Workshop Pre-requisites :

1. Windows, linux or Mac
2. Node.js v12 or later :
   Set up instructions at
   https://docs.npmjs.com/downloading-and-installing-node-js-and-npm
3. Metamask Wallet **Browser Extension***:*
   a. *Download at* https://metamask.io/download/
   b. Account *Set up Instructions* at
   https://myterablock.medium.com/how-to-create-or-import-a-metamask-wallet-a551fc2f5a6b

It would also be nice to know about Remix IDE (remix.ethereum.org/) & Truffle Development Environment (https://trufflesuite.com/docs/truffle/index.html) but not needed.

Workshop
Part 1

1. Asynchronously Setting up Node.js and npm for people who don't have it installed. Instructions at https://docs.npmjs.com/downloading-and-installing-node-js-and-npm & https://github.com/nvm-sh/nvm (optional)
1. Set up Metamask Wallet
   a. Enable Showing Test Networks
2. Get some test ETH from https://faucets.chain.link/rinkeby
   a. Be sure to connect your wallet first with the help of the button on the top right corner of the screen.
3. Create an Infura account -> Ethereum Project -> toggle to Rinkeby Testnet
4. Create an account on Postman
5. https://www.postman.com/downloads/
   a. Unselect the Content-Type under headers,
   b. Create your own Content-Type with the value as application/json.
   c. Use the link : http://localhost:3000/purchase/<api-key> to make a **POST** request.
   d. Put in the Body (Raw), an account address (2nd account from Metamask) for customer and a certain amount like so,

```
{
  "customer": "<your 2nd account address from metamask>",
  "amount": 1000
}
```

6. Remix Contract Deployment of the preset Minter Pauser ERC 20 at remix.ethereum.org/
   Steps :
   1. Importing of the contract from openzeppelin.
   2. Using the constructor to initialize the token contract
   3. Activate the Solidity Compiler on the Remix IDE in the plugin section.

4. Compile the contract.
    a. Make sure the solidity version being used by the solidity compiler is the same as the one specified in the smart contract at the top.
5. Go to Deploy & Run Transactions :
    a. Make sure the environment is set as Injected Web3
    b. Change the contact to Token.sol or whatever file name you have put the contract code in.
    c. Click on Deploy
    d. Metamask extension will pop up
        i. Make sure to change the network to Rinkeby Testnet
    e. Click on confirm
    f. Click on 'view on etherscan' in the Remix Console to view the transaction of the deployment.

Potential Code for Solidity [Steps 1 and 2 of Part 1]

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/presets/ERC20PresetMinterPauser.sol";

contract Learn is ERC20PresetMinterPauser {
  constructor(string memory name, string memory symbol) ERC20PresetMinterPauser("Learn",
"LRN"){
  }}
```

Part 2

1. Make a new folder
    a. mkdir <folder-name>
    b. cd <folder-name>

Start with setting up a nodeJS server. If node is not in path then go here:
https://www.tutorialspoint.com/nodejs/nodejs_environment_setup.htm

Run the following commands at the project root directory

1. npm init,
2. npm i -s express
3. npm i dotenv
4. npm i web3@1.3.4
5. npm i body-parser
6. npm i node-fetch@2.6.1

Create the following files and folders :

1. 'index.js' at the root directory
2. 'queries.js' at the root directory
3. '.env' at the root directory
4. Folder 'APIs' at the root directory
    a. Inside APIs, a file named 'brockChain.js'
5. Folder 'Artifacts' at the root directory
    a. <your-contract-name> & paste the ABI inside from Remix.

In your index.js file add

```
const express = require("express");
const bodyParser = require("body-parser");
const app = express();
const port = 3000;
const q = require("./queries.js");
require("dotenv").config();

app.use(express.json());
app.use(bodyParser.urlencoded({ extended: false }))

app.post(`/purchase/:api_key`, q.brockChain.purchase)
// app.get(`trackTransations/:api_key`, q.brockChain.trackTransations)
app.listen(port, () => {
console.log(`App running on port ${port}. API is pointed towards
${process.env.API}, with
```

```
${process.env.WALLET} as the wallet account address.
${process.env.FILE} is the ABI being used, on CHAINID
${process.env.CHAINID} WITH RPC ${process.env.RPC}`);
});
```

Inside queries.js add

```
var brockChain = require("./APIs/brockChain.js");
module.exports={brockChain}
```

Inside .env add

```
WALLET=yourwallet
PRIVATE_KEY=yourprivatekey
API_KEY=smashyourkeyboard
API=http://localhost:3000
RPC=https://rinkeby.infura.io/v3/{ProjectID}
FILE=Token // assuming the solidity contract name is Token.sol REMOVE THIS
COMMENT AFTER
CHAINID=4
ADDRESS=yourtokensaddress
COVELANTKEY=COVELANTKEY
```

Inside APIs -> brockChain.js file add the following

```
const { response } = require("express");
const { default: fetch } = require("node-fetch");
require('dotenv').config();
const Web3 = require('web3')
const FILE = process.env.FILE
const CHAINID = process.env.CHAINID
const RPC = process.env.RPC
const PRIVATE_KEY = process.env.PRIVATE_KEY
const API_KEY = process.env.API_KEY
const WALLET = process.env.WALLET
const API = process.env.API
const ADDRESS= process.env.ADDRESS
const abi = require(`../Artifacts/${FILE}.json`)

const purchase = async (request, response) =>
{
   const requestingKey = request.params.api_key
   if (requestingKey != API_KEY) {
       response.status(500).send({ error: 'not authorized' })
   }
   else {
       console.log(request.body.customer);
       console.log(request.body.amount);
```

```
        const {customer, amount} = request.body
        const web3 = new Web3(RPC)
        var gasprice = await web3.eth.getGasPrice()
        gasprice = gasprice * 2
        var nonce = await web3.eth.getTransactionCount(WALLET)
        const CT = new web3.eth.Contract(abi, ADDRESS)
        console.log(CT.methods);
        var string = amount.toString()
        const weiCount = web3.utils.toWei(string, 'ether')
        const payout = web3.utils.toHex(weiCount)
        ////this is what we transfer to them
        const txObject = {
            from: WALLET,
            nonce: "0x" + nonce.toString(16),
            to: customer,
            gas: 200000,
            value: "0x0",
            data: CT.methods.transfer(customer, payout).encodeABI(),
            gasPrice: gasprice
        }
        web3.eth.accounts.signTransaction(txObject, PRIVATE_KEY, (err, res) => {
            if (err) {
                console.log('err', err)
            }
            else {
                console.log('res', res)
            }
            const raw = res.rawTransaction
            web3.eth.sendSignedTransaction(raw, (err, txHash) => {
                if (err) {
                    console.log(err)
                }
                else {
                    console.log("txHash:", txHash)
                }
            })
        })
        response.status(200).send({ message: 'Great Job!' })
    }
}
const transactionTracker = (request, response) => {
    fetch(

`https://api.covalenthq.com/v1/${CHAINID}/address/${WALLET}/transfers_v2/?contract-address=${ADDRESS}&key=${COVELANTKEY}`,
            {
                method: "GET",
                mode: "cors",
                headers: {
                    "Content-Type": "application/json",
                    Accept: "application/json",
                }
```

```
            }
    )
    .then((response) => response.json())
    .then((covelantResponse) => {
        covelantResponse.map((transaction) => {
            console.log(transaction)
        })
    });
}



module.exports={purchase}
// module.exports={}
```

For running the code :-

1. Run ‘node index.js’ in terminal
2. Use postman to make a post request of http://localhost:3000/purchase/<api-key>

P.s : Not recommended to use but this is a github Repo set up with all files and folders : https://github.com/Lakshya021/WorkshopSetup
Use it only if you don't have any option. Just put in the env variables and you will be good.