Lab Session 3 (or 4)

# Theme of this Lab Session

Today the Lab is split into three parts. This first part involves introductory concepts and practical implementation of JavaScript enabling you to replicate practically the concepts discussed in the mini-lectures. The second part involves theoretical questions that enrich your understanding of JavaScript coding. Part three is an additional (optional) question that requires you to perform some JavaScript coding to the HTML file you created in Lab 1 of this Unit.

This particular Lab's setup is as follows:

* Sit in your arranged group in-person
* If one of the members in his/her designated group had already solved some of the questions it would be helpful to support your group colleagues and try to demonstrate the questions to them.   
  + Now in case you had questions yourself, try to explain it to other group members and check if collaboratively you all can solve it together or you could illustrate the issue. Speaking with other members help in expressing your ideas better and also to get acquainted with other members.
* Expect us to pass by asking you questions and listening to your answers

# Pre-requests skills for this Lab

* Having and running VS code editor with the configuration done in Lab session 1.

Main Lab Questions

# Part 1: Ethereum concepts replicated using NodeJS

In Lab 2 of this Unit, you had used the front-end user interface to generate private and public keys from a mnemonic phrase. You will be asked similar questions in this question, but you need to conduct this using Nodejs. So, you will act as the back-end developer from the user perspective. Please answer the following questions.

**Mini-lectures**: 6, 7, 8, 9, & 10. If you have not watched those mini-lectures it would be hard to solve the below questions.

**Expected time**: 1.5 hours

## Questions:

1. Installing Long-term service (LTS) version of NodeJS available in this [link](https://nodejs.org/en/download/)
2. Writing your first NodeJS Script. Please do the following
   1. Open VS Code editor
   2. Create a new folder (“First NodeJS Script”)
   3. Create a “HelloWorld.js” file
   4. Write the following code inside of it “Console.log(“Hello World”)
   5. Realize that the “console.log” is not part of the JavaScript programming language but part of the run time. To see this open this [link](https://nodejs.org/dist/latest-v16.x/docs/api/console.html#consolelogdata-args)
   6. Open Terminal in “First NodeJS Script” folder
   7. Write: node HelloWorld.js
3. Using this [link](https://www.freecodecamp.org/news/modular-programming-nodejs-npm-modules/) and this [one](https://docs.ethers.io/v5/getting-started/) as a resource, Install the module known by ethers, then require it in your NodeJS file called "Example1.js". Then answer the following list of simple questions.
   1. What is ether.js?
   2. Using ether module and on your NodeJS file, create a Wallet from a Mnemonic phrase you have done in Lab 2.
   3. Print out the private key of the Wallet created using the “.privateKey” method.
   4. Confirm whether the private key generated matches the private key of your MetaMask account
4. Another important NodeJS packages that provide developers with a collection of utility functions for Ethereum blockchain is [ethereumjs-util](https://www.npmjs.com/package/ethereumjs-util), [web3](https://web3js.readthedocs.io/en/v1.5.2/getting-started.html), and. ethereumjs-util and web3 needs to be installed using the node package manager (npm) i.e. "npm i --save-dev ethereumjs-util web3 @truffle/hdwallet-provider".   
     
   Web3js is one of the most powerful NodeJS libraries that will play a critical role in allowing developers to interact with the local or remote Ethereum node using HTTP. In particular, in this question you will use this library to connect to Infura and understand how we could use built-up functions to interact with a full node (Ethereum Client). Please answer the following set of sub-questions.  
   1. Copy the project ID of your Infura account and store it in a constant variable, "Project\_ID\_Infura" in your JS file.
   2. Using the methods privateToPublic, bufferToHex, and Buffer.from(‘your private key, 'hex') from packages ethereumjs-util and Buffer. Get the public key from the private key which you can download from MetaMask.
   3. How could one get the public address from public key?
   4. Using NodeJS, get the public address from the public key created in part b.
   5. Check if the found public address matches that of MetaMask
   6. Using Infura, web3 library, and [ether library](https://docs.ethers.io/v5/) check the balance of the account using public address in NodeJS. Do this using a callback function and using promises. Furthermore, is public address enough for allowing other network users to know how much fund there is in your account?
   7. Now repeat sub-question part f using asynchronous function.
   8. Why should we know how to get data using both promises and async-await?
5. In the main lectures, we discussed the concept of local blockchain simulation and that it is mainly used for testing purposes. Ganache is one of the most popular local blockchain simulations. In particular, Ganache is a personal blockchain for Ethereum development you can use to deploy contracts, develop your applications, and run tests. In this question, we shall use the command-line flavour of Ganache offered by ganache-cli as documented [here](https://www.trufflesuite.com/docs/ganache/overview). Therefore, please have a read of this [link](https://github.com/trufflesuite/ganache-cli-archive/blob/master/README.md) before solving the sub-questions in this question.   
   1. Does ganache run a full node?
   2. Install the command-line version of ganache on your machine using the following command “npm install ganache-core”
   3. Using the getAccounts() method, write an async function to get all the accounts on Ganache.
   4. Using Nodejs code, check the balance of the second account in this list of accounts provided in Ganache, how much ether is there?
   5. On your terminal write the following “ganache-cli -m "stereo consider quality wild fat farm symptom bundle laundry side one lemon"” comment on the output.
   6. Suppose you want to use your MetaMask for a local instance of a Dapp. This requires you to use the Ganache account from MetaMask. Therefore, it is crucial to set up Ganache onto your MetaMask. There are two ways of doing so. What are they? Try to do either of them.
   7. Define nonce of a wallet account, then find the nonce of the first account in Ganache list of addresses. Does it meet your expectation? Why?
   8. When doing local testing, such as sending ether from one account to another using Ganache-cli, is it required to sign the transaction, as is usually the case when dealing with testnets or the main net?
   9. Using the above sub-question demonstrate the concept of sending test ether from one account to another in ganache. Also within this code, show how the nonce of the account is changing after the transaction.

1. In this question, we shall focus on transactions using JavaScript. Mainly the focus here is on transactions from EOA to another EOA. Please answer the following sub-questions:
   1. What is a raw transaction?
   2. Create a transaction data whereby you want to send money from Wallet 1 to Wallet 2. Do you think you need a private key of any of these two wallets, to write the transaction data?
   3. Using similar concepts to questions 4 and 5, write a JS code that is able to send test Ether from Wallet 2 to Wallet 1 by signing the raw transaction using the private key and sending it to the Rinkeby testnet.
2. Mocha is a feature-rich JavaScript test framework[[1]](#footnote-2) running on Node.js and in the browser. Chai is an assertion library that is often used alongside Mocha. It provides functions and methods that help you compare the output of a certain test with its expected value. One of the main reasons why it is important to learn Mocha and chai is that they are both integrated into the truffle suite. Therefore, before we explore Truffle Suite in our next lab session, it is important to learn the basics of Mocha and Chai in this question. In this question, you are requested to demonstrate a test of a particular function.
   1. Follow these steps
      1. Create a new folder name it “testingJS”
      2. Open terminal at this location
      3. Create a package.json file using the command line “npm init -y”
      4. Please take a minute to see what this package.json file contain. Particularly, observe that it has “scripts”. You will need this information to answer one of the questions in part 3.
      5. On the terminal write “npm install --save-dev mocha chai’
      6. Re-observe the package.json file and see the changes that occur
      7. In the package.json file change the “"test": "echo \"Error: no test specified\" && exit 1"” to “"test": "mocha"” and save it.
      8. Create a two new folder called “scr” and “test”
   2. Create an “app.js” file which represents a Node module system that exports the average of three numbers at that folder “scr”
   3. Create an “app.test.js” in the test folder. These sub-questions are related to Mocha
      1. In this file, you need to require the Node.js assert module and the “app.js” file you created in part b. Test if the function created in app.js does what it is expected to do.
      2. In the terminal run “npm run test” in the folder “testingJS”. Now go back again and try to put false assertions and try to see what will happen.
      3. There is many important [asynchronous hooks](https://mochajs.org/#:~:text=%23-,ASYNCHRONOUS%20HOOKS,-All%20hooks%20(before) using mocha. One of them is beforeEach().
   4. These subquestions are related to chai.
      1. Amend the “app.test.js” file by requiring [chai](https://www.chaijs.com/).
      2. Do the same test you have done in Mocha i.e. test the average function.
      3. Do additional test i.e. using the it function to see whether in the average function you can put two numbers only.
      4. When running npm run test, what was the output in the terminal? If there was an error try to fix it and run it again. Furthermore, try to do more testing and see what amendments you could do to the app.js file so that all your test could run as expected.

# Part 2: Tutorial Questions

**Things to help to solve these questions**: Attend all mini-lectures and extra readings, you need to do some research about topics uncovered in class.

**Expected time**: 1.5 hours

## Questions:

1. Why is nodejs important in the field of blockchain development?
2. Why is modular programming is considered one of the pros of using NodeJS, and how is it being implemented in practice? Furthermore, how is this related to blockchain development?
3. React as well as HTML, CSS, and JavaScript are about building user interfaces, so what is the incentive of using React?
4. Using the reference available [here](https://nodejs.dev/learn/nodejs-buffers), and in many instances in some of the sub-questions of question 4 in Part 2 of this Lab, you were asked to use buffer.from() method. In particular, when attempting to get the public key from the private key, you have used buffer.from('your private key, 'hex'). Why was that? What is the concept of buffer in computer science, and how is it related to blockchain technology?
5. This question will contain a list of questions that helps in understanding the concepts related to testing particularly the testing framework Mocha. Please answer the following sub-questions
   1. What is software testing (particularly unit testing) and what is its purpose?
   2. Why unit testing is important and how is it related to Ethereum?
   3. What is a JavaScript Framework?
   4. What is a JavaScript Testing Framework?
6. What is a package.json file and how is it related to Node programming? Why do developers need it in projects?
7. What is ES6 and how are its modules different from Node modules?

# Part 3: Web Development using JavaScript

This question is a follow-on question from Lab session one. There you were asked to design a web page for a coin-guessing game. Here, we need to modify this web page to use javascript codes to make a tossing coin game. Therefore, please copy the folder you created in Lab session one to this Lab and answer the following questions.

**Mini-lectures**: 4 & 5 along with coin toss game in extra material

**Expected time**: 1.5 hours

## Questions:

1. Following the pre-recorded mini-lecture and the video [here](https://www.youtube.com/watch?v=yXEcd0eGrpw), adjust your webpage so that your HTML file interacts with the keys. In particular, do a spirit to your game. Note: Feel free to watch the pre-recorded videos using your ear/headphones in class. You can also use your own way of styling or writing a different algorithm that does what we expect it to do.

1. More detailed questions about testing framework are available in the next Part. [↑](#footnote-ref-2)