

Token Launch  
  
**Objective/Aim:**



To understand the process of creating and deploying a fungible token on the Ethereum blockchain using

the ERC-20 standard, and to simulate a basic token launch.

**Apparatus/Software Used:**

Laptop / PC

Node.js & npm

Hardhat / Remix IDE (for smart contract deployment)

**Theory/Concept:**

* What is a Token?  
   Tokens are digital assets created on a blockchain. Unlike coins (native to a blockchain like ETH), tokens are built using smart contracts.
* Types of Tokens:
* Fungible Tokens (ERC-20) → interchangeable (e.g., USDT, DAI).
* Non-Fungible Tokens (ERC-721) → unique (NFTs).
* ERC-20 Token Standard:  
   Defines basic functions every fungible token must implement, such as:
* totalSupply() → total number of tokens.
* balanceOf(address) → balance of a wallet.
* transfer(to, amount) → transfer tokens.
* approve() & transferFrom() → delegated transfers.
* Token Launch:  
   Involves deploying the ERC-20 contract, minting tokens, and distributing them to users.



**Procedure:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Set Up Environment
* Install Hardhat or open Remix IDE.
* Connect MetaMask to a local test network or Ethereum testnet (Goerli/Sepolia).
* Write ERC-20 Smart Contract  
   Example using OpenZeppelin library:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

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

contract MyToken is ERC20 {

constructor(uint256 initialSupply) ERC20("MyToken", "MTK") {

\_mint(msg.sender, initialSupply);

}

}

**Observation Table:**

* Token is created with a fixed initial supply.
* Each wallet shows correct balances after transfers.
* Transactions require gas fees (ETH).
* Smart contract prevents overspending beyond balance.

