Docbridge Project Journal

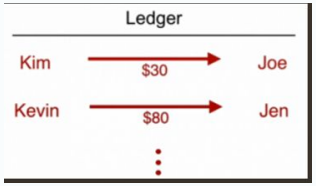
Blockchain Research - Revision 1

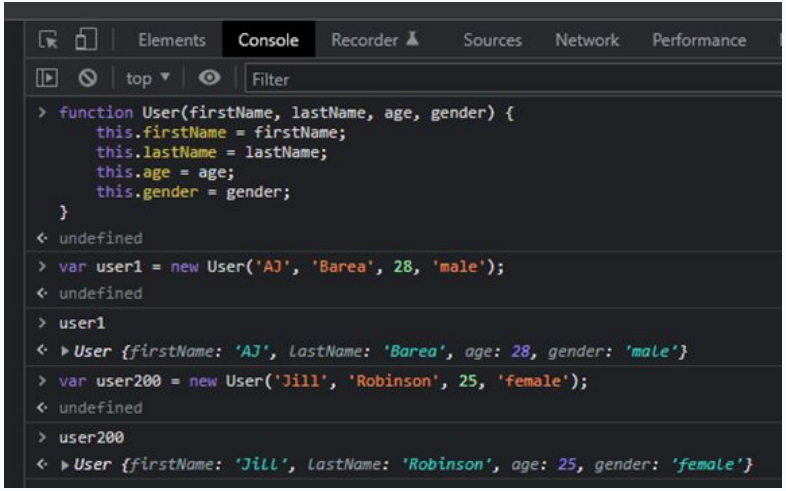
Blockchain Research Notes

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






```
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
```

3

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

4

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

1

```
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: 'KDGGKSWHGFKWR'
    },
    {
      index: 3,
      timestamp: 1643881606388,
      transactions: [],
      nonce: 9764,
      hash: 'fxyqwi497547',
      previousBlockHash: 'UXFAESWDTHHDG'
    }
  ],
  pendingTransactions: [
    {
      amount: 100,
      sender: 'ALEXHT845SJ5TKCJ2',
      recipient: 'JENN5BG5DF6HT8NG9'
    },
    {
      amount: 93,
      sender: 'KDGGKSWHGFKWR',
      recipient: 'JENN5BG5DF6HT8NG9'
    },
    {
      amount: 2,
      sender: 'UXFAESWDTHHDG',
      recipient: 'JENN5BG5DF6HT8NG9'
    }
  ]
}
```

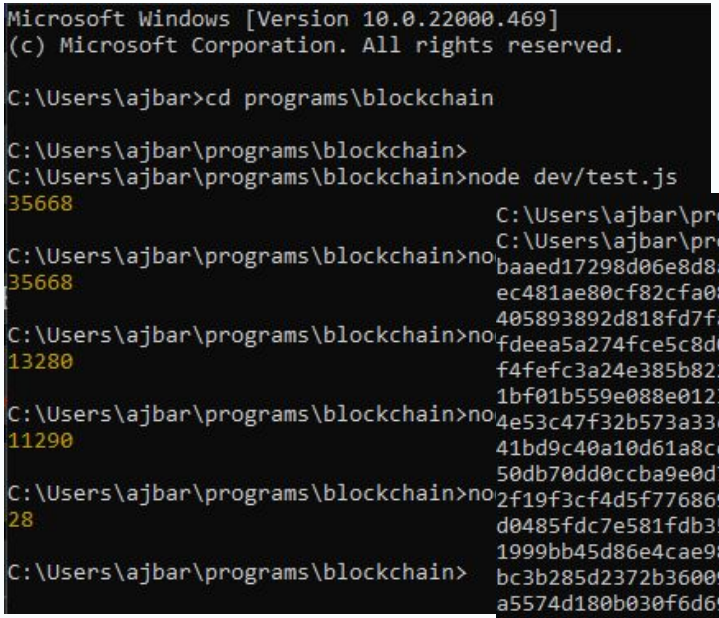
2

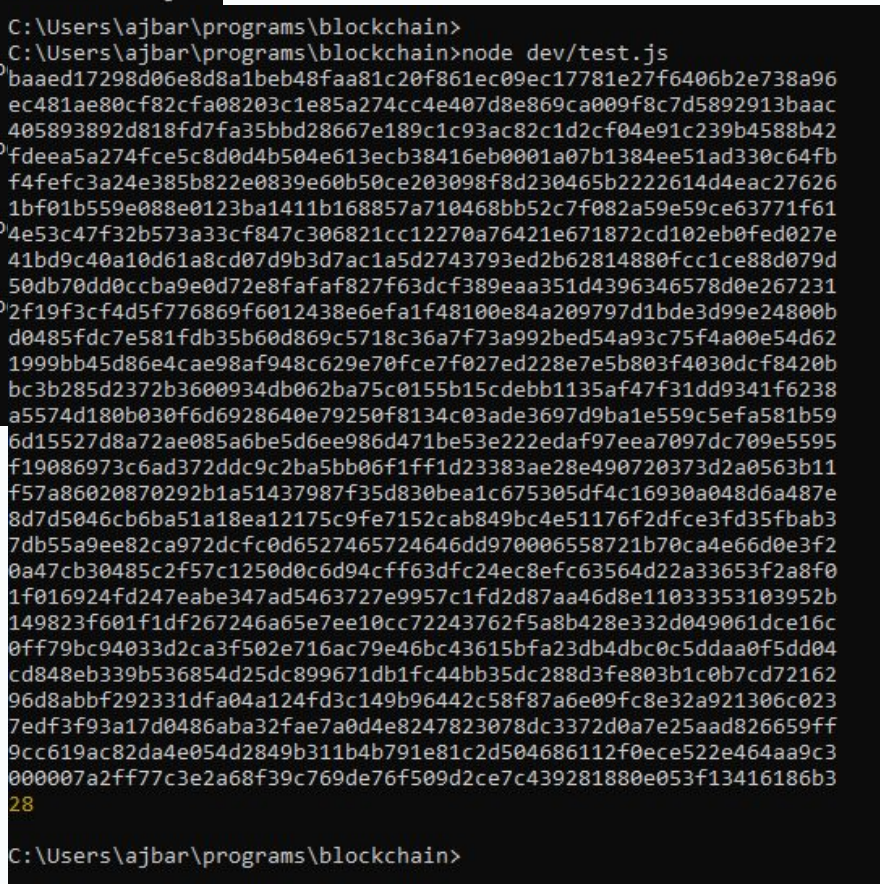
Blockchain Research Notes

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!





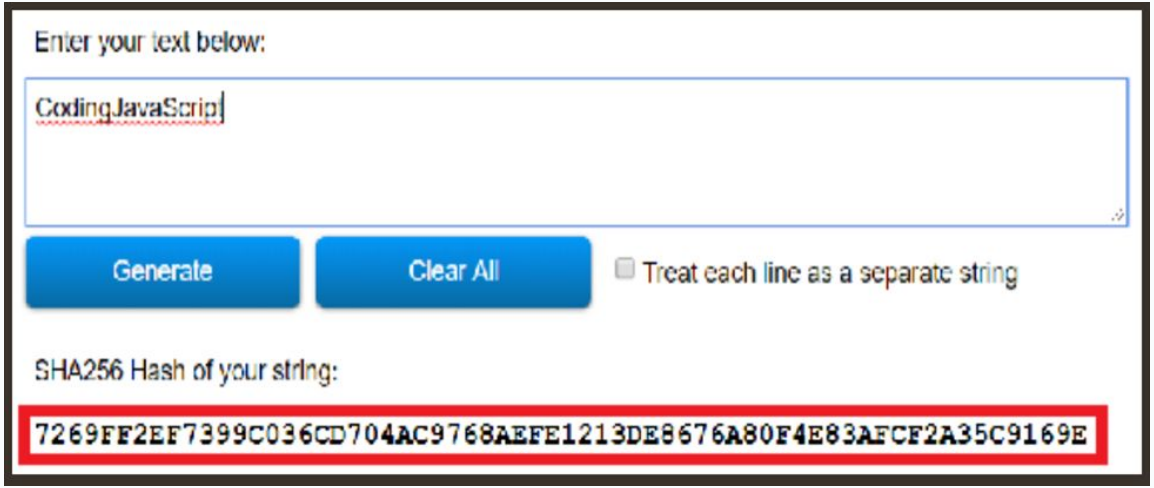
Blockchain Research Notes

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



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 ⇒

```
// 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;
}
```

Proof Of Work ⇒

```
// 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:

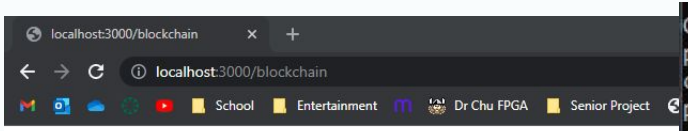
1. /blockchain - fetch entire blockchain to look at the data
   [app.get('/blockchain', function (req, res)]

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}.`});
});
```




```
{
  "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": "00002778916a7dadc7260a1b6cff17be291bda44445d157e48da55fe9dbb06b3",
      "previousBlockHash": "0000b9135b054d1131392c9eb9d03b0111d4b516824a03c35639e12858912100"
    },
    {
      "index": 4,
      "timestamp": 1644223737542,
      "transactions": [],
      "nonce": 92100,
      "hash": "0000d10859b91ab623a0bab0ddccbca68635b06f6fd582017b055e787303a569",
      "previousBlockHash": "00002778916a7dadc7260a1b6cff17be291bda44445d157e48da55fe9dbb06b3"
    },
    {
      "index": 5,
      "timestamp": 1644223830300,
      "transactions": [],
      "nonce": 125853,
      "hash": "0000dbb63de5d691a2f4926f58c06a4a116dc255c0c5093eff8ce6eb24072420",
      "previousBlockHash": "0000d10859b91ab623a0bab0ddccbca68635b06f6fd582017b055e787303a569"
    }
  ],
  "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>
```