Potential Answers to Lab 3

# Part One:

1. This has been done in class
2. Students should follow the steps with the guidance of the teaching team also the main lectures will cover this in more detail.
3. Answer:
   1. The ethers.js is a NodeJS library that aims (but is not) to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem. It has many features such as connecting to Ethereum nodes over JSON-RPC, INFURA, Etherscan, Alchemy, Cloudflare or MetaMask. Its full documentation is available here: <https://docs.ethers.io/v5/>.
   2. That is how  
      
   3. This is easy  
      
   4. This needs you to check the Metamask private key. You can get the MetaMask private key by following this [link](https://metamask.zendesk.com/hc/en-us/articles/360015289632-How-to-Export-an-Account-Private-Key). Then you need to compare it with what you got in part c.
4. Solutions to sub-questions



* 1. The code below will connect you to web3 and   
     
  2. The Ethereum address associated with this private key is the last 160 bit of the SHA3–256 (Keccak) hash of the public key. See this [link](https://medium.com/@codetractio/inside-an-ethereum-transaction-fa94ffca912f) for further details.
  3. This is done by following this line of codes  
     
  4. Should be able to get the same public address of MetaMask
  5. Using the public address only, anyone in the network is able to read the balance of the individual using the following code.

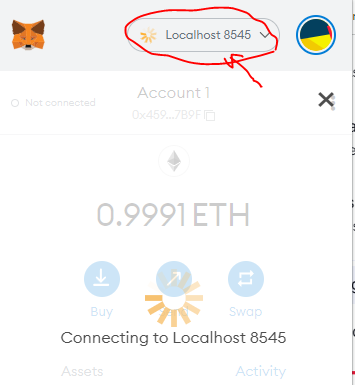


* 1. The code is available below  
     
  2. The answer is simple, in some applications, a developer might find it more feasible to use promises, and in others, they might prefer async-await. Therefore, in this module, you need to know both. If you want more details on when it is better to use async-await over promises and vice versa, this is a good [link](https://www.bitcoininsider.org/article/36058/should-i-use-promises-or-async-await) for further resources.

1. Solution
   1. No, ganache is a local blockchain it is only the developer who is running the blockchain. The main purpose is to be used as a way to test “Smart Contracts”. Particularly, “Ganache CLI uses ethereumjs to simulate full client behaviour and make developing Ethereum applications faster, easier, and safer.”
   2. Students are expected to do so on their terminal
   3. The code is available here  
        
      
   4. The student should be able to write a version of the code provided below, and they should show that each account contains 100 ETH.   
        
      
   5. The student should be able to see Available accounts, Private keys of these accounts, Mnemonic. Student should be able to relate those concepts with the ideas discussed in previous question.   
      Text

      Description automatically generated

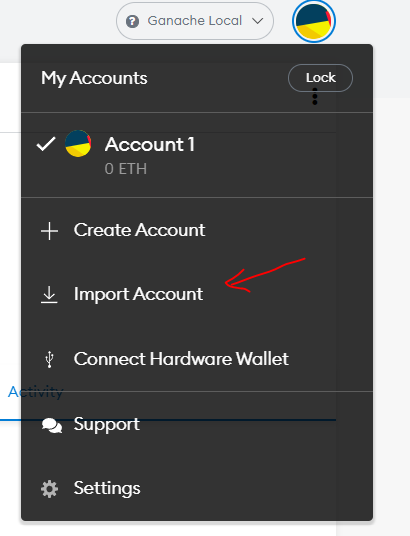
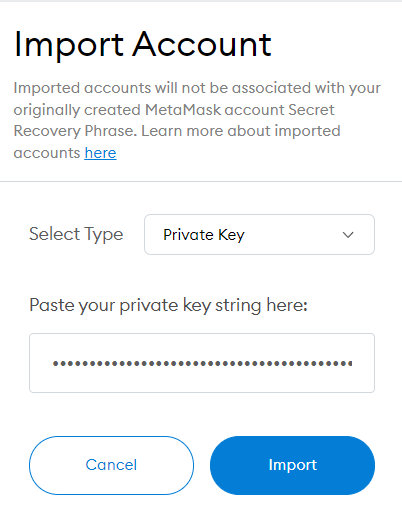
Student should be able to see that from one mnemonic phrase of 12 words, ganache was able to generate several accounts and that each account has an associated private key.   
  
A good idea to also realize is when you connect to MetaMask using the same mnemonic key, you should find that you have 100 eth in your account when the network is set to “Localhost 8545”

* 1. Setting up Ganache and MetaMask is very handy and essential, as we shall see in future labs. Most likely, you will be using the MetaMask account when testing deployed contracts, especially when dealing with front-end development. Therefore, having test ether using local blockchain is essential to pay gas fees (i.e., computational fees). There are two ways to do so using ganache. Let us start with the easy one.   
       
     First Method: If you know the wallet mnemonic of the MetaMask account you currently have, then it is helpful to run the same account mnemonic on your terminal. But before doing so, observe that the MetaMask account does not load Localhost 8545, as depicted in this photo.  
       
       
     Simply open terminal and write “ganache-cli -m "sign stadium slight inform frost harbor mix ghost process monkey wasp glance"”  
     A picture containing graphical user interface

     Description automatically generated  
     Now check your MetaMask account you should see that the Localhost 8545 is loaded with a 100 test Ether!  
     Graphical user interface, application

     Description automatically generated  
     Next Method: If you do not want to use your phrase. You can create a new network. You will need to add the following details.   
     A picture containing diagram

     Description automatically generated  
     This will create a new network. Then on your terminal you lunch ganache by writing “ganache-cli”   
     A picture containing text

     Description automatically generated  
     Then you import account  
       
     You copy and paste one of the private keys of the wallets generated by ganache   
       
       
     Then you are done!  
     Graphical user interface, application

     Description automatically generated
  2. The nonce is the number of transactions sent from a given address. In English, a nonce is a number that can only be used once. Having said this, if every time we load Ganache, I expect since no transaction on this account exist the nonce will be 0. The code that demonstrates this is available here.  
       
     
  3. No, there is no need to sign the transaction. This is mainly because everything is a fake currency or test ether, and usually, the developer is the only one dealing with the testing phase. Therefore, there is no need to sign the transaction using a private key of a particular address, as it is usually the case when more individuals are using a testnet or a main public net. Remember that getting test ether in testnet is not as easy as getting them on ganache.
  4. Here is the code  
     

1. Solution
   1. A raw transaction is not well defined in the Ethereum world. When this new field emerged, many developers were interested in getting things done with little attention to the historical academic relevance. For example, in 2017, in one of the web3 JavaScript old versions, developers infer from their codes that a raw transaction is "a machine representation of a transaction, with the signature attached to it." However, since Ethereum was built based on the Bitcoin blockchain concepts, raw transaction most likely meant Bitcoin's definition.

So, what is Bitcoin's definition of a raw transaction? According to Bitcoin, a raw format of a transaction (a.k.a. serialized transaction or raw transaction) is a transaction in a binary form that is often represented using a hexadecimal number [[link](https://btcinformation.org/en/glossary/serialized-transaction)].

* 1. When writing a transaction, no private key is required. This is similar to two people wanting to exchange money. Neither of the two individuals needs to share their bank passwords to write the transaction. The code of such a transaction in JS is available below.  
       
     
  2. The code is available here  
     

1. Solution
   1. This should be easy to follow. However, it is important that the student realize that there is “script” in package.json. This is because as the student will see in part 3, that package.json is important file for NodeJS development. As previously without package.json it was not straight forward to import modules written in ES6. Student should also realize that after installing the packages using npm, that this file updates the dependency property with the associated versions.
   2. Here is the code:  
      
   3. Solution
      1. Student should be able to write this code in app.test.js  
         
      2. Student should be able to run the code in the terminal using “npm run test”   
           
         
   4. Solution
      1. Here and
      2. Here  
         
      3. Students are expected to write this code:  
         
      4. The students are expected to report an error and they need to find out how to solve it by fixing the ‘app.js’ file. After doing this set of tests  
           
         One could have found that app.js file could be written as follow  
         

# Part Two:

1. To answer this question, it is important to define Nodejs and exactly understand its conventional use. By understanding this, it will be easy to see why NodeJS is ultimately important in blockchain development.

By definition, NodeJS is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. This is important to point to concepts here "JS runtime environment" and that it can run "outside a web browser".

The idea of a "runtime environment" means that Node JS is not a programming language, but it allows developers to use JavaScript, which is a programming language that allows users to build web applications. Why is that? Because web application requires to access to the file system and databases of the computer they are connected to (please refer to Lecture one about how the internet works).

The meaning of running "outside a web browser" is that NodeJS extends JavaScript language functionality beyond front-end usage only. In particular, with NodeJS the developer can access databases and other information within the internet to a connected server. This is especially important when a developer needs to run a certain code or interact with the server.

Having said this, it sounds clear that NodeJS is used for back-end development, meaning that for a developer to interact with other nodes (full node in the Ethereum blockchain), one may need to learn NodeJS, which is built on JavaScript programming language.

1. NodeJS has many advantages, and one of the main advantages is that many big corporations such as Uber, eBay, Netflix, PayPal, LinkedIn, among others, use it as their back-end web development. This implies that because of the many developers (authors)) that exist out there who are interested in supporting these companies, the NodeJS runtime environment has a rich set of modules that helps in making developers' future projects easier. As future developers may build their projects on existing modules.

Remember that the goal of modular programming is to allow large programs to be assembled using modules of code from diverse authors and sources and for all of that code to run correctly even in the presence of code that the various module authors did not anticipate."

It seems evident why NodeJS is one crucial concept that needs to be understood as re-inventing the wheel for the Ethereum blockchain technology may be cumbersome. Using existing technology to complement new technology is always important as it gives a good infrastructure and foundation.

With Node's modular functionality, we can import our external files, core (Native) node modules, and NPM modules. For more details on how this can be achieved, please read the following link: <https://www.freecodecamp.org/news/modular-programming-nodejs-npm-modules/>.

1. React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. The React basically allows developers to utilize individual parts of their application on both client-side and the server-side, which ultimately boosts the speed of the development process. It uplifts developers’ productivity as different developers can write individual parts and all changes made won’t cause the logic of the application. More details are available [here](https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html).   
     
   The main idea is the concept of components in React.js a component is a JavaScript class or function that optionally accepts inputs i.e., properties(props) and returns a React element that describes how a section of the UI (User Interface) should appear.

In particular, all user interfaces on all kinds of web applications can be split up into components. Such a separation of components helps developers manage large files that hold all the HTML codes and JavaScript logic. This is helpful, especially when each component has one apparent concern, focus, or task.

So, the beauty of React is connecting these components and translating them to the front-end web application. Hence, using React as opposed to html, CSS, and JavaScript may help in making the interface process easier.

1. In computer science, a data buffer (or just Buffer) is a region of physical memory storage (area of memory such as RAM compared to Hard Disk) used to temporarily store data while it is being moved from one place to another. Buffers are typically used when there is a difference between the rate at which data is received and the rate at which it can be processed. It represents a fixed-size chunk of memory.

A buffer can be thought of as an array of integers, representing a byte of data. On Node.js there is a Buffer Module already in place as described [here](https://www.w3schools.com/nodejs/ref_buffer.asp). It is worth having a look on the many properties and methods available for this Module. However, for the sake of this question, we shall focus on a particular method known by Buffer.from().   
  
Buffer.from() method creates a Buffer object from a JS object such as string, array, or Buffer. This method has two arguments, one essential and the other one optional. If the optional input is not inserted, then it uses the UTF-8 encoding method by default. This means that this method converts a string array for the sake of argument to a UTF-8, a variable-width character encoding used for electronic communication.

However, in the ethereumjs-util, we will be dealing with getting the public key from a private key. Therefore, it is crucial to understand how a private key is defined in the Ethereum yellow paper. Once this is known, it can be seen why there is a need for a buffer function. In particular, this will shed light on how a buffer is related to blockchain technology.

By definition, in the yellow paper specification, a private key is "a randomly selected positive integer (represented as a byte array of length 32 in big-endian form)". What does this mean? This means that there is a concrete definition of this unique type of data. In particular, Big endianness generally is the dominant ordering in networking protocols, such as in the internet protocol (IP) suite, where it is referred to as network order, transmitting the most significant byte first. Therefore, it is not difficult to see how blockchain technology embraced the functionality of existing technology, such as the big-endianness. In computing, endianness is the order or sequence of bytes of a word of digital data in computer memory.

On most modern computers, the byte is the basic unit of information in computer storage and processing. A byte consists of 8 adjacent binary digits (bits), each of which consists of a 0 or 1.

Since the private key is of length 32 bytes, the private key downloaded from the MetaMask of length 64 needs to be encoded into 32. This is achieved by using the Buffer.from() method, where the second argument needs to be 'hex'. This implies that the private key will be buffered using the Hexadecimal encoding, which converts the 8-bit data (1 byte) to 2 hex characters. This implies that although in characters there might be 64 in buffer bytes the buffer.from will be of length 32 bytes as illustrated below.  
  


1. Answers
   1. Software testing is the process of evaluating and verifying that a software product or application does what it's supposed to do. Its primary purpose is to test each unit or function. A unit is the smallest testable part of an application. It mainly has one or a few inputs and produces a single output. In object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class, or derived/child class.
   2. Unit testing is important because, as developers, we need to ensure that the function or method does what it is supposed to do, especially when money is involved and when one of the premises of Ethereum (Censorship-resistant) needs to be met. Censorship-resistant means that once a smart contract is deployed to the network, the developers can never claim it back to fix errors.

Unit testing allows developers to ensure that their developed modules are meeting the requirements specified by the smart contract. Therefore, as Smart Contract developers, it is important that tests are carried out after every new module (or a function of the smart contract) is created.

* 1. To answer this question, it is important to define a framework. Frameworks provide a template that handles common programming patterns. Each time a developer has to build an application, she does not need to write code for every single feature from scratch. Instead, she can build upon an existing feature set.

Therefore, a JavaScript framework is a collection of JavaScript code libraries that provide a web developer with pre-written code for routine programming tasks. Therefore, JavaScript frameworks, like most other frameworks, provide some rules and guidelines. Using these rules and guidelines, any developer can make complex applications faster and more efficiently than if they decided to build from scratch. The rules and guidelines help shape and organize developer's web application.

* 1. As said, JavaScript framework is a collection of JavaScript code libraries that provide a web developer with pre-written code for routine programming tasks. Therefore, the JavaScript Testing framework is a collection of JavaScript code libraries oriented towards testing units and functions.

1. The package.json file is a file named package with a JSON type format. What is JSON type format mean? JavaScript Object Notation (JSON) is a syntax that is a subset of JavaScript syntax (a set of rules that defines the combinations of symbols that are considered to be correctly structured statements or expressions in that language).

JSON is usually involved in the object serialization process. It is the process of converting an object's state to a string from with it can later restore. For example, consider the code below:  
  
  
  
Therefore, JSON is a text-based format for representing structured data based on JavaScript object syntax. As a matter of fact, on many occasions, it is being used as an alternative to Extensible Markup Language (XML) structured data. Hence, it could be used as an alternative to store and transport data over the internet (client to server).

Now how is this related to “package.json”? The main idea that you should have got was that JSON file is helpful, especially when "transport data over the internet". Now, this is a vital feature to keep in mind.

Generally, developers may need to import modules from different resources to run their codes in development mode. Once developers are pleased with their codes, they need to export the modules required over the internet. Therefore, a file that can store all these modules (packages) that is easy to transfer over the internet would be a virtue for a developer. Furthermore, it might be due to some technicalities (such as security restrictions) that the developer cannot import modules from different resources; therefore, if there is a file that can be read by NodeJS and be interpreted as the codes required by the developer in the development mode that would be wonderful for the developer. That is exactly what the package.json file do for NodeJS programming!

The package.json file is the heart of any Node project. This is because of the points raised in the previous paragraph. In particular, as a NodeJS developer, you may need to use ECMAScript version 6 (ES6) modules[[1]](#footnote-2) with NodeJS modules without the need of modular JavaScript files. Particularly, to call modules from ES6 to NodeJS, you need some workaround to get this done. Traditionally this was done by developers using .mjs file extension, among other things.

Now, however, NodeJS developers do not need to worry about this as Node usually interpret the package.json file and does this job for them.

In particular, this package contains serialized JavaScript objects that record important metadata[[2]](#footnote-3) about a project.

In particular, package.json can contain the ES6 modules required to run the file the developer’s need in development mode. Therefore, Node parses this package.json file and loads the ES6 modules required by the developer. Then once the developer is pleased with his development phase, he/she can easily send the file over the internet to the server. Hence, it is stored in JSON format.

1. ES6 is an abbreviation of ECMAScript version 6. ECMA stands for European Computer Manufacturer’s Association. JavaScript is considered as one of the most popular implementations of ECMAScript. Therefore, in practice, everyone just calls the language JavaScript.

Modules in ES6 work only in strict mode. This means variables or functions declared in a module will not be accessible globally. Furthermore, ES6 modules differ from Node modules in the syntax used for exporting and importing and also in the way that modules are defined in web browsers.

# Part Three:

Same as in the video mini-lectures!

1. JavaScript is considered as one of the most popular implementations of ECMAScript. Therefore, in practice, everyone just calls the language JavaScript. [↑](#footnote-ref-2)
2. What is metadata? Metadata is a set of data that describes and gives information about other data. For example, metadata for a document might include a collection of information like the author, file size, the date the document was created, and keywords to describe the document. [↑](#footnote-ref-3)