Compte-rendu tp1

package.json => dev pour commencer

les explications nécessaire pour comprendre votre démarche

findBlocks() :
La fonction findBlocks lit le contenu du fichier blockchain.json et retourne un tableau de blocs. Si le fichier n'existe pas ou n'est pas un JSON valide, elle initialise un tableau vide.

findLastBlock() :
La fonction findLastBlock retourne le dernier bloc de la chaîne.

calculateHash(data) :
La fonction calculateHash prend une chaîne de données, la convertit en hash SHA256, et retourne la représentation hexadécimale du hash.

createBlock(contenu) :
La fonction createBlock génère un nouvel ID, obtient la date actuelle, trouve le dernier bloc, calcule le hash, crée un nouveau bloc, ajoute le bloc au tableau existant, et enregistre le tableau mis à jour dans le fichier blockchain.json.

findBlock(partialBlock) :
La fonction findBlock recherche un bloc par son ID. Si le bloc est trouvé, elle vérifie la fiabilité du bloc en utilisant la fonction verifBlocks et renvoie le bloc ou un message d'erreur si le bloc n'est pas fiable ou n'est pas trouvé.

verifBlocks() :
La fonction verifBlocks vérifie l'intégrité de la chaîne en comparant les hashs des blocs successifs. Elle retourne true si la chaîne est intacte, sinon false.

```
/* Chemin du fichier de stockage des blocks */
const path = '../data/blockchain.json'
const filePath = new URL(path, import.meta.url)
```
⇒ n’oubliez pas new URL()

```
/* effectuez la lecture du fichier puis retournez les valeurs lues au format Json */
const content = await readFile(filePath, 'utf-8');
existingBlocks = JSON.parse(content);
```

d’un appel GET /blockchain définie dans server.js => case 'GET:/blockchain'

```
import {v4} from "uuid";
```
v4(); ⇒ générer ID unique, sécurité, à l'aide de nombres aléatoires

${(date.getMonth() + 1).toString().padStart(2, '0')}
⇒ +1 car getMonth() return valeur de 0 à 11
⇒ padStart pour s'assurer qu'il a toujours 2 chiffres, sinon ajoute un 0 à gauche
⇒ format ISO exemple : 2022-01-28 12:30:45

await fct()
codesuivants
/!\ … async … fct()
⇒ await : codesuivants ne continuent qu' après fct() return resultat.

vos choix, ce qui a marché facilement

Des fonctions comme const previousHash = lastBlock ? lastBlock.hash : null; peuvent être raccourcies.

La valeur const filePath = new URL(path, import.meta.url) peut être utilisée plusieurs fois dans le fichier.

vos difficultés, comment vous les avez surmonté

1. calculateHash()
    Au début, je n'avais pas le même previousBlock.hash et currentBlock.previousHash, et la raison en était que j'avais utilisé calculateHash(block) au lieu de calculateHash(block-hash).

    En plus, j'avais mal compris la question, pensant que le premier bloc devait avoir previousHash=calculateHash(monsecret), alors qu'en réalité, c'est son hash doit être égal à calculateHash(monsecret).

2. j'ai oublié de fermer {}

qu'est qu'on pourrait améliorer

code plus claire