Let's Mint a New Token

In this section, we are going to create a new Token called `REC`. This will be a subcurrency and will showcase reads and writes to the etherium blockchain.

First, create a new contract by pasting the following code in `contracts/Token.sol`:

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "hardhat/console.sol";

contract Token {
  string public name = "Recluze Token";
  string public symbol = "REC";
  uint public totalSupply = 1000;

  mapping(address => uint) balances;

  constructor() {
    balances[msg.sender] = totalSupply;
  }

  function transfer(address to, uint amount) external {
    require(balances[msg.sender] >= amount, "Insufficient tokens");
    balances[msg.sender] -= amount;
    balances[to] += amount;
  }

  function balanceOf(address account) external view returns (uint) {
    return balances[account];
  }
}
```

Compile the contract as before.

```
npx hardhat compile
```

Make the needed changes in `scripts/deploy.js` so that this contract is also deployed alongside the greeter.

```
// in function main, add the following
const Token = await hre.ethers.getContractFactory("Token");
const token = await Token.deploy();
await token.deployed();
console.log("Token deployed to:", token.address);
```

Then let's deploy it.

```
npx hardhat run scripts/deploy.js --network localhost
```

Take note of the token address. We'll need it.

Go to MetaMask and click on `Import Token` in the main window. Paste the address of the token we just created. Set `Decimal` to 0 if needed. Now you should have the REC token added with the amount properly set.

You can transfer REC token to another account as you can ETH. Of course, you'll need ETH for gas during transaction writing. Go ahead and transfer some RECs to `recly-test0x` account.

Sending and Receiving Tokens using Web Frontend

Add the following two functions to `src/App.js`.

```
// ...

import Token from './artifacts/contracts/Token.sol/Token.json'
// ...

const tokenAddress = "0x5FC8d32690cc91D4c39d9d3abcBD16989F875707"   // !!! Change this
// ...

// Within App()

const [userAccount, setUserAccount] = useState()
const [amount, setAmount] = useState()

async function getBalance() {
    if (typeof window.ethereum !== 'undefined') {
      const [account] = await window.ethereum.request({ method: 'eth_requestAccounts' }
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const contract = new ethers.Contract(tokenAddress, Token.abi, provider)
      const balance = await contract.balanceOf(account);
      console.log("Balance: ", balance.toString());
    }
  }

// ...

async function sendCoins() {
    if (typeof window.ethereum !== 'undefined') {
      await requestAccount()
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const signer = provider.getSigner();
      const contract = new ethers.Contract(tokenAddress, Token.abi, signer);
      const transation = await contract.transfer(userAccount, amount);
      await transation.wait();
      console.log(`${amount} Coins successfully sent to ${userAccount}`);
    }
  }

// ...
```

Also add the following in the React template in the same file:

```
<br />
<button onClick={getBalance}>Get Balance</button>
<button onClick={sendCoins}>Send Coins</button>
<input onChange={e => setUserAccount(e.target.value)} placeholder="Account ID" />
<input onChange={e => setAmount(e.target.value)} placeholder="Amount" />
```

Start the npm server again.

```
npm start
```

In the server, click on `Get Balance` to get the REC balance. Copy the Account ID of `recly-test0x` and send 20 REC to that account. MetaMask will ask you to connect your account if it's been a while. You can now go ahead and send the REC. Check the balance again on both accounts to ensure RECs have been transferred.