TP1 - Les blocks et la blockchain

L'objectif de ce TP est de vous faire implémenter les concepts de block et de blockchain.

[Temps estimé : ***50 minutes ~ 1 heure***.]

Introduction

1) Chargez dans un navigateur web le fichier ***/TP1/src/index.html***.

2) Chargez dans un éditeur de code le TP1 à partir de son répertoire racine ***/TP1***.

3) Rendez-vous dans le fichier du code métier ***/TP1/src/index.js*** *dans votre éditeur de code***.**

Exercice 1 : les blocks

1) Déclarez une classe ***Block***.

2) Déclarez un ***constructeur*** dans votre classe ***Block***. Il doit initialiser ces attributs :

- ***previousBlockHash*** : le hash en hexadécimal du block précédent (cet attribut peut valoir null pour un block qui n'a pas de block précédent, tel que par exemple le premier block d'une blockchain) ;

- ***transactionsHashs*** : le tableau des hashs en hexadécimal des transactions contenues et validées dans le block (le hash d'une transaction est issu du chiffrement de la transaction avec la clé privée de son expéditeur, c'est le principe de la signature électronique, les transactions sont signées par leur expéditeur) ;

- ***miner*** : la signature en hexadécimal du mineur qui a trouvé la preuve de travail ;

- ***proofOfWork*** : la preuve de travail en hexadécimal, qui a permis de valider le block dans la blockchain.

- ***creationDate*** : la date de création du block en hexadécimal (oui en hexadécimal, ça n'est pas une faute).

Le constructeur doit prendre en paramètre tous les attributs, sauf la date de création bien sûr, qui elle est déterminée directement dans le constructeur.

Pour stocker une date en hexadécimal, il y a deux fonctions auxquelles il faut penser : ***Date.now()*** et ***decimalToHex(number).***

Humm, comment faire ? Un indice : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Date/now.

Faites attention à l'orthographe pour les attributs et pour leur **getter** par la suite, cela aura une incidence.

3) Déclarez des **getters** pour chacun des attributs de la classe ***Block.***

4) Déclarez une fonction ***computeTransactionsMerkleRootHash*()**, qui retourne la concaténation des hashs de toutes les transactions du block. Cette fonction, dis de Merkle, a pour but de synthétiser en un seul hash, tous les hashs des transactions contenues dans un block. La bonne manière de l'écrire n'est pas de faire une concaténation, mais pour le moment, on va se suffire de ça.

Une certaine fonction ***Array.reduce(function, object)*** peut s'avérer utile ici, au lieu d'un for : https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce.

6) Déclarez un nouvel attribut ***transactionsMerkleRootHash*** dans le constructeur, qui doit prendre la valeur issue de la fonction ***computeTransactionsMerkleRootHash()***, et déclarez aussi un **getter** pour ce nouvel attribut.

6) Déclarez une fonction ***computeHash()*** qui calcule et retourne le hash du block. Voici comment on calcule le hash d'un block :

(1) On concatène les attributs d'un block, et pour notre implémentation on va faire dans cet ordre : ***previousBlockHash*** + ***transactionsMerkleRootHash*** + ***miner*** + ***proofOfWork*** + ***creationDate.***

(2) On hache la chaine de caractère issue de la concaténation avec une fonction de hachage (vous devez utilisé ici dans notre cas ***sha256(content)***).

7) Déclarez un nouvel attribut ***hash*** dans le constructeur, qui doit prendre la valeur issue de la fonction ***computeHash()***, et déclarez aussi un **getter** pour ce nouvel attribut.

8) Testez maintenant votre classe Block, en instanciant un objet Block dans la fonction ***handler*** (avec null en hash du block précédent, et des valeurs aléatoires pour les autres paramètres, ne

cherchez pas à donner des chaines contenant des nombres hexadécimaux), et stockez cet objet dans une variable.

9) A l'aide de la fonction **showBlock(Block)**, affichez votre variable, que voyez-vous ?

Si vous avez une erreur dans la console, faites signe au plus tôt à un des deux responsables de l'atelier, l'objectif n'est pas de bloquer sur une erreur.

Exercice 2 : la blockchain

1) Déclarez une classe ***Blockchain.***

2) Déclarez un constructeur dans votre classe ***Blockchain***. Il doit initialiser :
   - ***blocks*** : le tableau des blocks ;
   - ***currentBlock*** **:** le block courant.

   Le constructeur doit prendre en paramètre le block graine / d'origine de la blockchain, et initialiser avec celui-ci les deux attributs de la classe.

3) Déclarez des **getters** pour les deux attributs.

4) Déclarez une fonction ***addBlock(transactionsHashs, miner, proofOfWork)***. Elle doit permettre d'ajouter un nouveau block à la blockchain à partir de ses paramètres, et le nouveau block ajouté doit devenir le block courant.

5) Testez votre blockchain, dans la fonction ***handler***, en affectant à une variable un nouvel objet ***Blockchain*** (il doit prendre un objet ***Block*** graine qui peut être vide : hash du block précédent null, tableau de hash des transactions vide, mineur null, preuve de travail null).

6) A l'aide de la fonction ***showBlock(Block)***, affichez le bloc courant (graine du coup) de votre blockchain.

7) Ajoutez un block à votre blockchain, et affichez de nouveau le bloc courant de votre blockchain, qu'observez-vous d'intéressant ?