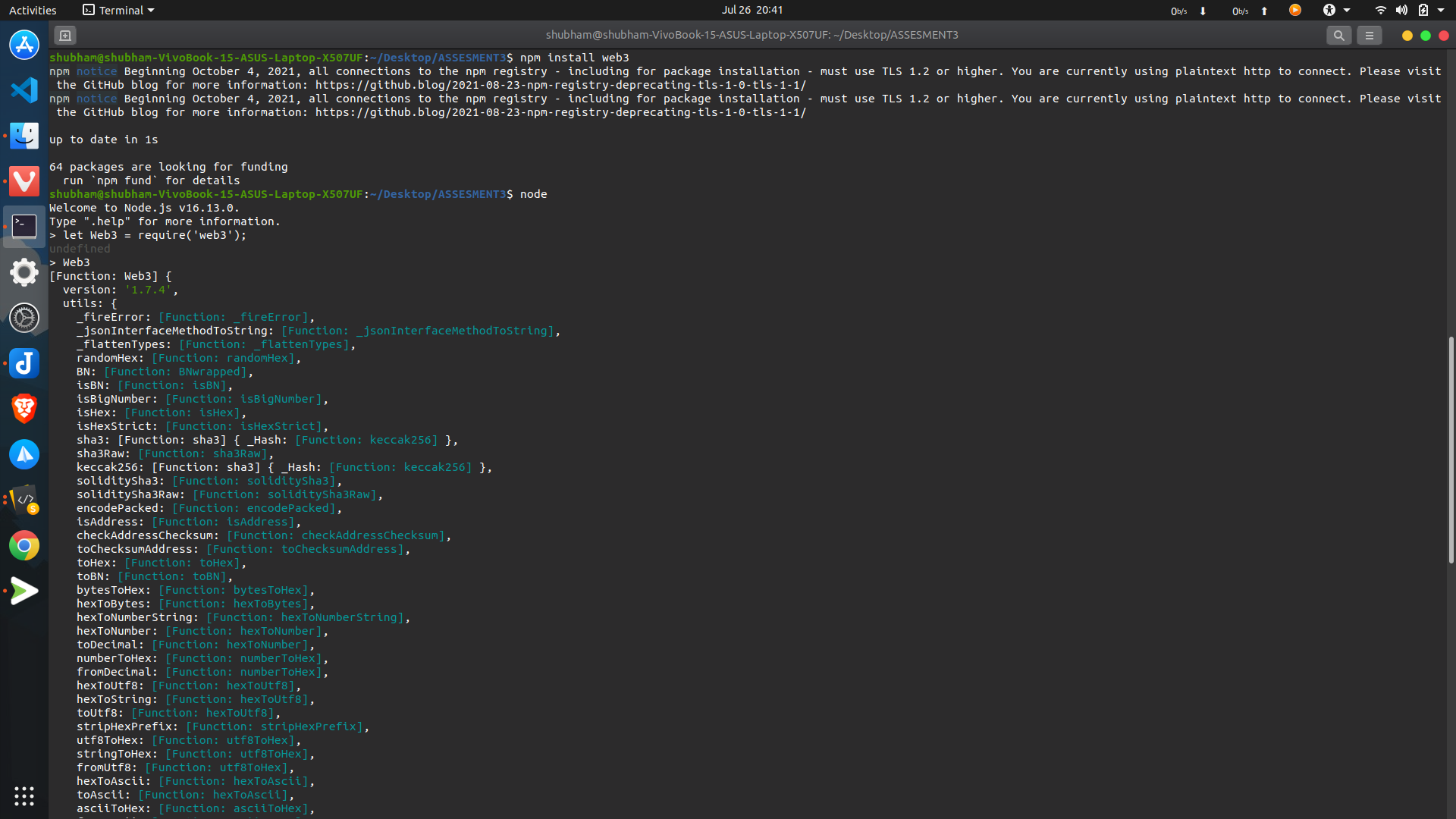
Prerequisites - ganache, terminal ,sublime text, Node

1. Initialize the project to use node. We are going to run the following command.
2. npm init
3. Next, we need to add a few dependencies. For this, we are going to run the following commands.
4. npm install web3 --save
5. npm install ethereumjs-tx --save
6. Now, we have installed dependencies, we are going to add the code. Lets us add a new file to project Election.js

**Dependency Objects**

1. We are going to add the following dependency objects.
2. const Web3 = require('web3');
3. const EthereumTx = require('ethereumjs-tx').Transaction;
4. These objects provide functions, which we are going to use to set up the connection, sign transactions, build transaction objects and other utility functions.



**Initializing Connection**

1. We are going to set up a connection with our local Ethereum blockchain.
2. const rpcURL = 'http://127.0.0.1:7545';
3. const web3 = new Web3(rpcURL);
4. In this case, the local Ethereum node is providing following rpc endpoint *http://127.0.0.1:7545*
5. Let us add these details to code.
6. let abi = '[{"constant":false,"inputs":[{"name":"value","type":"uint256"}],"name":"update\_quantity","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"get\_quantity","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]'

let bytecode = '608060405234801561001057600080fd5b50606460008190555060ca806100276000396000f3fe6080604052348015600f57600080fd5b506004361060325760003560e01c806380219655146037578063ed0109a5146062575b600080fd5b606060048036036020811015604b57600080fd5b8101908080359060200190929190505050607e565b005b6068608c565b6040518082815260200191505060405180910390f35b806000540160008190555050565b6000805490509056fea265627a7a7230582002f975dfd70c1b1f649671805826a83fc9b92457fe7dd245527f56b7776d043464736f6c634300050a0032';

**Contract Object**

1. Next, we are going to create a contract object.
2. In order to create the contract object, we will use the contract abi.
3. // Contact ABI
4. let deploy\_contract = new web3.eth.Contract(JSON.parse(abi));
5. We are going to need an Ethereum account for contract deployment.
6. For account information, we are going to use one of the accounts in Ganache.
7. Next, we are going to add this to our code.
8. // address from Ganache

let account = '**0xc2d961F35550B30a1bB85362e3355Fe603bC8E29** ';

1. Next, we also need to provide metadata for contract deployment.
   * First of all,  *from:* The account which will be used to deploy the contract. The account needs to have ether, in order to deploy the contract.
   * Next, *gas*: The max limit on the cost, the account is willing to incur for contract deployment. If the cost goes above this value. The contract deployment will be reverted.
   * Finally, *gasPrice*: The price of each unit of gas the account is willing to pay. The transactions get prioritized Based on the *gasPrice*.
2. let parameter = {
3. from: account,
4. gas: web3.utils.toHex(800000),
5. gasPrice: web3.utils.toHex(web3.utils.toWei('30', 'gwei'))

}

**Deploying the Smart Contract**

1. Now, we have all the bits and pieces required.
2. Finally, the last piece of the puzzle, let us add the last piece of code.
3. deploy\_contract.deploy(payload).send(parameter, (err, transactionHash) => {
4. console.log('Transaction Hash :', transactionHash);
5. }).on('confirmation', () => {}).then((newContractInstance) => {
6. console.log('Deployed Contract Address : ', newContractInstance.options.address);

})

## ****Putting it all together****

1. This is how our server.js file looks, after putting all the individual pieces together.
2. // Library Imports
3. const Web3 = require('web3');
4. const EthereumTx = require('ethereumjs-tx').Transaction;
5. // Connection Initialization
6. const rpcURL = "http://127.0.0.1:7545";
7. const web3 = new Web3(rpcURL);
8. // Data set up
9. let abi = '[{"constant":false,"inputs":[{"name":"value","type":"uint256"}],"name":"update\_quantity","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"get\_quantity","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]'
10. let bytecode = '608060405234801561001057600080fd5b50606460008190555060ca806100276000396000f3fe6080604052348015600f57600080fd5b506004361060325760003560e01c806380219655146037578063ed0109a5146062575b600080fd5b606060048036036020811015604b57600080fd5b8101908080359060200190929190505050607e565b005b6068608c565b6040518082815260200191505060405180910390f35b806000540160008190555050565b6000805490509056fea265627a7a7230582002f975dfd70c1b1f649671805826a83fc9b92457fe7dd245527f56b7776d043464736f6c634300050a0032';
11. //Contract object and account info
12. let deploy\_contract = new web3.eth.Contract(JSON.parse(abi));
13. let account = '0xd935580Ce80986aD46D31e2dA55564Eb93A09318';
14. // Function Parameter
15. let payload = {
16. data: bytecode
17. }
18. let parameter = {
19. from: account,
20. gas: web3.utils.toHex(800000),
21. gasPrice: web3.utils.toHex(web3.utils.toWei('30', 'gwei'))
22. }
23. // Function Call
24. deploy\_contract.deploy(payload).send(parameter, (err, transactionHash) => {
25. console.log('Transaction Hash :', transactionHash);
26. }).on('confirmation', () => {}).then((newContractInstance) => {
27. console.log('Deployed Contract Address : ', newContractInstance.options.address);

})

1. In order to execute this code, run this from the terminal window

node server.js

