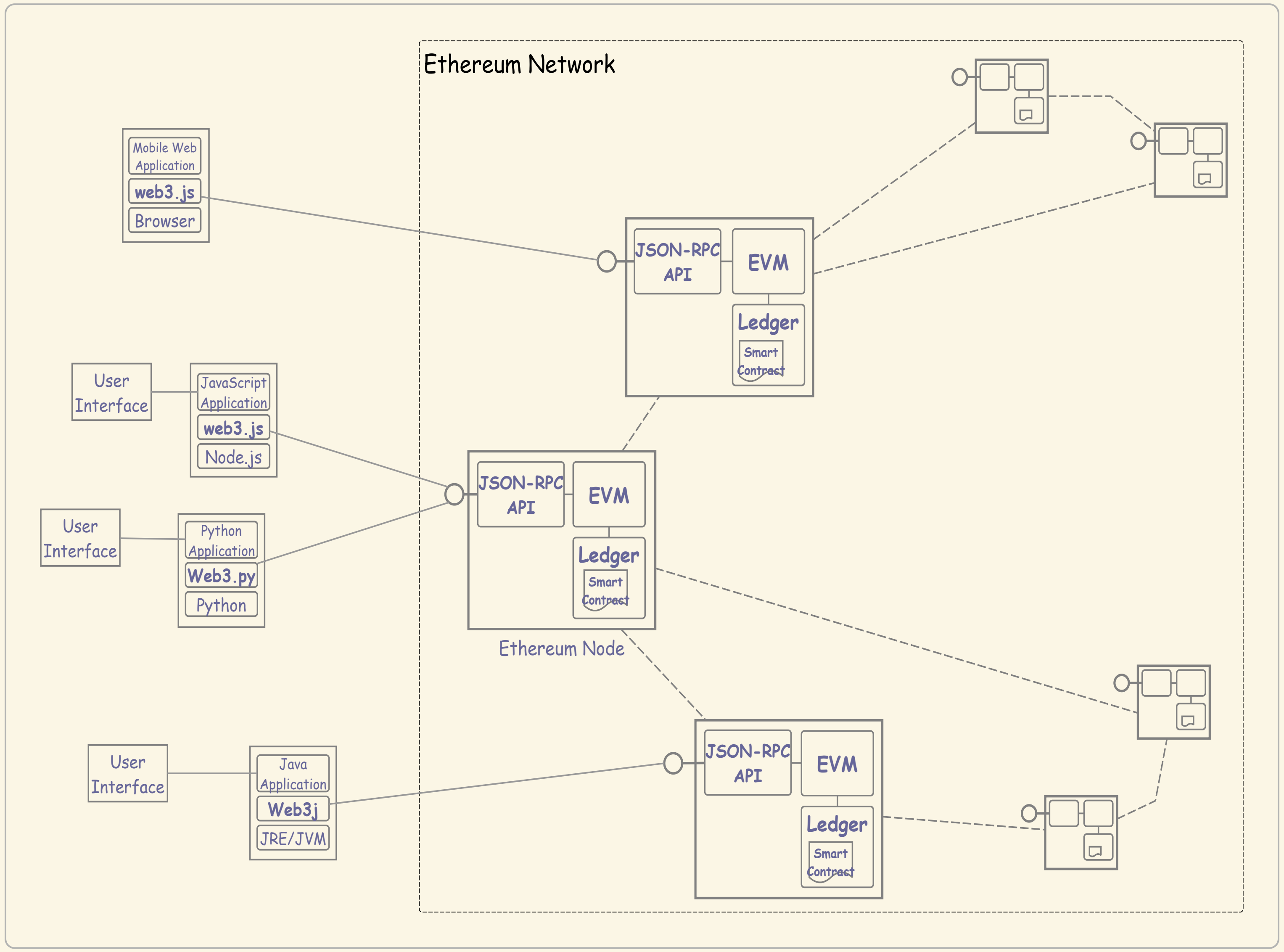
https://github.com/3rdstage/smart-contract-dev-101-sample/blob/master/smart-contract-dev-101.pdf



ToC

I. 개발환경 구성 및 실습

DApp Architecture



- **참고자료** 🗎
  - JSON-RPC API
  - Web3.js API
  - Web3.py API
  - Introduction to the Ethereum Stack

Smart Contract 개발 환경

Category		Tool/Service	Remarks
Editing/Build		Remix IDE	Web based
Build/Deploy/Unit Testing		Truffle	JavaScript based
		Brownie	Python based
		Hardhat	
Local Node		Ganache	Ganache CLI
Mainnet/Testnet Gateway		Infura	
Library		OpenZeppelin Contracts	
Block Explorer		Etherscan	Mainnet
		Etherscan/Rinkeby	
Wallet		MetaMask	

- Development Frameworks

사전준비사항 (Prerequisite)

- **O/S : Windows, macOS, Linux**
  - Windows의 경우 Git Bash 권장
    - 윈도우에서 Git Bash 설치하기

- **Node.js v14.x**
  - Node.js 내려받기
  - `Node.js` Releases

- **NVM or NVM for Windows**

  - 2개 이상 다수 version 의 Node.js 를 단일 장비에 설치하고 필요에 따라 version을 바꾸어가며 사용

Software	O/S	Guide
`NVM`	Linux, macOS	nvm 소개 및 설치 방법
`NVM for Windows`	Windows	nvm-windows, node.js 및 npm 설치 (Microsoft)

Truffle 설치

Smart contract 개발 환경 - Compile, Build, Deploy, Test

- **Node.js 버전 확인**

```
$ node --version
v14.19.0
```

- **Truffle 설치**

```
$ npm ls -g truffle          # Truffle 설치 여부 확인
...
$ npm ls -g --depth 0        # Global scope으로 설치된 모든 package 확
...
$ truffle version            # 설치된 Truffle 버전 확인
...
$ npm uninstall -g truffle  # 현재 설치된 Truffle 제거(uninstall)
...
$ npm install -g truffle
...
```

- **Truffle 설치 확인**

```
$ truffle version
Truffle v5.5.27 (core: 5.5.27)
Ganache v7.4.0
Solidity v0.5.16 (solc-js)
Node v14.19.0
Web3.js v1.7.4
```

Truffle Command-line 명령어

`truffle help`, `truffle help <subcommand>`

- Truffle이 제공하는 모든 명령어는 `truffle help` 명령으로 확인 가능

```
$ truffle help
Truffle v5.5.4 - a development framework for Ethereum

Usage: truffle <command> [options]

Commands:
  build       Execute build pipeline (if configuration present)
  compile     Compile contract source files
  config      Set user-level configuration options
  console     Run a console with contract abstractions and commands avail
  create      Helper to create new contracts, migrations and tests
  ...
  deploy      (alias for migrate)
  develop     Open a console with a local development blockchain
  exec        Execute a JS module within this Truffle environment
  help        List all commands or provide information about a specific c
  init        Initialize new and empty Ethereum project
  install     Install a package from the Ethereum Package Registry
  migrate     Run migrations to deploy contracts
  networks    Show addresses for deployed contracts on each network
  ...
  run         Run a third-party command
  test        Run JavaScript and Solidity tests
  unbox       Download a Truffle Box, a pre-built Truffle project
  version     Show version number and exit
  watch       Watch filesystem for changes and rebuild the project automa
```

- 개별 하위 명령어에 대한 상세 구문과 도움말은 `truffle help <subcommand>` 명령으로 확인

```
$ truffle help deploy

truffle deploy [--reset] [-f <number>] [--compile-all] [--verbose-rpc] [
(alias for migrate)

--reset
    Run all migrations from the beginning, instead of running from the l
--f <number>
    Run contracts from a specific migration. The number refers to the pr
--to <number>
    Run contracts to a specific migration. The number refers to the pref
--compile-all
    Compile all contracts instead of intelligently choosing which contra
--compile-none
    Do not compile any contracts before migrating.
--verbose-rpc
    Log communication between Truffle and the Ethereum client.
...
--dry-run
    Only perform a test or 'dry run' migration.
--skip-dry-run
    Do not run a test or 'dry run' migration.
...
--network <name>
    Specify the network to use. Network name must exist in the configura
...
```

Truffle Project 생성

`truffle init`

- Truffle 은 smart contract 개발을 위한 표준 디렉토리 layout 을 정의하고 있음.
- 프로젝트 base 디렉토리 아래에 용도별로 `contracts` , `migrations` , `test` , `build` 의 하위 디렉토리를 활용하고,
  Truffle 환경설정은 `truffle-config.js` 파일에 지정함.

```
$ mkdir first-contracts && cd first-contracts
...
first-contracts $ truffle init

Starting init...
================
...

first-contracts $ ls -R
.:
./  ../  .gitattributes  contracts/  migrations/  test/  truff

./contracts:
./  ../  .gitkeep

./migrations:
./  ../  .gitkeep

./test:
./  ../  .gitkeep
first-contracts $ cat truffle-config.js
...
```

```
first-contracts $ npm init -y
...

first-contracts $ cat package.json
...
```

- **참고자료**
  - Truffle Commands

Truffle Compile 명령

`truffle compile`

- **Contract Source 파일 생성**

```
first-contracts $ touch contracts/MetaCoin.sol  # 빈 sourc
first-contracts $ vi contracts/MetaCoin.sol
...
first-contracts $ cat contracts/MetaCoin.sol
...
first-contracts $
```

- **Contract Compile**

```
first-contracts $ truffle compile --all
...
- Downloading compiler. Attempt #1.
...
> Compiled successfully using:
   - solc: 0.8.16+commit.07a7930e.Emscripten.clang
first-contracts $
```

- **Contract Compile 결과 산출물**

```
first-contracts $ ls build/contracts/
./  ../  ConvertLib.json  MetaCoin.json
first-contracts $
```

- **Contract Source** (원본 📄)

```
// SPDX-License-Identifier: MIT
// from 'https://github.com/truffle-box/metacoin-box'
pragma solidity ^0.8.13;

library ConvertLib {
  function convert(uint amount, uint conversionRate)
      public pure returns (uint convertedAmount) {
    return amount * conversionRate;
  }
}

contract MetaCoin {
  mapping (address => uint) balances;

  event Transfer(address indexed _from,
      address indexed _to, uint256 _value);

  constructor() { balances[tx.origin] = 10000; }

  function sendCoin(address receiver, uint amount)
      public returns(bool sufficient) {
    if (balances[msg.sender] < amount) return false;
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
    emit Transfer(msg.sender, receiver, amount);
    return true;
  }

  function getBalance(address addr) public view returns(uint) {
    return balances[addr];
  }

  function getBalanceInEth(address addr) public view returns(uint) {
    return ConvertLib.convert(getBalance(addr),2);
  }
}
```

Truffle Compile 산출물

bytecode, ABI

```
first-contracts $ node        # Node.js shell (REPL) 진입
Welcome to Node.js v14.19.0.
Type ".help" for more information.
```

```
>
> a = fs.readFileSync('./build/contracts/MetaCoin.json')
...
> b = JSON.parse(a.toString())
...
> b.bytecode
'0x608060405234801561001057600080fd5b506127106000803273ffffffffffffffff...'

> console.dir(b.abi, {depth : null})
[
  { inputs: [], stateMutability: 'nonpayable', type: 'constructor' },
  ...
  {
    inputs: [
      { internalType: 'address', name: 'receiver', type: 'address' },
      { internalType: 'uint256', name: 'amount', type: 'uint256' }
    ],
    name: 'sendCoin',
    outputs: [ { internalType: 'bool', name: 'sufficient', type: 'bool' } ],
    stateMutability: 'nonpayable',
    type: 'function'
  },
  ...
]
> .exit
first-contracts $
```

Ganache

Local standalone Ethereum simulator for testing purpose

- **설치**

```
first-contracts $ npm ls -g --depth 0 ganache    # 설치 여부 확인
...
first-contracts $ npm install -g ganache         # 설치
...
first-contracts $ ganache --version              # 버전 확인
```

- **사용법**

```
first-contracts $ ganache --help | less
...
```

- **참고자료** 📄
    - Ganache Startup Options

Ganache 실행

```
first-contracts $ mnemonic="army van defense carry jealous true garbage claim echo media make crunch
first-contracts $ ganache --database.dbPath run/ganache/data -m "$mnemonic" -a 10 -n false -e 10000
...
```

```
ganache v7.4.1 (@ganache/cli: 0.5.1, @ganache/core: 0.5.1)
Starting RPC server

Available Accounts
==================
(0) 0x2161DedC3Be05B7Bb5aa16154BcbD254E9e9eb68 (10000 ETH)
(1) 0x9595F373a4eAe74511561A52998cc6fB4F9C2bdD (10000 ETH)
(2) 0x67F439f1ba85f86e1e405810675c06bC4020596D (10000 ETH)
(3) 0xf319c1A07c173800a5A3532195A8804bd90d997E (10000 ETH)
...

Private Keys
==================
(0) 0x73bf21bf06769f98dabcfac16c2f74e852da823effed12794e56876ede02d45d
(1) 0x9b1dd6e4ee4f1895e9191e626bd61081cf7f4cfe63e16024faeac73aa829cfcb
(2) 0x1b129af25984b49d1be37ddcefaccf05eefc934fe56c2529caa77e85d161d3db
(3) 0xd0ba4cd486a6d63c02c1d83b187ae1169397710572e73211c97e6b769d283adb
...

HD Wallet
==================
Mnemonic:      army van defense carry jealous true garbage claim echo media make crunch
Base HD Path:  m/44'/60'/0'/0/{account_index}

...

RPC Listening on 127.0.0.1:8545
```

Ganache 접속

- **Truffle 환경설정 파일 Update**

```
first-contracts $ vi truffle-config.js
...
```

- `truffle-config.js` **내용**

```
module.exports = {
  networks: {
    development: {
      network_id: "*",
      host: '127.0.0.1',
      port: 8545,
      gasPrice: 0
    }
  },

  compilers: {
    solc: {
      version: "pragma"
    }
  }
}
```

- **Ganache 접속**

```
first-contracts $ cat truffle-config.js
...

first-contracts $ truffle console
truffle(development)>
```

- **참고자료**
  - Truffle Configuration
  - `truffle console` Command
  - Compiler(`solc`) Configuration

Web3.js on Ganache

```
truffle(development)> web3.eth.getNodeInfo()
'Ganache/v7.4.1/EthereumJS TestRPC/v7.4.1/ethereum-js'
truffle(development)> web3.eth.net.getPeerCount()
0
truffle(development)> web3.eth.getBlockNumber()
0
truffle(development)> web3.eth.getAccounts()
...
truffle(development)> accounts
...
truffle(development)> web3.eth.getBalance(accounts[0])
'10000000000000000000000'
truffle(development)> web3.eth.getBalance(accounts[1])
'10000000000000000000000'
truffle(development)> web3.eth.sign("message", accounts[0])
  '0x707c8c0210364d6a07349d298d43f21ccc52cd27fb45bc73760a66dc81f61b25221aab284ab5eb3528f7b12a02350332dec15bd75994013c
truffle(development)> web3.eth.sign("message", accounts[1])
'0xf4eda169175b9dfbbefeb59f0abf77792bda22e5f1828a223b2342fafa1f691554cebb08d24fa58edf030cf0111a516dd5f8d2067e258c9ff5
truffle(development)> web3.eth.sendTransaction({from: accounts[0], data: "0x0"})
...
truffle(development)> web3.eth.sendTransaction({from: accounts[1], data: "0x0"})
...
truffle(development)> web3.eth.getTransaction('...')
...
truffle(development)> web3.eth.getBlockNumber()
...
truffle(development)> web3.eth.getBlock('latest')
...
```

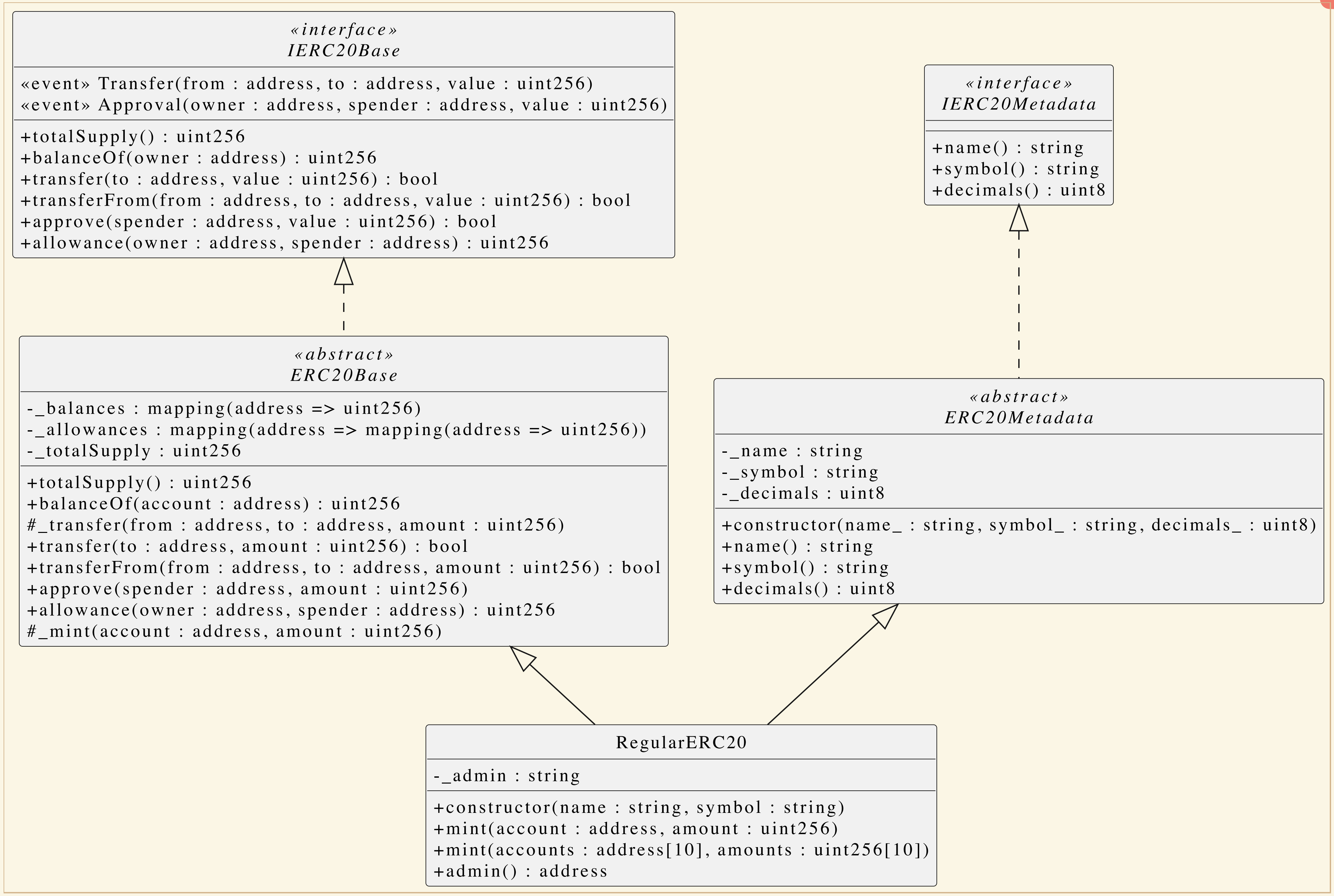
- `web3.js` API

II. Solidity Programming

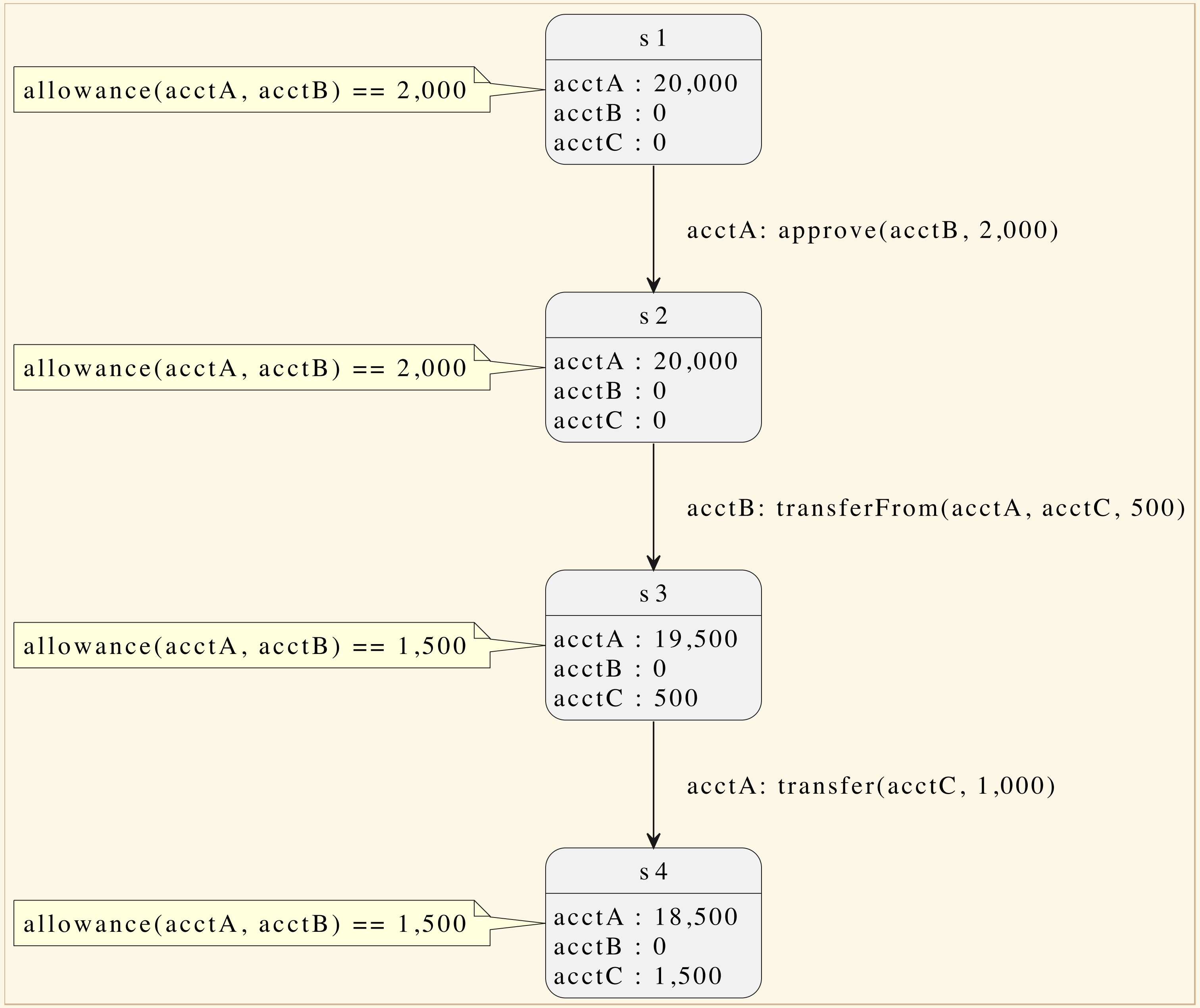
Solidity

- ***Programming language for EVM***
  - Compiled into bytecode and executed as a number of EVM Opcodes.
  - Ethereum Yellowpaper
- ***Object-Oriented***
  - `contract` / `class`
  - `interface`, `abstract contract`,
  - multiple inheritance, polymorphism(function overriding)
- ***Statically typed***
  - Compile-time type safety
  - `bool`, `uint<M>` (unsigned int), `int<M>` (signed int), `address`, `bytes<N>` (fixed-sized byte array), `enum`
  - `bytes` (dynamic-sized byte array), `string` (unicode string)
  - `T[M]`, `T[]`, `mapping[K => V]` (hash table), `struct`
- ***C++, Java, JavaScript like Syntax***
  - Curly-brace block (`{` ... `}`)
  - Control statements
    - `if` / `else if` / `else`
    - `for`, `do`, `while`, `break`, `continue`
  - Exception handling statement : `try` / `catch`
  - Operators
    - arithmetic : `+` `-` `*` `/` `%` `**` `++` `--`
    - comparison : `==` `!=` `<` `>` `<=` `=>`
    - logical : `&&` `||` `!`,
    - bitwise : `&` `|` `^` `~` `<<` `>>`
    - assignment : `=` `+=` `-=` `*=` `/=` `%=` `<<=` `=>>`
    - ternary : `<condition> ? <if-true> : <if-false>`
  - Visibility : `external`, `public`, `internal`, `private`

Sample Contracts Model


«interface»
IERC20Base
«event» Transfer(from : address, to : address, value : uint256)
«event» Approval(owner : address, spender : address, value : uint256)
+totalSupply() : uint256
+balanceOf(owner : address) : uint256
+transfer(to : address, value : uint256) : bool
+transferFrom(from : address, to : address, value : uint256) : bool
+approve(spender : address, value : uint256) : bool
+allowance(owner : address, spender : address) : uint256
«interface»
IERC20Metadata
+name() : string
+symbol() : string
+decimals() : uint8
«abstract»
ERC20Base
-_balances : mapping(address => uint256)
-_allowances : mapping(address => mapping(address => uint256))
-_totalSupply : uint256
+totalSupply() : uint256
+balanceOf(account : address) : uint256
#_transfer(from : address, to : address, amount : uint256)
+transfer(to : address, amount : uint256) : bool
+transferFrom(from : address, to : address, amount : uint256) : bool
+approve(spender : address, amount : uint256)
+allowance(owner : address, spender : address) : uint256
#_mint(account : address, amount : uint256)
«abstract»
ERC20Metadata
-_name : string
-_symbol : string
-_decimals : uint8
+constructor(name_ : string, symbol_ : string, decimals_ : uint8)
+name() : string
+symbol() : string
+decimals() : uint8
RegularERC20
-_admin : string
+constructor(name : string, symbol : string)
+mint(account : address, amount : uint256)
+mint(accounts : address[10], amounts : uint256[10])
+admin() : address

ERC-20 : `approval` / `allowance` and `transferFrom`


s1
acctA : 20,000
acctB : 0
acctC : 0
allowance(acctA, acctB) == 2,000
acctA: approve(acctB, 2,000)
s2
acctA : 20,000
acctB : 0
acctC : 0
allowance(acctA, acctB) == 2,000
acctB: transferFrom(acctA, acctC, 500)
s3
acctA : 19,500
acctB : 0
acctC : 500
allowance(acctA, acctB) == 1,500
acctA: transfer(acctC, 1,000)
s4
acctA : 18,500
acctB : 0
acctC : 1,500
allowance(acctA, acctB) == 1,500

Remix IDE

- `remixd` **설치**

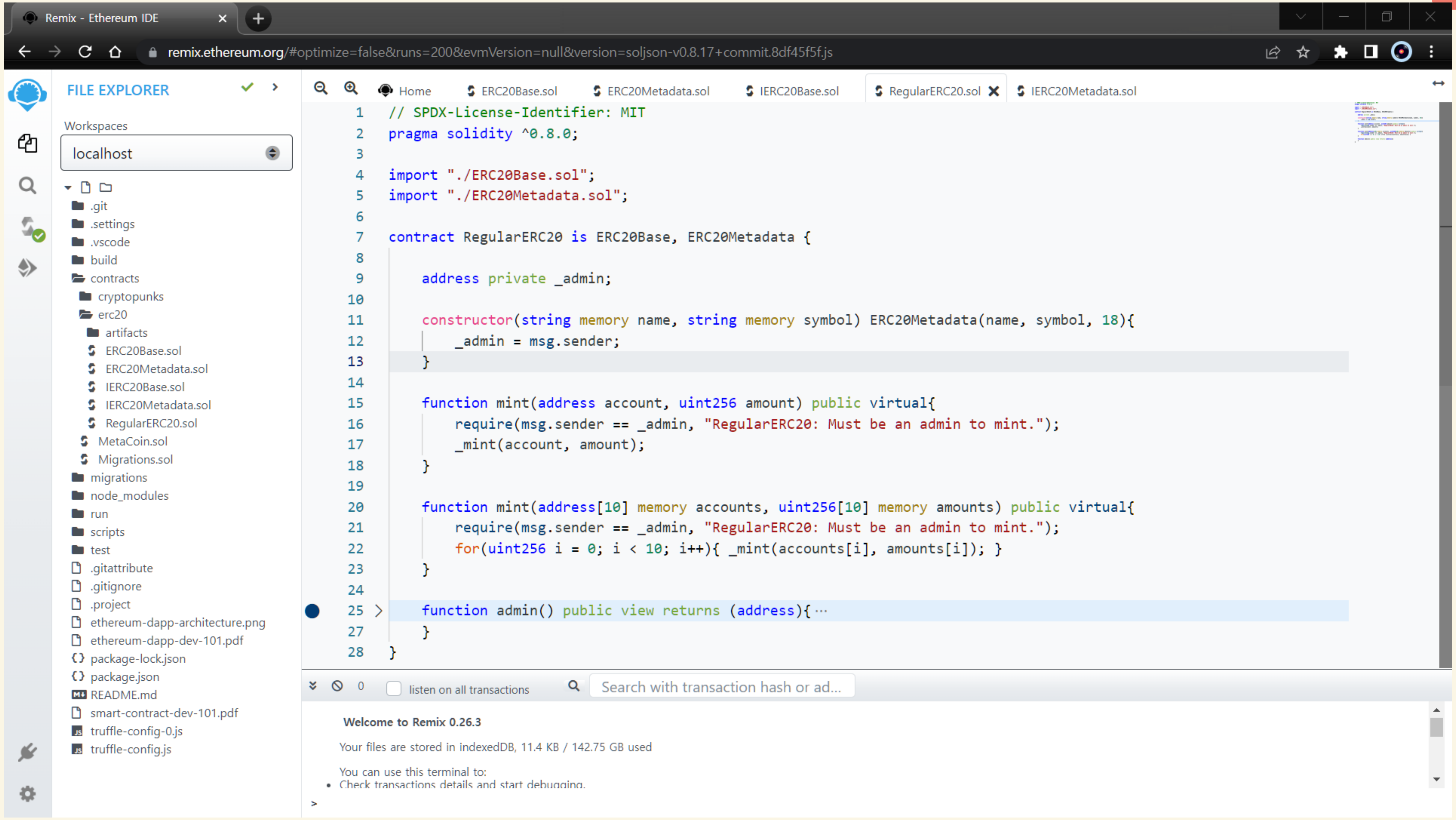
```
first-contracts $ npm install @remix-project/remixd -D    # 설치
...
first-contracts $ npm ls --depth 0                        # 설치 여부 확인
...
first-contracts $ npx remixd --version                    # 버전 확인
...
...
```

- `remixd` **실행**

```
first-contracts $ npx remixd --shared-folder ./ --remix-ide https://remix.ethereum.org
...
[INFO] Fri Sep 16 2022 17:45:17 GMT+0900 (대한민국 표준시) remixd is listening on 127.0.0.1:65520
```

- **Remix IDE 사용**
  - Browser를 사용하여 https://remix.ethereum.org 접속
  - Main panel 중앙의 **File** section 아래에 있는 **Connect to Localhost** 클릭
  - 왼쪽 panel 의 **File Explorer** 에서 `contracts` 디렉토리에 오른쪽 마우스 클릭으로 popup 메뉴를 열고 **New Folder** 를 실행하여 `erc20` 디렉토리를 생성
  - 생성한 `erc20` 디렉토리에 오른쪽 마우스 클릭으로 popup 메뉴를 열고 **New File**을 실행하여 `IERC20Base.sol` 파일 생성
    - 같은 방법으로 `erc20` 디렉토리에 `IERC20Metadata.sol`, `ERC20Base.sol`, `ERC20Metadata.sol`, `RegularERC20.sol` 파일들을 차례로 생성
  - 아래 GitHub repository에서 각 contract file의 source를 복사
    - https://github.com/3rdstage/smart-contract-dev-101-sample/tree/master/contracts/erc20

Remix IDE



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./ERC20Base.sol";
import "./ERC20Metadata.sol";

contract RegularERC20 is ERC20Base, ERC20Metadata {

    address private _admin;

    constructor(string memory name, string memory symbol) ERC20Metadata(name, symbol, 18){
        _admin = msg.sender;
    }

    function mint(address account, uint256 amount) public virtual{
        require(msg.sender == _admin, "RegularERC20: Must be an admin to mint.");
        _mint(account, amount);
    }

    function mint(address[10] memory accounts, uint256[10] memory amounts) public virtual{
        require(msg.sender == _admin, "RegularERC20: Must be an admin to mint.");
        for(uint256 i = 0; i < 10; i++){ _mint(accounts[i], amounts[i]); }
    }

    function admin() public view returns (address){…
    }
}
```

```
Welcome to Remix 0.26.3

Your files are stored in indexedDB, 11.4 KB / 142.75 GB used

You can use this terminal to:
• Check transactions details and start debugging.
```

- **참고자료** 🗎
  - Remix Project (Remix IDE, Remix Desktop)
  - Remix IDE Documentation
  - Remixd

Sample Contracts - `IERC20Metadata.sol` , `ERC20Metadata.sol`

- `interface IERC20Metadata`

```
pragma solidity ^0.8.0;

interface IERC20Metadata {

    function name() external view returns (string memory);
    function symbol() external view returns (string memory);
    function decimals() external view returns (uint8);
}
```

- `abstract contract ERC20Metadata`

```
pragma solidity ^0.8.0;
import "./IERC20Metadata.sol";

abstract contract ERC20Metadata is IERC20Metadata {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor(string memory name_, string memory symbol_, uint8 decimals_){
        _name = name_;
        _symbol = symbol_;
        _decimals = decimals_;
    }

    function name() public view override returns (string memory){
        return _name;
    }

    function symbol() public view virtual override returns (string memory){
        return _symbol;
    }

    function decimals() public view virtual override returns (uint8){
        return _decimals;
    }
}
```

Visibility and Mutability

- **Visibility** : state variable 또는 function에 접근할 수 있는 범위를 지정

Visibility	Applied-to	Description	Samples
`private`	state variable, function	현재 contract 내부에서만 접근이 가능	`string private _name` `function _init() private{ ... }`
`internal`	state variable, function	현재 contract 또는 상속받은 contract 에서 접근 가능	`function _mint(...) internal { ... }`
`public`	state variable, function	현재 contract 내부, 상속받은 contract, 다른 contract 에서 모두 접근 가능	`function balanceOf(address) public view returns (uint256){ ... }`
`external`	function	다른 contract에서만 접근 가능	`function transfer(address to, uint256 value) external returns (bool)`

- **Mutability** : function이 state variable 에 접근(읽기, 쓰기)하는 방식을 지정

Mutability	Description	Consensus	JSON-RPC API	Samples
	state variable 값을 변경하거나, event 를 생성	O	`eth_sendTransaction`, `eth_sendRawTransaction`	`transfer(address to, uint256 amount) public virtual override returns (bool)`
`view`	state variable 값을 읽기만 하며, 변경하거나 event를 생성하지 않음	X	`eth_call`	`function balanceOf(address account) public view virtual override returns (uint256)`
`pure`	state variable 값을 읽지도 변경하지도 않음.	X	`eth_call`	`function add(uint256 a, uint256 b) public pure returns (uint256)`

Sample Contracts - `IERC20Base.sol`

- `interface IERC20Base`

```
// SPDX-License-Identifier: MIT
// https://eips.ethereum.org/EIPS/eip-20

pragma solidity ^0.8.0;

interface IERC20Base {

    event Transfer(address indexed _from, address indexed _to, uint256 _value);

    event Approval(address indexed _owner, address indexed _spender, uint256 _value);

    function totalSupply() external view returns (uint256);

    function balanceOf(address owner) external view returns (uint256 balance);

    function transfer(address to, uint256 value) external returns (bool success);

    function transferFrom(address from, address to, uint256 value) external returns (bool success);

    function approve(address spender, uint256 value) external returns (bool success);

    function allowance(address owner, address spender) external view returns (uint256 remaining);
}
```

Sample Contracts - `ERC20Base.sol`

- `abstract contract ERC20Base`

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./IERC20Base.sol";

abstract contract ERC20Base is IERC20Base {

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;
    uint256 private _totalSupply;

    function totalSupply() public view
          virtual override returns (uint256) {
        return _totalSupply;
    }

    function balanceOf(address account) public view
          virtual override returns (uint256) {
        return _balances[account];
    }

    function _transfer(address from,
          address to,
          uint256 amount) internal virtual {
        require(from != address(0),
            "ERC20: transfer from the zero address");
        require(to != address(0),
            "ERC20: transfer to the zero address");

        uint256 fromBalance = _balances[from];
        require(fromBalance >= amount,
            "ERC20: transfer amount exceeds balance");
        _balances[from] = fromBalance - amount;
        _balances[to] += amount;

        emit Transfer(from, to, amount);
    }
```

```solidity
    function transfer(address to, uint256 amount) public
          virtual override returns (bool) {
        address owner = msg.sender;
        _transfer(owner, to, amount);
        return true;
    }

    function transferFrom(address from, address to, uint256 amount)
          public virtual override returns (bool) {
        address spender = msg.sender;
        uint256 currentAllowance = allowance(from, spender);
        require(currentAllowance >= amount,
            "ERC20: insufficient allowance");
        _allowances[from][spender] = currentAllowance - amount;
        emit Approval(from, spender, currentAllowance - amount);

        _transfer(from, to, amount);
        return true;
    }

    function approve(address spender, uint256 amount)
          public virtual override returns (bool) {
        address owner = msg.sender;
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
        return true;
    }

    function allowance(address owner, address spender)
          public view virtual override returns (uint256) {
        return _allowances[owner][spender];
    }

    function _mint(address account, uint256 amount) internal virtual {
        require(account != address(0),
            "ERC20: mint to the zero address");
        _totalSupply += amount;
        unchecked { _balances[account] += amount; }
        emit Transfer(address(0), account, amount);
    }
}
```

Sample Contracts - `RegularERC20.sol`

- `contract RegularERC20`

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./ERC20Base.sol";
import "./ERC20Metadata.sol";

contract RegularERC20 is ERC20Base, ERC20Metadata {

    address private _admin;

    constructor(string memory name, string memory symbol) ERC20Metadata(name, symbol, 18){
        _admin = msg.sender;
    }

    function mint(address account, uint256 amount) public virtual{
        require(msg.sender == _admin, "RegularERC20: Must be an admin to mint.");
        _mint(account, amount);
    }

    function mint(address[10] memory accounts, uint256[10] memory amounts) public virtual{
        require(msg.sender == _admin, "RegularERC20: Must be an admin to mint.");
        for(uint256 i = 0; i < 10; i++){ _mint(accounts[i], amounts[i]); }
    }

    function admin() public view returns (address){
        return _admin;
    }
}
```

Using `RegularERC20`

```
first-contracts $ npm install @truffle/hdwallet-provider -D
...
```

```
truffle(development)> token = await RegularERC20.new("Color Token", "RGB", {from: accounts[0
...

truffle(development)> token.totalSupply()
BN { negative: 0, words: [ 0, <1 empty item> ], length: 1, red: null }
truffle(development)> token.mint(accounts[1], 15000, {from: accounts[0]})
...
//transaction receipt
truffle(development)> token.mint(accounts[2], 12000, {from: accounts[0]})
...
//transaction receipt
truffle(development)> token.totalSupply().then(t => t.toLocaleString())
'27000'
truffle(development)> token.balanceOf(accounts[1]).then(b => b.toLocaleString())
'15000'
truffle(development)> token.balanceOf(accounts[2]).then(b => b.toLocaleString())
'12000'
truffle(development)> token.transfer(accounts[1], 5000, {from: accounts[2]})
...
truffle(development)> token.balanceOf(accounts[1]).then(b => b.toLocaleString())
'20000'
truffle(development)> token.balanceOf(accounts[2]).then(b => b.toLocaleString())
'7000'
truffle(development)> token.allowance(accounts[1], accounts[3]).then(b => b.toLocaleString()
'0'


...
```