Writing a tiny blockchain in JavaScript

Almost everyone has heard about cryptocurrencies like Bitcoin and Ethereum, but few people actually know how the technology behind these work. In this blog post I'll create a simple blockchain in JavaScript to demonstrate how they work internally. I'll call it SavjeeCoin!

This blog post is part of a whole series:

- **Part 1: Implementing a basic blockchain**

- Part 2: Implementing proof-of-work.
- Part 3: Transactions & mining rewards

Blockchains

A blockchain is a public database that consists out of blocks that anyone can read. Nothing special, but they have an interesting property: they are immutable. Once a block has been added to the chain, it cannot be changed anymore without invalidating the rest of the chain.

That is the reason why cryptocurrencies are based on blockchains. You don't want people changing their transactions after they've made them!

Building a Block

A blockchain consists out of many blocks that are linked together (that makes a lot of sense, right?). The chaining of blocks happens in a way that allows us to detect when someone has manipulated any of the previous blocks.

So how do we ensure the integrity? Well each block contains a hash that is computed based on its contents. It also contains the hash of the previous block.

This is what a Block class could look like in JavaScript:

```
const SHA256 = require("crypto-js/sha256");
class Block {
    constructor(index, timestamp, data, previousHash = '') {
        this.index = index;
        this.previousHash = previousHash;
        this.timestamp = timestamp;
        this.data = data;
        this.hash = this.calculateHash();
    }
```

```
    calculateHash() {
        return SHA256(this.index + this.previousHash + this.timestamp + JSON.string
    }
}
```

I start by requiring the [crypto-js library](#) because the sha256 hash function is not available in JavaScript. After that I define a constructor that initializes the properties of my block. Each block is given an `index` that tells us at what position the block sits on the chain. We also include a timestamp, some data to store in our block and finally the hash of the previous block.

Building the chain

Now we can start chaining blocks together in a Blockchain class! Here's what that could look like in JavaScript:

```
class Blockchain{
    constructor() {
        this.chain = [this.createGenesisBlock()];
    }

    createGenesisBlock() {
        return new Block(0, "01/01/2017", "Genesis block", "0");
    }

    getLatestBlock() {
        return this.chain[this.chain.length - 1];
    }

    addBlock(newBlock) {
        newBlock.previousHash = this.getLatestBlock().hash;
        newBlock.hash = newBlock.calculateHash();
        this.chain.push(newBlock);
    }

    isChainValid() {
        for (let i = 1; i < this.chain.length; i++){
            const currentBlock = this.chain[i];
            const previousBlock = this.chain[i - 1];
```

```
            if (currentBlock.hash !== currentBlock.calculateHash()) {
                return false;
            }

            if (currentBlock.previousHash !== previousBlock.hash) {
                return false;
            }
        }
        return true;
    }
}
```

In the constructor I initialize the chain by creating an array that contains the genesis block. The first block is special because it cannot point to a previous block. I've also added two methods:

- `getLatestBlock()` returns the latest block on our blockchain.
- `addBlock()` is responsible for adding a new block to our chain. To do that we add the hash of the previous block to our new block. That way we preserve the integrity of the chain. Because we changed the contents of our new block, we need to recalculate it's hash. When that's done, I push the block onto the chain (array).

Finally I've created a method `isChainValid()` to make sure that nobody has messed with the blockchain. It loops over all the blocks and checks if the hash of each block is correct. It also checks if each block points to the correct previous block by comparing the `previousHash` value. If everything checks out it returns true and if something is wrong it returns false.

Using the blockchain

With our Blockchain class finished, we can actually start using it!

```
let savjeeCoin = new Blockchain();
savjeeCoin.addBlock(new Block(1, "20/07/2017", { amount: 4 }));
savjeeCoin.addBlock(new Block(2, "20/07/2017", { amount: 8 }));
```

Here I'm just creating a new instance of a Blockchain and naming it SavjeeCoin. Afterwards I add some dummy blocks onto the chain. Blocks can contain any data that you want, but in this case I opted for an object with an `amount` property.

Trying to manipulate it

In the introduction I said that blockchains are immutable. Blocks cannot be changed once they are added. Let's test that!

```
// Check if chain is valid (will return true)
console.log('Blockchain valid? ' + savjeeCoin.isChainValid());

// Let's now manipulate the data
savjeeCoin.chain[1].data = { amount: 100 };

// Check our chain again (will now return false)
console.log("Blockchain valid? " + savjeeCoin.isChainValid());
```

I'll start by verifying the integrity of our chain by running `isChainValid()`. I haven't manipulated any blocks so it returns `true`.

After that I take the first block on the chain (index = 1) and I manipulate the amount. I then recheck the integrity of the chain and this time it detects that something is wrong. Our chain is no longer valid.

Conclusion

This implementation is far from complete. It doesn't implement proof-of-work or a P2P network to communicate with other miners.

It does however demonstrate how a blockchain works. Many people think that it's very complex, but this post demonstrates that the basic concepts of a

blockchain are easy to understand and to implement.

Next up

This blockchain is not complete and not fully secure. Keep reading:

- **Part 1: Implementing a basic blockchain**
- Part 2: Implementing proof-of-work.
- Part 3: Transactions & mining rewards

Posted on 19 Jul 2017

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Oliver Mensah • 7 months ago

Awesome work bro

4 ^ v • Reply • Share ›

Arun Kumar • 4 months ago

I am a dotnet developer, and average in english. watched many video to understand blockchain but failed :-(. but the way you explained with code is very very useful for developers like me to understand it. Thanks Savjee.

1 ^ v • Reply • Share ›

Ridwaan Maharaj • 4 months ago

This is spot on and easy to understand, thanks man

1 ^ v • Reply • Share ›

ryderx3 • 4 months ago

Thanks for the effort. To the point.

1 ^ v • Reply • Share ›

Angel Rodriguez • 7 months ago

Thank you soo much for this! Beautifully done and easy to comprehend! THANKS!!!

1 ^ v • Reply • Share ›

KWC Coin • 18 days ago

Great work. Finally understand it instead of just doing it!!!

^ v • Reply • Share ›

Israel Tomilayo • 21 days ago

changing the first block's data tend to make the chain valid still.
i.e savjeeCoin,chain[0].data = { amount: 100 }. when I check the chain again, it was still valid.

Reply • Share ›

Gaitchs Gangmei • a month ago

please make it POS not POW

Reply • Share ›