Blockchain & Cryptocurrency

Eng: Islam Ahmed

Lab1

Name: Ahmed Ismail Farouk

Section. 1

Code: 220100279

Code and Output :-

```
"use strict";
const SHOW = "SHOW_PRICE";
const UPDATE = "UPDATE_USD_PRICE";
let fs = require('fs');
let EventEmitter = require('events');
function readJsonFromFile(fileName) {
    let data = fs.readFileSync(fileName, 'utf8');
    return JSON.parse(data);
}
class CurrencyConverter extends EventEmitter {
    static calculateRates(usdPrices) {
       let rates = {};
       let usdMap = {};
       // Calculate USD conversion rates and store them for cross conversion
        for (let i in usdPrices) {
           let o = usdPrices[i];
```

```
let sym = o['asset_id_quote'];
        let usdRate = o['rate'];
       rates[`USD-${sym}`] = usdRate;
        rates[`${sym}-USD`] = 1 / usdRate;
        usdMap[sym] = usdRate;
    }
    // Calculate direct crypto-to-crypto conversion rates
   let symbols = Object.keys(usdMap);
    for (let from of symbols) {
       for (let to of symbols) {
            if (from !== to) {
                let tag = `${from}-${to}`;
                rates[tag] = usdMap[to] / usdMap[from];
                rates[`${to}-${from}`] = usdMap[from] / usdMap[to];
            }
        }
    }
   return rates;
constructor(coin2USD) {
    super();
    this.rates = this.constructor.calculateRates(coin2USD.rates);
    this.on(SHOW, (o) => {
        console.log("SHOW event received.");
       console.log(o);
        const { from, to } = o;
```

}

```
try {
        let rate = this.convert(1, from, to);
        console.log(`1 ${from} is worth ${rate} ${to}`);
    } catch (e) {
        console.error(e.message);
   }
});
this.on(UPDATE, (o) => {
    const { sym, usdPrice } = o;
    if (!sym || !usdPrice || usdPrice <= 0) {</pre>
        console.error("Invalid update parameters.");
        return;
    }
    console.log(`Updating ${sym} price to ${usdPrice} USD.`);
    // Update USD rates
    // complete the equality
    this.rates[`USD-${sym}`] = usdPrice;
    this.rates[`${sym}-USD`] = 1 / usdPrice;
   // Recalculate all crypto-to-crypto rates
    const symbols = Object.keys(this.rates)
        .filter(key => key.startsWith('USD-'))
        .map(key => key.split('-')[1]);
    console.log("symbols", symbols);
    for (let from of symbols) {
        for (let to of symbols) {
            if (from !== to) {
```

```
this.rates[`${from}-${to}`] = this.rates[`USD-${to}`] / this.rates[`USD-
${from}`];
                  }
               }
           }
           console.log("Rates updated successfully.");
       });
   }
   convert(amount, fromUnits, toUnits) {
       let tag = `${fromUnits}-${toUnits}`;
       let rate = this.rates[tag];
       if (rate === undefined) {
           throw new Error(`Rate for ${tag} not found`);
       }
       return rate * amount;
   }
}
// All prices listed are in USD
// write here your JSON File Path (rates.json)
const PATH = './rates.json';
let cnv = new CurrencyConverter(readJsonFromFile(PATH));
console.log(cnv.rates);
console.log("======="");
function test(amt, from, to) {
   console.log(`${amt} ${from} is worth ${cnv.convert(amt, from, to)} ${to}.`);
}
```

```
test(4000, 'ETH', 'BTC');
test(200, 'BTC', 'EOS');
// Test event handling
cnv.emit(SHOW, { from: "EOS", to: "BTC" });
console.log("======="");
cnv.emit(SHOW, { from: "EOS", to: "ETH" });
cnv.emit(SHOW, { from: "ETC", to: "ETH" });
console.log("========"");
cnv.emit(SHOW, { from: "LTC", to: "BTC" });
cnv.emit(UPDATE, { sym: "BTC", usdPrice: 50000 });
cnv.emit(SHOW, { from: "LTC", to: "BTC" });
```

```
🥞 | 📖 🛅 🍃 🍅 🖭 ~ | 1 2 3 4 | 🕟 🖸
                                         ~/Desktop/tasks JS/Lab1.js - Mousepad
                                                                                                                                                  File Actions Edit View Help
File Edit Search View Document Help
■ ■ ■ ■ C × b c × □ i Q & q
                                                                                                                                                      'USD-EIC': 0.03093/365914358225,
'USD-BTC': 0.0002807956773388707,
'BTC-USD': 3561.308384363684,
'USD-EOS': 0.4121926588487459,
'EOS-USD': 2.42605,
'USD-ETC': 0.23186021515115565,
'ETC-USD': 4.3129434661659145,
'USD-ETT': 0.00869119484991826,
'ETL-USD': 115.080697392170482
             ist SHOW = "SHOW_PRICE";
               t UPDATE = "UPDATE_USD_PRICE";
             fs = require('fs');
              EventEmitter = require('events');
              ction readJsonFromFile(fileName) {
                                                                                                                                                       'ETH-USD': 115.05897834170464,
'USD-USDT': 1.001974323186529,
'USDT-USD': 0.9980295670848628,
               let data = fs.readFileSync(fileName, 'utf8');
10  let data = fs.readFileSync(fileName, "utf8")
11  return JSON.parse(data);
12 }
13
14 class CurrencyConverter extends EventEmitter {
15
16  static calculateRates(usdPrices) {
17  let rates = {};
18  let usdMap = {};
19
20  // Calculate USD conversion rates and s
cross conversion
                                                                                                                                                       'LTC-BTC': 0.00919515056165485,
'BTC-LTC': 108.75297726718573,
'LTC-EOS': 13.497976872161685,
                                                                                                                                                       'EOS-LTC': 0.07408517657652877,
'LTC-ETC': 7.592672393598242,
'ETC-LTC': 0.13170593279424903,
                                                                                                                                                       'LTC-ETH': 0.28460853088279475,
'ETH-LTC': 3.513598123352853,
'LTC-USDT': 32.81141949166661,
    oross conversion

for (let i in usdPrices) {

let o = usdPrices[i];

let sym = o['asset_id_quote'];

let usdRate = o['rate'];
21
22
23
24
25
26
                                                                                                                                                        'BTC-EOS': 1467.9451719311985,
'EOS-BTC': 0.0006812243530079674,
                                                                                                                                                        'BTC-ETC': 825.7257282181782,
'ETC-BTC': 0.0012110558819063148,
'BTC-ETH': 30.952025089143703,
                               rates[`USD-${sym}`] = usdRate;
                                                                                                                                                                                                                                                       🗾 🔘 🌬 🗗 🤌 🔤 🔲 🚰 🔞 🚱 🛂 Right Ct
```

First part : -

Function: readJsonFromFile

```
9 function readJsonFromFile(fileName) {
10    let data = fs.readFileSync(fileName, 'utf8');
11    return JSON.parse(data);
12 }
```

What does this function do?

This function reads a JSON file and converts its content into a JavaScript object.

Step-by-step explanation:

1. fs.readFileSync(fileName, 'utf8');

- Reads the file synchronously using the **fs** (File System) module.
- 'utf8' ensures the file is read as a UTF-8 encoded string.
- The file content is stored in the **data** variable as a string.

2. JSON.parse(data);

- Converts the JSON string into a JavaScript object.
- return JSON.parse(data);

■ Returns the parsed object, allowing us to use the data in JavaScript.

Second part :-

Calculating Crypto-to-Crypto Conversion Rates:

```
rates[tag] = usdMap[to] / usdMap[from];
rates[`${to}-${from}`] = usdMap[from] / usdMap[to];
```

What does this code do?

This code calculates conversion rates between different cryptocurrencies based on their USD values.

- Step-by-step explanation:
- 1. usdMap[to] / usdMap[from]
 - usdMap is an object containing the USD price of each cryptocurrency.
 - usdMap[to] represents the USD value of the target currency.
 - usdMap[from] represents the USD value of the source currency.
 - The division calculates the **conversion rate** between the two cryptocurrencies.
- 2. rates[tag] = usdMap[to] / usdMap[from];
 - Saves the conversion rate from **from** → **to** in the **rates** object.
 - Example: If tag = "BTC-ETH", it stores how many ETH are equal to 1 BTC.
- 3. rates[\${to}-\${from}] = usdMap[from] / usdMap[to];
 - Saves the reverse conversion rate from to \rightarrow from.
 - Example: If tag = "ETH-BTC", it stores how many BTC are equal to 1 ETH.
- Third part : -
- Updating Cryptocurrency Prices:

```
// Update USD rates
// complete the equality
this.rates[`USD-${sym}`] = usdPrice;
this.rates[`${sym}-USD`] = 1 / usdPrice;
```

• What does this code do?

This code updates the exchange rate of a cryptocurrency against USD and calculates its reverse rate.

- > Step-by-step explanation:
- 1. this.rates[USD-\${sym}] = usdPrice;
 - Updates the USD price of the cryptocurrency in the rates object.

```
this.rates[${sym}-USD] = 1 / usdPrice;
```

■ Calculates the reverse exchange rate, determining how much of the cryptocurrency is equal to 1 USD.