# Building My Own Blockchain

Before starting off as a beginner blockchain developer, one of my main goals was to create my very own cryptocurrency. Creating my very own bitcoin or Ethereum always seemed like a wonderful idea that would be more than challenging and worthy of my efforts.

I built my very own Ethereum-based cryptocurrency a few weeks ago, and I must say, it wasn’t so hard once such a handy library like open Zeppelin came into the picture. I even built my very own centralized exchange and everything, and got bored after creating the second or third crypto.

Now, it seems, a new opportunity has presented itself for me to completely stress myself and pull my own hair out. I have pulled several study materials together, and am going to building my very own blockchain over the next few months (so help me God).

I thought I’d document the process just for laughs, so that I could maybe stumble on it one day in the future when I’m at Vitalik’s level, or big enough for Hoskinson to know my name (because I built some revolutionary piece of tech that helps millions). Who knows what might happen?

So… here goes nothing.

## Day 1 (10:44 pm, November 8th 2022)

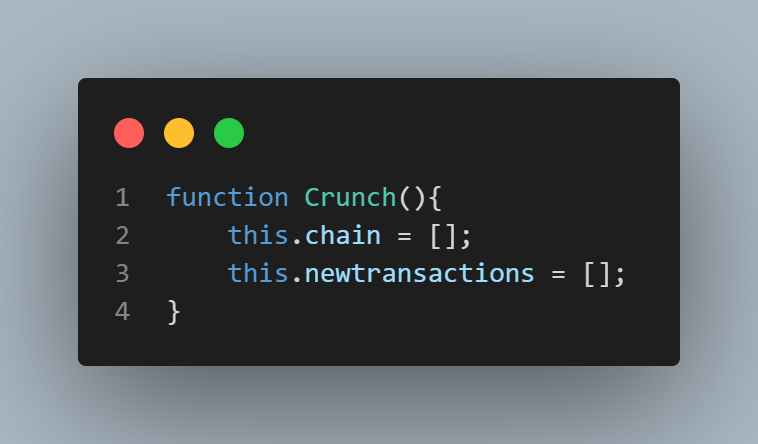
* First days are for introductions.
* The blockchain I’m going to build is going to be a Proof of Work (PoW) blockchain like bitcoin and Ethereum (before the merge).
* It will have the ability to accommodate new blocks through a mining process, and will be able to hold new and immutable transactions that are capable of being validated.
* Its users will also be able to retrieve data from it like addresses and transaction data. It will feature a decentralized blockchain network that hosts the blockchain. The first step in creating one of these blockchains would be to create an API or server to interact with the blockchain via the internet.
* The API will also be upgradable to accommodate several nodes for true decentralization.
* The blockchain will feature a consensus algorithm of sorts, that will ensure that all nodes contain the correct and perfectly synchronized information. There will even be a block explorer webpage to explore the data on the chain via a GUI.
* I’ll call by blockchain “Crunch”.

## Day 2 (10:42 pm, November 9th 2022)

* First of all, I create a new folder called “crunch”, and then create two new JavaScript (or typescript) files in it.
* The first of these files will be called “crunch.js”, and will be the main blockchain file
* The second of these files will be the test file called test.js and will be the file we use to write tests against the blockchain

## Day 2 (9:28 pm, November 14th 2022)

* In the crunch.js file, I created my constructor function called “Crunch”, and added the blockchain array and the “new transaction” array.



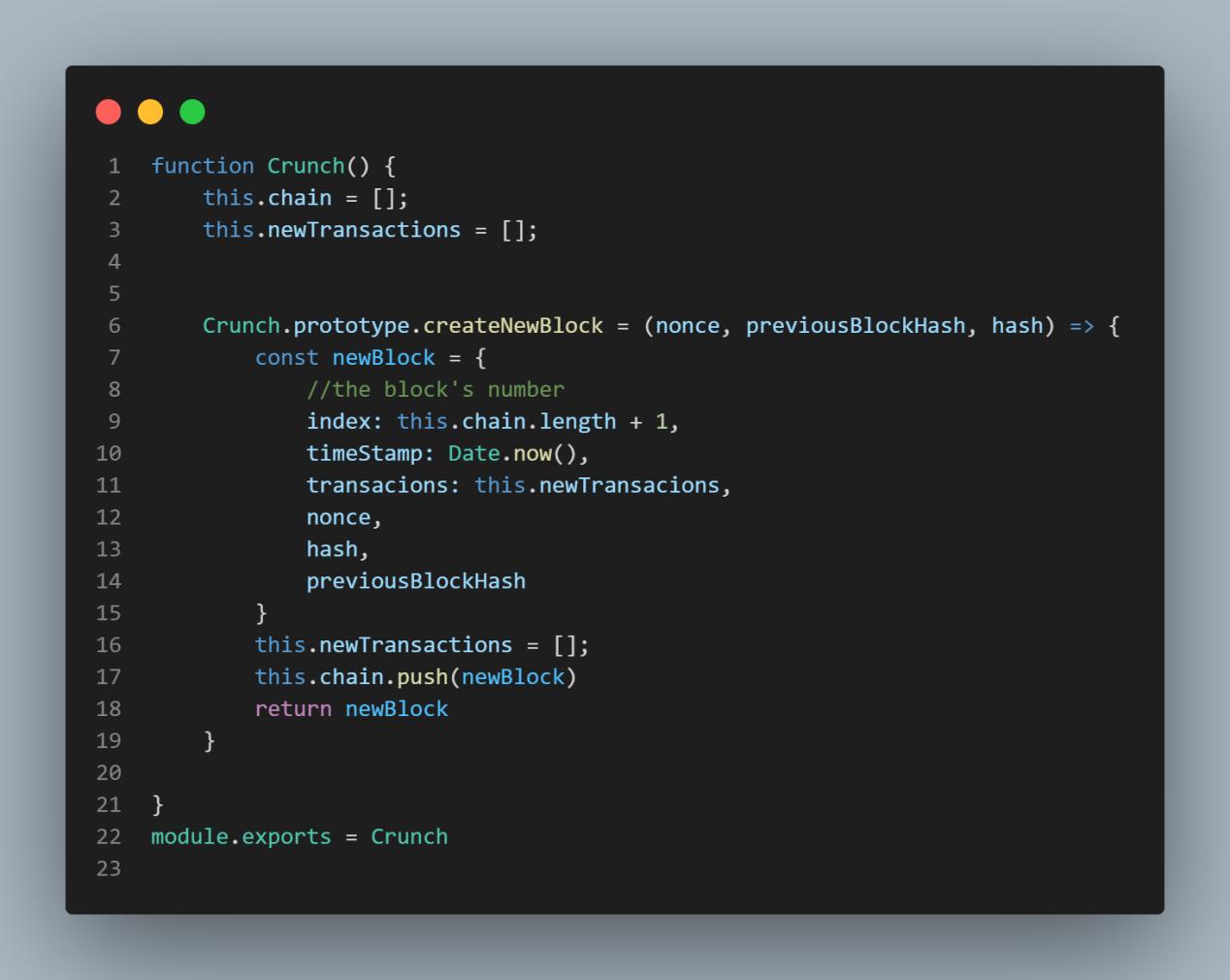
* Next, I create a new protoype method using Crunch.prototype.createNewBlock
* I initialize this to an anonymous function that takes three parameters

1. Nonce
2. The previous block’s hash
3. The hash of the new block

* This protoype method creates a new constant (an object) that has six properties:

1. The index of the block (or its number in the chain)
2. The timestamp of the block
3. The new transaction we wish to add to the block we’re creating.
4. The nonce of the block
5. The previous block’s hash
6. The hash of the new block

* The prototype reinitializes the newTransactions array to an empty array.
* It also pushes the new block to the new chain.



## Day 3 (8:25 pm, November 14th 2022)

* Today, I write tests against my blockchain.
* In my test.js file, I require the crunch file in which I created by blockchain and initialize it to a constant called “Crunch”.
* Next, I initialize a new instance of this blockchain constructor with the “new” keyword and set it to a variable called “crunch” (notice the small letter difference).
* After this, I create a new function called “mainTest” and write my tests inside it.
* Inside this function, I call the “createNewBlock” prototype function on the crunch blockchain instance and pass it arbitrary values as the nonce, previousBlockHash and the hash of the new block.

1. 72637 as the nonce
2. 82YRIUY274YR2 as an arbitrary hash of the previous block
3. 726287463248H as the hash of this new block being created.
4. Then I console.log() the response of the prototype object and see what it spits out.

