

# Docbridge Project Journal

Blockchain Research - Revision 1

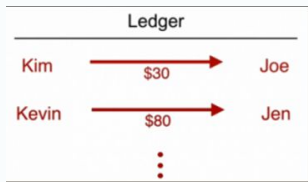


## Blockchain Research Notes

Blockchain is a ledger in which the transactions can never be changed and the blockchain itself is distributed across the network and run by thousands of independent people, groups, or nodes

Creating constructor for objects in JavaScript was very similar to C classes using 'this' keyword!

The prototype object is simply an object that multiple other objects can refer to in order to get any information or functionality that they need. Prototype function in constructor can allow hidden function to be accessed by all objects



```
function User(firstName, lastName, age, gender) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.gender = gender;
}

var user1 = new User('AJ', 'Barea', 28, 'male');

user1
// User {firstName: 'AJ', lastName: 'Barea', age: 28, gender: 'male'}

var user200 = new User('Jill', 'Robinson', 25, 'female');

user200
// User {firstName: 'Jill', lastName: 'Robinson', age: 25, gender: 'female'}
```

```
User.prototype.getEmailAddress = function () {
  return this.firstName + this.lastName + this.emailDomain;
}
```

## Blockchain Research Notes

```
function Blockchain () {  
  
  this.chain = [];  
  
  this.pendingTransactions = [];  
  
  this.createNewBlock(100, '0', '0');  
  
}
```

### Blockchain Prototypes:

- createNewBlock
- createNewTransaction
- getLastBlock
- hashBlock
- proofOfWork

```
// blockchain constructor function  
function Blockchain () {  
  this.chain = [];  
  this.pendingTransactions = [];  
}  
  
// this prototype object is used to create a new BLOCK for the chain  
Blockchain.prototype.createNewBlock = function (nonce, previousBlockHash, hash) {  
  const newBlock = {  
    index: this.chain.length + 1,           // block number  
    timestamp: Date.now(),                  // time block was created  
    transactions: this.pendingTransactions, // new transactions waiting to be placed into a block  
    nonce: nonce,                          // function param used for proof of work; NONCE = "N-umber only used-ONCE"  
    hash: hash,                             // transaction data for the current block compressed into a string  
    previousBlockHash: previousBlockHash,   // transaction data from the previous block  
  };  
  
  this.pendingTransactions = [];           // emptying the newTransactions array for the next newBlock  
  this.chain.push(newBlock);               // add new block to Blockchain  
  
  return newBlock;  
}
```

```
// this prototype object is used to create a new TRANSACTION  
//FIXME: tweak transaction for PhiQuest - improve 'document' relevance  
Blockchain.prototype.createNewTransaction = function (amount, sender, recipient) {  
  const newTransaction = {  
    amount: amount,  
    sender: sender,  
    recipient: recipient,  
  };  
  
  this.pendingTransactions.push(newTransaction); // add new transaction to array  
  return(this.getLastBlock()['index'] + 1);      //return the block that the newTransaction will be added to  
}
```

## Blockchain Research Notes

EVERY prototype function gets tested before building onto it!!!

Blockchain function constructor? Test.

createNewBlock? Test. 1

createNewTransaction? Test. 2

getLastBlock? Test.

hashBlock? Test. 3

proofOfWork? Test.

genesisBlock? Test. 4

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
35481fa18a181135a359c1f2b33da3b4e81f0de06b5100801add6a1493e8f26e
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
35481fa18a181135a359c1f2b33da3b4e81f0de06b5100801add6a1493e8f26e
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
35481fa18a181135a359c1f2b33da3b4e81f0de06b5100801add6a1493e8f26e
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
879792388b549882423e14595a1ad1d9c4e41983e6f53e355d0d721513323968
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
4921669d69e1a1714972a858a95fa2be9511ec6a6c683199432bb34d0f056beb
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
722394475aa0f146169af78c4a5a05aeea789da579288890da1db0ca121eb464
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
Blockchain {
  chain: [
    {
      index: 1,
      timestamp: 1643881606388,
      transactions: [],
      nonce: 2389,
      hash: '78s97d4x6dsf',
      previousBlockHash: 'OIUOEREDHKHKD'
    },
    {
      index: 2,
      timestamp: 1643881606388,
      transactions: [Array],
      nonce: 6704,
      hash: 'js83jtgf6fwj',
      previousBlockHash: 'KDGGSWHGFKWR'
    },
    {
      index: 3,
      timestamp: 1643881606388,
      transactions: [],
      nonce: 9764,
      hash: 'fxyqwi497547',
      previousBlockHash: 'UXFAESWDTHHDG'
    }
  ],
  pendingTransactions: [
    {
      amount: 100,
      sender: 'ALEXHT845S35TKCJ2',
      recipient: 'JENNSBG5DF6HT8NG9'
    },
    {
      amount: 93,
      sender: 'KDGGSWHGFKWR',
      recipient: 'JENNSBG5DF6HT8NG9'
    },
    {
      amount: 2,
      sender: 'UXFAESWDTHHDG',
      recipient: 'JENNSBG5DF6HT8NG9'
    }
  ]
}
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
Blockchain {
  chain: [
    {
      index: 1,
      timestamp: 1643882571407,
      transactions: [],
      nonce: 2389,
      hash: '78s97d4x6dsf',
      previousBlockHash: 'OIUOEREDHKHKD'
    }
  ],
  pendingTransactions: []
}
```

```
C:\Users\ajbar\programs\blockchain>
C:\Users\ajbar\programs\blockchain>node dev/test.js
Blockchain { chain: [], pendingTransactions: [] }
```



## Blockchain Research Notes

Every time a new block is created, we first make sure that it is a legitimate block by mining it through

**Proof of Work.** Here we run our hashBlock method many times until we end up generating a hash that has four zeros at the beginning.

Since SHA is random we do this repeated trial and error while constantly incrementing the nonce value, allowing us to change the hash string without changing the block data

Running hashBlock to get the right fit uses up a lot of energy and computing power! Maybe 28 loops, maybe 35,669 loops!

```
Microsoft Windows [Version 10.0.22000.469]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\ajbar>cd programs\blockchain
```

```
C:\Users\ajbar\programs\blockchain>
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
```

```
35668
```

```
C:\Users\ajbar\programs\blockchain>
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js  
35668  
baaed17298d06e8d8a1beb48faa81c20f861ec09ec17781e27f6406b2e738a96  
ec481ae80cf82cfa08203c1e85a274cc4e407d8e869ca009f8c7d5892913baac  
405893892d818fd7fa35bbd28667e189c1c93ac82c1d2cf04e91c239b4588b42
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js  
13280  
fdeea5a274fce5c8d0d4b504e613ecb38416eb0001a07b1384ee51ad330c64fb  
f4fefc3a24e385b822e0839e60b50ce203098f8d230465b2222614d4eac27626  
1bf01b559e088e0123ba1411b168857a710468bb52c7f082a59e59ce63771f61
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js  
11290  
4e53c47f32b573a33cf847c306821cc12270a76421e671872cd102eb0fed027e  
41bd9c40a10d61a8cd07d9b3d7ac1a5d2743793ed2b62814880fcc1ce88d079d  
50db70dd0ccb9e0d72e8fafaf827f63dcf389eaa351d4396346578d0e267231
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js  
28  
2f19f3cf4d5f776869f6012438e6efaf1f48100e84a209797d1bde3d99e24800b  
d0485fdc7e581fdb35b60d869c5718c36a7f73a992bed54a93c75f4a00e54d62  
1999bb45d86e4cae98af948c629e70fce7f027ed228e7e5b803f4030dcf8420b
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js  
bc3b285d2372b3600934db062ba75c0155b15cdebb1135af47f31dd9341f6238  
a5574d180b030f6d6928640e79250f8134c03ade3697d9ba1e559c5efa581b59  
6d15527d8a72ae085a6be5d6ee986d471be53e222edaf97eea7097dc709e5595
```

```
f19086973c6ad372ddc9c2ba5bb06f1ff1d23383ae28e490720373d2a0563b11  
f57a86020870292b1a51437987f35d830bea1c675305df4c16930a048d6a487e  
8d7d5046cb6ba51a18ea12175c9fe7152cab849bc4e51176f2dfce3fd35fbab3
```

```
7db55a9ee82ca972dcfc0d6527465724646dd970006558721b70ca4e66d0e3f2  
0a47cb30485c2f57c1250d0c6d94cff63dfc24ec8efc63564d22a33653f2a8f0  
1f016924fd247eabe347ad5463727e9957c1fd2d87aa46d8e11033353103952b
```

```
149823f601f1df267246a65e7ee10cc72243762f5a8b428e332d049061dce16c  
0ff79bc94033d2ca3f502e716ac79e46bc43615bfa23db4dbc0c5ddaa0f5dd04  
cd848eb339b536854d25dc899671db1fc44bb35dc288d3fe803b1c0b7cd72162
```

```
96d8abbf292331dfa04a124fd3c149b96442c58f87a6e09fc8e32a921306c023  
7edf3f93a17d0486aba32fae7a0d4e8247823078dc3372d0a7e25aad826659ff  
9cc619ac82da4e054d2849b311b4b791e81c2d504686112f0ece522e464aa9c3
```

```
000007a2ff77c3e2a68f39c769de76f509d2ce7c439281880e053f13416186b3  
28
```

```
C:\Users\ajbar\programs\blockchain>
```

## Blockchain Research Notes

When a new block is created, that's when all pending transactions become recorded on our blockchain, and they can never be changed.

SHA 256 - npm library JavaScript component to compute the SHA256 of strings or bytes.

<https://www.npmjs.com/package/sha256>

Blockchain - npm library node.js module to access the blockchain websocket api

<https://www.npmjs.com/package/blockchain>

Enter your text below:

CodingJavaScript

Generate Clear All ☐ Treat each line as a separate string

SHA256 Hash of your string:

7269FF2EF7399C036CD704AC9768AEFE1213DE8676A80F4E83AFCF2A35C9169E

The SHA256 hashing function takes in any string of text, hashes that text, and returns a fixed-length hashed string. Any one-character change has a drastic effect on the hash, but given the same string the hash will always be the same, making it very secure

The **WebSocket API** is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

## Blockchain Research Notes

Hash Block ⇒

Proof Of Work ⇒

```
// this prototype object is used to hash all of a block's data
Blockchain.prototype.hashBlock = function(previousBlockHash, currentBlockData, nonce) {
  //all block data is concatenated into one long string
  const dataAsString = previousBlockHash + nonce.toString() + JSON.stringify(currentBlockData);
  const hash = sha256(dataAsString); // convert data string to hash
  return hash;
}

// this prototype object repeatedly runs hashBlock until "safe" string is randomly generated
Blockchain.prototype.proofOfWork = function(previousBlockHash, currentBlockData) {
  let nonce = 0;
  let hash = this.hashBlock(previousBlockHash, currentBlockData, nonce);
  // run hashBlock with all data intact and NEW nonce value
  while(hash.substring(0, 4) !== '0000') {
    nonce++;
    hash = this.hashBlock(previousBlockHash, currentBlockData, nonce);
  }
  return nonce;
}
```

## Installs:

1. express - minimalist web framework for node
2. nodemon - automatically restart node app when file changes in the directory are detected
3. postman - make calls to any of our post endpoints
4. body-parser - if a request comes in with JSON data or with form data, we simply want to parse that data so that we can access it in any of the endpoints

## endpoints:

1. /blockchain - fetch entire blockchain to look at the data  
`[app.get('/blockchain', function (req, res))]`
2. /transaction - create a new transaction  
`[app.post('/transaction', function(req, res))]`
3. /mine - mine a new block by using the proofOfWork method  
`[app.get('/mine', function(req, res))]`
4. /consensus - validate chain using the chainIsValid method  
`[app.get('/consensus), function(req, res))]`



```

// BLOCKCHAIN endpoint is used to fetch entire blockchain
app.get('/blockchain', function(req, res) {
  res.send(bitcoin);
});

// TRANSACTION endpoint is used to create a new transaction; use 'post' because web server accepts data
app.post('/transaction', function(req, res) {
  const blockIndex = bitcoin.createNewTransaction(req.body.amount, req.body.sender, req.body.recipient)
  res.json({ note: `Transaction will be added in block ${blockIndex}.` }); //transaction creation success note
});

// MINE endpoint is used to mine and create a new block using previous block data
app.get('/mine', function(req, res) {
  //previous block's index and hash
  const lastBlock = bitcoin.getLastBlock();
  const previousBlockHash = lastBlock['hash'];

  //current block data has current index (previous + 1) and pending transactions array
  const currentBlockData = {
    transactions: bitcoin.pendingTransactions,
    index: lastBlock['index'] + 1
  };

  const nonce = bitcoin.proofOfWork(previousBlockHash, currentBlockData); //acquire 'safe' hash nonce
  const blockHash = bitcoin.hashBlock(previousBlockHash, currentBlockData, nonce); //hash data with safe nonce
  const newBlock = bitcoin.createNewBlock(nonce, previousBlockHash, blockHash); //create new block

  res.json({ note: "New block mined successfully", block: newBlock }); //new block creation success note
});

// CONSENSUS endpoint is used to call chainIsValid method to check for inconsistencies
app.get('/consensus', function(req, res) {
  const value = bitcoin.chainIsValid(bitcoin.chain);
  res.json({ note: `Blockchain security check complete. VALID: ${value}.` });
});

```

```
localhost:3000/blockchain x +
localhost:3000/blockchain
School Entertainment Dr Chu FPGA Senior Project
{
  "chain": [
    {
      "index": 1,
      "timestamp": 1644223521345,
      "transactions": [],
      "nonce": 100,
      "hash": "0",
      "previousBlockHash": "0"
    },
    {
      "index": 2,
      "timestamp": 1644223670793,
      "transactions": [],
      "nonce": 18140,
      "hash": "0000b9135b054d1131392c9eb9d03b0111d4b516824a03c35639e12858912100",
      "previousBlockHash": "0"
    },
    {
      "index": 3,
      "timestamp": 1644223731919,
      "transactions": [],
      "nonce": 92894,
      "hash": "00002778916a7dad7260a1b6c7f17be291bda44445d157e48da55fe9dbb06b3",
      "previousBlockHash": "0000b9135b054d1131392c9eb9d03b0111d4b516824a03c35639e12858912100"
    },
    {
      "index": 4,
      "timestamp": 1644223737542,
      "transactions": [],
      "nonce": 92100,
      "hash": "0000d10859b91ab623a0bab0ddccbc6a68635b06f6fd582017b055e787303a569",
      "previousBlockHash": "00002778916a7dad7260a1b6c7f17be291bda44445d157e48da55fe9dbb06b3"
    },
    {
      "index": 5,
      "timestamp": 1644223830300,
      "transactions": [],
      "nonce": 125853,
      "hash": "0000dbb63de5d691a2f4926f58c06a4a116dc25c05093eff8ce6eb24072420",
      "previousBlockHash": "0000d10859b91ab623a0bab0ddccbc6a68635b06f6fd582017b055e787303a569"
    }
  ],
  "pendingTransactions": []
}
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
previousBlockHash => 0
currentBlockHash => 0000b9135b054d1131392c9eb9d03b0111d4b516824a03c35639e12858912100
previousBlockHash => 0000b9135b054d1131392c9eb9d03b0111d4b516824a03c35639e12858912100
currentBlockHash => 000003f7f57c94c7d13bfa11220b58ec8db1a5cbfd7b43288d69a16fafac4470
previousBlockHash => 000003f7f57c94c7d13bfa11220b58ec8db1a5cbfd7b43288d69a16fafac4470
currentBlockHash => 00005f77a7b004b9b8c9a7173f8d4049707e70801975835b5ea3dbfcd840380e
previousBlockHash => 00005f77a7b004b9b8c9a7173f8d4049707e70801975835b5ea3dbfcd840380e
currentBlockHash => 0000062ff3d935285727c3998fce46b06772dc1de0a73948db651270fc2baa65
previousBlockHash => 0000062ff3d935285727c3998fce46b06772dc1de0a73948db651270fc2baa65
currentBlockHash => 0000ef093d26eb926dae5227967be7b774a9de2b5157a987eec0c05e4722abc2
VALID: true
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: true
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: false
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: true
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: false
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: true
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: false
```

```
C:\Users\ajbar\programs\blockchain>node dev/test.js
VALID: true
```

```
C:\Users\ajbar\programs\blockchain>
```