Ethereum
Blockchain Development
Week 23

Introduction

The intention of this lab is to look at functions and how variables are stored. All the exercises are completed in Remix IDE or the VM provided.

Code Completion

Writing code in a new language can be a steep learning curve. The approach here is to provide some code with underscores (_) that you are required to complete. These underscores are there to help you. By completing these exercises you will be improving your skills and knowledge of Solidity.

Each exercise starts on a new page. The red numbers in the right-hand margin are estimated minutes you should spend on each exercise.

1 Full Stack 1

20–25

This is runs a simple contract and displays the owner's address and balance. The key to this is setup and execution. It relies on the previous 10 weeks work and doesn't introduce anything new. However, it does combine all these parts and produce a full stack for a web3 smart contract.

Follow the instruction below:

- Make a directory, `Bank` and `cd Bank`
- `truffle init`
- Move any Solidity files to contract directory
- Update Migrations folder with associated migrations files
- `truffle compile`
- Update `truffle-config` file
- Run Network on Ganache
- Add the accounts from Ganache to the MetaMask wallet
- `truffle migrate`
- `npx create-react-app bank`
- Download any CSS and Components
- Remove default code
- Rearrange files ensuring all are included in `App.js`
- `cd bank`

The code can be downloaded using `wget` command from links below:

- https://blockchain.smerf.net/23/ex1/App.js
- https://blockchain.smerf.net/23/ex1/Header.js
- https://blockchain.smerf.net/23/ex1/Row.js

- https://blockchain.smerf.net/23/ex1/bulma2.css
- https://blockchain.smerf.net/23/ex1/overflowGood.sol
- https://blockchain.smerf.net/23/ex1/safeMath.sol
- https://blockchain.smerf.net/23/ex1/1_overflowGood.js
- https://blockchain.smerf.net/23/ex1/truffle-config.js

Move the files to their respective directories and type `npm start`.

2 Wallet Interaction

20–25

This is another implementation of the Simple contract. It has two functions `increment` and `decrement` and a state variable `x`. The objective is to complete full stack development for a wallet to interact with the Smart Contract and increment and decrement the state variable, and display that variable value.

2.1 Set up

- Make a directory, `Simple` and `cd Bank`
- `truffle init`
- Move any Solidity files to contract directory
- Update Migrations folder with associated migrations files
- `truffle compile`
- Update `truffle-config` file
- Run Network on Ganache
- Add the accounts from Ganache to the MetaMask wallet
- `truffle migrate`

Example Environment Code – **Do not copy**

```
export const CONTRACT_ADDRESS ='0xeFa348aEB2eF80855E2136Fe7bFFF16Cdb4953f9'
export const CONTRACT_ABI =  [
    {
      "inputs": [],
      "name": "x",
      "outputs": [
        {
          "internalType": "uint256",
          "name": "",
```

Figure 2.1: Environment Code. Stop and think - do NOT cut and paste. Entering this contract address will result in an error for your network. Enter the contract address displayed by migrating the contract, not this one. The ABI here is only a sample and incomplete.

- Add the contract address from the above migration command as shown in Fig. 2.1
- `npx create-react-app simple`
- Download any CSS and Components
- Remove default code
- Rearrange files ensuring all are included in `App.js`
- `cd simple`

3 Raffle

45–60

3.1 Raffle Setup

- **I**nit: `truffle init`
- **R**eact: `npx create-react-app raffle`
- **U**pdate: Complete the following
  - Download raffle.sol and move to contracts directory

  - Download 1_raffle.js to the migrate directory
  - Remove `Migrate.sol` and `1_initial_migration.js` from `contracts` and `migrations` directory, respectively

- **M**igrate: `truffle migrate`
- **G**anache Setup: Complete the following:
  - Update the `truffle-config.js` and uncomment the development network. Ensure that the port is changed to 7545 to match the same port Ganache is operating.
  - Run Ganache and add a new workspace, `w23-ex3`, and run. Look at the 10 accounts created.
  - Ensure that your wallet is connect to `Localhost`
- **I**mport: Import the first account, and at least 3 other accounts to your *Metamask* Wallet (see Lecture/Lab 18). Name these accounts as follows:
  - RaffleOwner
  - Raffle1
  - Raffle2
  - Raffle3
- **E**xport: the contract address and ABI to a separate file, `environment.js`, as shown in Lecture 23. Set up the Web front-end as shown in previous exercise.
- **R**un: the front-end, `npm start`

3.2 Raffle Front Page for accounts

- Migrate the contract using truffle, make a note of the contract address and the ABI.
- Run the truffle console and call the `createAccount` for three accounts other than the owner.
- Connect the accounts in Metamask to the localhost

- Download the `App.js` and replace the current `App.js` from App.js
- Download 1_raffle.js to the migrate directory
- Review the frontpage in a browser?
- Enter the three accounts into the array in `App.js` as shown in Fig. **??**
- Review the frontpage in a browser, are they the same accounts imported?
- The accounts should all have the same balance of 10000
- In the `console`, buy a raffle ticket for each of the accounts, Raffle1, 2, 3.
- Check the balance of the accounts in the `console`, does it match the accounts in the frontpage?

3.3 Further exercise

Write a further component to execute the Winner and Reset functions.
Write a further component to execute the buyTicket functions.

References