Name : Abdel-Rhaman Refaey Abdullah Ali

Section : 2

```
"use strict";

const SHOW = "SHOW_PRICE";
const UPDATE = "UPDATE_USD_PRICE";

let fs = require('fs');
let EventEmitter = require('events');

function readJsonFromFile(fileName) {
    // Read from the specified file (using the fs module),
    // pass the contents to JSON.parse, and return the
    // resulting object.
    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}`;
```

```
                    // Set the rates for trading different cryptocurrencies
directly,
                    // calculating the relative prices based off of their USD
prices.
                    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) {
                console.error("Invalid update parameters.");
                return;
            }
            console.log(`Updating ${sym} price to ${usdPrice} USD.`);

            // Update USD rates
            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');
```

```
console.log("========================================================================
");


// Test event handling
cnv.emit(SHOW, { from: "EOS", to: "BTC" });
console.log("========================================================================
");

cnv.emit(SHOW, { from: "EOS", to: "ETH" });
console.log("========================================================================
");

cnv.emit(SHOW, { from: "ETC", to: "ETH" });
console.log("========================================================================
");

cnv.emit(SHOW, { from: "LTC", to: "BTC" });
console.log("========================================================================
");

cnv.emit(UPDATE, { sym: "BTC", usdPrice: 50000 });
console.log("========================================================================
");

cnv.emit(SHOW, { from: "LTC", to: "BTC" });
```

السطران التاليان من الكود يقومان بقراءة ملف JSON وتحويله إلى كائن:JavaScript

```
let data = fs.readFileSync(fileName, 'utf8');
      return JSON.parse(data);
```

شرح الكود:

1. **fs.readFileSync(fileName, 'utf8')**
   - يستخدم **fs.readFileSync** من مكتبة **File System** لقراءة محتوى الملف بشكل **متزامن**.
   - fileName هو مسار الملف الذي نريد قراءته.
   - الخيار 'utf8' يحدد ترميز النص، بحيث يتم إرجاع البيانات كنص عادي بدلاً من **Buffer**.
2. **JSON.parse(data)**
   - يأخذ النص المقروء من الملف (data) و**يحلله** ليصبح كائن **JavaScript** يمكن التعامل معه برمجياً.

```
3.  let symbols = Object.keys(usdMap);
4.          for (let from of symbols) {
5.              for (let to of symbols) {
6.
7.                  if (from !== to) {
8.                      let tag = `${from}-${to}`;
9.                      // ***YOUR CODE HERE***
10.                 // set the rates for trading different cryptocurrencies
    directly,
11.                 // calculating the relative prices based off of their USD
    prices.
12.                 // For example, if `sym` is "BTC", calculate the values
    for
13.                 // "BTC-ETH", "ETH-BTC", "BTC-EOS", "EOS-BTC", etc.
14.                 rates[tag] = usdMap[to] / usdMap[from];
15.                 rates[`${to}-${from}`] = usdMap[from] / usdMap[to];
16.                 }
17.             }
18.         }
```

هذا الجزء من الكود يقوم بحساب أسعار التحويل المباشرة بين العملات المشفرة بناءً على أسعارها مقابل الدولار الأمريكي .(USD) دعنا نشرحه بالتفصيل:

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

المتغير tag يستخدم لإنشاء اسم التحويل بين العملات

حساب معدل التحويل بين العملات:

rates[tag] = usdMap[to] / usdMap[from];

هذا الجزء من الكود يقوم بتحديث أسعار العملات المشفرة مقابل الدولار الأمريكي (USD) عندما يتم إرسال حدث UPDATE_USD_PRICE، وهو مهم جدًا لضمان أن الأسعار تكون محدثة دائمًا. دعونا نشرحه بالتفصيل

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

حيث:

- sym → هو رمز العملة المشفرة (مثل BTC, ETH, LTC).
- usdPrice → هو السعر الجديد لهذه العملة بالدولار الأمريكي.

https://github.com/abdorefaey/blockchain-and-cryptocurrency/tree/49f6c337413fc6d06b6e44c9294701310bcfcfde/lab1