**Currency Converter Program**

Name : Abdullah Serag Ewies Boriek

Section : 2

Code : 220100377

**Code:**

"use strict";  
  
const SHOW = "SHOW\_PRICE";  
const UPDATE = "UPDATE\_USD\_PRICE";  
const fs = require("fs");  
const EventEmitter = require("events");  
  
function readJsonFromFile(fileName) {  
 try {  
 let data = fs.readFileSync(fileName, "utf8");  
 return JSON.parse(data);  
 } catch (err) {  
 console.error("Error reading JSON file:", err);  
 return {};  
 }  
}  
  
class CurrencyConverter extends EventEmitter {  
 static calculateRates(usdPrices) {  
 let rates = {};  
 let usdMap = {};  
  
 // Calculate exchange rates against USD  
 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 exchange 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];  
 }  
 }  
 }  
 return rates;  
 }  
  
 constructor(coin2USD) {  
 super();  
 this.rates = this.constructor.calculateRates(coin2USD.rates);  
  
 this.on(SHOW, ({ from, to }) => {  
 console.log("SHOW event received.");  
 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, ({ sym, usdPrice }) => {  
 if (!sym || !usdPrice || usdPrice <= 0) {  
 console.error("Invalid update parameters.");  
 return;  
 }  
  
 console.log(`Updating ${sym} price to ${usdPrice} USD.`);  
  
 // Update USD exchange rates  
 this.rates[`USD-${sym}`] = usdPrice;  
 this.rates[`${sym}-USD`] = 1 / usdPrice;  
  
 // Recalculate crypto-to-crypto exchange rates  
 const symbols = Object.keys(this.rates)  
 .filter((key) => key.startsWith("USD-"))  
 .map((key) => key.split("-")[1]);  
  
 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;  
 }  
}  
  
// JSON file path containing currency rates  
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}.`);  
}  
  
// Testing conversions  
test(4000, "ETH", "BTC");  
test(200, "BTC", "EOS");  
  
console.log("========================================================");  
  
// Testing 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" });

**Explanation:**

1. Required Modules:

- \*\*fs\*\*: Used for reading data from a JSON file.

- \*\*events\*\*: Used to create and handle custom events.

2. Constants:

- \*\*SHOW = "SHOW\_PRICE"\*\*: Used as an event identifier for displaying currency prices.

- \*\*UPDATE = "UPDATE\_USD\_PRICE"\*\*: Used as an event identifier for updating the price of a specific currency.

3. readJsonFromFile Function:

- Reads a JSON file and parses it into a JavaScript object.

- If an error occurs, it prints an error message and returns an empty object.

4. CurrencyConverter Class:

- Handles exchange rates and currency conversions.

5. calculateRates Method:

- Calculates exchange rates for cryptocurrencies against USD.

- Computes direct conversion rates between different cryptocurrencies.

6. Constructor :

- Reads price data from a JSON file.

- Computes conversion rates.

- Listens for SHOW and UPDATE events.

7. convert Method :

- Converts a given amount from one currency to another.

- If no exchange rate is found, it throws an error.

8. Testing Conversions and Events :

- Converts example amounts of cryptocurrency.

- Handles dynamic price updates and exchange rate calculations.