

MyToken (MTK) | ERC-20 Smart Contract

1. Project Overview

MyToken (MTK) is a fully functional ERC-20 cryptocurrency token built on the Ethereum blockchain using Solidity. This project demonstrates a foundational understanding of smart contract development, token economics, and the Ethereum Virtual Machine (EVM).

The contract implements the standard ERC-20 interface, allowing for wallet compatibility, token transfers, and spending allowances, while also including robust error handling and event logging.

2. Technical Stack

- **Language:** Solidity (Version 0.8.x)
- **IDE:** Remix Ethereum IDE
- **Environment:** Remix VM (Cancun)
- **Standard:** ERC-20

3. Token Specifications

Based on the deployment configuration:

- **Token Name:** MyToken
- **Token Symbol:** MTK
- **Decimals:** 18
- **Total Supply:** 1,000,000 MTK (1,000,000 * 10¹⁸ wei)

4. Key Features Implemented

- **Core ERC-20 Functions:** transfer, approve, transferFrom, balanceOf, allowance.
- **Helper Functions:** getTokenInfo() for quick frontend data retrieval.
- **Event Logging:** Emits Transfer and Approval events to the blockchain logs.
- **Security Validation:**
 - Prevents transfers to the zero address (burn prevention).
 - Checks for sufficient balances before transfers.
 - Checks for sufficient allowance during delegated transfers.

5. Development & Testing Log

This section documents the successful compilation, deployment, and stress-testing of the smart contract.

A. Compilation

The contract was successfully compiled using the Solidity 0.8.30 compiler with no errors or warnings.

The screenshot shows the Remix IDE interface. On the left, the Solidity Compiler panel displays code for a token contract named MyToken.sol. The code includes functions for emitting transfers, getting total supply, and getting token info. Below the code are buttons for running analysis, publishing to IPFS, Swarm, or GitHub, and viewing compilation details. The middle section shows the default_workspace with tabs for Home and MyToken.sol. The RemixAI Assistant on the right provides personalized guidance, and the Explain contract feature at the bottom allows users to interact with their contracts via AI copilot.

B. Deployment

The contract was deployed to the local Remix VM environment. The state variables were initialized correctly upon deployment (Name, Symbol, Decimals, and Supply).

The screenshot shows the Remix IDE interface. On the left, the sidebar includes sections for 'DEPLOY & RUN TRANSACTIONS' (with tabs for 'ENVIRONMENT' and 'Reset State'), 'Remix VM (Cancun)', 'ACCOUNT' (set to 0x5B3...eddC4), 'GAS LIMIT' (set to 'Estimated Gas' or 'Custom 3000000'), and 'VALUE' (set to 0 Wei). The 'CONTRACT' section shows 'MyToken - contracts/MyToken.sol' and 'evm version: prague'. It has buttons for 'Deploy' (with a hex address) and 'At Address' (with a placeholder 'Load contract from Address'). Below these are sections for 'Transactions recorded' (with 1 transaction) and 'Deployed Contracts' (listing 'MYTOKEN AT 0xD91...39138 (M)').

The main workspace displays the Solidity code for 'MyToken.sol' with line numbers 90 to 107. The code includes a constructor emitting a Transfer event, helper functions for total supply, and a function to get token details.

```
90 emit Transfer(_from, _to, _value);
91 }
92
93 // --- 9. HELPER FUNCTIONS ---
94
95 // Returns the total number of tokens in existence
96 // (This is redundant because 'totalSupply' is public, but good for learning)
97 function getTotalSupply() public view returns (uint256) {
98     return totalSupply;
99 }
100
101 // Returns all token details in a single call
102 // Useful for websites to load data quickly without making 4 separate calls
103 function getTokenInfo() public view returns (string memory, string memory, uint256, uint256) {
104     return (name, symbol, decimals, totalSupply);
105 }
106
107 }
```

The screenshot shows the REMIX IDE interface with the following components:

- Top Bar:** REMIX 1.3.0, default_workspace, and window control buttons.
- Left Sidebar:** Icons for file operations (New, Open, Save, Import, Export, Find, Replace, Undo, Redo, Help).
- Middle Left Panel:** "DEPLOY & RUN TRANSACTIONS" section with a tree view of function signatures and their parameters. Functions listed include `decimals`, `getTokenInfo`, `name`, `symbol`, `totalSupply`, `getTotalSupply`, and `getTotalSupply` again.
- Middle Right Panel:** Code editor for `MyToken.sol`. The code defines a token contract with functions for emitting transfers, getting total supply, and returning token details.
- Bottom Panel:** "Explain contract" button, transaction history (CALL, [call] from: 0x5B38Da6a701c568545dCfcB03FcB87f5f6beddC4 to: MyToken.decimals() data: 0x31...ce567), and a "Debug" button.

C. Core Functionality: Transfers & Events

The standard transfer function was tested. The transaction succeeded, balances were updated, and the blockchain emitted the correct Transfer event logs.

The screenshot shows the REMIX IDE interface with the following details:

- Top Bar:** REMIX 1.3.0, default_workspace
- Left Sidebar:** Includes icons for File, Deploy, Run, Transactions, Contracts, and Explain.
- Deploy & Run TRANSACTIONS:** Shows "Transactions recorded" (2) and "Deployed Contracts" (1).
- Deployed Contracts:** MYTOKEN at address 0xd91...39138 (0 ETH). It lists several functions:
 - TRANSFER:** To: 0xAb8483f64d9C6d1Ecf9b849Ae6, Value: 100000000000000000000000000000000
 - transferFrom:** From: address _from, To: address _to
 - allowance:** From: address , To: address
 - BALANCEOF:** Address: 0xAb8483f64d9C6d1Ecf9b849Ae6
- Code Editor:** Solidity code for MyToken.sol. The code includes functions for emitting Transfer events, creating tokens, and getting total supply, token info, and balance of an address.
- Explain contract:** A modal window showing transaction details for calls to get Token Info, decimals, transfer, and balanceOf.
- Bottom Bar:** Buttons for Calldata, Parameters, and call (highlighted), along with a Debug button for each transaction.

The screenshot shows the Remix IDE interface with the following details:

- Contract:** MyToken.sol
- Code:**

```

90     emit Transfer(_from, _to, _value);
91     return true;
92 }
93
94 // --- 9. HELPER FUNCTIONS ---
95
96 // Returns the total number of tokens in existence
97 // (This is redundant because 'totalSupply' is public, but good for learning)
98 function getTotalSupply() public view returns (uint256) {
99     return totalSupply;
100 }
101
102 // Returns all token details in a single call
103 // Useful for websites to load data quickly without making 4 separate calls
104 function getTokenInfo() public view returns (string memory, string memory, uint256, uint256) {
105     return (name, symbol, decimals, totalSupply);
106 }
107 }
```
- Deploy & Run Transactions:**
 - Approve:** Spender: "0xAb8483F64d9C6d1EcF9b849Ae6", Value: "50000000000000000000000000000000". Transaction status: Pending.
 - TransferFrom:** From: "0x5B38Da6a701c568545dCfcB03FcB875f56beddc4", To: "0x4B20993Bc481177ec/E8f571ceCa", Value: "50000000000000000000000000000000". Transaction status: Pending.
- Logs:**
 - [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddc4 to: MyToken.balanceOf(address) data: 0x70a...35cb2
 - [vm] from: 0x5B3...edd4 to: MyToken.approve(address,uint256) 0xd91...39138 value: 0 wei data: 0x095...80000 logs: 1 hash: 0xc63...a3fa3
 - [vm] from: 0xAb8...35cb2 to: MyToken.transferFrom(address,address,uint256) 0xd91...39138 value: 0 wei data: 0x23b...80000 logs: 1 hash: 0xF55...9c0d7

D. Security & Error Handling

To ensure contract integrity, specific edge cases were tested to verify that transactions revert (fail) under invalid conditions.

1. Zero Address Validation: Attempting to transfer tokens to the 0x000... address correctly reverts the transaction to prevent accidental token burning.

The screenshot shows the Remix IDE interface with the following details:

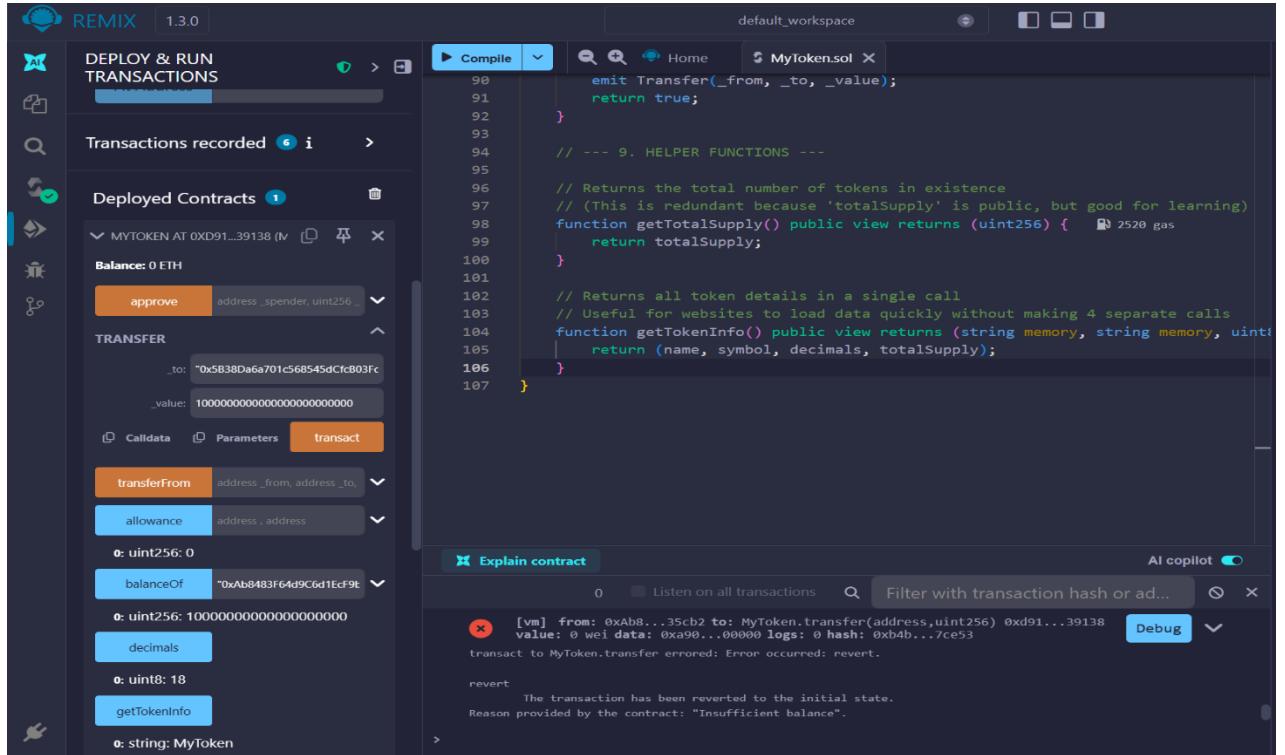
- Contract:** MyToken.sol
- Code:**

```

90     emit Transfer(_from, _to, _value);
91     return true;
92 }
93
94 // --- 9. HELPER FUNCTIONS ---
95
96 // Returns the total number of tokens in existence
97 // (This is redundant because 'totalSupply' is public, but good for learning)
98 function getTotalSupply() public view returns (uint256) {
99     return totalSupply;
100 }
101
102 // Returns all token details in a single call
103 // Useful for websites to load data quickly without making 4 separate calls
104 function getTokenInfo() public view returns (string memory, string memory, uint256, uint256) {
105     return (name, symbol, decimals, totalSupply);
106 }
107 }
```
- Deploy & Run Transactions:**
 - Transfer:** To: "0x00000000000000000000000000000000", Value: "10". Transaction status: Reverted.
- Logs:**
 - [vm] from: 0x5B3...edd4 to: MyToken.transfer(address,uint256) 0xd91...39138 value: 0 wei data: 0xa90...00000 logs: 0 hash: 0xf9a...7d8aa

The log message indicates: "The transaction has been reverted to the initial state. Reason provided by the contract: "Cannot transfer to zero address". If the transaction failed for not having enough gas, try increasing the gas limit gently."

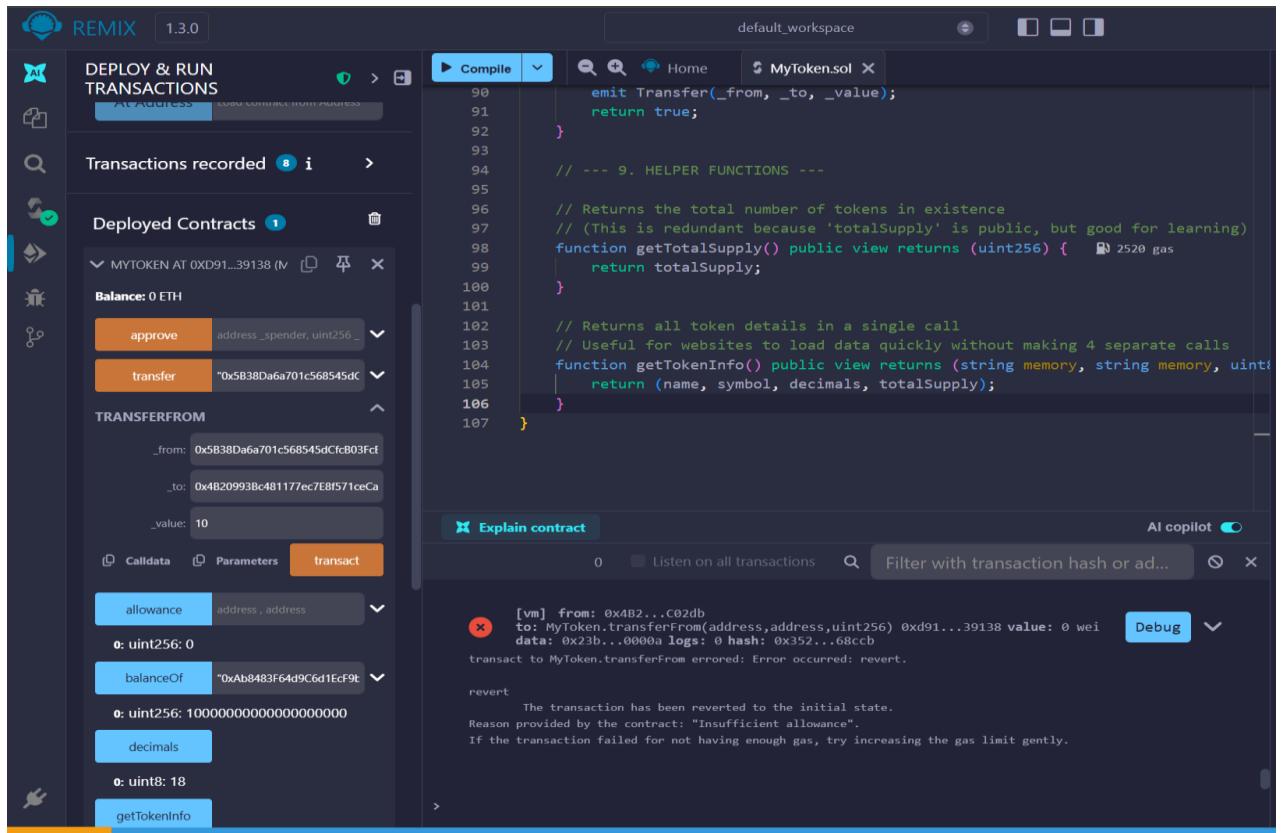
2. Balance Validation: Attempting to transfer more tokens than the sender possesses correctly reverts the transaction.



The screenshot shows the REMIX IDE interface with the following details:

- Deploy & Run Transactions:** Shows a transaction record for "MYTOKEN at 0xd91...39138" with a revert message: "transact to MyToken.transfer errored: Error occurred: revert".
- Deployed Contracts:** Displays the state of the deployed contract "MYTOKEN" with "Balance: 0 ETH".
- Contract Code:** The Solidity code for "MyToken.sol" includes functions like `approve`, `transfer`, and `getTotalSupply`.
- Explain contract:** A modal window provides an explanation of the revert: "The transaction has been reverted to the initial state. Reason provided by the contract: 'Insufficient balance'."

3. Allowance Validation: Attempting to spend tokens on behalf of another user without sufficient approval (transferFrom) correctly reverts.



The screenshot shows the REMIX IDE interface with the following details:

- Deploy & Run Transactions:** Shows a transaction record for "MYTOKEN at 0xd91...39138" with a revert message: "transact to MyToken.transferFrom errored: Error occurred: revert".
- Deployed Contracts:** Displays the state of the deployed contract "MYTOKEN" with "Balance: 0 ETH".
- Contract Code:** The Solidity code for "MyToken.sol" includes functions like `approve`, `transfer`, and `getTotalSupply`.
- Explain contract:** A modal window provides an explanation of the revert: "The transaction has been reverted to the initial state. Reason provided by the contract: 'Insufficient allowance'. If the transaction failed for not having enough gas, try increasing the gas limit gently."

6. How to Run This Project

1. **Open Remix:** Navigate to [Remix IDE](#).
 2. **Create File:** Create MyToken.sol in the contracts folder.
 3. **Compile:** Use the Solidity Compiler tab (ensure version 0.8.x is selected).
 4. **Deploy:**
 - o Go to the "Deploy & Run Transactions" tab.
 - o Select "Remix VM" as the environment.
 - o Deploy the contract (Input initial supply if required by constructor).
 5. **Interact:** Use the deployed contract buttons to execute transfers and check balances.
-

7. What I've Learned

Building this ERC-20 token provided practical insight into the mechanics of the Ethereum blockchain. Beyond just writing code, I gained a deeper understanding of:

- **The Power of Standards:** I learned why the ERC-20 interface is critical. By strictly following function names like transfer and approve, my token is automatically compatible with crypto wallets and exchanges.
 - **State Management in Solidity:** I mastered using mapping to track user balances and allowances (mapping(address => uint256)), which is far more efficient than array loops used in traditional programming.
 - **Security & Validation:** My testing highlighted the importance of require() statements. Without explicitly checking for the **Zero Address** or **Insufficient Balance**, the contract would be vulnerable to burning tokens or invalid arithmetic.
 - **The "Decimals" Concept:** I learned that Solidity does not handle floating-point numbers. Managing 18 decimals means interacting with huge integers (1 Token = 1,000,000,000,000,000 wei), which requires careful calculation.
 - **Event Logging:** I discovered that smart contracts are often "blind" to the outside world, but by emitting Events (like Transfer), I can create a trail of data that frontend applications can read and display to users.
-

8. Conclusion

This project successfully transformed a theoretical concept into a live, functional digital asset. By deploying **MyToken (MTK)**, I have not only built a cryptocurrency from scratch but also validated its security and functionality through rigorous testing in the Remix IDE.

This project serves as the "Hello World" of my blockchain journey, proving that I can write, deploy, and debug smart contracts. It lays the groundwork for more advanced development, such as building Decentralized Applications (DApps), integrating with frontend frameworks, or exploring complex DeFi protocols.