UNIVERSITY OF MANNHEIM

Public Blockchain: Exercises

Hands-on Intro to Blockchains

OpenZeppelin

# ERC Standards, Open Zeppelin

ERC20

Dr. Stefano Balietti

## Discord Channel of the Course

Most DAOs manage their community through a public discord channel, sometimes even used for off-chain voting.

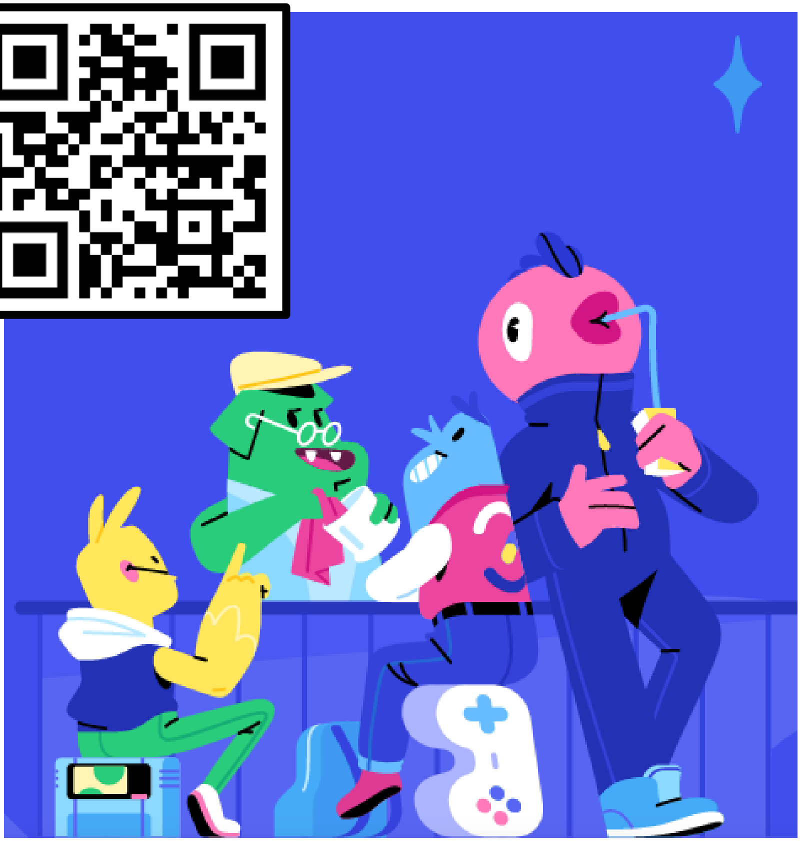
This course also has a Discord channel!

1. Download Discord: https://discord.com/
2. Course Channel:

**https://discord.gg/4xcnqVYb9y**

**What can you use the channel for?**

Ask questions, connect with other students, share news, events, and material related to the course, and as well off-topic posts.



## Recap: What did we learn last week?

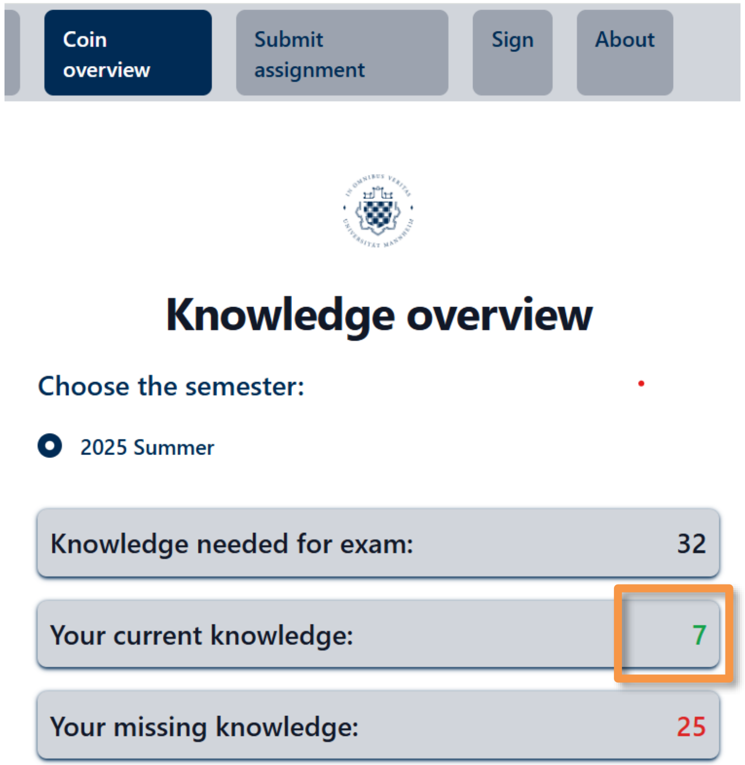
## Assignment 1: Statistics

21 addresses answered at least one correct answer
0 addresses answered incorrect answers only

13 out 21 addresses were also registered on ILIAS

**If you expected tokens and did not receive them, it’s time to check with your instructor!**

## How many coins do I have?



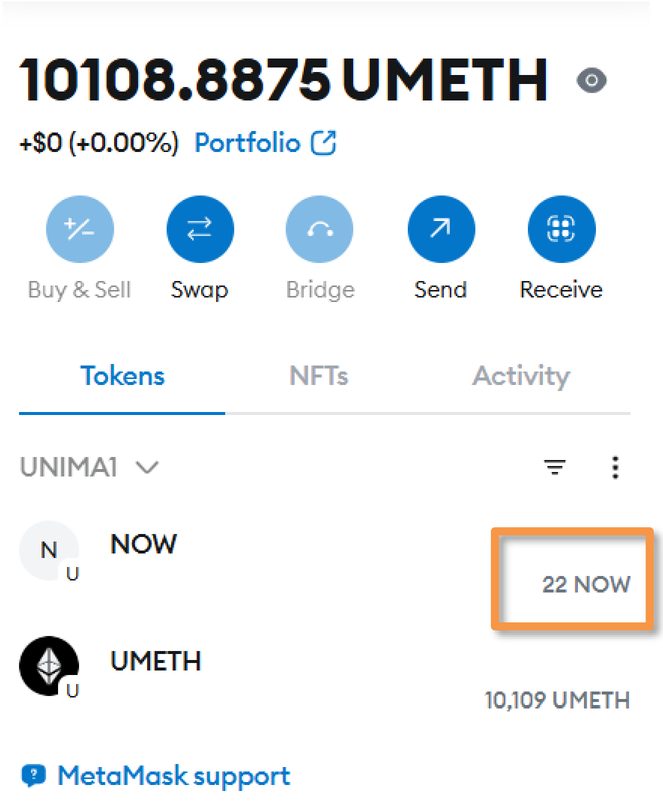
Coin overview | Submit assignment | Sign | About

**Knowledge overview**

Choose the semester:

● 2025 Summer

| | |
|---|---|
| Knowledge needed for exam: | 32 |
| Your current knowledge: | 7 |
| Your missing knowledge: | 25 |



10108.8875 UMETH

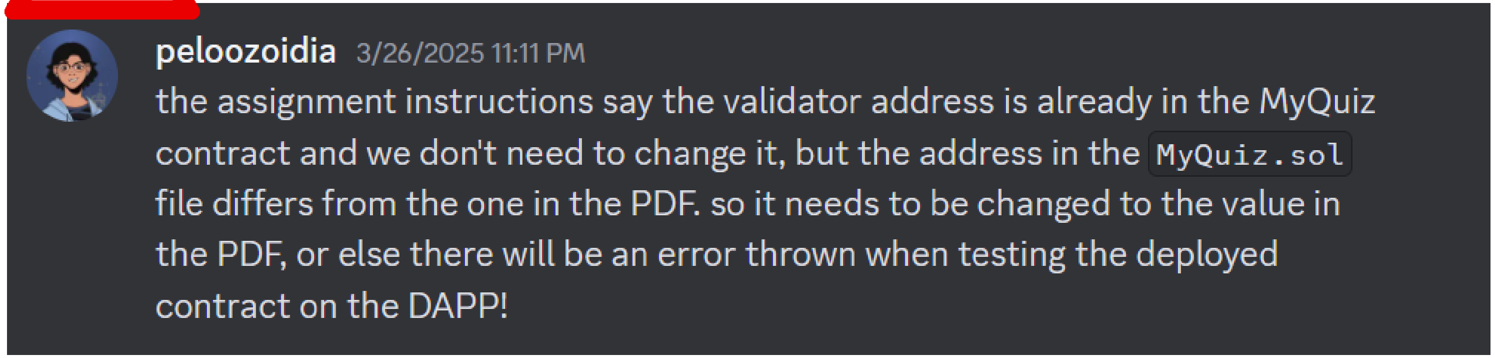
+$0 (+0.00%) Portfolio

| | |
|---|---|
| NOW | 22 NOW |
| UMETH | 10,109 UMETH |

Coin overview in Dapp shows your coin in *this semester*

Metamask shows all the coins that you have ever collected (the two values might differ if you are a returning student)

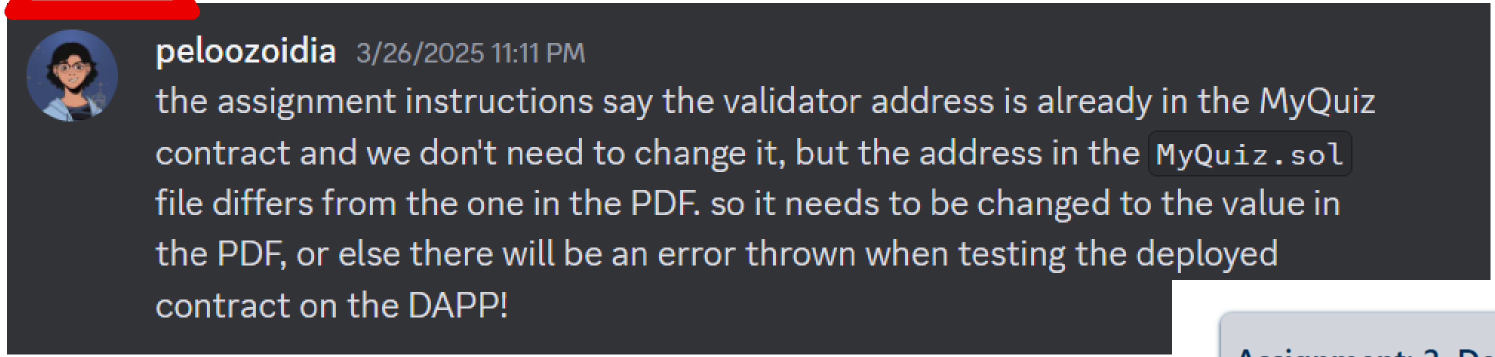
## Assignment 2 Attention!



peloozoidia 3/26/2025 11:11 PM

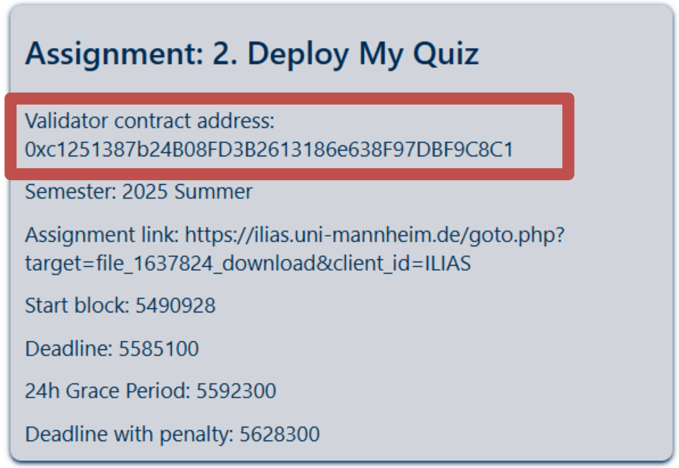
the assignment instructions say the validator address is already in the MyQuiz contract and we don't need to change it, but the address in the `MyQuiz.sol` file differs from the one in the PDF. so it needs to be changed to the value in the PDF, or else there will be an error thrown when testing the deployed contract on the DAPP!

## Assignment 2 Attention!


peloozoidia 3/26/2025 11:11 PM

the assignment instructions say the validator address is already in the MyQuiz contract and we don't need to change it, but the address in the `MyQuiz.sol` file differs from the one in the PDF. so it needs to be changed to the value in the PDF, or else there will be an error thrown when testing the deployed contract on the DAPP!

Make sure the contract address for BaseAssignment is the same as the one on the Submission Dapp


Assignment: 2. Deploy My Quiz

Validator contract address: 0xc1251387b24B08FD3B2613186e638F97DBF9C8C1

Semester: 2025 Summer

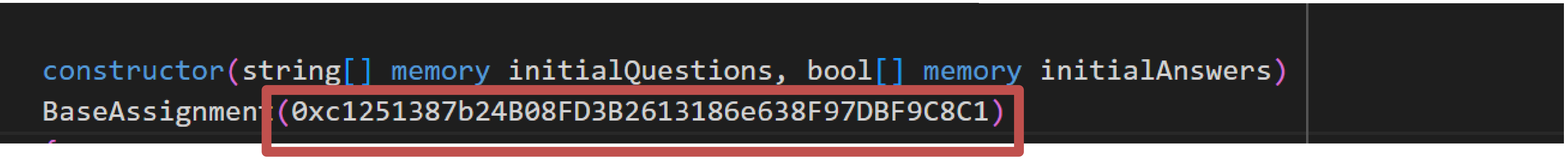
Assignment link: https://ilias.uni-mannheim.de/goto.php?target=file_1637824_download&client_id=ILIAS

Start block: 5490928

Deadline: 5585100

24h Grace Period: 5592300

Deadline with penalty: 5628300

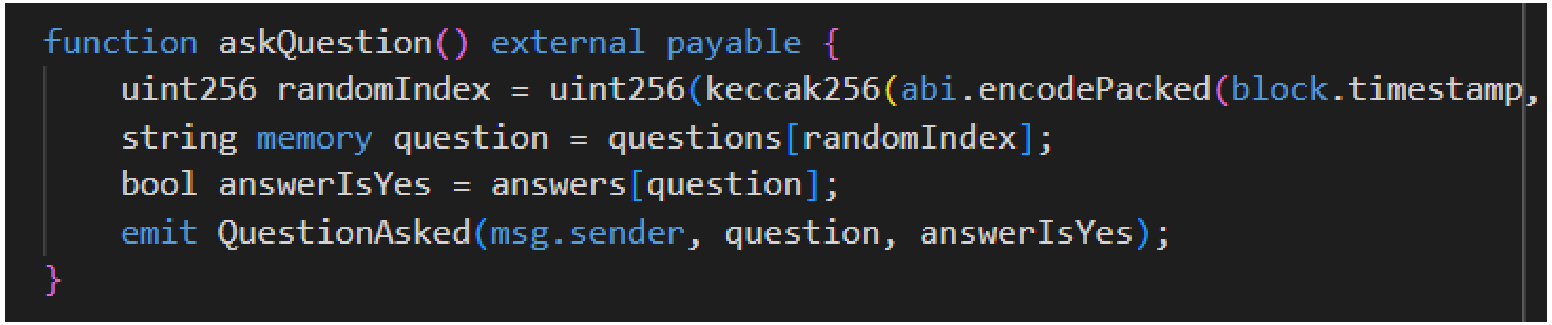

```
constructor(string[] memory initialQuestions, bool[] memory initialAnswers)
BaseAssignment(0xc1251387b24B08FD3B2613186e638F97DBF9C8C1)
```

## Recap: What did we learn last week?

- **What does *payable* means?**
- What does *require* do?
- What happens when a transaction *reverts*?
- What type of *variables* are there in solidity?
- What are *events*?
- What are *mappings*?

## Payable in the MyQuiz Contract

The method is accepting incoming Ether…how much did we send ?

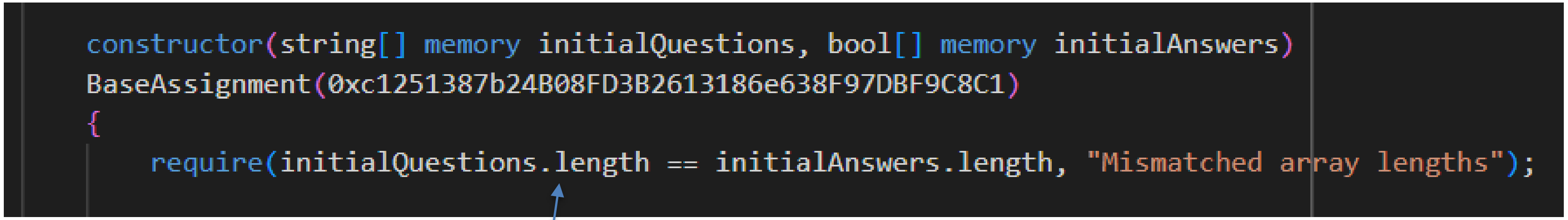


```
function askQuestion() external payable {
    uint256 randomIndex = uint256(keccak256(abi.encodePacked(block.timestamp,
    string memory question = questions[randomIndex];
    bool answerIsYes = answers[question];
    emit QuestionAsked(msg.sender, question, answerIsYes);
}
```

## Recap: What did we learn last week?

- What does *payable* means?
- **What does *require* do?**
- **What happens when a transaction *reverts*?**
- What type of *variables* are there in solidity?
- What are *events*?
- What are *mappings*?

## Require in the MyQuiz Contract



```
constructor(string[] memory initialQuestions, bool[] memory initialAnswers)
BaseAssignment(0xc1251387b24B08FD3B2613186e638F97DBF9C8C1)
{
    require(initialQuestions.length == initialAnswers.length, "Mismatched array lengths");
```

Number of answers and questions must match

## Recap: What did we learn last week?

- What does *payable* means?
- What does *require* do?
- What happens when a transaction *reverts*?
- **What type of *variables* are there in solidity?**
- What are *events*?
- What are *mappings*?

## What type of *variables* are there?

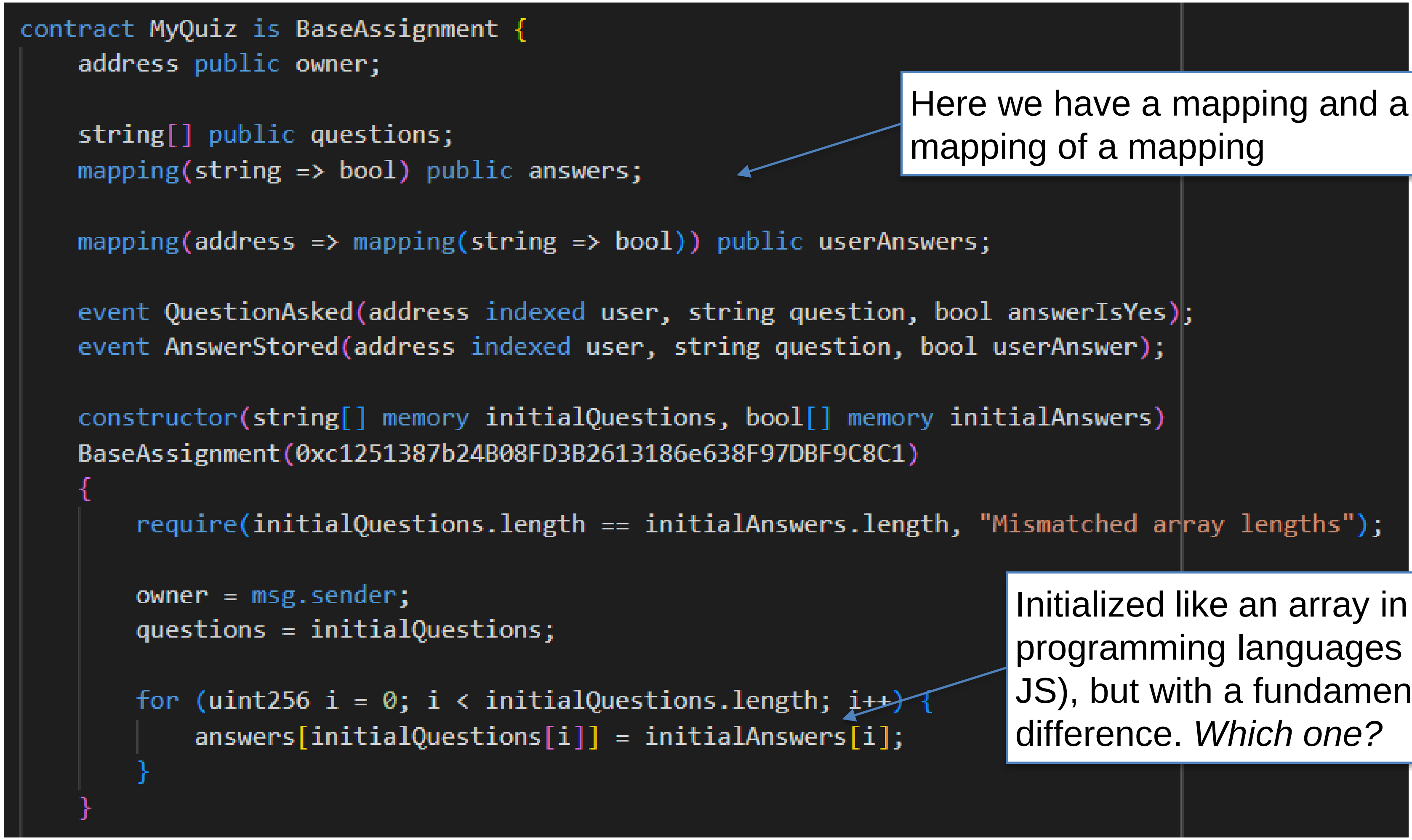
Solidity By Example is your friend!

https://solidity-by-example.org/variables/

## Recap: What did we learn last week?

- What does *payable* means?
- What does *require* do?
- What happens when a transaction *reverts*?
- What type of *variables* are there in solidity?
- **What are *events*?**
- **What are *mappings*?**

# Mappings in the MyQuiz Contract

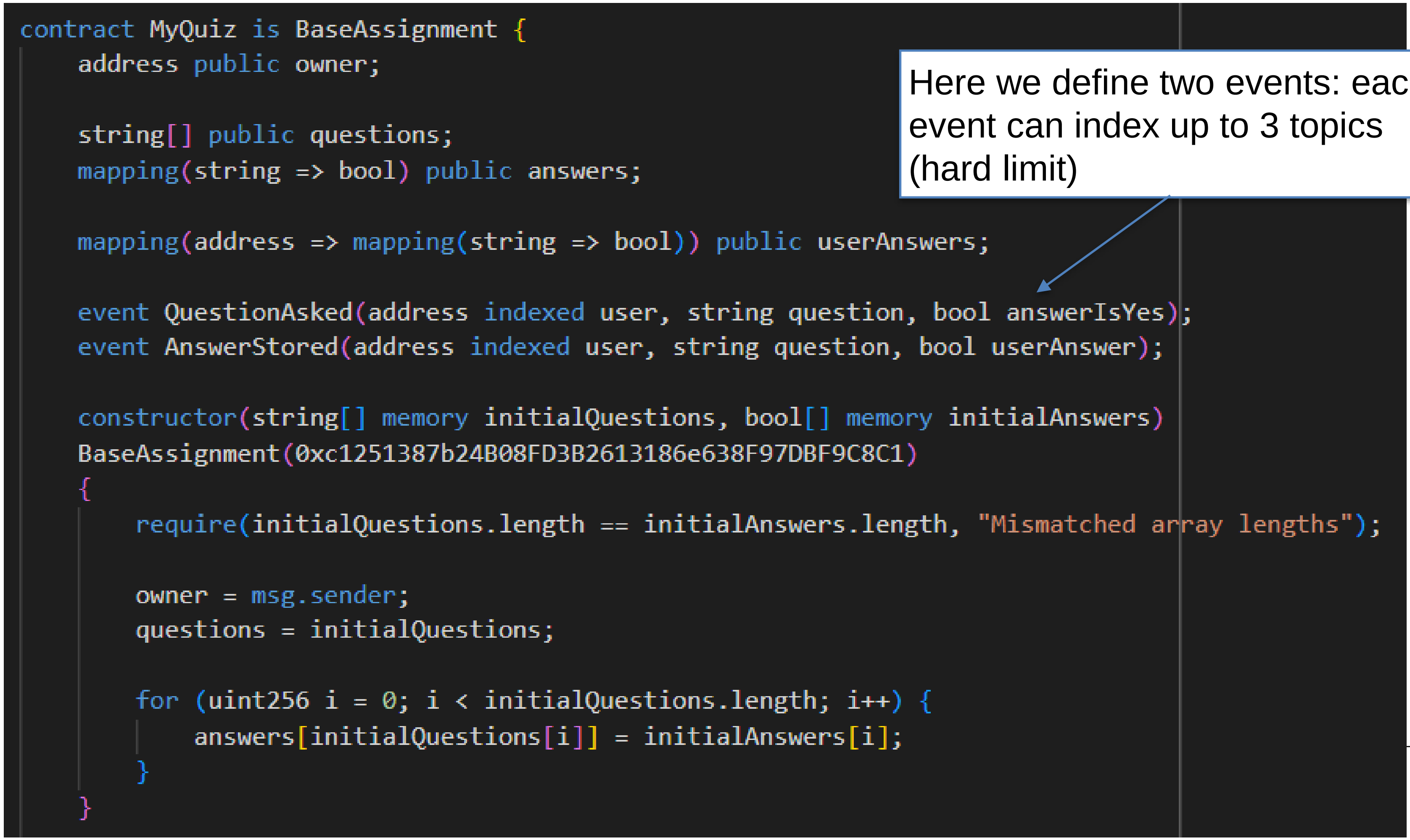


```
contract MyQuiz is BaseAssignment {
    address public owner;

    string[] public questions;
    mapping(string => bool) public answers;

    mapping(address => mapping(string => bool)) public userAnswers;

    event QuestionAsked(address indexed user, string question, bool answerIsYes);
    event AnswerStored(address indexed user, string question, bool userAnswer);

    constructor(string[] memory initialQuestions, bool[] memory initialAnswers)
    BaseAssignment(0xc1251387b24B08FD3B2613186e638F97DBF9C8C1)
    {
        require(initialQuestions.length == initialAnswers.length, "Mismatched array lengths");

        owner = msg.sender;
        questions = initialQuestions;

        for (uint256 i = 0; i < initialQuestions.length; i++) {
            answers[initialQuestions[i]] = initialAnswers[i];
        }
    }
```

Here we have a mapping and a mapping of a mapping

Initialized like an array in other programming languages (e.g., JS), but with a fundamental difference. *Which one?*

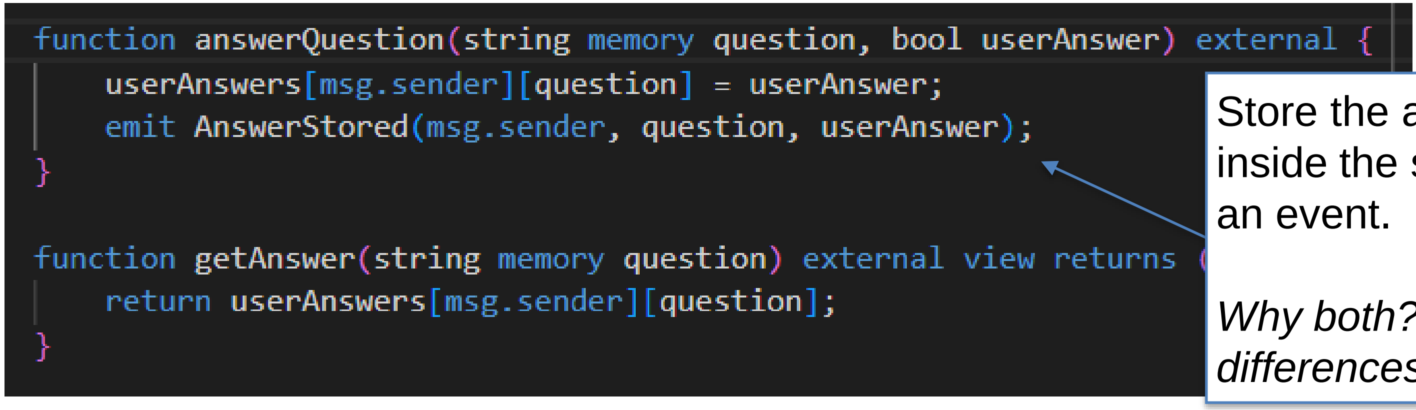
## Events in the MyQuiz Contract



```
contract MyQuiz is BaseAssignment {
    address public owner;

    string[] public questions;
    mapping(string => bool) public answers;

    mapping(address => mapping(string => bool)) public userAnswers;

    event QuestionAsked(address indexed user, string question, bool answerIsYes);
    event AnswerStored(address indexed user, string question, bool userAnswer);

    constructor(string[] memory initialQuestions, bool[] memory initialAnswers)
    BaseAssignment(0xc1251387b24B08FD3B2613186e638F97DBF9C8C1)
    {
        require(initialQuestions.length == initialAnswers.length, "Mismatched array lengths");

        owner = msg.sender;
        questions = initialQuestions;

        for (uint256 i = 0; i < initialQuestions.length; i++) {
            answers[initialQuestions[i]] = initialAnswers[i];
        }
    }
```

Here we define two events: each event can index up to 3 topics (hard limit)

## Mappings and Events in the MyQuiz Contract

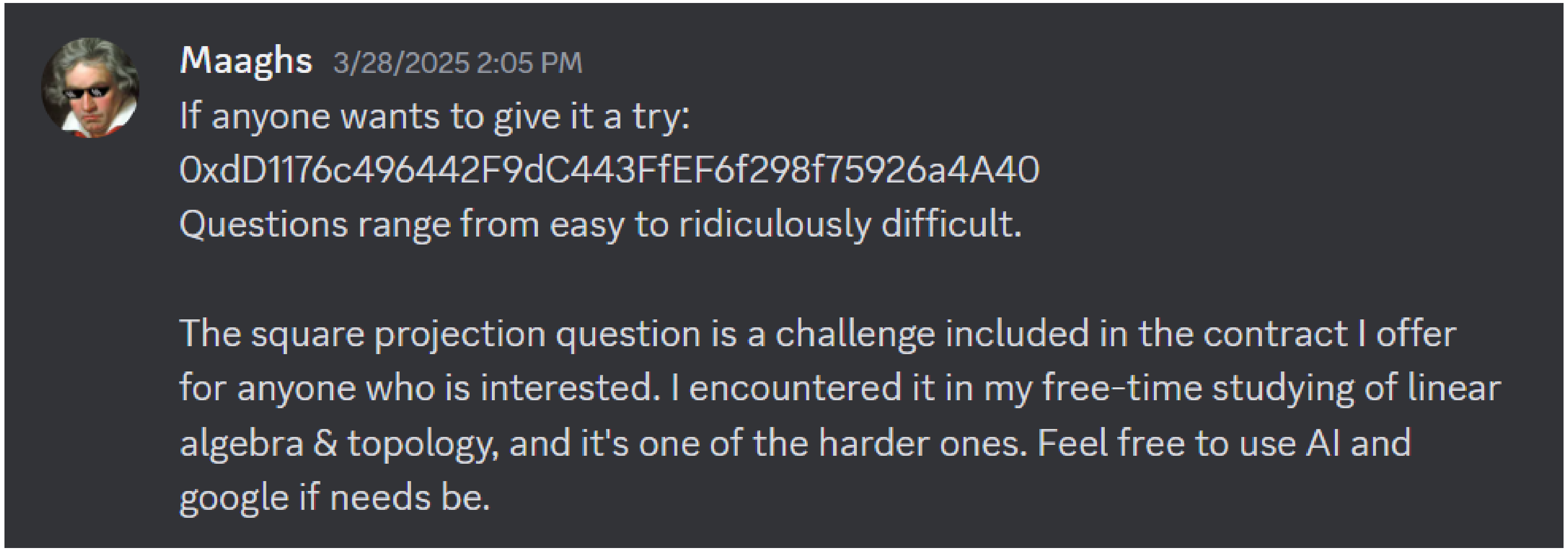


```
function answerQuestion(string memory question, bool userAnswer) external {
    userAnswers[msg.sender][question] = userAnswer;
    emit AnswerStored(msg.sender, question, userAnswer);
}

function getAnswer(string memory question) external view returns (
    return userAnswers[msg.sender][question];
}
```

Store the answer in the mapping inside the smart contract AND in an event.

*Why both? What are the differences?*

## Some Fun!

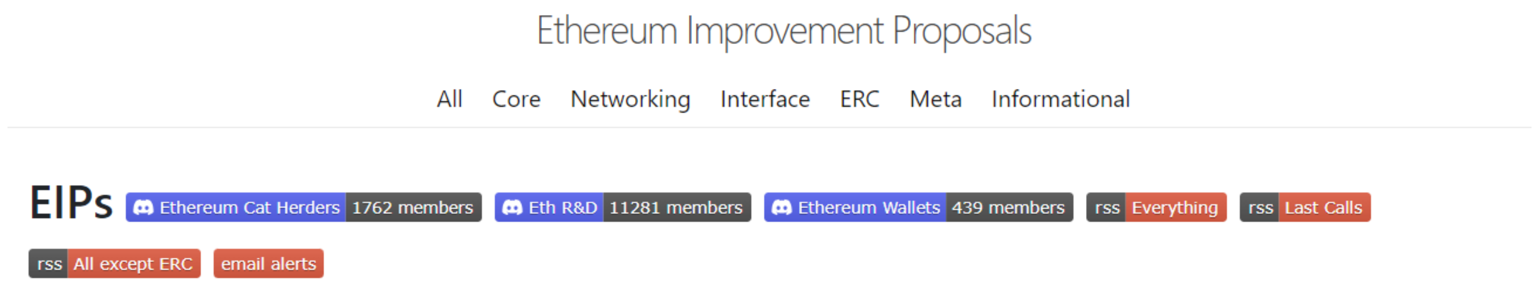

Maaghs 3/28/2025 2:05 PM

If anyone wants to give it a try:
0xdD1176c496442F9dC443FfEF6f298f75926a4A40
Questions range from easy to ridiculously difficult.

The square projection question is a challenge included in the contract I offer for anyone who is interested. I encountered it in my free-time studying of linear algebra & topology, and it's one of the harder ones. Feel free to use AI and google if needs be.

| # | Date | Content | Details | Assignment |
|---|---|---|---|---|
| 1 | 25 Feb | Basic Intro | Intro, Setting Up MetaMask, Connecting to Faucets, Transactions, Wallet Mnemonic, Keypairs, ENS. | **0** |
| 2 | 4 Mar | Setup, Intro EthersJS | Setting up local environment, Basic programming, Ethers.JS | |
| 3 | 11 Mar | EthersJS | Ethers.JS Wallets, Dotenv | |
| 4 | 18 Mar | EthersJS | Ethers.JS: Providers and Signers | **1** |
| 5 | 25 Mar | Hardhat, Solidity | First steps with Hardhat, Basic contract in Solidity. | |
| 6 | 26 Mar | Solidity | Review Assignment 1. Mappings, data structures, payable, modifiers. UniMa Blockchain Submission System. | **2** |
| 7 | 1 Apr | ERC Standards | ERC standards, Open Zeppelin | |
| 8 | 8 Apr | Testing | Review Assignment 2. Testing smart contracts. | **3** |
| 9 | 29 Apr | Keccak256 | Keccak256, abiEncode, abiEncodePacked | **4** |
| 10 | 6 May | ABI Encodings | Review Assignment 3. Raw Transactions | |
| 11 | 6 May | Contract to Contract | Contract to Contract | |
| 12 | 13 May | Upgradable Contracts | Review Assignment 4. Static calls, proxy contracts, implementations, storage clashes | **5** |
| 13 | 20 May | Exam Preparation | Conducting exercises like what could be found in the exam | |
| 14 | 27 May | Optimizing Solidity | Review Assignment 5. Optimization, open issues. | |

## EIP: Ethereum Improvement Proposal


Ethereum Improvement Proposals

All Core Networking Interface ERC Meta Informational

EIPs Ethereum Cat Herders 1762 members Eth R&D 11281 members Ethereum Wallets 439 members rss Everything rss Last Calls rss All except ERC email alerts

https://eips.ethereum.org/

- Document providing info to the Ethereum community or **describing a new feature**
- Should provide a **concise technical specification** of the feature and a rationale for it
- Several **approval steps** from “Idea” to “Final”

See also: https://ethereum.org/en/eips/

Bitcoin has BIPs

## Main EIP Types

- **Core**: improvements requiring a consensus fork (e.g. EIP-5, EIP-101)
- **Networking**: improvements around transport protocol among Ethereum nodes (e.g., EIP-8)
- **Interface**: improvements around client API/RPC calls (e.g., EIP-6), and contract ABIs.
- **ERC**: *application-level* standards and conventions, including contract standards such as token standards (ERC-20), name registries (ERC-137), URI schemes, library/package formats, and wallet formats.

## EIP Types

ERC stands Ethereum Requests for Comments.
token interfaces. These standards help ensure smart contracts remain composable, so for instance when a new project issues a token, that it remains compatible with existing decentralized exchanges

## ERC: Request for Comments

**Tokens:**

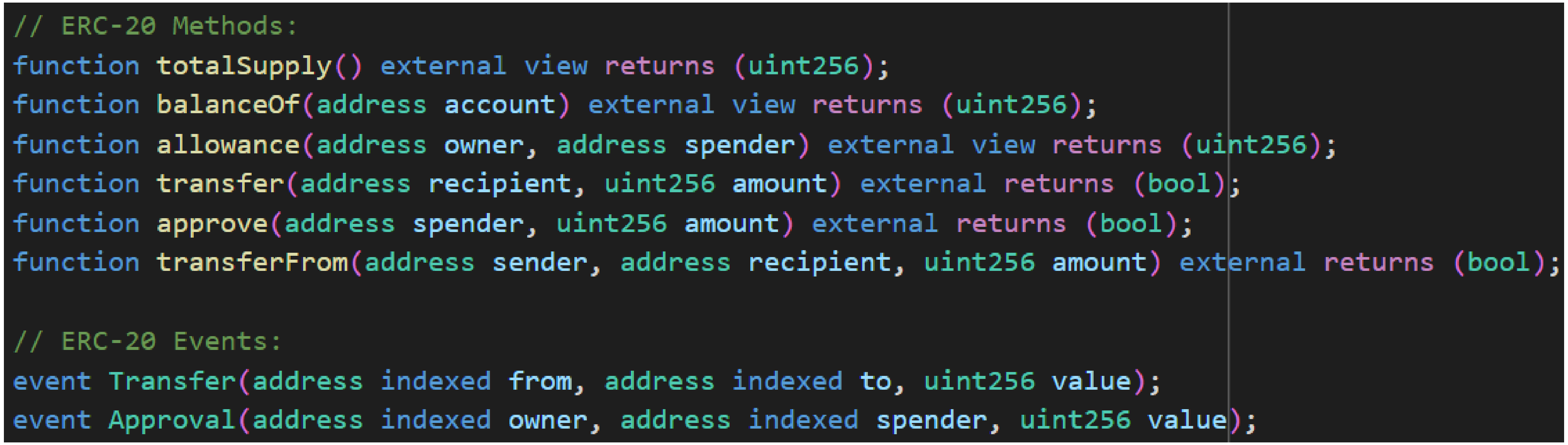
- ERC-20 - A standard interface for fungible (interchangeable) tokens, like voting tokens, staking tokens or virtual currencies.
- ERC-721 - A standard interface for non-fungible tokens, like a deed for artwork or a song.

And more: https://ethereum.org/en/developers/docs/standards/tokens/

More info also here: https://medium.com/codex/token-standards-erc-20-vs-erc-721-vs-erc-1155-2e4a09dc0f8a

## ERC 20

A smart contract that implements the following **6 methods** and **2 events** is an *ERC-20 compatible token*:

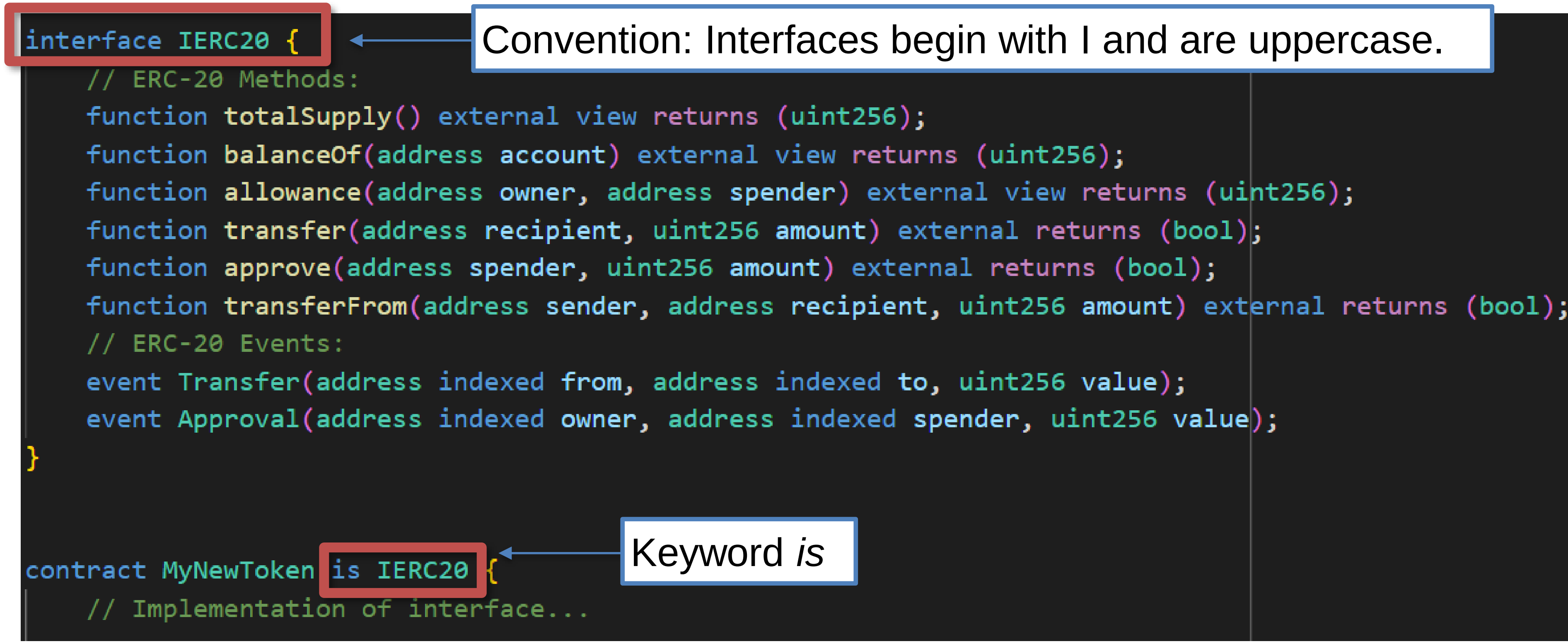


```
// ERC-20 Methods:
function totalSupply() external view returns (uint256);
function balanceOf(address account) external view returns (uint256);
function allowance(address owner, address spender) external view returns (uint256);
function transfer(address recipient, uint256 amount) external returns (bool);
function approve(address spender, uint256 amount) external returns (bool);
function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

// ERC-20 Events:
event Transfer(address indexed from, address indexed to, uint256 value);
event Approval(address indexed owner, address indexed spender, uint256 value);
```

https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

## Interfaces

A “standard” approach to implement a standard is by defining an interface and then declaring that the contract `is` implementing



```
interface IERC20 {
    // ERC-20 Methods:
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function allowance(address owner, address spender) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    // ERC-20 Events:
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}


contract MyNewToken is IERC20 {
    // Implementation of interface...
```

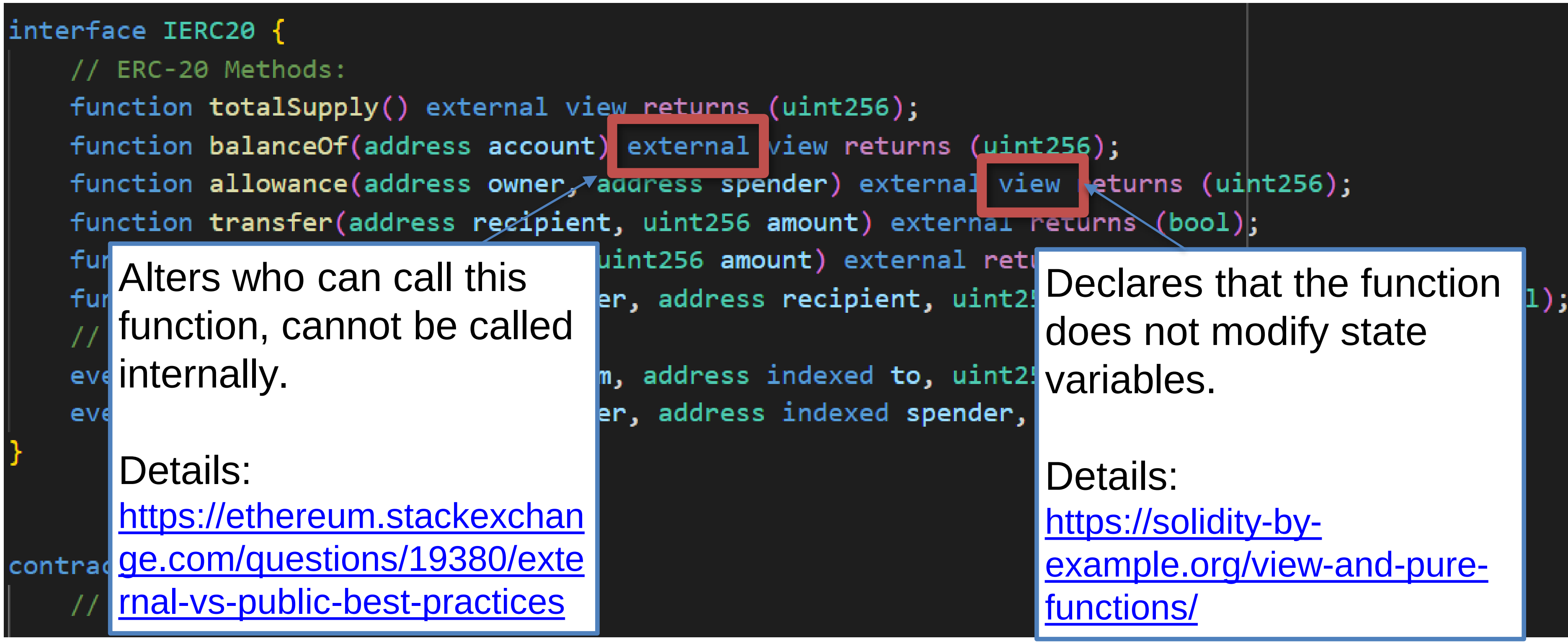
Convention: Interfaces begin with I and are uppercase.

Keyword *is*

*See full code in Exercises Repo*

## Function Modifiers

- Keywords added after the input parameters.
- Alter function’s behavior, usually for optimization or access control.



```
interface IERC20 {
    // ERC-20 Methods:
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function allowance(address owner, address spender) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    fun                 uint256 amount) external retu
    fun                 er, address recipient, uint25                    1);
    //
    eve                 m, address indexed to, uint25
    eve                 er, address indexed spender,
}


contrac
    //
```

Alters who can call this function, cannot be called internally.

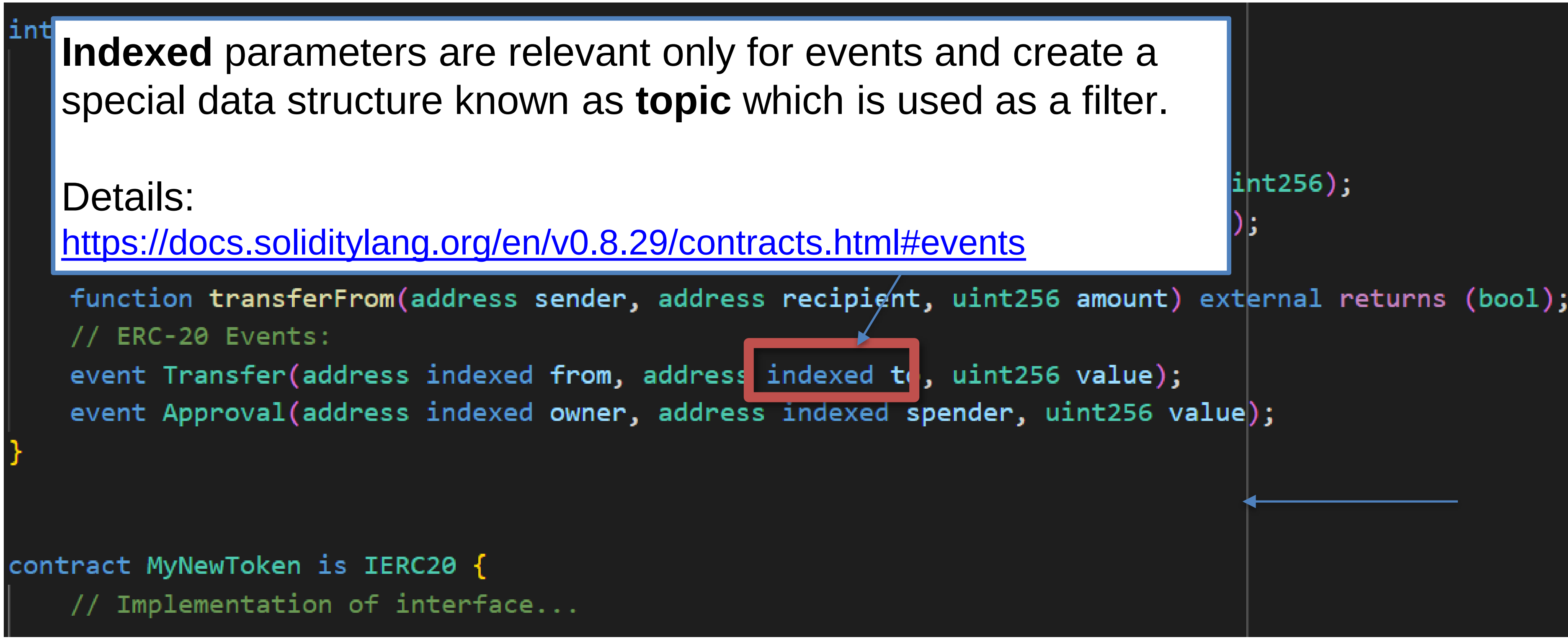
Details: [https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices](https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices)

Declares that the function does not modify state variables.

Details: [https://solidity-by-example.org/view-and-pure-functions/](https://solidity-by-example.org/view-and-pure-functions/)

## Function Modifiers

- Keywords added after the input parameters.
- Alter function's behavior, usually for optimization or access control.



**Indexed** parameters are relevant only for events and create a special data structure known as **topic** which is used as a filter.

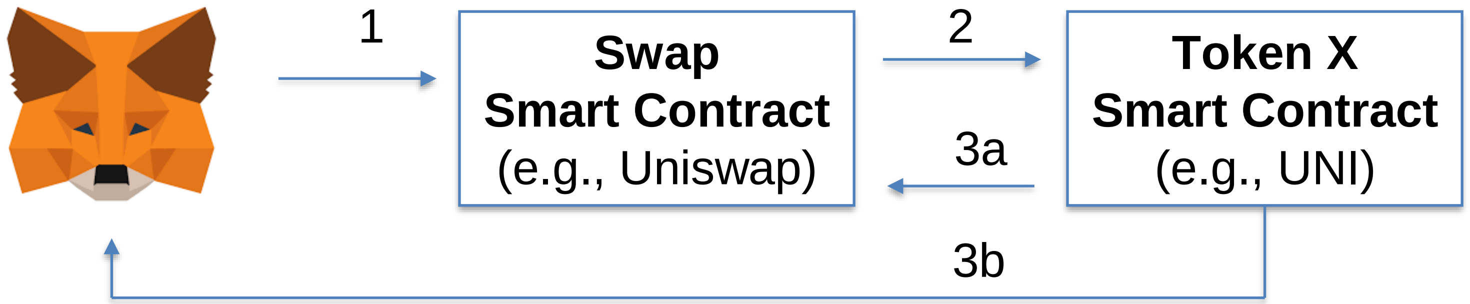
Details:
https://docs.soliditylang.org/en/v0.8.29/contracts.html#events

```
int
                                                                              int256);
                                                                              );
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    // ERC-20 Events:
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}


contract MyNewToken is IERC20 {
    // Implementation of interface...
```

## Understanding the ERC-20‘s Approve Pattern

https://eips.ethereum.org/EIPS/eip-20

The **Approve** pattern allows smart contracts to *spend your tokens.*

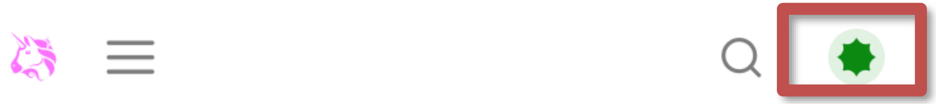


## Exercise: Testing Approve Pattern on Uniswap (Sepolia)

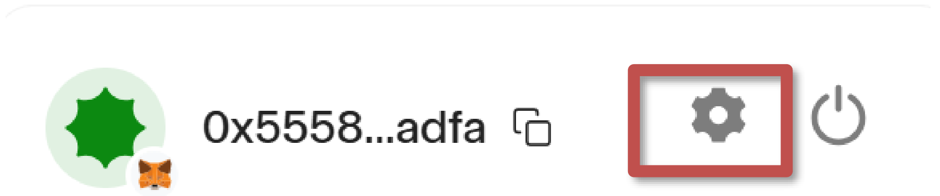
0) Select the Sepolia Network on MetaMask; you will need *some Sepolia ETH*.

1) Go to https://app.uniswap.com/swap

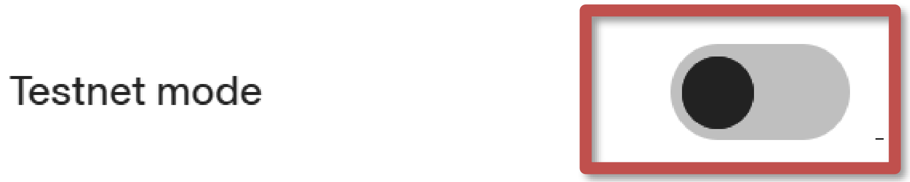
2) Tap the “green sun” for “Account”



3) Tap the gear icon for “Settings”



4) Enable Testnet mode

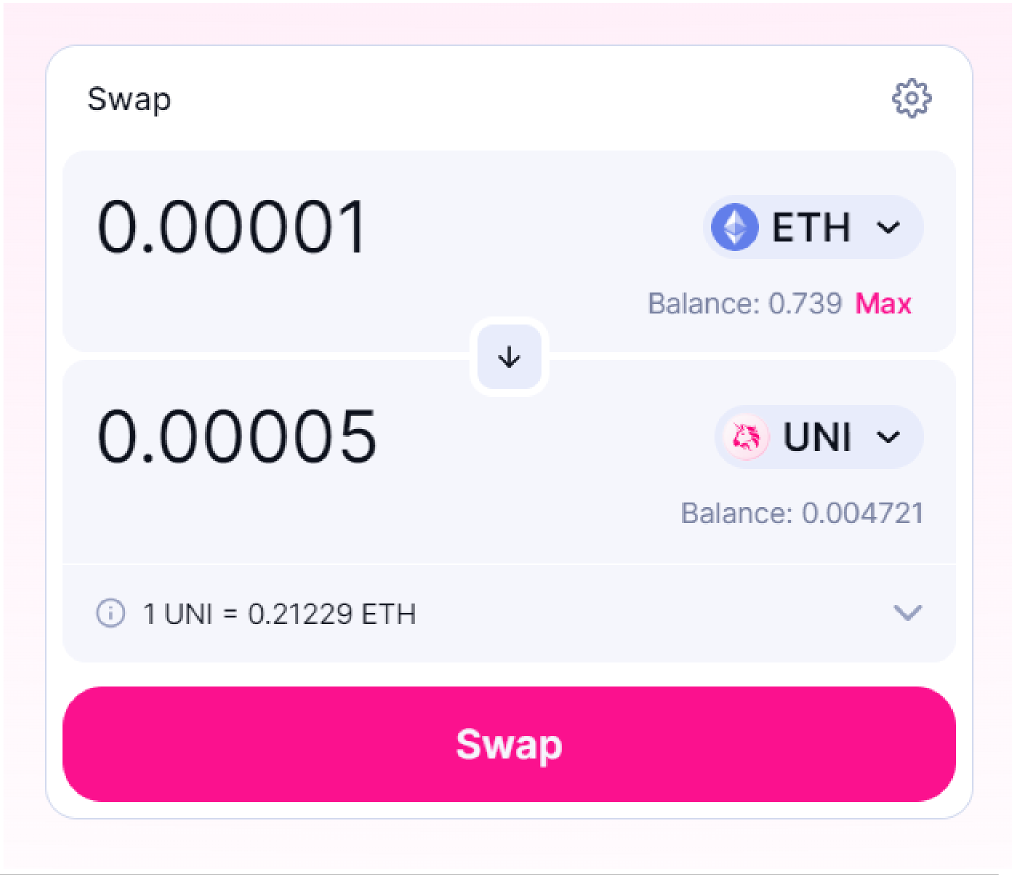


## Exercise: Testing Approve Pattern on Uniswap (Sepolia)

**ETH to UNI**

**Swap** some ETH for UNI (you might need to select UNI from a dropdown menu)

Click on Swap and start the transaction.


Swap

0.00001 ETH

Balance: 0.739 Max

0.00005 UNI

Balance: 0.004721

1 UNI = 0.21229 ETH

## Exercise: Testing Approve Pattern on Uniswap (Sepolia)

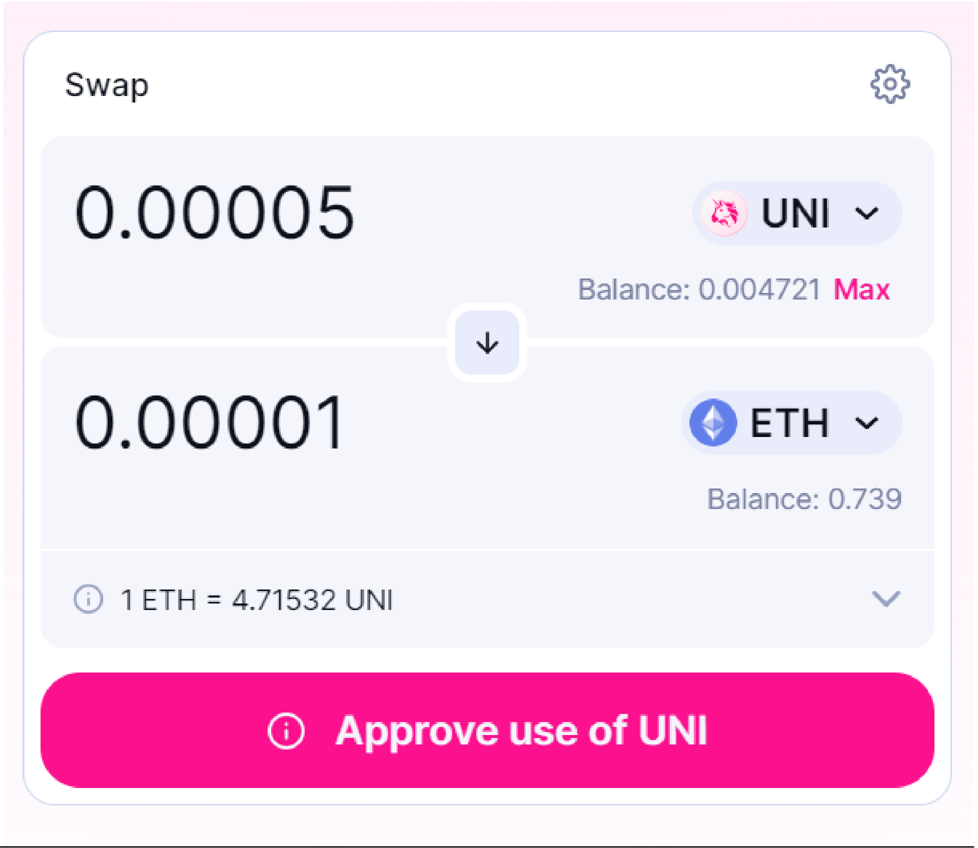
**UNI to ETH**

You got UNI, now do the opposite trade

Swap some UNI for ETH

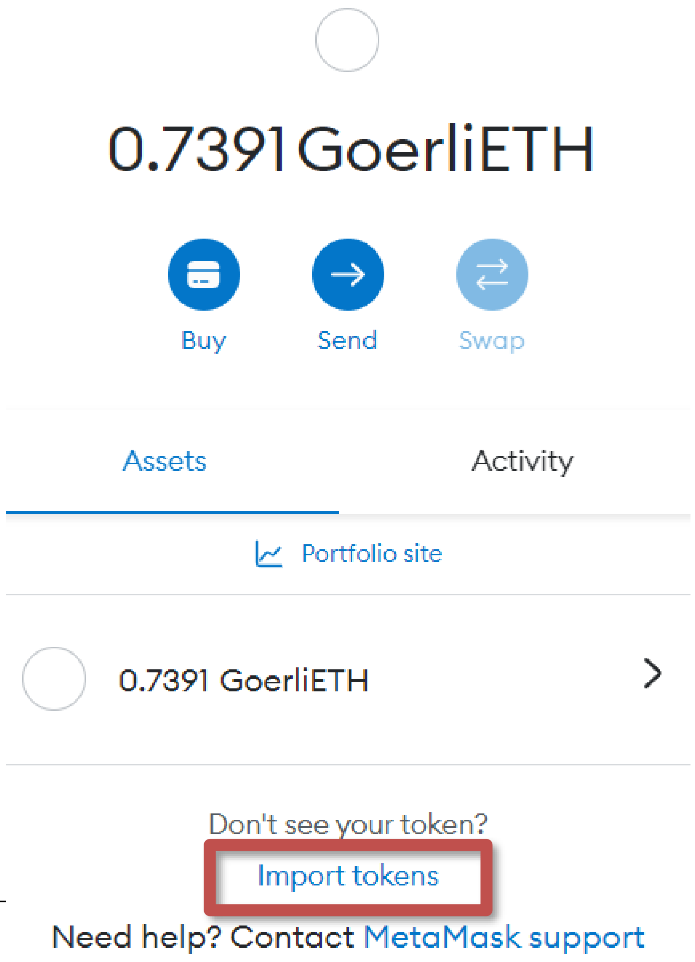
You need now to do **two** transactions:

1. **approve** first the Uniswap smart contract to take your UNI
2. **swap** UNI for ETH



## Add UNI to Metamask

- By default, Metamask will **NOT** show you your newly swapped UNI.
- You need to **manually** import the token into Metamask, so that it will track your balance.
- To add it, you need to know the **contract address** of the UNI token.
- You can easily find the contract address from the transaction page on **Etherscan**.

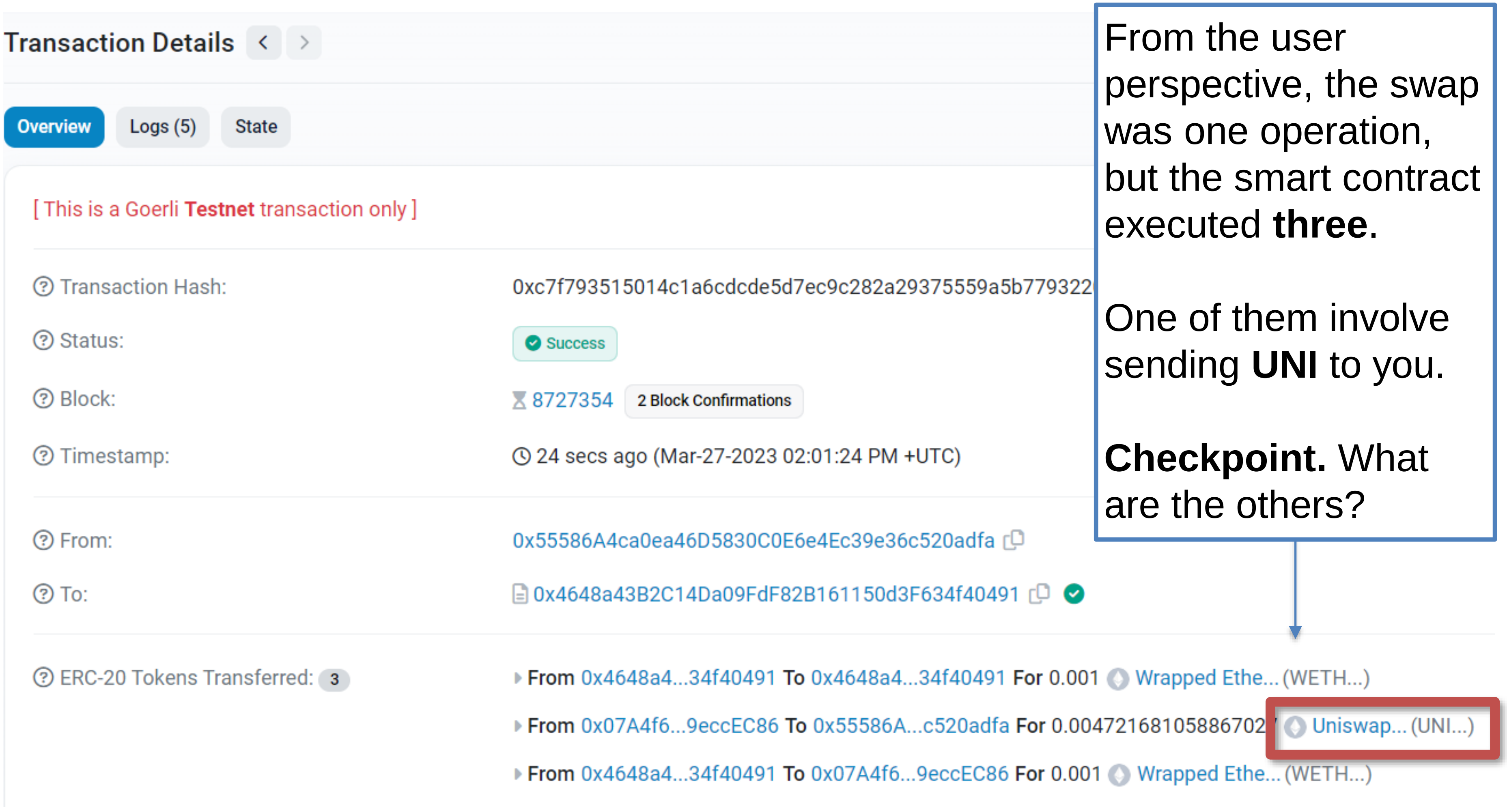

0.7391 GoerliETH

0.7391 GoerliETH

Don't see your token?

Need help? Contact MetaMask support

## Add UNI to Metamask



Transaction Details

Overview | Logs (5) | State

[ This is a Goerli **Testnet** transaction only ]

Transaction Hash: 0xc7f793515014c1a6cdcde5d7ec9c282a29375559a5b779322

Status: Success

Block: 8727354 2 Block Confirmations

Timestamp: 24 secs ago (Mar-27-2023 02:01:24 PM +UTC)

From: 0x55586A4ca0ea46D5830C0E6e4Ec39e36c520adfa

To: 0x4648a43B2C14Da09FdF82B161150d3F634f40491

ERC-20 Tokens Transferred: 3

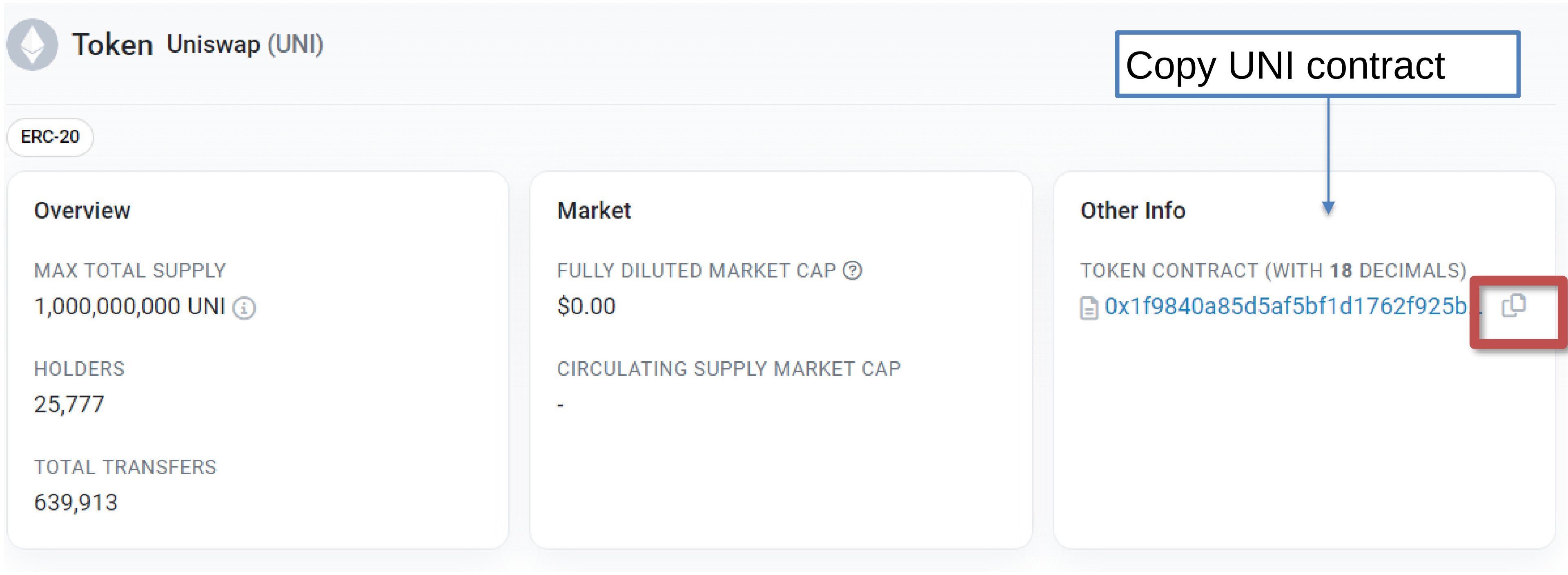
- **From** 0x4648a4...34f40491 **To** 0x4648a4...34f40491 **For** 0.001 Wrapped Ethe... (WETH...)
- **From** 0x07A4f6...9eccEC86 **To** 0x55586A...c520adfa **For** 0.004721681058867027 Uniswap... (UNI...)
- **From** 0x4648a4...34f40491 **To** 0x07A4f6...9eccEC86 **For** 0.001 Wrapped Ethe... (WETH...)

From the user perspective, the swap was one operation, but the smart contract executed **three**.

One of them involve sending **UNI** to you.

**Checkpoint.** What are the others?

## Add UNI to Metamask



Token Uniswap (UNI)

Copy UNI contract

ERC-20

**Overview**

MAX TOTAL SUPPLY
1,000,000,000 UNI

HOLDERS
25,777

TOTAL TRANSFERS
639,913

**Market**

FULLY DILUTED MARKET CAP
$0.00

CIRCULATING SUPPLY MARKET CAP
-

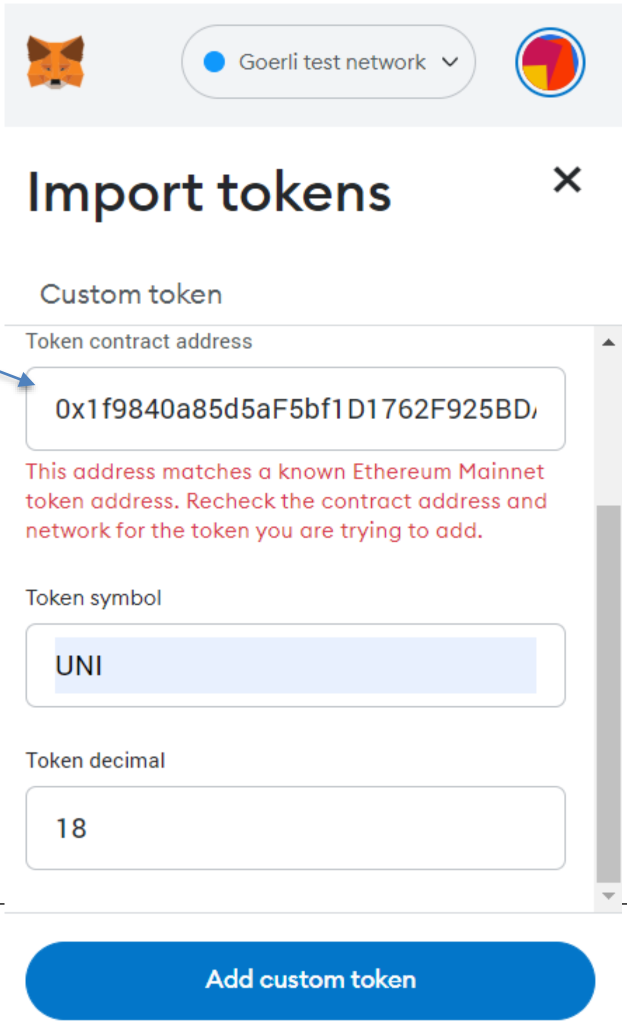
**Other Info**

TOKEN CONTRACT (WITH 18 DECIMALS)
0x1f9840a85d5af5bf1d1762f925b

*Note: contract address may differ*

## Add UNI to Metamask

UNI contract



Goerli test network

Import tokens

Custom token

Token contract address

0x1f9840a85d5aF5bf1D1762F925BD/

This address matches a known Ethereum Mainnet token address. Recheck the contract address and network for the token you are trying to add.

Token symbol

UNI

Token decimal

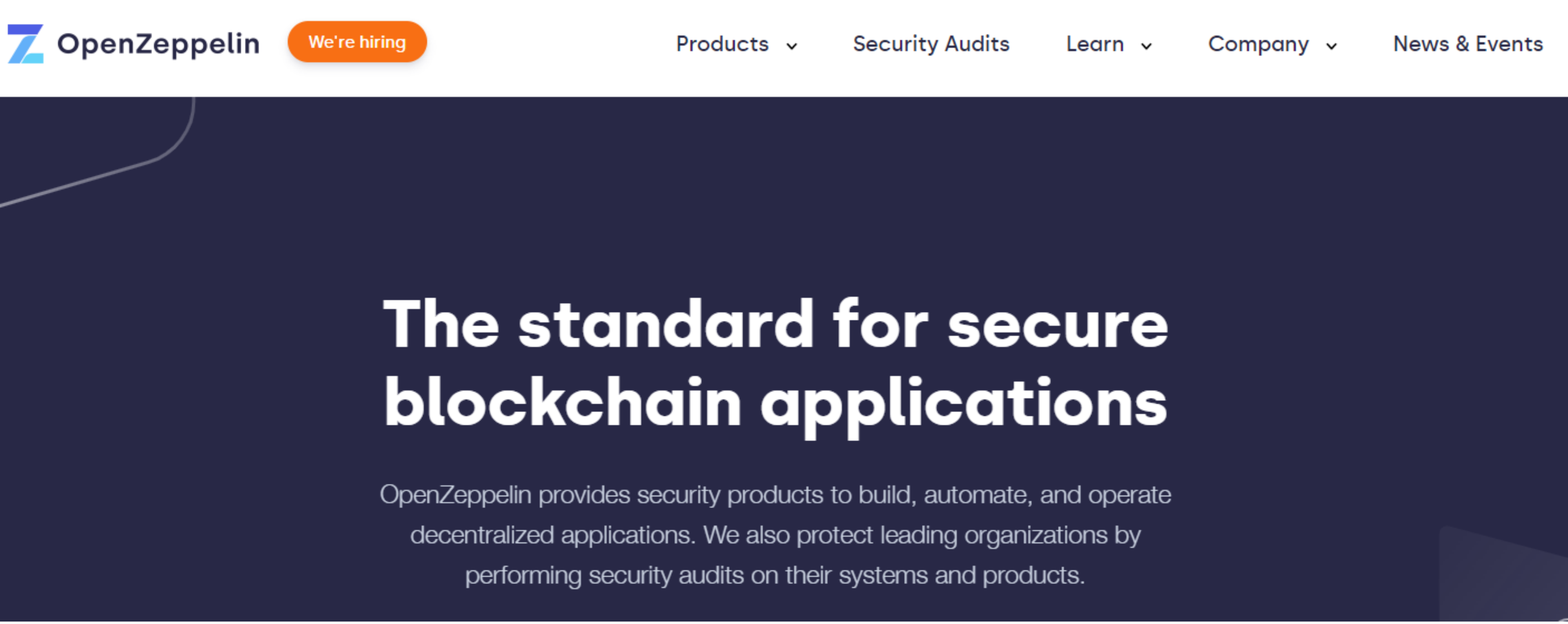
18

**BONUS:**
You can also get some SEPOLIA LINK and swap it:
[https://faucets.chain.link/sepolia](https://faucets.chain.link/sepolia)

**Checkpoint.**
If you switch LINK to UNI how many approvals are required?

*Note: contract address may differ*

## Open Zeppelin



The standard for secure blockchain applications

OpenZeppelin provides security products to build, automate, and operate decentralized applications. We also protect leading organizations by performing security audits on their systems and products.

https://openzeppelin.com

## Open Zeppelin Contracts Library

Audited and reliable, Open Zeppelin offers a collection of:

- Implementations of standards like ERC20 and ERC721
- Flexible role-based permissioning scheme
- Reusable Solidity components

To use it in your application, you need to install it:

```
npm install @openzeppelin/contracts
```

You can also install it with yarn

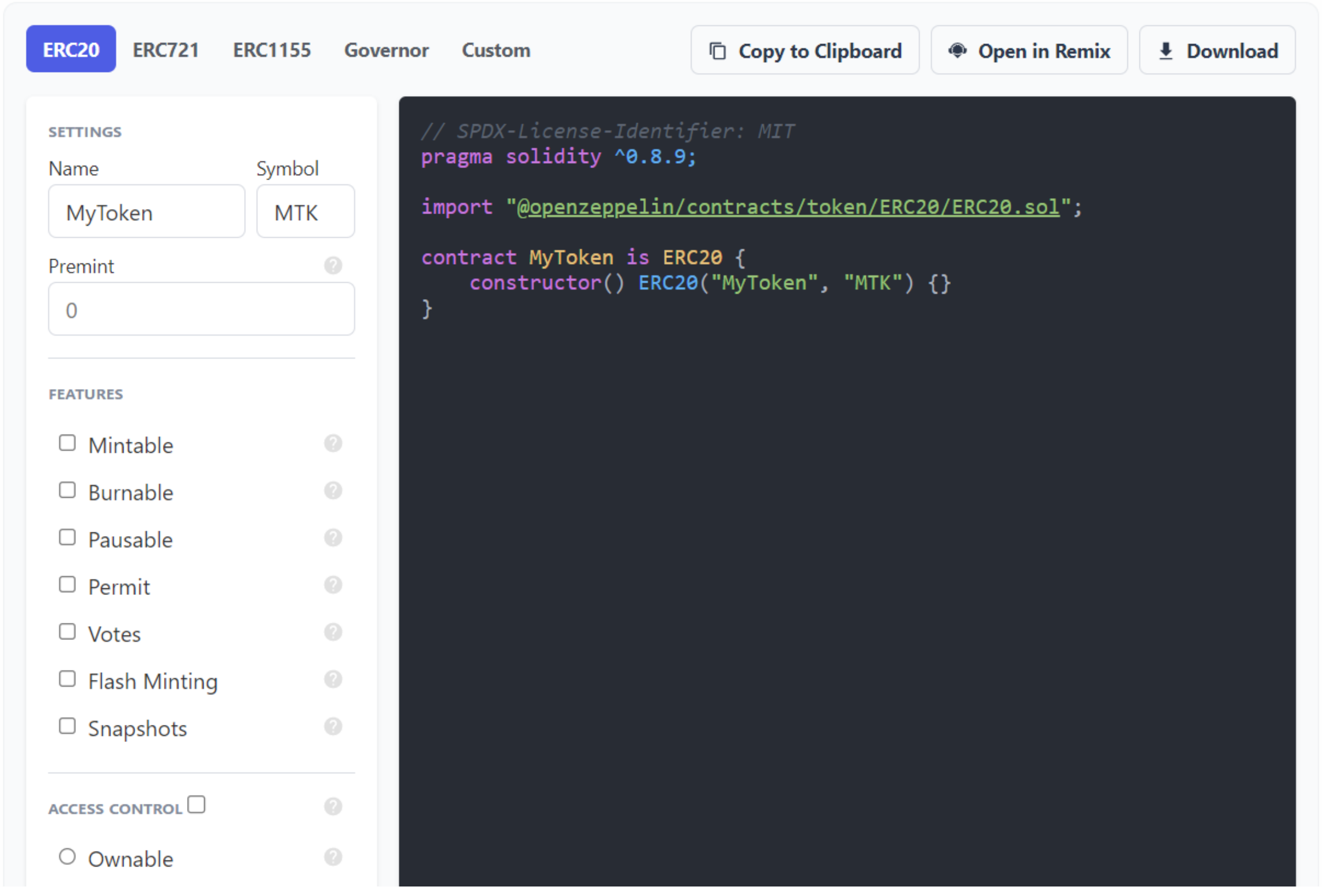
Details: https://docs.openzeppelin.com/contracts/5.x

## Open Zeppelin Contracts Library

- Let's play with Open Zeppelin Wizard to understand what this library offers.

**Exercise:**

- Deploy an Open Zeppelin ERC20 contract to the (not) UniMa blockchain.



SETTINGS

Name: MyToken

Symbol: MTK

Premint: 0

FEATURES

- ☐ Mintable
- ☐ Burnable
- ☐ Pausable
- ☐ Permit
- ☐ Votes
- ☐ Flash Minting
- ☐ Snapshots

ACCESS CONTROL ☐

- ○ Ownable

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract MyToken is ERC20 {
    constructor() ERC20("MyToken", "MTK") {}
}
```

[https://docs.openzeppelin.com/contracts/5.x/wizard](https://docs.openzeppelin.com/contracts/5.x/wizard)

## Time for Exercises…