BlockChain

BlockChain

Proof of Work:

Proof of Work and the 51 Attack:

Proof of Work System

- A system that requires miners to do computational work to add blocks.
- Any peer can replace the blockchain.
- The proof-of-work makes it expensive to generate corrupt chains.
- Manageable to submit one block, unproductive to generate an entire chain.

Proof of Work System

- Hashcash was a proof-of-work system to prevent email spamming.

Difficulty = 6

Hash = 000000haxi2910jasdflk

- Generate hashes until a one with the matching leading 0's is found.
- A "nonce" value adjusts in order to generate new hashes.
- This computational work is "mining."

Proof of Work System

- The difficulty sets a rate of mining.
- Bitcoin sets the rate to a new block around every 10 minutes.

BlockChain

51% Attack

- A dishonest miner has more than at least 51% of the network's power.
- A 51% attack for bitcoin would be more than $6 billion (start of 2018).

Proof of Work and the Nonce:

```
const DIFFICULTY = 4;
```

Blockchain > JS block.js > Block

```
        this.lastHash = lastHash;
        this.hash = hash;
        this.data = data;
        this.nonce = nonce;
    }
```

Blockchain > JS block.js > Block

```
        Hash: ${this.hash.substr
        Nonce: ${this.nonce};
        Data: ${this.data}`;
```

Blockchain > JS block.js > Block

```
    }

    static genesis(){
        return new this('Genesis time', '-----', 'f1r57-h45h',[],0);
```

BlockChain

Blockchain > JS block.js > Block

```js
        let nonce = 0;
        do{
            nonce++;
            timestamp = Date.now();
            hash = Block.hash(timestamp, lastHash, data, nonce);


        } while(hash.substring(0, DIFFICULTY) !== '0'.repeat(DIFFICULTY)



        return new this(timestamp, lastHash, hash, data, nonce);

    }
```

```js
        return new this(timestamp, lastHash, hash, data, nonce);

    }

    static hash(timestamp, lastHash, data, nonce){
        return SHA256(`${timestamp}${lastHash}${data} ${nonce}`).toString();

    }

    static blockHash(block){
        const {timestamp, lastHash, data, nonce} = block;
        return Block.hash(timestamp, lastHash, data, nonce);
    }
}
```

BlockChain

Test the Nonce Functionality

```
JS config.js > ...
const DIFFICULTY = 4;
                   (property) DIFFICULTY: number

module.exports = {DIFFICULTY};
```

```
const { DIFFICULTY} = require('../config');
```

```
Blockchain > JS block.test.js > ...


    if('generates a hash that matches the diffiulty', ()=>{
        expect(block.hash.substring(0, DIFFICULTY)).toEqual('0'.repeat(DIFFICULTY))

    });
});
```

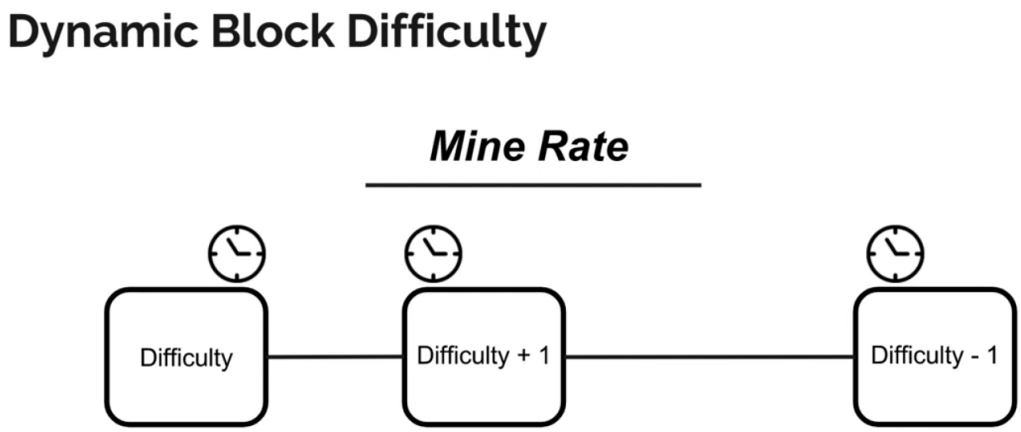
```
Blockchain > JS block.test.js > describe('Block') callback
    });


    if('generates a hash that matches the diffiulty', ()=>{
        expect(block.hash.substring(0, DIFFICULTY)).toEqual('0'.repeat(DIFFICULTY));
        console.log(block.toString());

    });
});
```

BlockChain

Dynamic Block Difficulty:




```
PASS  project/Build the Blockchain/block.test.js
PASS  project/Develop the Blockchain Application/Blockchain/block.test.js
PASS  project/Create the Blockchain Network/Blockchain/block.test.js
PASS  project/Build the Blockchain - the Chain/block.test.js

RUNS  Blockchain/index.test.js
RUNS  Blockchain/block.test.js

Test Suites: 7 passed, 7 of 9 total
Tests:       21 passed, 21 total
Snapshots:   0 total
Time:        209 s
```

JS config.js > ...

```js
const DIFFICULTY = 4;

const MINE_RATE = 3000;

module.exports = { DIFFICULTY, MINE_RATE};
```

```js
const { DIFFICULTY, MINE_RATE} = require('../config');
```

BlockChain

chain > JS block.js > SHA256

class Block

```
class Block {
    constructor(timestamp, lastHash, hash, data, nonce, difficulty){
        this.timestamp = timestamp;
        this.lastHash = lastHash;
        this.hash = hash;
        this.data = data;
        this.nonce = nonce;
        this.difficulty = difficulty || DIFFICULTY;
    }
```

kage.json | JS dev-test.js | JS block.js × | JS config.js | JS blo

Blockchain > JS block.js > SHA256

```
    toString(){
        return `Block -
        Timestamp: ${this.timestamp}
        Last Hash: ${this.lastHash.substring(0,10)}
        Hash: ${this.hash.substring(0,10)}
        Nonce: ${this.nonce};
        Difficulty: ${this.difficulty}
        Data: ${this.data}`;
    }
```

Blockchain > JS block.js > SHA256

(method) Block.genesis(): Block

```
    static genesis(){
        return new this('Genesis time', '-----', 'f1r57-h45h',[],0, DIFFICULTY);
    }
```

BlockChain

Blockchain > JS block.js > SHA256

```js
    static mineBlock(lastBlock, data){
        let hash, timestamp;
         const lastHash = lastBlock.hash;
         let {difficulty} =lastBlock;
         let nonce = 0;
         do{
             nonce++;
            timestamp = Date.now();
            difficulty = Block.adjustDifficulty(lastBlock, timestamp);
             hash = Block.hash(timestamp, lastHash, data, nonce, difficulty);


            } while(hash.substring(0, difficulty) !== '0'.repeat(difficulty));;



         return new this(timestamp, lastHash, hash, data, nonce, difficulty);

    }
```

Blockchain > JS block.js > Block > mineBlock

```js

    static adjustDifficulty(lastBlock, currentTime){
        let {difficulty} = lastBlock;
        difficulty = lastBlock.timestamp +MINE_RATE > currentTime ? difficulty +1 : difficulty -1;
        return difficulty;
    }
}
```

BlockChain

Test Difficulty Adjustment:

```
PASS  project/Build the Blockchain/block.test.js
PASS  project/Build the Blockchain - the Chain/block.test.js
PASS  Blockchain/block.test.js
  ● Console

    console.log
      Block -
              Timestamp: 1661942040073
              Last Hash: f1r57-h45h
              Hash: 0007ade38d
              Nonce: 3153;
              Difficulty: 3
              Data: bar

      at Object.log (Blockchain/block.test.js:31:17)

PASS  project/Create the Blockchain Network/Blockchain/block.test.js
PASS  project/Develop the Blockchain Application/Blockchain/block.test.js

Test Suites: 9 passed, 9 total
Tests:       28 passed, 28 total
Snapshots:   0 total
Time:        1.836 s, estimated 2 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Blockchain > JS block.test.js > describe('Block') callback

```
27
28
29    it('generates a hash that matches the difficulty', ()=>{
30        expect(block.hash.substring(0, block.difficulty)).toEqual('0'.repeat(block.difficulty));
31        console.log(block.toString());
32
33    });
34
35    it('lowers the difficulty for slowly mined blocks', () =>{
36
37        expect(Block.adjustDifficulty(block, block.timestamp + 360000)).toEqual(block.difficulty -1);
38
39    });
40
```

BlockChain

index.js > ...

```js
const Blockchain = require('./blockchain');

const bc = new Blockchain();

for (let i = 0; i <10; i++){
    console.log(bc.addBlock(`foo ${i}`).toString());
}

```

```
[Function: toString]
[Function: toString]
[Function: toString]
[Function: toString]
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Block -
        Timestamp: 1661942597592
        Last Hash: f1r57-h45h
        Hash: 0005d317b5
        Nonce: 5832;
        Difficulty: 3
        Data: foo 0
Block -
        Timestamp: 1661942597996
        Last Hash: 0005d317b5
        Hash: 000091cd16
        Nonce: 61810;
        Difficulty: 4
        Data: foo 1
Block -
        Timestamp: 1661942599618
        Last Hash: 000091cd16
        Hash: 000009c032
        Nonce: 212555;
        Difficulty: 5
        Data: foo 2
Block -
        Timestamp: 1661942602910
        Last Hash: 000009c032
        Hash: 0000fc360e
        Nonce: 518157;
        Difficulty: 4
        Data: foo 3
Block -
        Timestamp: 1661942605358
        Last Hash: 0000fc360e
```

BlockChain