Front-end for the dApp

Table of Content

Build your first decentralized application on Ethereum

This is a step-by-step tutorial on how to create a simple frontend website, create and deploy a Solidity smart contract, and connect them together. We will be using MetaMask, Remix IDE, and Ethers.js. By the end of this tutorial, you will be able to create a simple frontend with two buttons that can interact with smart contract functions.

The tutorial takes place in three stages:

1. Create a basic HTML web page
2. Create a basic Solidity smart contract
3. Connect the web page with the smart contracts using Ethers.js

Prerequisites

1. Download and setup MetaMask if you do not have it already
2. Click on the User Icon at the top-right of MetaMask, go to Settings, and turn on Show Test Networks
3. Click on Ethereum Mainnet at the top of MetaMask and change it to Sepolia test network
4. Request some Sepolia Testnet ETH from a faucet.
5. Install a code editor
6. Install Node.js if you do not have it already
7. Open up a terminal (Command Prompt on Windows, Terminal on macOS or Linux) and install lite-server so you can run your website

```
npm install -g lite-server
```

Building the webpage

The first thing we will do is build out a basic webpage.

Create a new folder on your computer, and open that folder in your code editor. Inside that folder, create a new file - index.html - and open that in your code editor.

Each HTML page has some basic boilerplate we need to write to tell the web browser that this is an HTML document. Add the following to index.html.

```
<!DOCTYPE html>
<html lang="en">
```

```
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>LearnWeb3 First dApp</title>
  </head>
  <body>
  <!-- We will add more code here -->
  </body>
</html>
```

Inside the body tag, we will start writing the rest of our code.

The smart contract we create will be very simple - just reading and writing a value to and from the Sepolia testnet. Let's create a textbox and a few buttons on the page.

Update your body tag to look like this:

```
<body>
  <div>
    <h1>This is my dApp!</h1>
    <p>Here we can set or get the mood:</p>
    <label for="mood">Input Mood:</label> <br />
    <input type="text" id="mood" />

    <button onclick="getMood()">Get Mood</button>
    <button onclick="setMood()">Set Mood</button>
    <p id="showMood"></p>
  </div>
</body>
```

We can also write some CSS inside the head tag to make our website look a little better. This step is optional, but helps!

```
<style>
  body {
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
  }

  div {
    width: 20%;
    margin: 0 auto;
    display: flex;
    flex-direction: column;
  }

  button {
    width: 100%;
    margin: 10px 0px 5px 0px;
  }
</style>
```

Now save this file, and navigate to the folder where this index.html file is through your Terminal. Once you're there, run the following command:

```
lite-server
```

If everything was set up properly, your webpage should now be accessible! Go to http://localhost:3000/ to see your page!

We have our boilerplate frontend website ready!