EarthSaver

COLLABORATION EARTH

Project-2

(Smart Contract)

Griffith UNIVERSITY

Table of Contents

DESIGN

The smart contract used for the website is coded in Solidity and can be found through the provided Bank.sol file. The deployed contract has the following features:

- Name of the token: EarthSaver (ESV)
- Initial Minting: 100,000 ESV tokens are minted to the contract creator's address.

Functionalities

1. Deposit Money:

- Users can deposit their ESV tokens into the bank. This is done by burning the specified amount of ESV tokens from the user's balance.
- Parameters:
  - `amount`: The number of ESV tokens to deposit.
- Requirement: User must have a balance equal to or greater than the deposit amount.

2. Withdraw Money:

- Users can withdraw their ESV tokens from the bank. This is done by minting the specified amount of ESV tokens to the user's balance.
- Parameters:
  - `amount`: The number of ESV tokens to withdraw.

3. Transfer Money:

- Users can transfer ESV tokens to another address.
- Parameters:
  - `recipient`: The address to receive the ESV tokens.
  - `amount`: The number of ESV tokens to transfer.

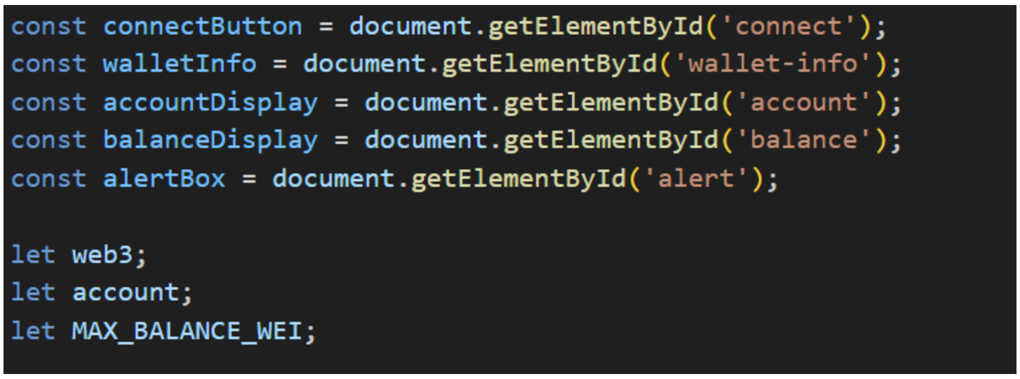
Once deployed, using the ABI, Binary, and contract address, individuals can interact with the contract to deposit, withdraw, and transfer funds. Additionally, they can view relevant information such as their token balance and transaction history.

This contract serves as a basic implementation of a banking system using ERC20 tokens, allowing users to manage their tokens within a decentralized platform.

FUNCTIONS

Initializer

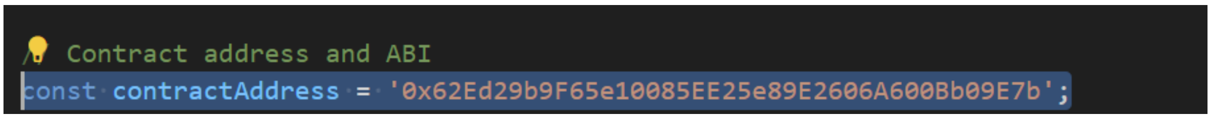
This JavaScript code snippet initializes variables and references to HTML elements for a web application that interacts with a cryptocurrency wallet. `connectButton`, `walletInfo`, `accountDisplay`, `balanceDisplay`, and `alertBox` are DOM elements for connecting to the wallet, displaying wallet information, account details, balance, and alert messages, respectively. Variables `web3`, `account`, and `MAX_BALANCE_WEI` are placeholders for the Web3 instance, the user's account, and the maximum balance in Wei (smallest Ethereum unit).



```
const connectButton = document.getElementById('connect');
const walletInfo = document.getElementById('wallet-info');
const accountDisplay = document.getElementById('account');
const balanceDisplay = document.getElementById('balance');
const alertBox = document.getElementById('alert');

let web3;
let account;
let MAX_BALANCE_WEI;
```

Contract Address

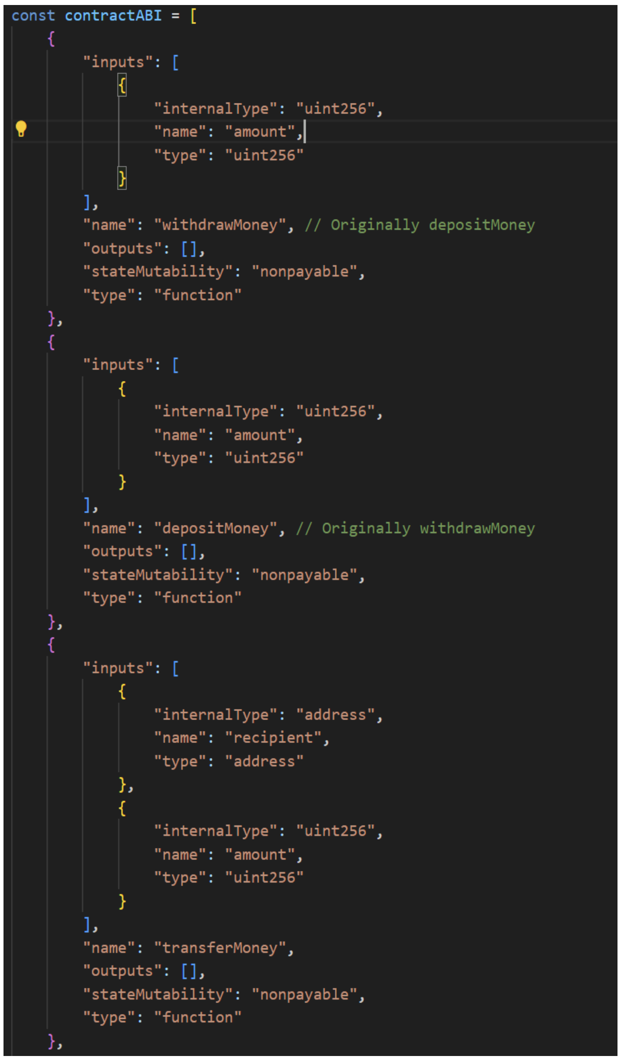
The `contractAddress` variable stores the specific Ethereum blockchain address '0x62Ed29b9F65e10085EE25e89E2606A600Bb09E7b' for a smart contract. This address is used to interact with the deployed contract, enabling functions like transactions, contract calls, and data retrieval.



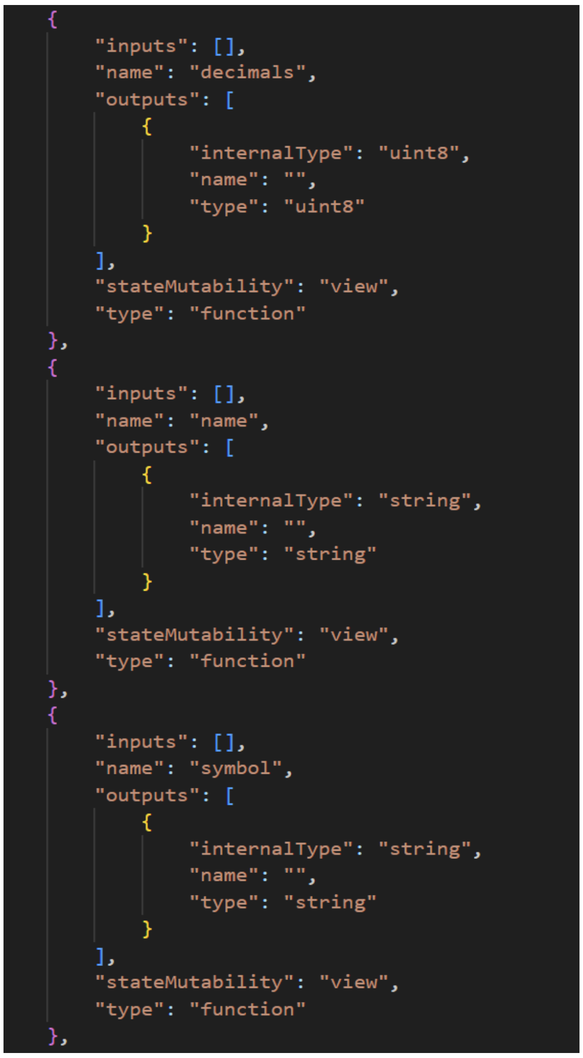
```
Contract address and ABI
const contractAddress = '0x62Ed29b9F65e10085EE25e89E2606A600Bb09E7b';
```

Token Manager

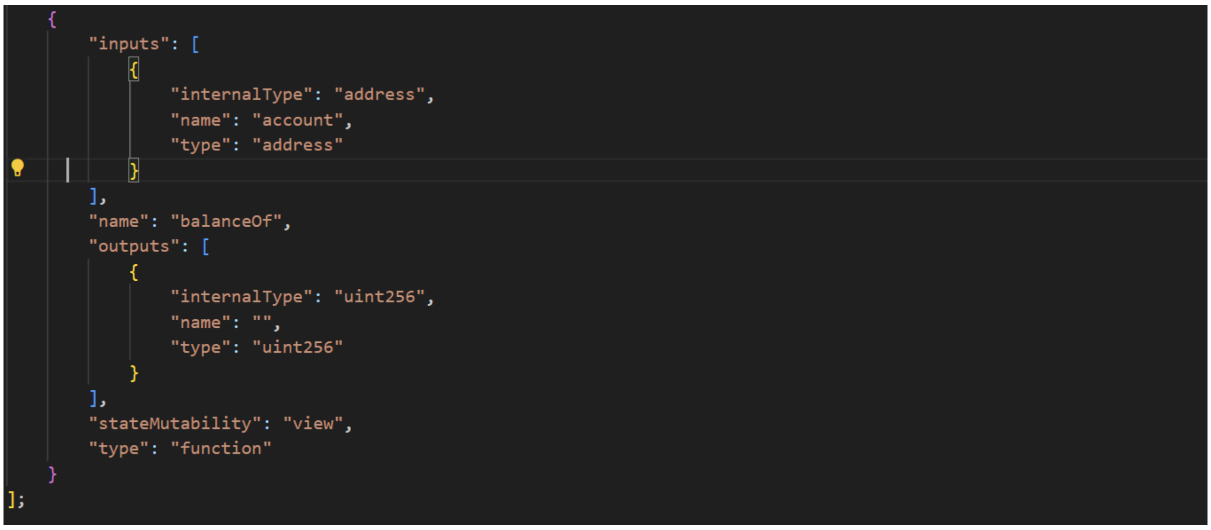
This contract ABI describes an Ethereum smart contract for handling a token. It includes functions for depositing, withdrawing, and transferring tokens, as well as querying token metadata (name, symbol, decimals) and checking account balances.



```
const contractABI = [
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "amount",
                "type": "uint256"
            }
        ],
        "name": "withdrawMoney", // Originally depositMoney
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "amount",
                "type": "uint256"
            }
        ],
        "name": "depositMoney", // Originally withdrawMoney
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "recipient",
                "type": "address"
            },
            {
                "internalType": "uint256",
                "name": "amount",
                "type": "uint256"
            }
        ],
        "name": "transferMoney",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
```



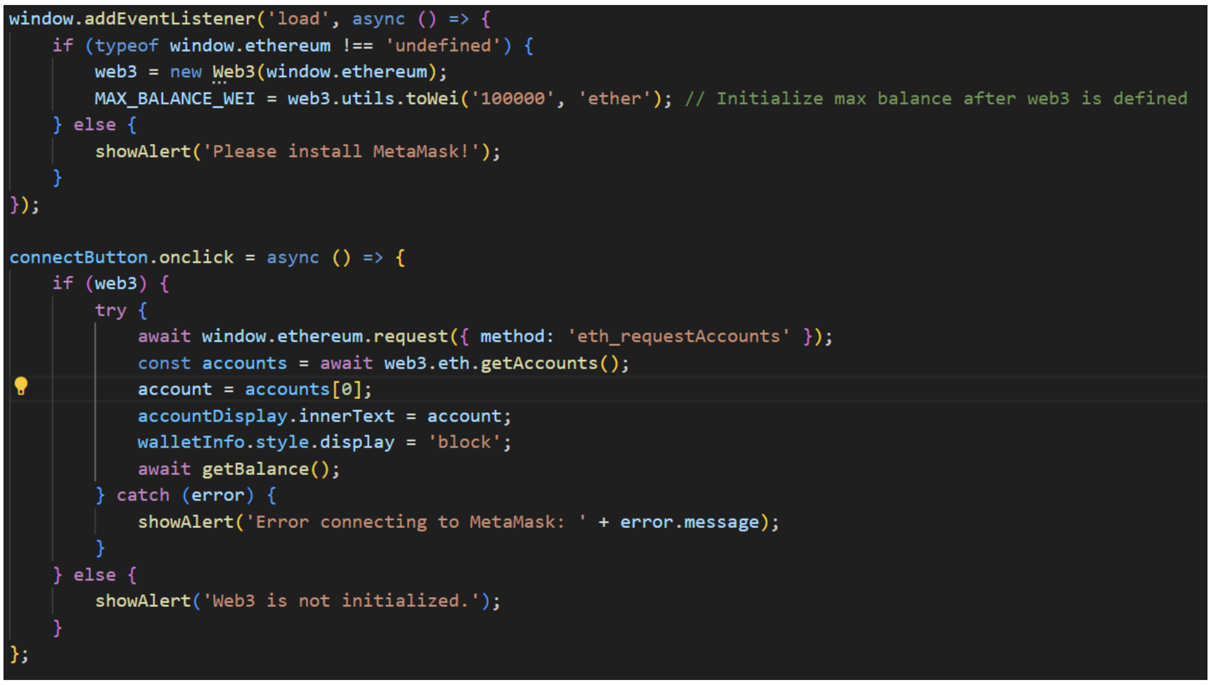
```
    {
        "inputs": [],
        "name": "decimals",
        "outputs": [
            {
                "internalType": "uint8",
                "name": "",
                "type": "uint8"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "name",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "symbol",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
```



```
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "account",
                "type": "address"
            }
        ],
        "name": "balanceOf",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];
```

MetaMaskAuthenticator

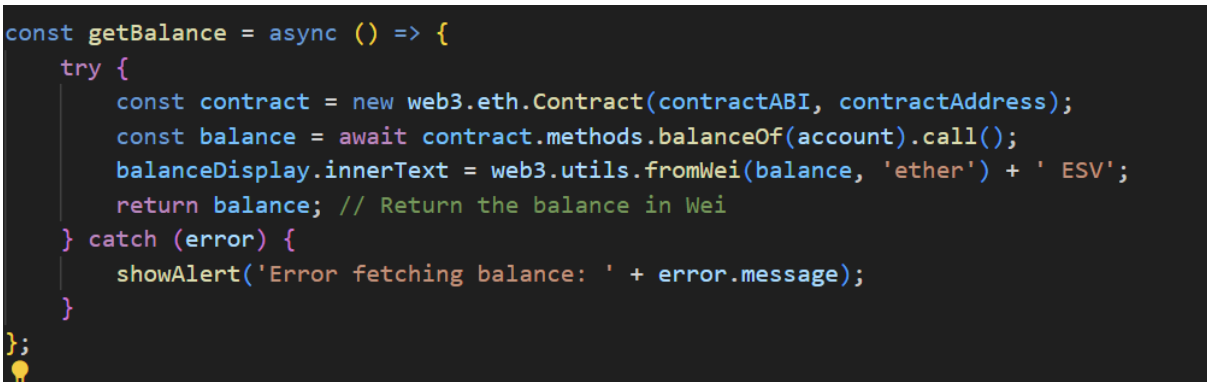
This script initializes Web3 when the window loads, checks if MetaMask is installed, and connects to the user's MetaMask account upon a button click. It then retrieves and displays the user's account and balance, handling potential connection errors.



```
window.addEventListener('load', async () => {
    if (typeof window.ethereum !== 'undefined') {
        web3 = new Web3(window.ethereum);
        MAX_BALANCE_WEI = web3.utils.toWei('100000', 'ether'); // Initialize max balance after web3 is defined
    } else {
        showAlert('Please install MetaMask!');
    }
});

connectButton.onclick = async () => {
    if (web3) {
        try {
            await window.ethereum.request({ method: 'eth_requestAccounts' });
            const accounts = await web3.eth.getAccounts();
            account = accounts[0];
            accountDisplay.innerText = account;
            walletInfo.style.display = 'block';
            await getBalance();
        } catch (error) {
            showAlert('Error connecting to MetaMask: ' + error.message);
        }
    } else {
        showAlert('Web3 is not initialized.');
    }
};
```

BalanceChecker

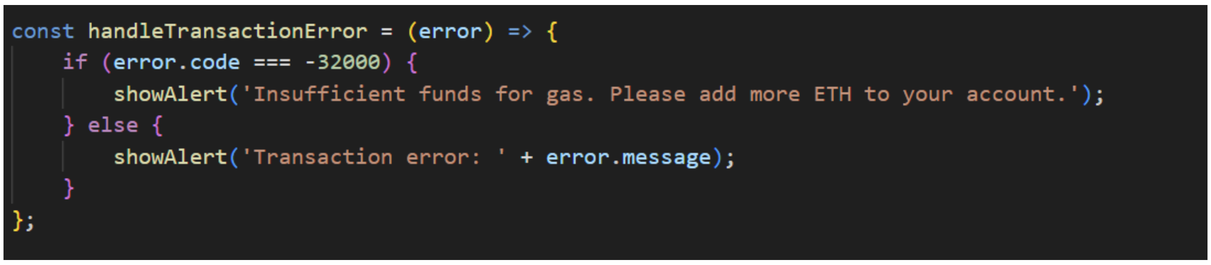
The getBalance function asynchronously retrieves the balance of the specified account from the smart contract using Web3. It converts the balance from Wei to Ether for display, updating the UI with the balance. If an error occurs, it shows an alert with the error message.



```
const getBalance = async () => {
    try {
        const contract = new web3.eth.Contract(contractABI, contractAddress);
        const balance = await contract.methods.balanceOf(account).call();
        balanceDisplay.innerText = web3.utils.fromWei(balance, 'ether') + ' ESV';
        return balance; // Return the balance in Wei
    } catch (error) {
        showAlert('Error fetching balance: ' + error.message);
    }
};
```

ErrorHandler

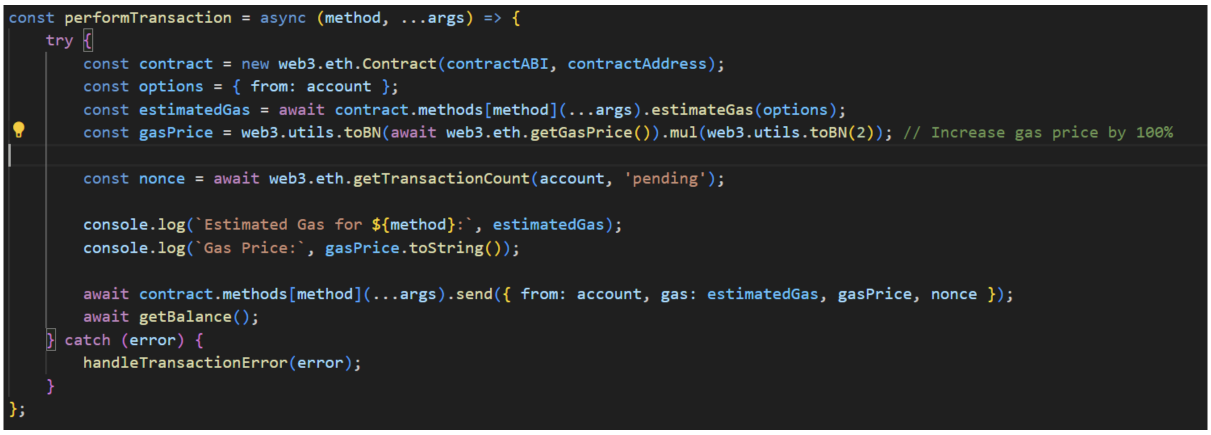
The handleTransactionError function manages transaction errors by checking the error code. If the code is -32000, it alerts the user about insufficient funds for gas. For other errors, it displays a general transaction error message.



```javascript
const handleTransactionError = (error) => {
    if (error.code === -32000) {
        showAlert('Insufficient funds for gas. Please add more ETH to your account.');
    } else {
        showAlert('Transaction error: ' + error.message);
    }
};
```

TransactionHandler

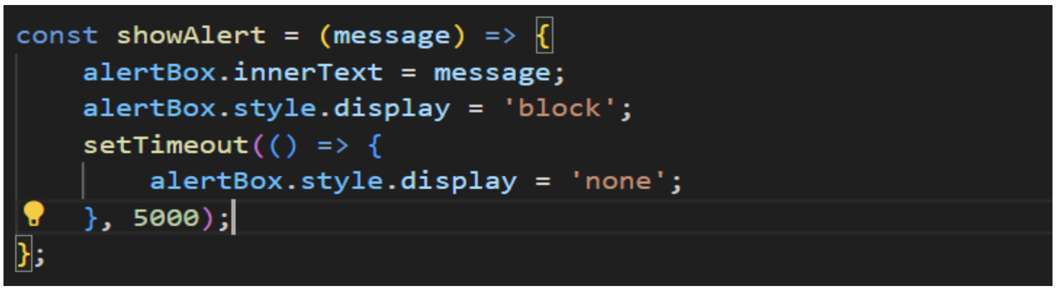
The performTransaction function in Web3.js estimates gas, doubles the gas price for faster transaction execution, and retrieves the nonce. It then sends the transaction, handles errors, and updates the balance.



```javascript
const performTransaction = async (method, ...args) => {
    try {
        const contract = new web3.eth.Contract(contractABI, contractAddress);
        const options = { from: account };
        const estimatedGas = await contract.methods[method](...args).estimateGas(options);
        const gasPrice = web3.utils.toBN(await web3.eth.getGasPrice()).mul(web3.utils.toBN(2)); // Increase gas price by 100%

        const nonce = await web3.eth.getTransactionCount(account, 'pending');

        console.log(`Estimated Gas for ${method}:`, estimatedGas);
        console.log(`Gas Price:`, gasPrice.toString());

        await contract.methods[method](...args).send({ from: account, gas: estimatedGas, gasPrice, nonce });
        await getBalance();
    } catch (error) {
        handleTransactionError(error);
    }
};
```

AlertHandler

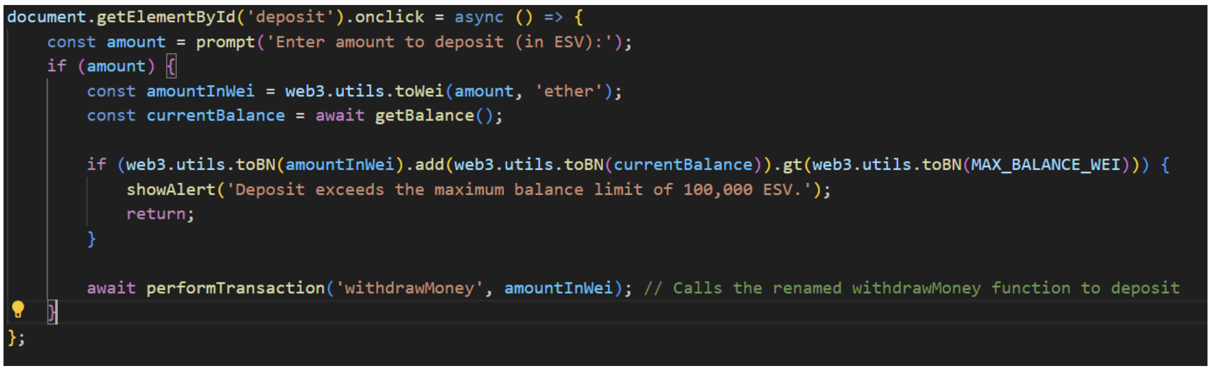
The function showAlert displays a message in an alert box for 5 seconds. It updates the innerText of the alertBox element with the provided message, makes the alert box visible by setting its display style to 'block', and hides it again after 5 seconds using setTimeout.



```javascript
const showAlert = (message) => {
    alertBox.innerText = message;
    alertBox.style.display = 'block';
    setTimeout(() => {
        alertBox.style.display = 'none';
    }, 5000);
};
```

DepositHandler

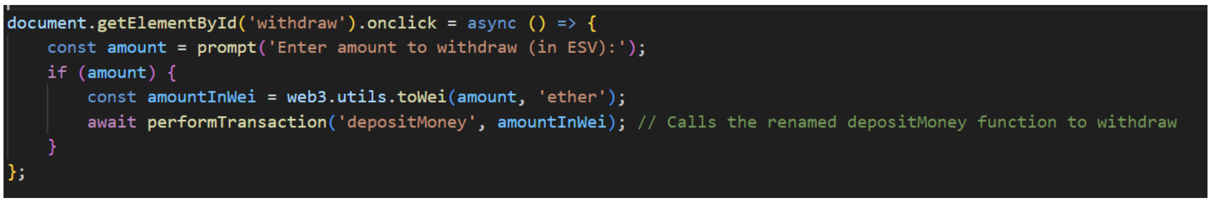
The DepositHandler function captures a user's deposit amount, converts it to Wei, checks if it exceeds the maximum balance limit, and then calls the withdrawMoney function to perform the deposit transaction if the limit is not exceeded.



```
document.getElementById('deposit').onclick = async () => {
    const amount = prompt('Enter amount to deposit (in ESV):');
    if (amount) {
        const amountInWei = web3.utils.toWei(amount, 'ether');
        const currentBalance = await getBalance();

        if (web3.utils.toBN(amountInWei).add(web3.utils.toBN(currentBalance)).gt(web3.utils.toBN(MAX_BALANCE_WEI))) {
            showAlert('Deposit exceeds the maximum balance limit of 100,000 ESV.');
            return;
        }

        await performTransaction('withdrawMoney', amountInWei); // Calls the renamed withdrawMoney function to deposit
    }
};
```

WithdrawHandler

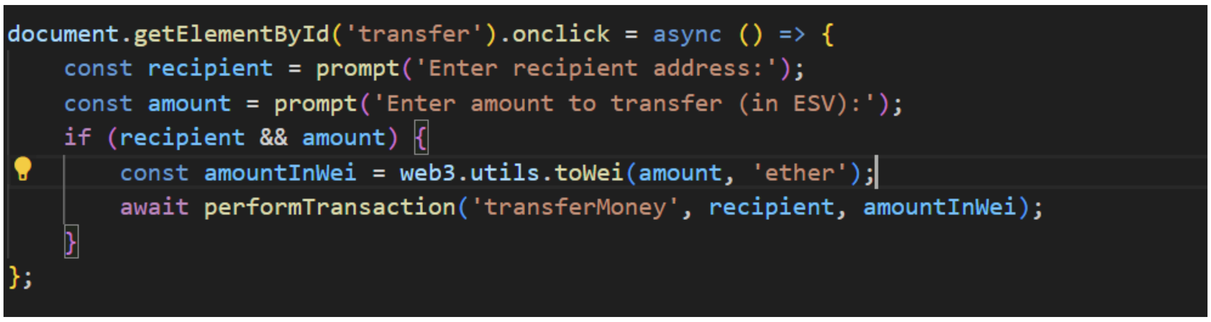
The code attaches an event listener to an HTML element with the ID 'withdraw'. When clicked, it prompts the user to enter an amount in ESV to withdraw, converts this amount to Wei using Web3.js utilities, and then calls the performTransaction function to execute the transaction, passing the converted amount.



```
document.getElementById('withdraw').onclick = async () => {
    const amount = prompt('Enter amount to withdraw (in ESV):');
    if (amount) {
        const amountInWei = web3.utils.toWei(amount, 'ether');
        await performTransaction('depositMoney', amountInWei); // Calls the renamed depositMoney function to withdraw
    }
};
```

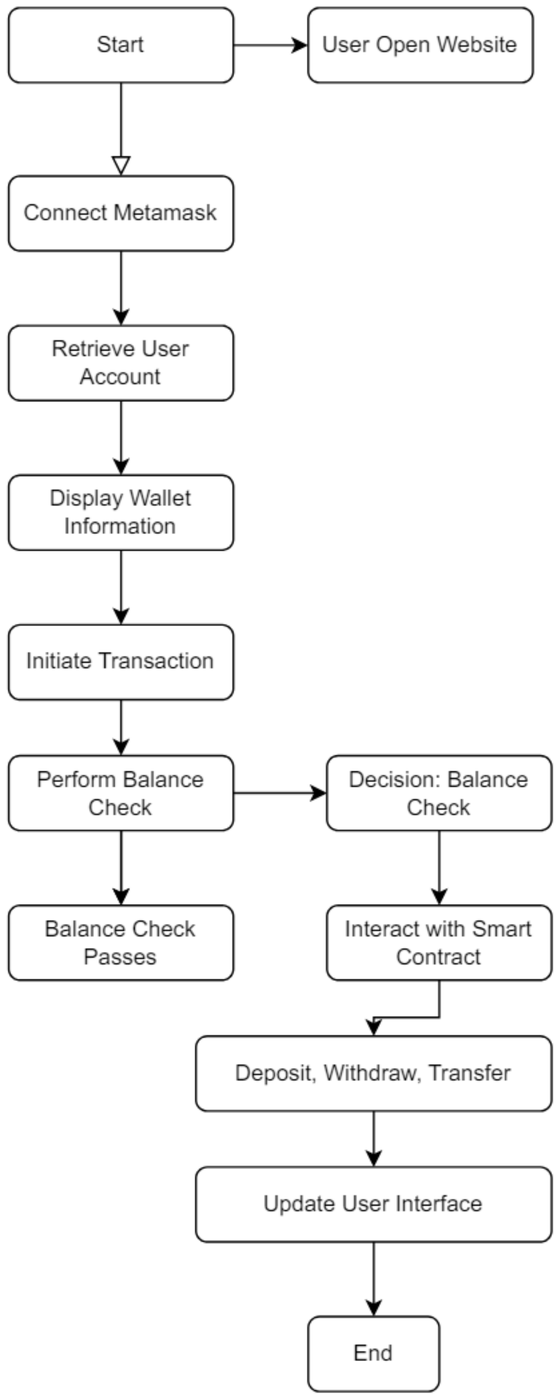
TransactionHandler

The code attaches an event listener to a button with ID 'transfer'. When clicked, it prompts the user to input a recipient address and amount of ESV to transfer. It converts the amount to Wei and calls the performTransaction function to execute the transfer.



```
document.getElementById('transfer').onclick = async () => {
    const recipient = prompt('Enter recipient address:');
    const amount = prompt('Enter amount to transfer (in ESV):');
    if (recipient && amount) {
        const amountInWei = web3.utils.toWei(amount, 'ether');
        await performTransaction('transferMoney', recipient, amountInWei);
    }
};
```

FLOW DIAGRAM


Start
User Open Website
Connect Metamask
Retrieve User Account
Display Wallet Information
Initiate Transaction
Perform Balance Check
Decision: Balance Check
Balance Check Passes
Interact with Smart Contract
Deposit, Withdraw, Transfer
Update User Interface
End