Centurion University Shaping Lives... Empowering Communities...	School: .... Campus: .... Academic Year: .... Subject Name: .... Subject Code: .... Semester: .... Program: .... Branch: .... Specialization: .... Date: ....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Read the Chain – Web3.js Basics

Objective/Aim:

To read blockchain data using **Web3.js** by connecting to an Ethereum blockchain node and fetching details like the latest block, block number, and account balances.

Apparatus/Software Used:

- Node.js
- Web3.js library
- Ethereum test network (e.g., Ganache / Sepolia / Goerli)
- MetaMask (optional, for account management)
- VS Code or any code editor

Theory/Concept:

Web3.js is a JavaScript library that allows interaction with Ethereum blockchain nodes using RPC (Remote Procedure Calls).

- **Blockchain** is a decentralized ledger consisting of blocks.
- Each block has a block number, timestamp, transactions, and hash.
- Using **Web3.js**, developers can:
  - Connect to an Ethereum node (via HTTP or WebSocket).
  - Read chain data such as current block number, gas price, or balances.
  - Interact with smart contracts.

In this experiment, we focus on **reading the chain**, i.e., fetching basic blockchain information.

Procedure:

Step 1: Write Token Contract (in Remix IDE)
Step 2: Open Remix IDE → Select compiler **0.8.x** → Compile MyToken.sol.
Step 3: **Deploy Token**

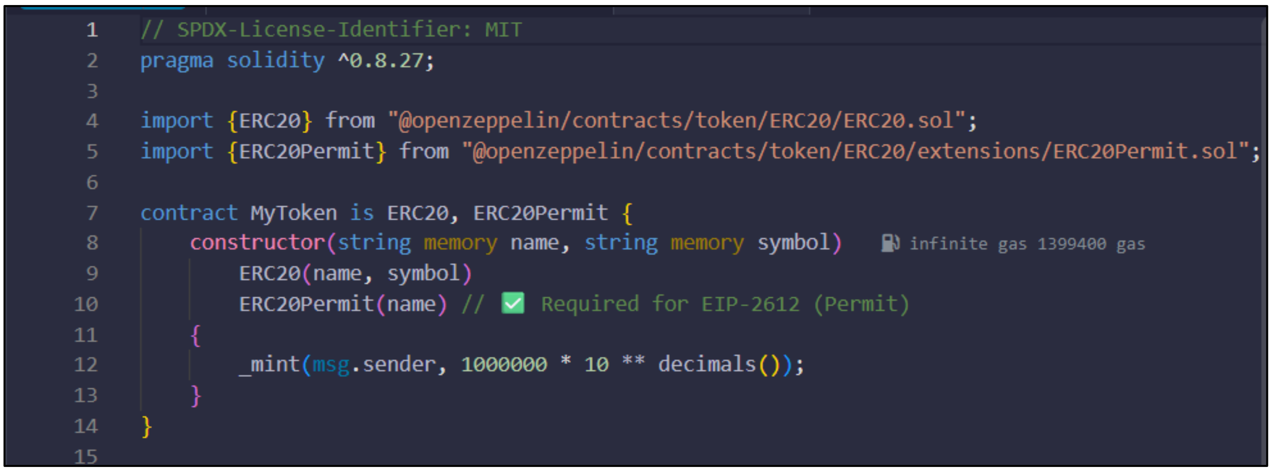
- In Remix → "Deploy & Run Transactions" tab.
- Select **Injected Web3** (MetaMask connected to local testnet).

Step 4: **Verify on Avee.net**

- Copy deployed contract address.
- Go to **Avee.net** → Add Token → Paste contract address.
- Token appears in the dashboard.

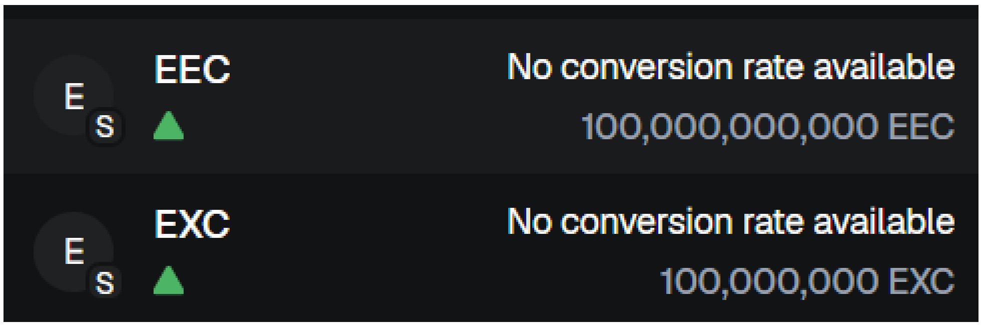
Step 5: **Interact with Token**

  - Check total supply.
  - Transfer tokens between MetaMask accounts.
  - Use Avee.net to approve and transfer tokens on behalf of another account.



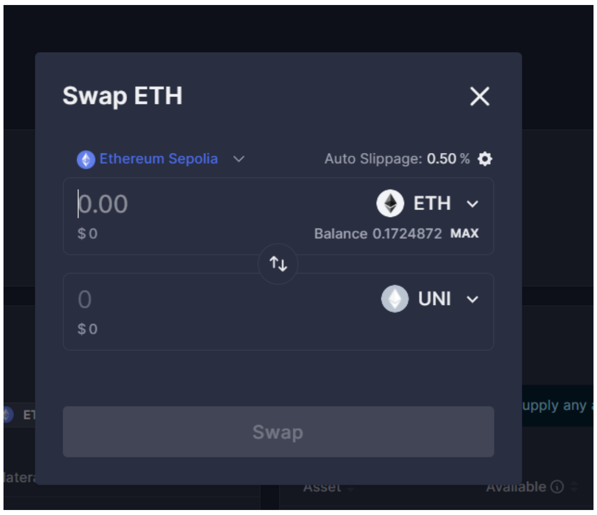
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.27;

import {ERC20} from "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import {ERC20Permit} from "@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol";

contract MyToken is ERC20, ERC20Permit {
    constructor(string memory name, string memory symbol)    infinite gas 1399400 gas
        ERC20(name, symbol)
        ERC20Permit(name) // ✅ Required for EIP-2612 (Permit)
    {
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }
}
```

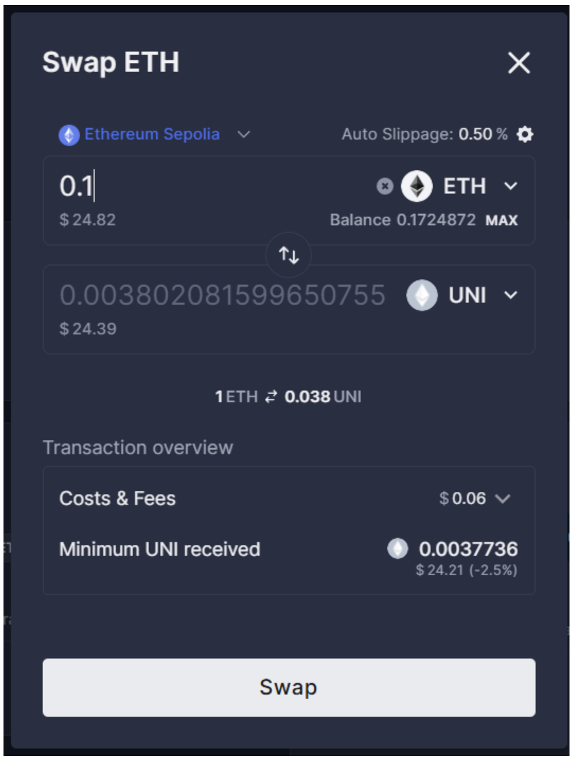
contract



```
EEC    No conversion rate available
       100,000,000,000 EEC
EXC    No conversion rate available
       100,000,000 EXC
```

Token




Swap ETH

Ethereum Sepolia | Auto Slippage: 0.50 %

0.1 ETH
$ 24.82 | Balance 0.1724872 MAX

0.003802081599650755 UNI
$ 24.39

1 ETH ⇄ 0.038 UNI

Transaction overview

Costs & Fees	$ 0.06
Minimum UNI received	0.0037736 $ 24.21 (-2.5%)

Observation

- Token contract deployed successfully.
- Contract address was visible and verified through Avee.net.
- Initial supply was credited to the deployer's wallet (MetaMask).
- Able to transfer tokens and check balances on Avee.net interface.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty: