Project Deployment

Table of Content

Connect your webpage to your smart contract

Now, let's go back to index.html in your code editor. Here, we will do a few things:

1. Add ethers.js as an external library to your code
2. Use ethers.js to create a reference to the deployed Solidity contract
3. Call your contract's functions through MetaMask with the help of ethers.js

At the beginning (or end) of your HTML file, import ethers.js library as follows, and add a second empty script tag for your own JavaScript code:

```
<script
  src="https://cdn.ethers.io/lib/ethers-5.7.2.umd.min.js"
  type="application/javascript"
/>

<script>
// Further code will go here
</script>
```

Inside the second script tag, assign your contract address and contract ABI to the relevant variables, and declare two more variables for the Contract and Signer

```
// Replace the following two values
const MoodContractAddress = ...;
const MoodContractABI = ...;

// Currently these two are undefined, we will use Ethers to assign them values
let MoodContract = undefined;
let signer = undefined;
```

If you are wondering what the Contract ABI is, see this section on Solidity's documentation. We will also go in-depth into ABIs a few lessons later.

For the Contract ABI, we want to specifically navigate to the JSON Section and describe our smart contract in JSON format. Since we have two functions in our contract, this should be an array with two objects:

```
const MoodContractABI = [ {...}, {...} ];
```

Each object inside the array should have the following fields: constant, inputs, name, outputs, payable, stateMutability, and type.

For setMood, we describe each field below:

- name: setMood, self-explanatory
- type: function, self-explanatory
- outputs: should be [] because this does not return anything
- stateMutability: This is nonpayable because this function does not accept Ether
- inputs: this is an array of inputs to the function. Each object in the array should have internalType, name and type, and these are string, _mood and string respectively

For getMood, we describe each field below:

- name: getMood, self-explanatory
- type: function, self-explanatory
- outputs: this has the same type as inputs in setMood. For internalType, name and type, this should be string, "", and string respectively
- stateMutability: This is view because this is a view function
- inputs: this has no arguments so this should be []

Your end result should look like this:

```
const MoodContractABI = [{
      "inputs": [],
      "name": "getMood",
      "outputs": [{
         "internalType": "string",
         "name": "",
         "type": "string"
      }],
      "stateMutability": "view",
      "type": "function"
   },
   {
      "inputs": [{
         "internalType": "string",
         "name": "_mood",
         "type": "string"
      }],
      "name": "setMood",
      "outputs": [],
      "stateMutability": "nonpayable",
      "type": "function"
   }
]
```

Next, in your code, define a Web3 Provider - this is our connection to the Ethereum Network (Sepolia Testnet) - and it happens through MetaMask.

```
const provider = new ethers.providers.Web3Provider(window.ethereum, "sepolia");
```

Request access to the user's wallet and assign values to MoodContract and signer that were previously set to undefined.

```
provider.send("eth_requestAccounts", []).then(() => {
  provider.listAccounts().then((accounts) => {
    signer = provider.getSigner(accounts[0]);
    MoodContract = new ethers.Contract(
      MoodContractAddress,
      MoodContractABI,
      signer
    );
  });
});
```

Now that we have the signer and MoodContract - we can create two functions for calling the two smart contract functions

```
async function getMood() {
  const mood = await MoodContract.getMood();
  document.getElementById("showMood").innerText = `Your Mood: ${mood}`;
  console.log(mood);
}

async function setMood() {
  const mood = document.getElementById("mood").value;
  await MoodContract.setMood(mood);
}
```

Finally, update your Buttons such that they call these functions when clicked

```
<button onclick="getMood()">Get Mood</button>
<button onclick="setMood()">Set Mood</button>
```

Save your code, and now it's time to test it!