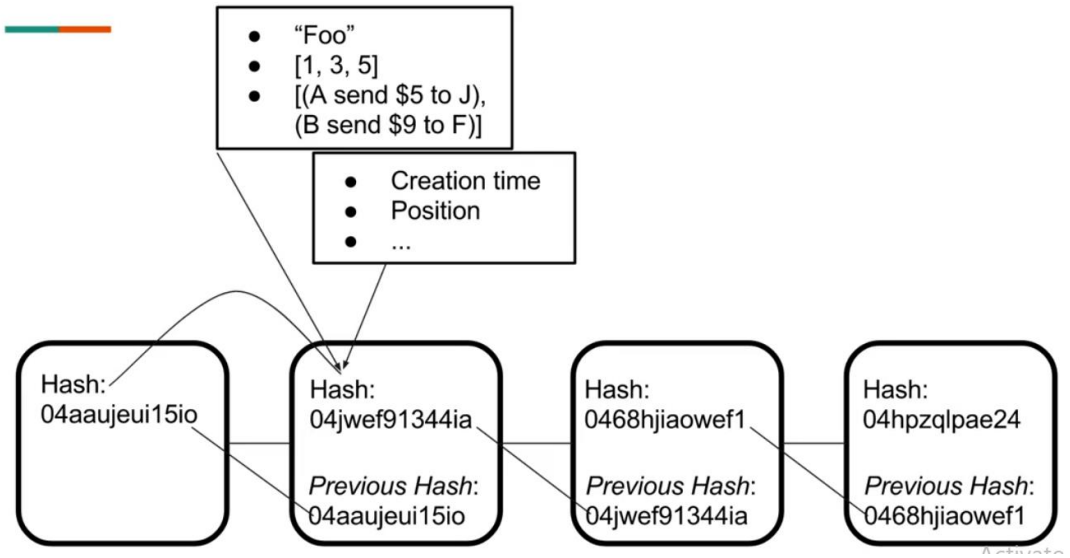
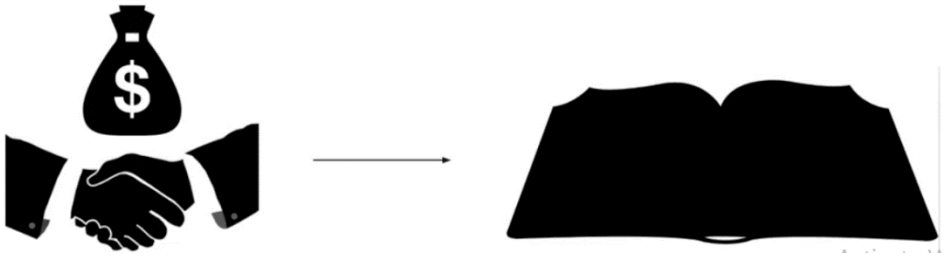
BlockChain

What is the Blockchain and Why Use it:

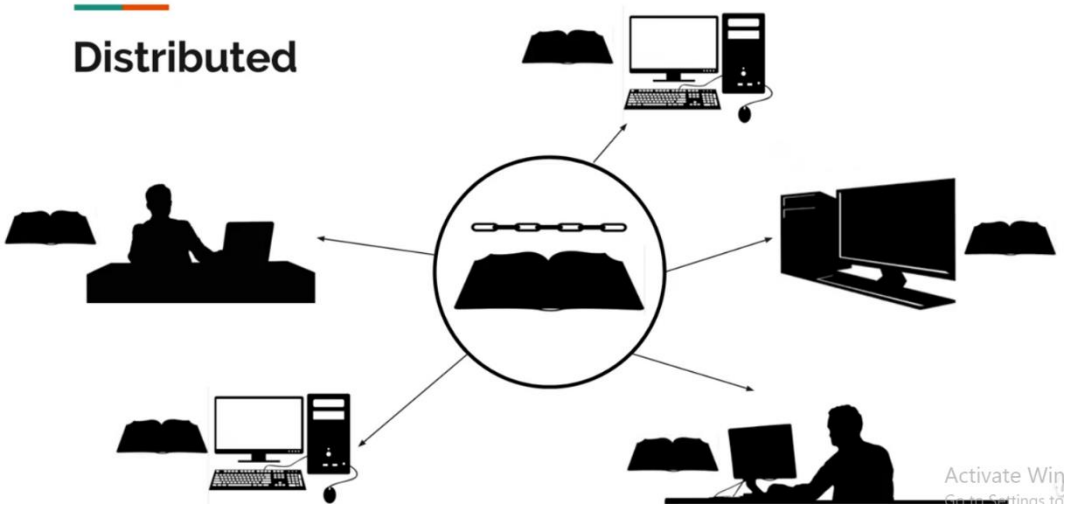


Ledger

A ledger is a record-keeping book that stores all the transactions of an organization.



Distributed



BlockChain

Centralized

- One entity records the data.
- The central entity has a lot of power.
- Full authority to fine or reward.
- Complete trust with the entity.

Decentralized

- Everyone records the data.
- Everyone has equal power.
- Fair and transparent system.
- *Trustless.*

Why use the Blockchain?

- Decentralization leads to a trustless system.
- No middle men and no fees.
- Highly secure and no central point of failure.
- Dependable data.

Blockchain

The blockchain is a distributed and decentralized ledger that stores data such as transactions, and that is publicly shared across all the nodes of its network.

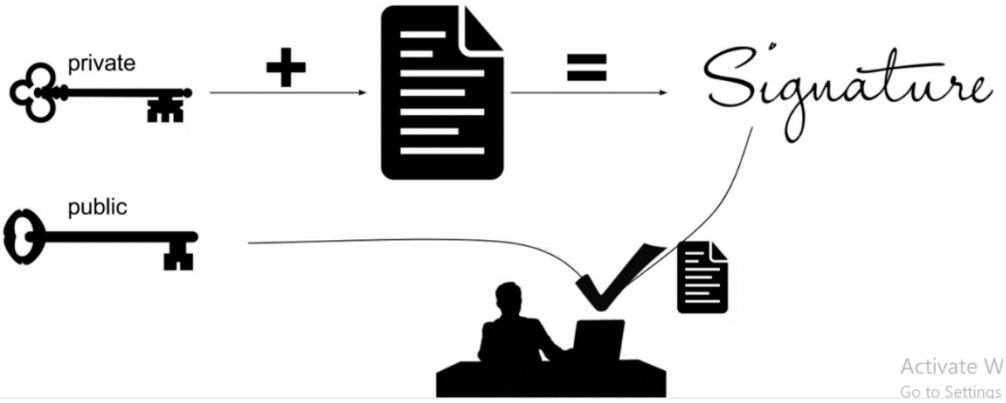
BlockChain

The Blockchain in Practice Cryptocurrencies and Cryptography:

Leverages the blockchain

- How is this secure?
- Uses cryptography to generate digital signatures.

Digital Signatures

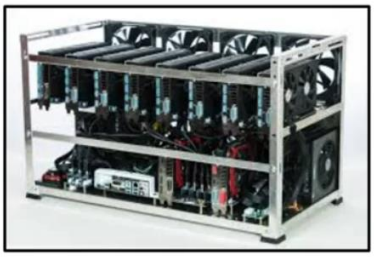


Wallets

- Objects that store the private and public key of an individual.
- The public key is the address of the wallet.
- Help sign transations.

BlockChain

The Blockchain in Practice Mining and Bitcoin:



Mining

- Transactions are temporarily "unconfirmed."
- Include blocks of transactions by solving a "proof of work" algorithm.
  - Difficult to solve, and computationally expensive.
  - Once solved, the miner can add the block and other miners will verify.
  - Miners are rewarded for adding a block to the chain.
  - The difficulty can adjust to control the rate of new blocks coming in.

Bitcoin

- The first decentralized cryptocurrency in 2009.
- Great growth, and widespread adoption - $$$

ripple IOTA ethereum litecoin

Additional Use Cases

- Blockchain-based voting registers.
- Blockchain-supported documentation and identification systems.
- *"Will this really change the world?"*

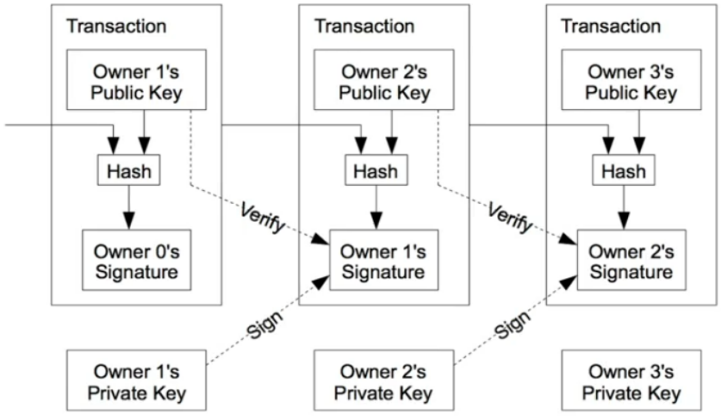
BlockChain

Roadmap to Building the Blockchain Guided by the Bitcoin White Paper:

bitcoin.pdf

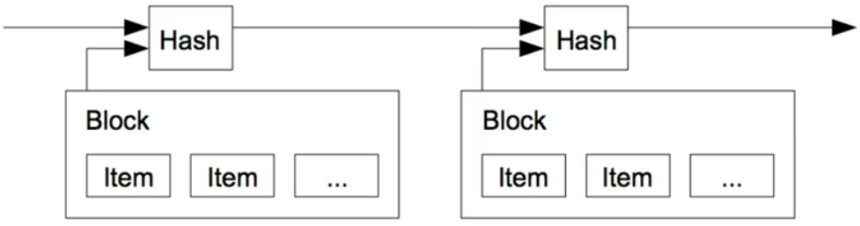
- Published on October 31st, 2008.
- White paper is an official document outlining proposals on an issue.
- Satashi Nakamoto

2. Transactions



3. Timestamp Server

- Timestamps order blocks in the chain.

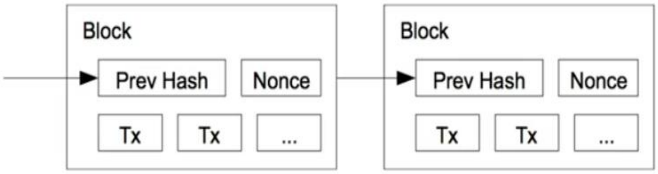


BlockChain

4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



5. Network

6. Incentive

7. Reclaiming Disk Space

8. Simplified Payment Verification

9. Combining and Splitting Value

10. Privacy

11. Calculations

BlockChain

BlockChain

02 Build the Blockchain – Blocks:

Set Up the Blockchain Application:

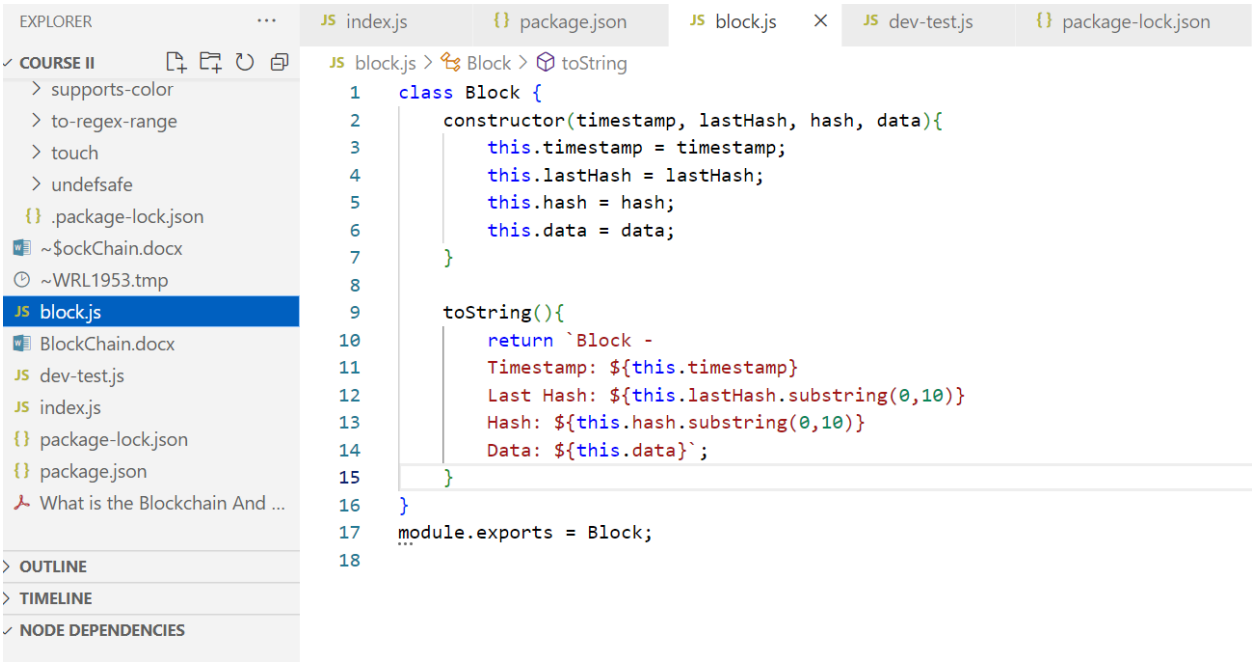
```
C:\Users\Dell>cd C:\Users\Dell\Desktop\BlockChain

C:\Users\Dell\Desktop\BlockChain>cd course II

C:\Users\Dell\Desktop\BlockChain\Course II>npm init -y
Wrote to C:\Users\Dell\Desktop\BlockChain\Course II\package.json:

{
  "name": "course-ii",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



C:\Users\Dell\Desktop\BlockChain\Course II>npm i nodemon --save-dev

added 32 packages, and audited 33 packages in 2s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Dell\Desktop\BlockChain\Course II>
```

BlockChain

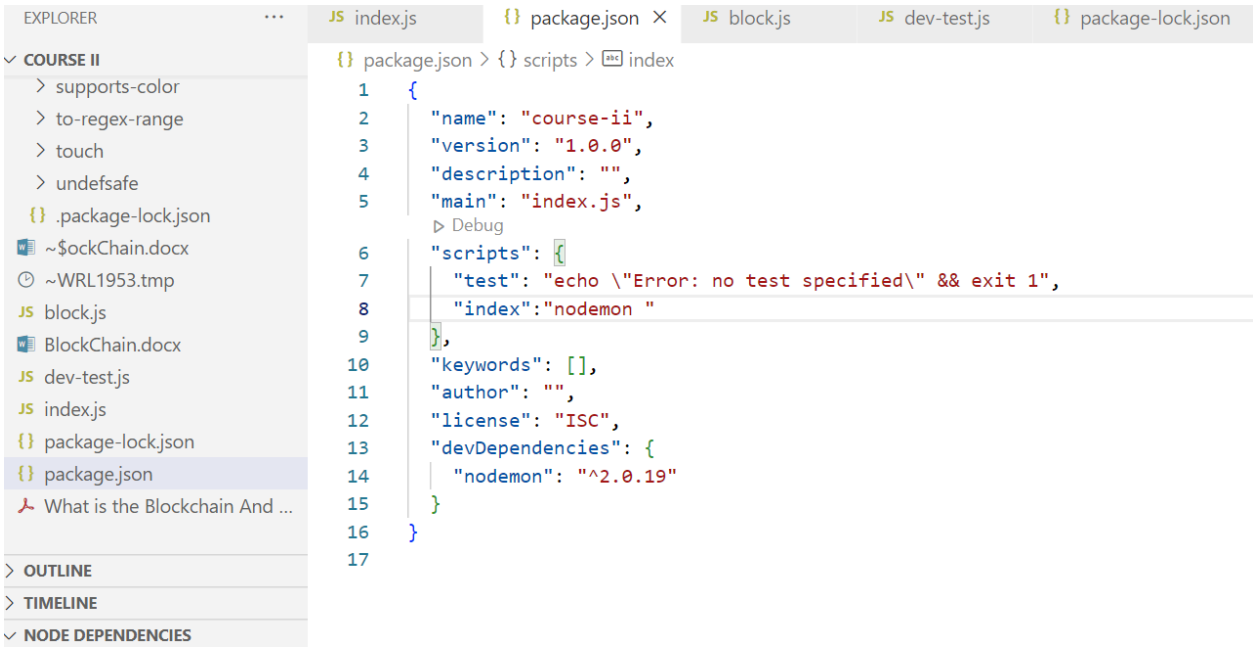
Create a Block:

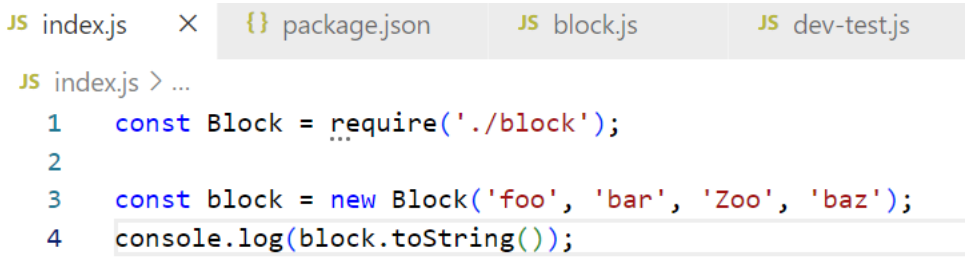
Block

- Timestamp in milliseconds.
- lastHash - the hash of the block before it.
- hash - based on its own data.
- The data to store.



BlockChain

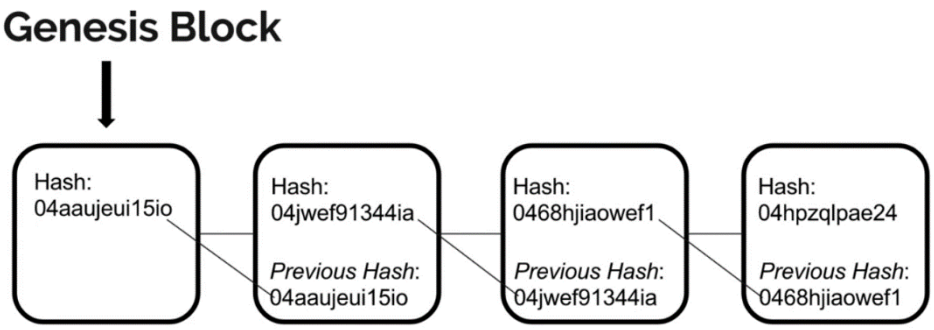





```
C:\Users\Dell\Desktop\BlockChain\Course II>npm run index

> course-ii@1.0.0 index
> nodemon

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Block -
        Timestamp: foo
        Last Hash: bar
        Hash: Zoo
        Data: baz
[nodemon] clean exit - waiting for changes before restart
```

BlockChain

Genesis Block:




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

```
const block = new Block('foo', 'bar', 'Zoo', 'baz');
console.log(block.toString());
console.log(Block.genesis().toString());
```

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm run index

> course-ii@1.0.0 index
> nodemon

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Block -
        Timestamp: foo
        Last Hash: bar
        Hash: Zoo
        Data: baz
Block -
        Timestamp: Genesis time
        Last Hash: -----
        Hash: f1r57-h45h
        Data:
[nodemon] clean exit - waiting for changes before restart
```

BlockChain

Mine Blocks:

```
static mineBlock(lastBlock, data){
    const timestamp = Date.now();
    const lastHash = lastBlock.hash;
    const hash = 'todo-hash';

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

}
```

```
const fooBlock = Block.mineBlock(Block.genesis(), 'foo');
console.log(fooBlock.toString());
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Block -
        Timestamp: 1661751440935
        Last Hash: f1r57-h45h
        Hash: todo-hash
        Data: foo
[nodemon] clean exit - waiting for changes before restart
```

BlockChain

SHA256 Hash Function:

Block Hashes and SHA-256

- The hash is generated from the timestamp, lastHash, and stored data.
- We'll use an algorithm called SHA-256.
  - Produces a unique 32-byte (256 bit) hash value for unique data inputs.
  - One-way hash.
- Useful for block validation.

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i crypto-js --save

added 1 package, and audited 34 packages in 848ms

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
const SHA256 = require('crypto-js/sha256');

 static mineBlock(lastBlock, data){
     const timestamp = Date.now();
     const lastHash = lastBlock.hash;
     const hash = Block.hash(timestamp, lastHash, data);

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

 }
```

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

}
```

```
[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Block -
        Timestamp: 1661752741755
        Last Hash: f1r57-h45h
        Hash: 0ca2b0c81a
        Data: foo
[nodemon] clean exit - waiting for changes before restart
```

BlockChain

Test the Block:

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i jest --save-dev

added 275 packages, and audited 309 packages in 14s

32 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Dell\Desktop\BlockChain\Course II>
```

```
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.735 s, estimated 1 s
Ran all test suites.

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

```
"scripts": {
  "test": "jest --watchAll",
  "index": "nodemon "
},
```

JS block.test.js > ...

```
1   // const { expect } = require('@babel/types');
2   // const { it } = require('node:test');
3   // const { describe } = require('yargs');
4   const Block = require('./block');
5
6   describe('Block', () => {
7
8       let data, lastBlock, block;
9
10      beforeEach(()=> {
11          data = 'bar';
12          lastBlock = Block.genesis();
13          block = Block. mineBlock(lastBlock,data);
14      });
15      it('sets the `data` to match the input', ()=> {
16          expect(block.data).toEqual(data);
17      });
18
19      it('sets the `lastHash` to match the hash of the last block', ()=> {
20
21          expect(block.lastHash).toEqual(lastBlock.hash);
22
23
24      });
25  });
```

BlockChain

BlockChain

Build the Blockchain - the Chain:

Build the Blockchain Class:

blockchain.js > ...

```
1   const Block = require('./block');
2
3   class Blockchain{
4       constructor(){
5           this.chain = [Block.genesis()];
6       }
7
8       addBlock(data){
9
10          const block = Block.mineBlock(this.chain[this.chain.length-1], data);
11          this.chain.push(block);
12
13          return block;
14
15      }
16  }
17
18  module.exports = Blockchain;
```

Test the Blockchain:

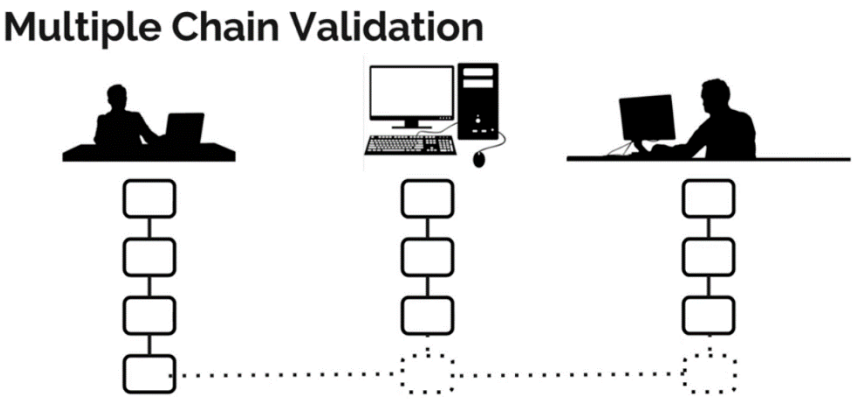
```
Watch Usage: Press w to show more.
 PASS  Build the Blockchain/block.test.js
 PASS  ./blockchain.test.js
 PASS  ./block.test.js

Test Suites: 3 passed, 3 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        0.859 s, estimated 1 s
Ran all test suites.

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

BlockChain

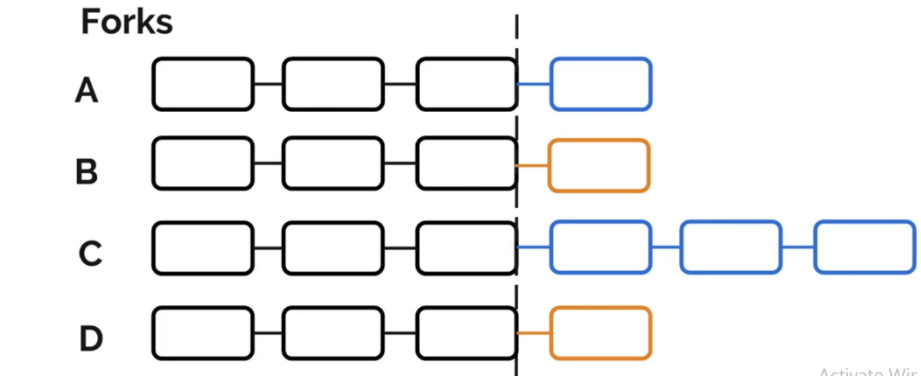
JS blockchain.test.js > describe('Blockchain') callback > it('starts with genesis block') callback

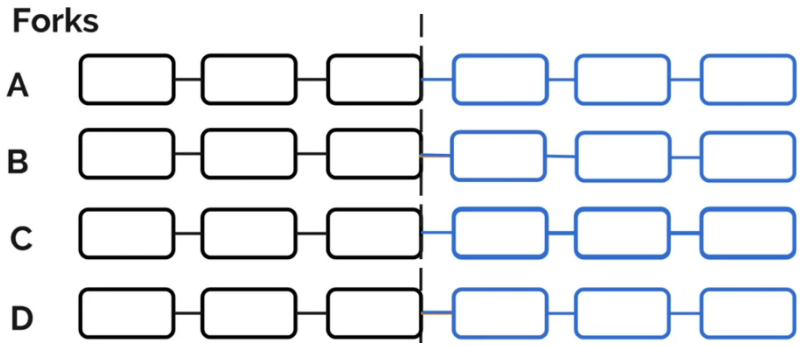
```
1
2   const Blockchain = require('./blockchain');
3   const Block = require('./block');
4
5
6   describe('Blockchain', ()=> {
7       let bc;
8
9       beforeEach(() =>{
10          bc = new Blockchain();
11
12      });
13
14      it('starts with genesis block', () => {
15          expect(bc.chain[0]).toEqual(Block.genesis());
16
17
18      });
19
20      it('adds a new block', () =>{
21          const data = 'foo';
22          bc.addBlock(data);
23
24          expect(bc.chain[bc.chain.length-1].data).toEqual(data);
25      });
26
27  });
```

Multiple Chain Validation:



BlockChain





Chain Validation:

blockchain.js > ...

```
1   const Block = require('./block');
2
3   class Blockchain{
4       constructor(){
5           this.chain = [Block.genesis()];
6       }
7
8       addBlock(data){
9
10          const block = Block.mineBlock(this.chain[this.chain.length-1], data);
11          this.chain.push(block);
12
13          return block;
14
15      }
16
17      isValidChain(chain){
18          if(JSON.stringify(chain[0]) !== JSON.stringify(Block.genesis()))
19          return false;
20
21          for(let i=1; i<chain.length; i++){
22              const block = chain[i];
23              const lastBlock = chain[i-1];
24
```

BlockChain

```
24
25              if (block.lastHash !== lastBlock.hash ||
26                  block.hash !== Block.blockHash(block)){
27                  return false;
28              }
29          }
30          return true;
31      }
32  }
33
34  module.exports = Blockchain;
```

JS block.js > Block > blockHash

```
38
39      static blockHash(block){
40          const {timestamp, lastHash, data} = block;
41          return Block.hash(timestamp, lastHash, data);
42      }
```

Test Chain Validation:

JS blockchain.test.js > describe('Blockchain') callba

```
9
10      beforeEach(() =>{
11          bc = new Blockchain();
12          bc2 = new Blockchain();
13
14      });
```

```
PASS  ./block.test.js
PASS  Build the Blockchain/block.test.js
PASS  ./blockchain.test.js

Test Suites: 3 passed, 3 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        0.954 s, estimated 1 s
Ran all test suites.

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

BlockChain

blockchain.test.js > describe('Blockchain') callback

```
28
29      it('validates a valid chain', () => {
30          bc2.addBlock('foo');
31
32          expect(bc.isValidChain(bc2.chain)).toBe(true);
33
34      });
35
36      if('invalidates a chain with a corrupt genesis block', () => {
37          bc2.chain[0].data = 'Bad data';
38
39          expect(bc.isValidChain(bc2.chain)).toBe(false);
40      });
41
42      if('invalidates a corrupt chain', () =>{
43          bc2.addBlock('foo');
44          bc2.chain[1].data = 'Not foo';
45
46          expect(bc.isValidChain(bc2.chain)).toBe(false);
47      });
48
49  });
```

Replace the Chain:

blockchain.js > ...

```
32
33      replaceChain(newChain){
34          if(newChain.length <= this.chain.length){
35              console.log('Received chain is not longer than the current chain ');
36              return;
37
38          } else if(!this.isValidChain(newChain)){
39              console.log('The received chain is not valid.');
40              return;
41          }
42
43          console.log('Replacing blockchain with the new chain.');
44          this.chain = newChain;
45      }
```

BlockChain

Test Chain Replacement:

JS blockchain.test.js > Blockchain

```
49
50      it('replaces the chain with a valid chain',() =>{
51          bc2.addBlock('goo');
52          bc.replaceChain(bc2.chain);
53
54          expect(bc.chain).toEqual(bc2.chain);
55      });
56
57      if('does not replace the chain with one of less than or equal to length', () =>{
58          bc.addBlock('foo');
59          bc.replaceChain(bc2.chain);
60
61          expect(bc.chian).not.toEqual(bc2.equal);
62      });
63
64  });
```

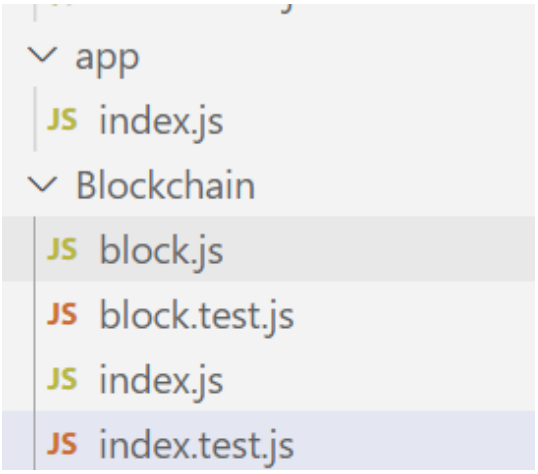
```
Test Suites: 3 passed, 3 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        0.938 s, estimated 1 s
Ran all test suites.

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

BlockChain

BlockChain

Develop the Blockchain Application:

Organize the Project:




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

    console.log
      Replacing blockchain with the new chain.

      at Blockchain.log [as replaceChain] (Blockchain/in


Test Suites: 5 passed, 5 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        1.158 s
Ran all test suites.

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

BlockChain

Blockchain API - Get Blocks:

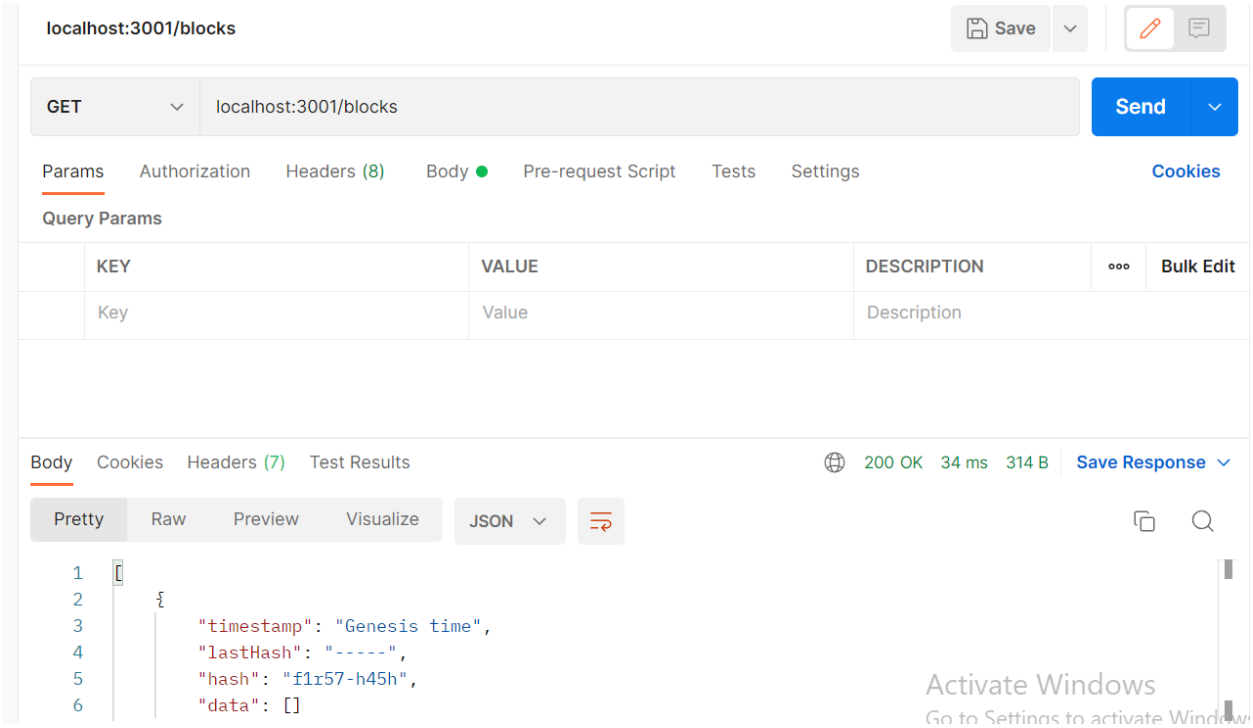
```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i express --save

added 60 packages, and audited 369 packages in 2s

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm run dev

> course-ii@1.0.0 dev
> nodemon ./app

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./app`
Listening on port 3001
```

app > JS index.js > ...

```
1   const express = require ('express');
2   const Blockchain = require('../blockchain');
3
4   const HTTP_PORT = process.env.HTTP_PORT || 3001;
5
6   const app = express();
7   const bc = new Blockchain();
8
9   app.get('/blocks', (req, res)=>{
10      res.json(bc.chain);
11  });
12
13  app.listen(HTTP_PORT, ()=> console.log(`Listening on port ${HTTP_PORT}`));
14
```

BlockChain



Mine Blocks Post Request:

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i body-parser --save

up to date, audited 369 packages in 811ms

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

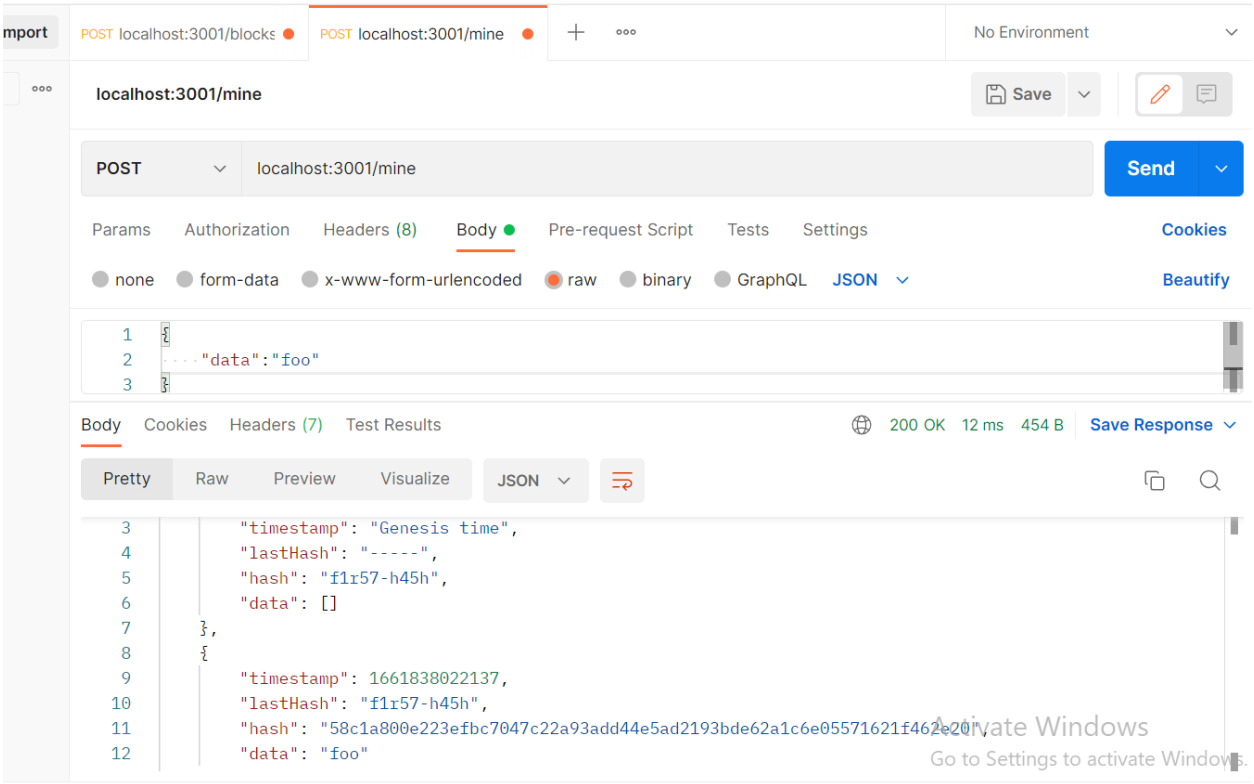
app > JS index.js > app.post('/mine') callback

```
1  const express = require ('express');
2
3  const bodyParser = require('body-parser');
4
```

BlockChain

app > JS index.js > app.post('/mine') callback

```
17
18  app.post('/mine', (req, res)=>{
19      const block = bc.addBlock(req.body.data);
20      console.log(`New block added : ${block.toString()}`);
21
22      res.redirect('/blocks');
23  });
24
25
26  app.listen(HTTP_PORT, ()=> console.log(`Listening on port ${HTTP_PORT}`));
27
```



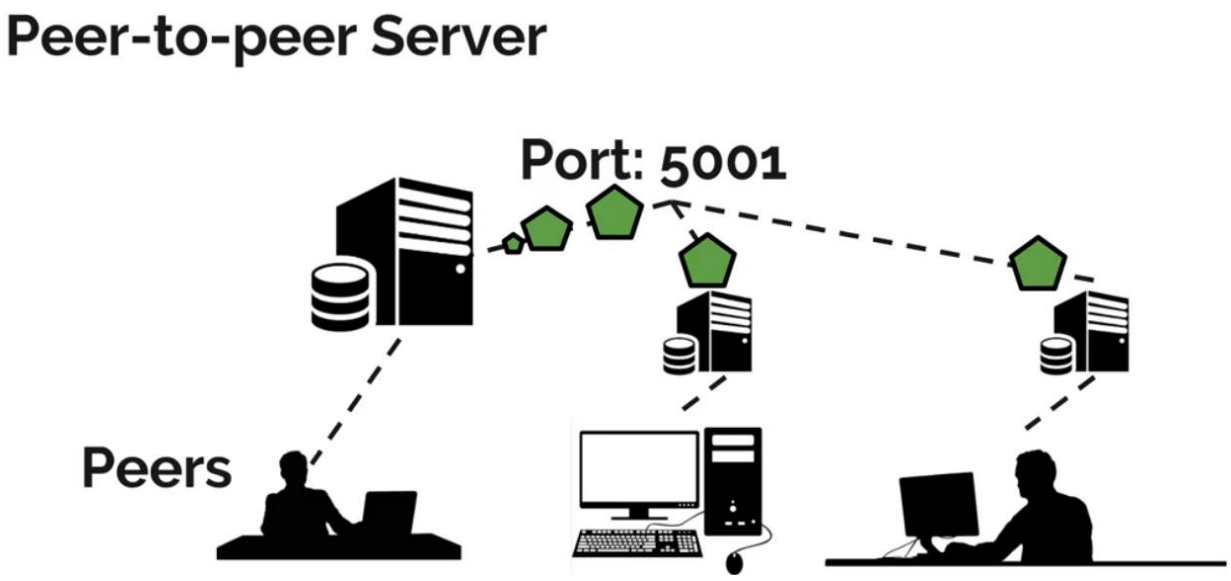
BlockChain

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm run dev

> course-ii@1.0.0 dev
> nodemon ./app

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./app`
Listening on port 3001
New block added : Block -
        Timestamp: 1661838022137
        Last Hash: f1r57-h45h
        Hash: 58c1a800e2
        Data: foo
```

BlockChain

Create the Blockchain Network:

Peer to Peer Server:



1. The first app to start the peer-to-peer server.

2. A later server, connecting to the original.

Create the WebSocket Server:

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i ws --save

added 1 package, and audited 370 packages in 925ms

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

BlockChain

app > JS p2p-server.js > P2pServer > listen

```
1   const Websocket = require('ws');
2
3   const P2P_PORT = process.env.P2P_PORT || 5001;
4   const peers= process.env.PEERS ? process.env.PEERS.split(',') : [];
5
6
7   class P2pServer {
8       constructor(blockchain){
9           this.blockchain = blockchain;
10          this.sockets = [];
11
12      }
13
14      listen(){
15          const server = new Websocket.Server({ port: P2P_PORT });
16          server.on('connection', socket => this.connectSocket(socket));
17          console.log(`Listening for peer-to-peer connections on: ${P2P_PORT}`);
18      }
19
20      connectSocket(socket){
21          this.sockets.push(socket);
22          console.log('Socket connected');
23      }
24  }
25
26  //$ HTTP_PORT=3002 P2P_PORT= ws://localhost:5001,ws://localhost:5002
```

Connect to Blockchain Peers:

app > JS p2p-server.js > P2pServer > connectToPeers > peers.forEach() callback

```
22
23      connectToPeers(){
24          peers.forEach(peer =>{
25              const socket = new Websocket(peer);
26
27              socket.on('open', () => this.connectSocket(socket) );
28          });
29      }
30
31      connectSocket(socket){
32          this.sockets.push(socket);
33          console.log('Socket connected');
34      }
35  }
36
37  module.exports = P2pServer;
```

BlockChain

```
const P2pServer = require('./p2p-server');

const p2pServer = new P2pServer(bc);

p2pServer.listen();
```

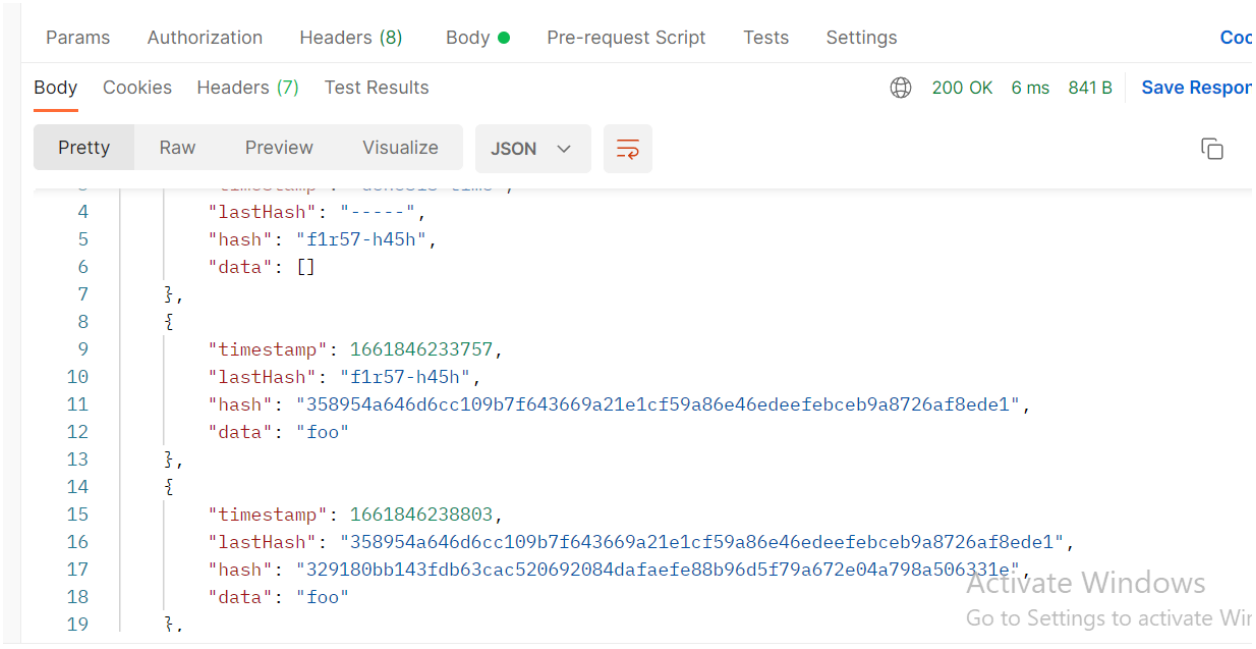
```
C:\Users\Dell\Desktop\BlockChain\Course II>npm run dev

> course-ii@1.0.0 dev
> nodemon ./app

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./app`
Listening for peer-to-peer connections on: 5001
Listening on port 3001
```

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm install -g http-server

added 39 packages, and audited 40 packages in 4s

9 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
C:\Users\Dell\Desktop\BlockChain\Course II> HTTP_PORT = 3002 P2P_PORT = 5002 PEERS = ws://localhost:5001 npm run dev
```

BlockChain

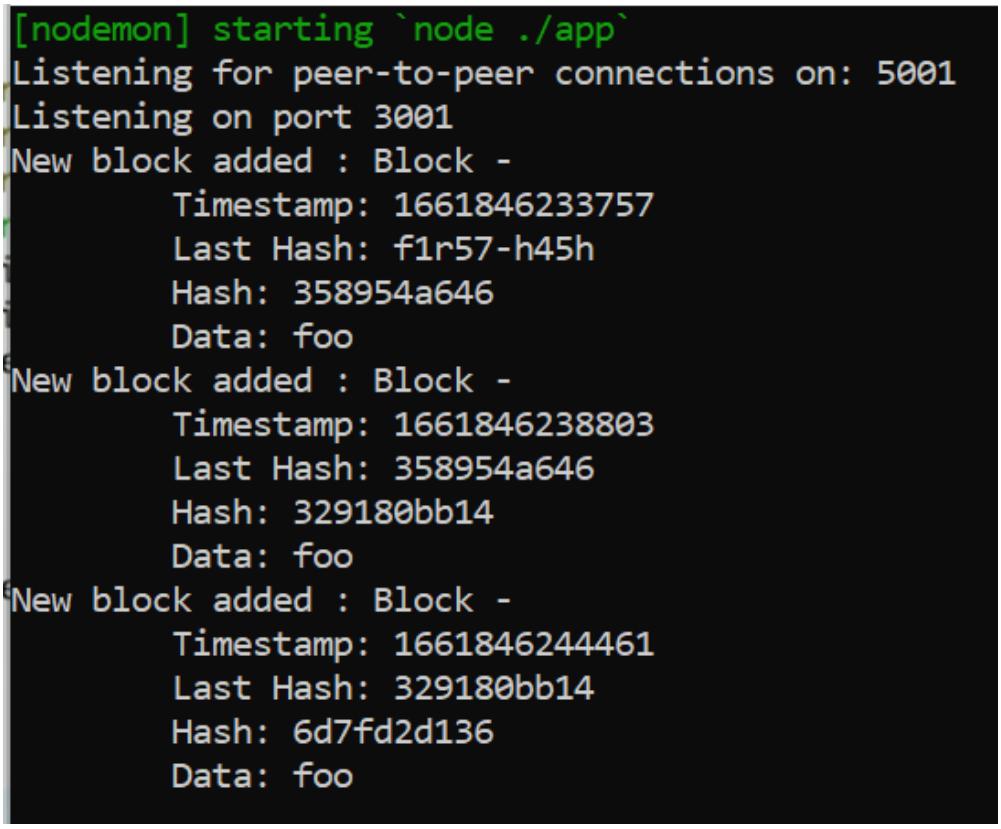
Handle Messages from Peers:



app > JS p2p-server.js > P2pServer > connectSocket

```
31      connectSocket(socket){
32          this.sockets.push(socket);
33          console.log('Socket connected');
34
35
36          this.messageHandler(socket);
37
38          socket.send(JSON.stringify(this.blockchain.chain));
39      }
40
41      messageHandler(socket){
42          socket.on('message', message =>{
43              const data = JSON.parse(message);
44              console.log('data',data);
45          } );
46      }
47  }
```

BlockChain



HTTP_PORT = 3002 P2P_PORT = 5002 PEERS = ws:// localhost:5001 npm run dev

set HTTP_PORT=3002 && set P2P_PORT=5002 && set PEERS=ws://localhost:5001 && npm run dev

BlockChain

Synchronize the Blockchain across Peers:

app > JS index.js > app.post('/mine') callback

```
21  });
22
23  app.post('/mine', (req, res)=>{
24      const block = bc.addBlock(req.body.data);
25      console.log(`New block added : ${block.toString()}`);
26
27      p2pServer.syncChains();
28
29      res.redirect('/blocks');
30  });
```

JS index.js Blockchain | JS index.test.js | JS index.js app | JS p2p-server.js

app > JS p2p-server.js > ...

```
44                  const data = JSON.parse(message);
45                  console.log('data',data);
46
47                  this.blockchain.replaceChain(data);
48              } );
49      }
50  sendChains(socket){
51      socket.send(JSON.stringify(this.blockchain.chain));
52  }
53  syncChains(){
54      this.sockets.forEach(socket =>this.sendChains(socket));
55  }
```

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

```
10        this.hash = hash;
11        this.data = data;
12        this.nonce = nonce;
13    }
```

Blockchain > JS block.js > Block

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

Blockchain > JS block.js > Block

```
22    }
23
24    static genesis(){
25        return new this('Genesis time', '-----', 'f1r57-h45h',[],0);
```

BlockChain

Blockchain > JS block.js > Block

```
31          let nonce = 0;
32          do{
33              nonce++;
34              timestamp = Date.now();
35              hash = Block.hash(timestamp, lastHash, data, nonce);
36
37
38          } while(hash.substring(0, DIFFICULTY) !== '0'.repeat(DIFFICULTY)
39
40
41
42          return new this(timestamp, lastHash, hash, data, nonce);
43
44      }
```

```
        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

config.js > ...

```
1   const DIFFICULTY = 4;
2
3   module.exports = {DIFFICULTY};
```

(property) DIFFICULTY: number

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

Blockchain > block.test.js > ...

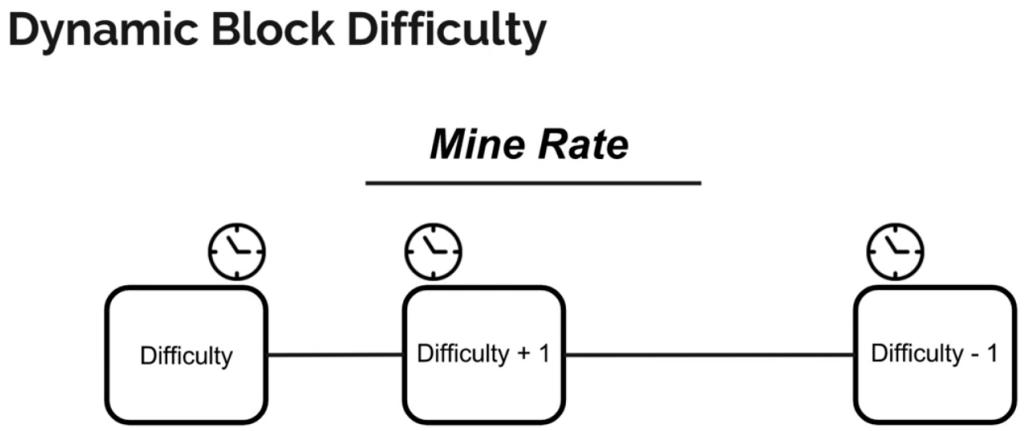
```
26
27
28      if('generates a hash that matches the diffiulty', ()=>{
29          expect(block.hash.substring(0, DIFFICULTY)).toEqual('0'.repeat(DIFFICULTY))
30
31      });
32  });
```

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

```
25      });
26
27
28      if('generates a hash that matches the diffiulty', ()=>{
29          expect(block.hash.substring(0, DIFFICULTY)).toEqual('0'.repeat(DIFFICULTY));
30          console.log(block.toString());
31
32      });
33  });
```

BlockChain

Dynamic Block Difficulty:



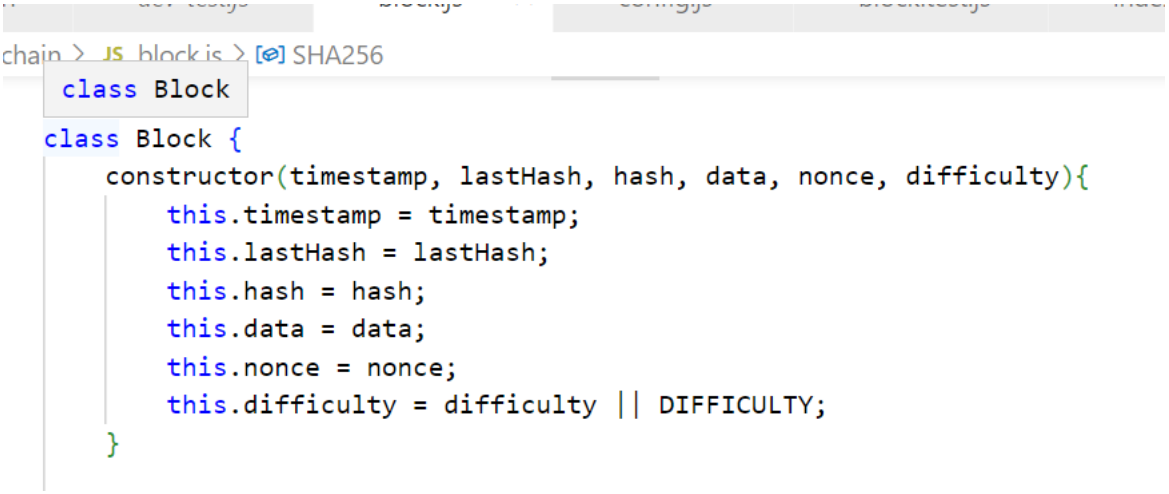

```
PASS  project/Build the Blockchain/block.test.js
PASS  project/Develop the Blockchain Application/Blockchain/block.test.js
PASS  project/Create the Blockchain Network/Blockchain/block.test.js
PASS  project/Build the Blockchain - the Chain/block.test.js

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
```

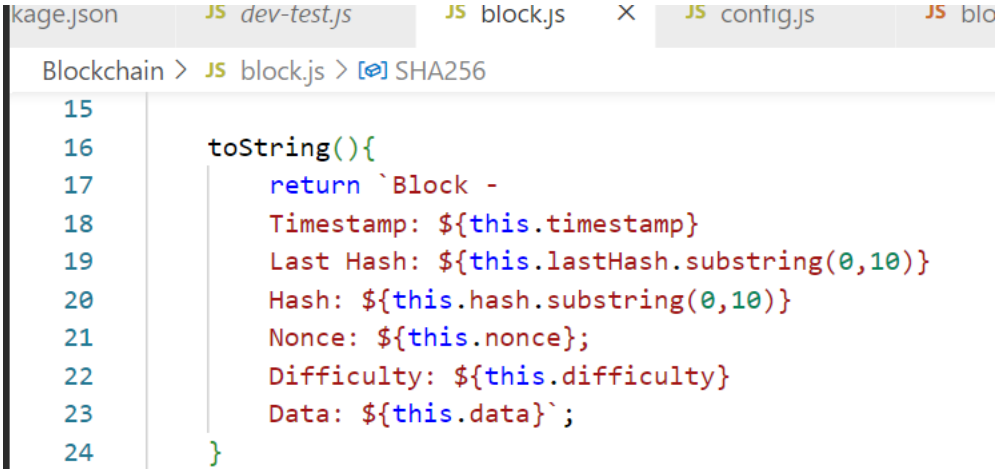
config.js > ...

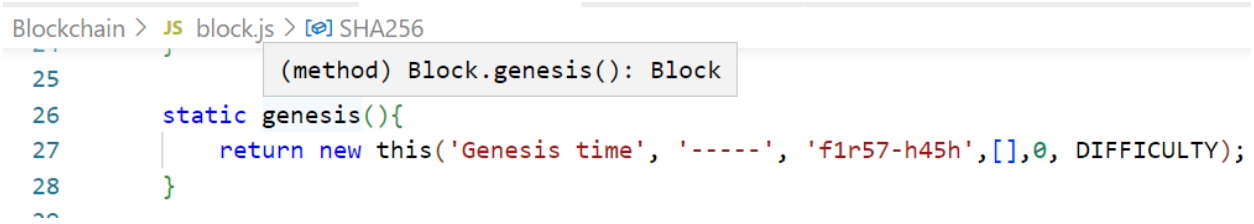
```
1   const DIFFICULTY = 4;
2
3   const MINE_RATE = 3000;
4
5   module.exports = { DIFFICULTY, MINE_RATE};
```

```
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
dev-test.js
block.js
config.js
Blockchain > JS block.js > SHA256
15
16 toString(){
17 return `Block -
18 Timestamp: ${this.timestamp}
19 Last Hash: ${this.lastHash.substring(0,10)}
20 Hash: ${this.hash.substring(0,10)}
21 Nonce: ${this.nonce};
22 Difficulty: ${this.difficulty}
23 Data: ${this.data}`;
24 }


Blockchain > JS block.js > SHA256
(method) Block.genesis(): Block
25
26 static genesis(){
27 return new this('Genesis time', '-----', 'f1r57-h45h',[],0, DIFFICULTY);
28 }

BlockChain

Blockchain > JS block.js > SHA256

```
30      static mineBlock(lastBlock, data){
31          let hash, timestamp;
32           const lastHash = lastBlock.hash;
33           let {difficulty} =lastBlock;
34           let nonce = 0;
35           do{
36                nonce++;
37               timestamp = Date.now();
38               difficulty = Block.adjustDifficulty(lastBlock, timestamp);
39                hash = Block.hash(timestamp, lastHash, data, nonce, difficulty);
40
41
42               } while(hash.substring(0, difficulty) !== '0'.repeat(difficulty));;
43
44
45
46           return new this(timestamp, lastHash, hash, data, nonce, difficulty);
47
48      }
```

Blockchain > JS block.js > Block > mineBlock

```
59
60      static adjustDifficulty(lastBlock, currentTime){
61          let {difficulty} = lastBlock;
62          difficulty = lastBlock.timestamp +MINE_RATE > currentTime ? difficulty +1 : difficulty -1;
63          return difficulty;
64      }
65  }
```

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

JS index.js > ...

```
12
13    const Blockchain = require('./blockchain');
14
15    const bc = new Blockchain();
16
17    for (let i = 0; i <10; i++){
18        console.log(bc.addBlock(`foo ${i}`).toString());
19    }
20
```

```
[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

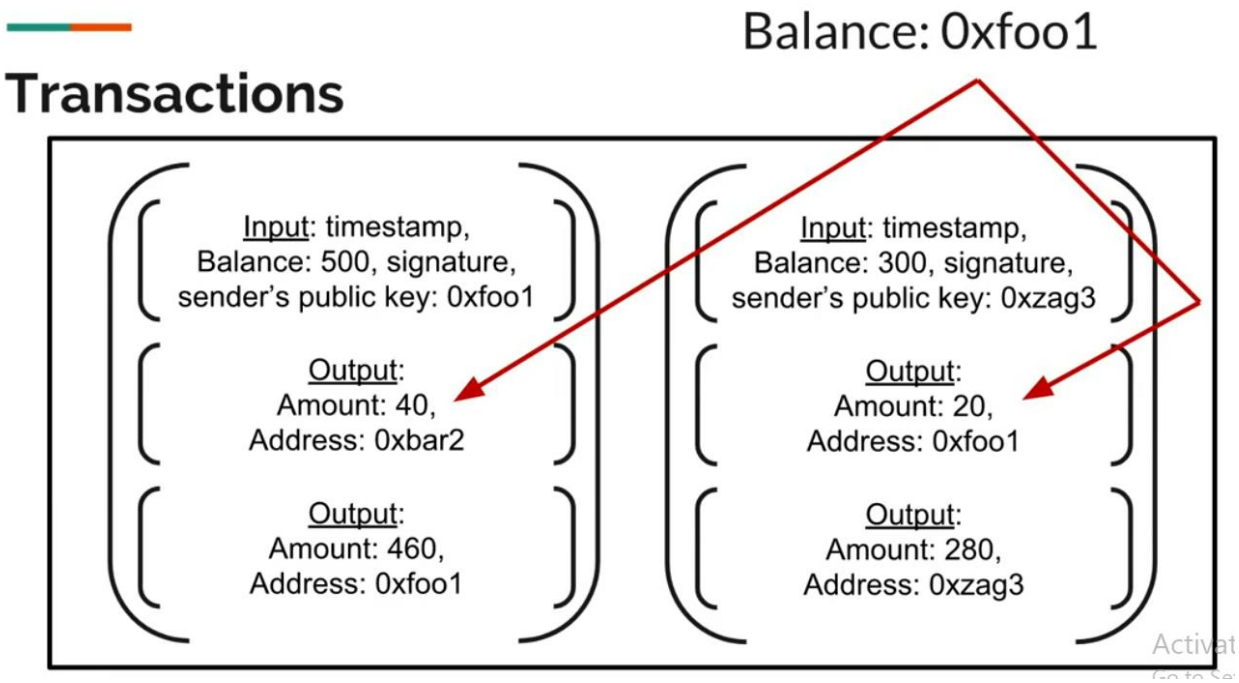
Wallets and Transactions on the Blockchain:

Wallets Keys and Transactions:

What is a wallet?

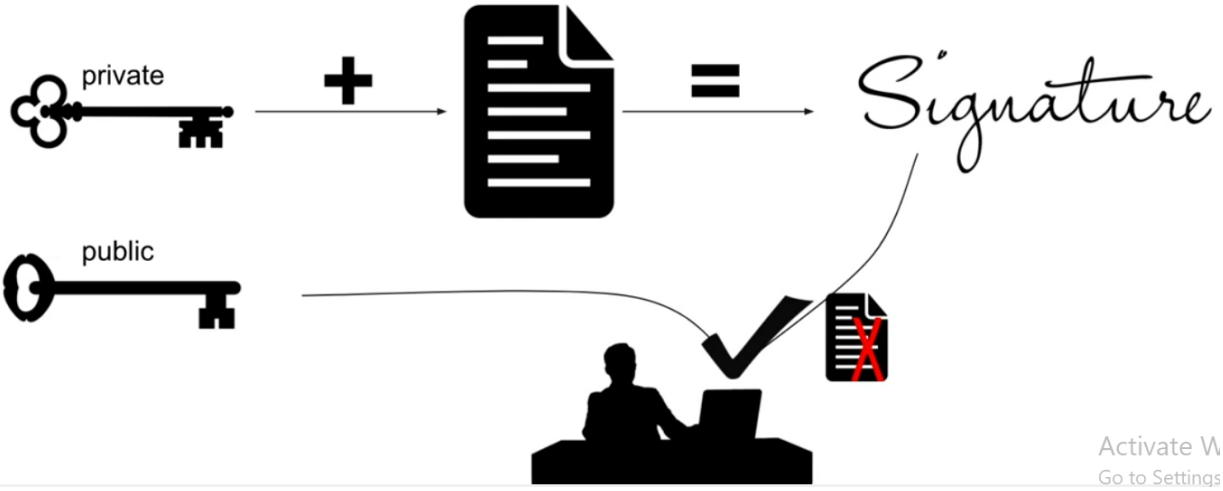
- Wallets store the balance of an individual.
- They store an individual's keys.

Private key	Public key
Used to generate signatures.	Used to verify signatures. Also the public address.



BlockChain

Digital Signatures



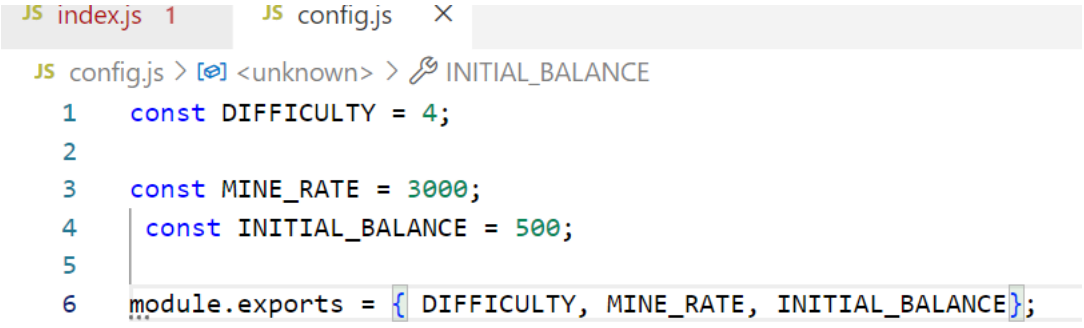
Blockchain-powered Cryptocurrencies

- Contain wallet objects.
- Keys for digital signatures and verification.
- Have transactions objects to represent currency exchange.

BlockChain

Create Wallet:

```
wallet > JS index.js > Wallet

1   const { INITIAL_BALANCE } = require('../config');
2
3
4   class Wallet {
5       constructor(){
6           this.balance = ;
7           this.keyPair = null;
8           this.publickey = null;
9       }
10
11      toString(){
12          return `Wallet -
13          publicKey: ${this.publickey.toString()}
14          balance : ${this.balance}`
15      }
16  }
17
18  module.exports = Wallet;
```



```
JS config.js > <unknown> > INITIAL_BALANCE

1   const DIFFICULTY = 4;
2
3   const MINE_RATE = 3000;
4   const INITIAL_BALANCE = 500;
5
6   module.exports = { DIFFICULTY, MINE_RATE, INITIAL_BALANCE};
```

BlockChain

Chain Util and Key Generation

npm i elliptic --save

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i elliptic --save

added 7 packages, and audited 377 packages in 2s

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

index.js > ...

```
21
22  const Wallet = require('./wallet');
23  const wallet = new Wallet();
24  console.log(wallet.toString());
25
```

wallet > JS index.js > Wallet

```
1   const ChainUtil = require('../chain-util');
2   const { INITIAL_BALANCE } = require('../config');
3
4
5   class Wallet {
6       constructor(){
7           this.balance = INITIAL_BALANCE;
8           this.keyPair = ChainUtil.genKeyPair;
9           this.publicKey = this.keyPair.getPublic().encode('hex');
10      }
11
12      toString(){
13          return `Wallet -
14          publicKey: ${this.publicKey.toString()}
15          balance : ${this.balance}`
16      }
17  }
18
19  module.exports = Wallet;
```

BlockChain

Create a Transaction

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i uuid --save

added 1 package, and audited 378 packages in 819ms

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

wallet > transactions.js > Transaction > constructor

```
1   const ChainUtil = require('../chain-util');
2
3   class Transaction {
4       constructor(){
5           this.id = ChainUtil.id();
6           this.input = null;
7           this.outputs = [];
8       }
9
10      static newTransaction(senderWallet, recipient, amount){
11          const transaction = new this();
12
13          if(amount >senderWallet.balance){
14              console.log(`Amount : ${amount} exceeds balance.`);
15              return;
16          }
17
18
19          transaction.outputs.push(...[
20              {amount: senderWallet.balance = amount, address: senderWallet.publicKey },
21              {amount, address: recipient}
22          ])
23
24          return transaction;
25
26      }
27  }
28
29  module.exports = Transaction;
```

BlockChain

Test the Transaction:

chain-util.js > ChainUtil > genKeyPair

```
const EC = require('elliptic').ec;
const uuidV1 = require('uuid/v1');
const ec = new EC('secp256k1');

class ChainUtil {
    static genKeyPair(){
        return ec.genKeyPair();
    }

    static id() {
        return uuidV1();
    }
}

module.exports = ChainUtil;
```

wallet > JS transaction.test.js > ...

```
const Transaction = require('./transactions');
const Wallet = require('./index');


describe('Transcation', ()=> {
    let transaction, wallet, recipient, amount;

    beforeEach(() => {
        wallet = new Wallet();
        amount = 50;
        recipient = 'r3c1p13nt';
        transactions = Transaction.newTransaction(wallet, recipient,amount);

    });

    it('outputs the `amount` subtracted from the wallet balance', () =>{
        expect(Transaction.outputs.find(output => output.address === wallet.publicKey).amount)
        .toEqual(wallet.balance - amount);

    });

    it('outputs the `amount` added to the recipient', () =>{
        expect(transaction.outputs.find(output.address === recipient).amount)
        .toEqual(amount);
    });
})
```

{} package.json > {} dependencies

```
  "jest":{
    "testEnvironment": "node"
  },
```

wallet > JS transaction.test.js > describe('Transcation') callback > describe('transacting with an amount that exceeds the balance')

```
17
18      it('outputs the `amount` subtracted from the wallet balance', () =>{
19          expect(Transaction.outputs.find(output => output.address === wallet.publicKey).amount)
20          .toEqual(wallet.balance - amount);
21
22      });
23
24      it('outputs the `amount` added to the recipient', () =>{
25          expect(transaction.outputs.find(output.address === recipient).amount)
26          .toEqual(amount);
27      });
28
29      describe('transacting with an amount that exceeds the balance', ()=>{
30          beforeEach(() =>{
31              amount = 50000;
32              transaction = Transaction.newTransaction(wallet, recipient, amount);
33          });
34
35          it('does not create the transaction', () =>{
36              expect(transaction).toEqual(undefined);
37          });
38      });
39  })
```

Sign a Transaction:

BlockChain

JS chain-util.js > ChainUtil > hash

```
1   const EC = require('elliptic').ec;
2   const SHA256 = require('crypto-js/sha256');
3   const { v1: uuidV1 } = require('uuid');
4   //const uuidV1 = require('uuid/v1');
5
6
7   // const uuidV1 = require('uuid');
8
9   //    console.log(uuidV1.v1());
10
11
12  // var uuid = require('uuid');
13  // const uuidV1 = require('uuid/v1');
14
15  const ec = new EC('secp256k1');
16
17  class ChainUtil {
18      static genKeyPair(){
19          return ec.genKeyPair();
20      }
21
22      static id() {
23          return uuidV1();
24      }
25
26      static hash(data){
27          return SHA256(JSON.stringify(data)).toString();
28      }
29  }
```

BlockChain

wallet > JS index.js > Wallet > sign

```
1   const ChainUtil = require('../chain-util');
2   const { INITIAL_BALANCE } = require('../config');
3
4
5   class Wallet {
6       constructor(){
7           this.balance = INITIAL_BALANCE;
8           this.keyPair = ChainUtil.genKeyPair;
9           this.publicKey = this.keyPair.getPublic().encode('hex');
10      }
11
12      toString(){
13          return `Wallet -
14          publicKey: ${this.publicKey.toString()}
15          balance : ${this.balance}`
16      }
17
18      sign(dataHash){
19          return this.keyPair.sign(dataHash);
20      }
21  }
22
23  module.exports = Wallet;
24
25  //getPublic()
```

BlockChain

wallet > transactions.js > Transaction > newTransaction

```
16          }
17
18
19          transaction.outputs.push(...[
20              {amount: senderWallet.balance = amount, address: senderWallet.publicKey },
21              {amount, address: recipient}
22          ])
23
24          Transaction.signTransaction(transaction, senderWallet);
25
26          return transaction;
27
28      }
29
30      static signTransaction(transaction, senderWallet){
31          transaction.input ={
32              timestamp: Date.now(),
33              amount: senderWallet.balance,
34              address: senderWallet.publicKey,
35              signature: senderWallet.sign(ChainUtil.hash(transaction.outputs))
36          }
37      }
38  }
```

Test the Transaction Input:

wallet > transaction.test.js > describe('Transcation') callback > it('inputs the balance of the

```
28
29      it('inputs the balance of the wallet', ()=>{
30          expect(transaction.input.amount).toEqual(wallet.balance);
31      })
```

Verify Transactions:

chain-util.js > ChainUtil > verifySignature

```
29
30      static verifySignature(publicKey, signature, dataHash){
31          return ec,keyFromPublic(publicKey, 'hex').verify(dataHash, signature);
32      }
33  }
```

BlockChain

wallet > JS transactions.js > Transaction > verifyTransaction

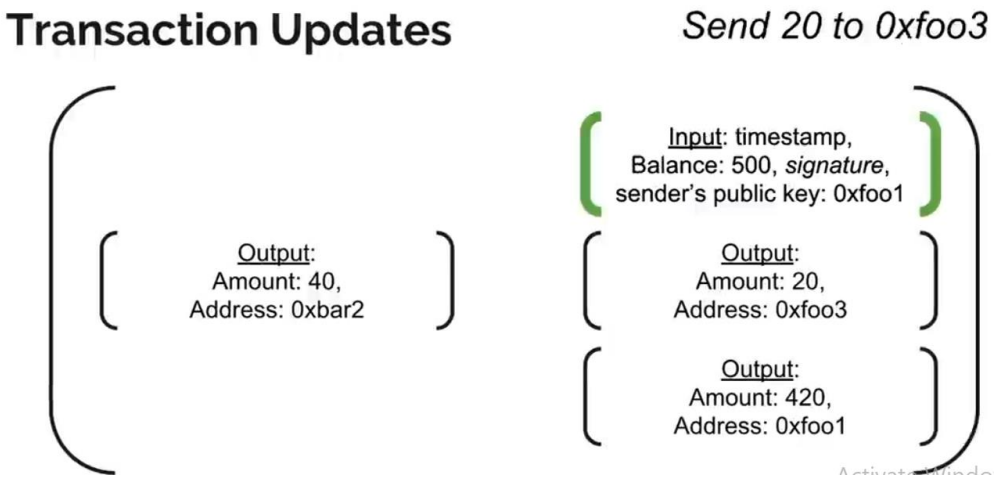
```
39
40      static verifyTransact any ransaction){
41          return ChainUtil.verigySignature(
42              transaction.input.address,
43              transaction.input.signature,
44              ChainUtil.hash(transaction.outputs)
45          );
46      }
```

Test Transaction Verification:

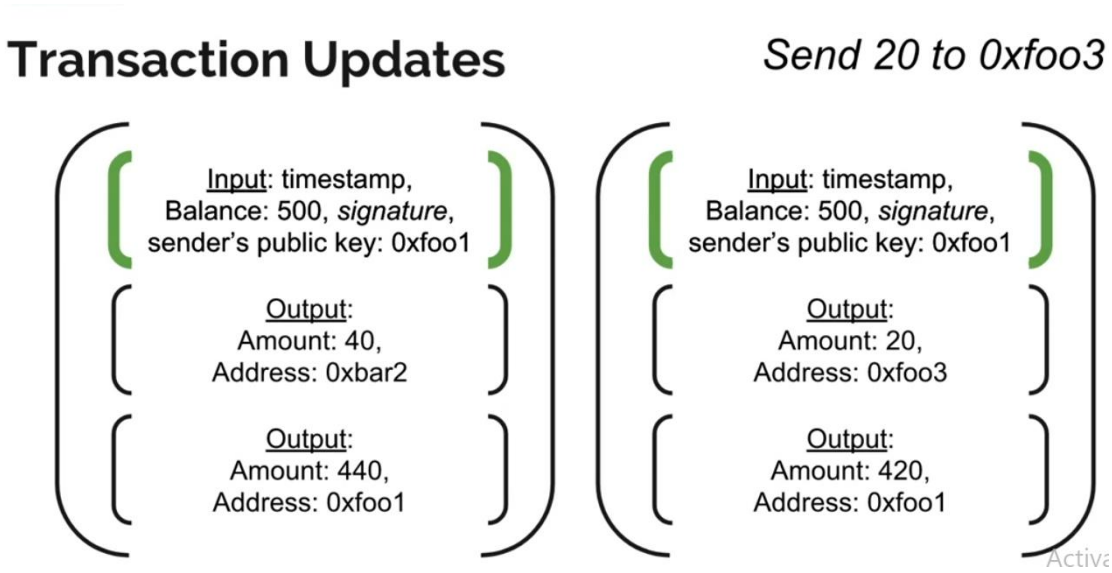
wallet > JS transaction.test.js > describe('Transcation') callback

```
33
34      it('validates a valid transaction', () => {
35          expect(Transaction.verifyTransaction(transaction)).toBe(true);
36      });
37
38      it('invalidates a corrupt transaction', () =>{
39          transaction.outputs[0].amount = 50000;
40          expect(Transaction.verifyTransaction(transaction)).toBe(false);
41      });
```

Transaction Updates:



BlockChain



wallet > JS transactions.js > Transaction > update

```
9
10      update(senderWallet, recipient, amount){
11          const senderOutput = this. outputs.find(output => output.address == senderWallet.publicKey);
12
13          if (amount > senderOutput.amount) {
14              console.log(`Amount : ${amount} exceeds balance.`);
15              return ;
16          }
17
18          senderOutput.amount = senderOutput.amount - amount;
19          this.outputs/push({amount, address: recipient});
20          Transaction.signTransaction(this, senderWallet);
21
22          return this;
23
24      }
```

Test Transaction Updates:

BlockChain

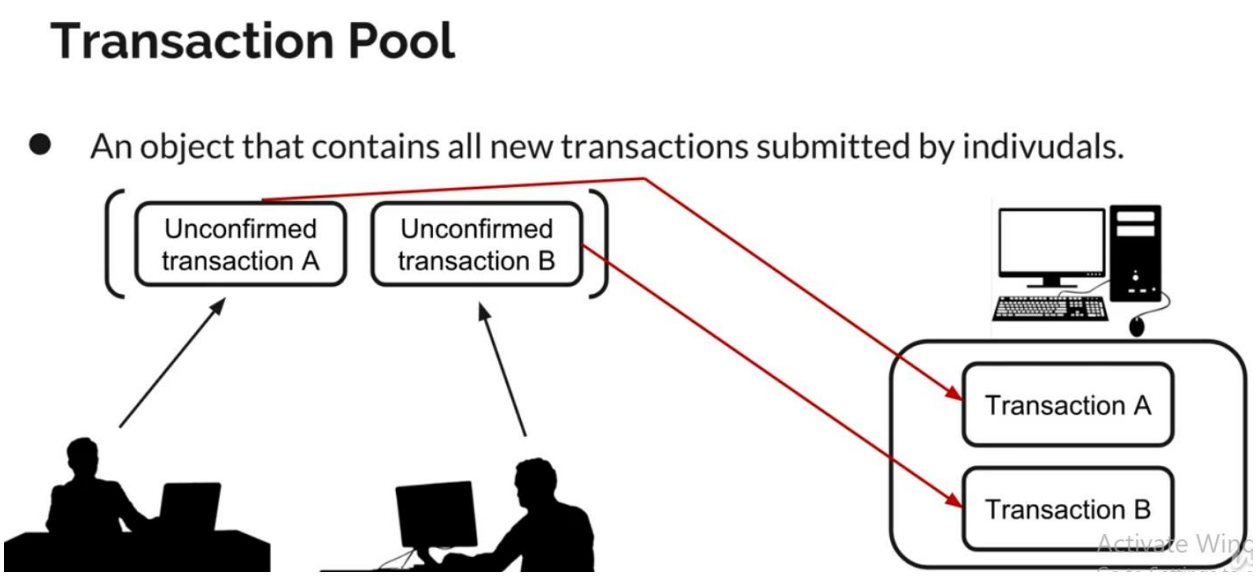
wallet > JS transaction.test.js > ...

```
54      describe('and updating a transaction', () =>{
55          let nextAmount, nextRecipient;
56
57          beforeEach(() =>{
58              nextAmount = 20;
59              nextRecipient = 'n3xt-4ddr355';
60              transaction = transaction.update(wallet, nextRecipient, nextAmount);
61          });
62
63          it('subtrancts the next amount from the senders output', ()=>{
64              expect(transaction.outputs.find(output => output.address === wallet.publicKey).amount)
65              .toEqual(wallet.balance - amount - nextAmount);
66
67          });
68
69          it('outputs an amount for the next recipient', ()=>{
70              expect(transaction.outputs.find(output => output.address == nextRecipient).amount)
71              .toEqual(nextAmount);
72          });
73
74      });
75  });
```

BlockChain

Collect Transactions in a Pool:

Transaction Pool:



Transaction Pool - Add Transaction:

wallet > JS transaction-pool.js > ...

```
1   class TransactionPool {
2       constructor(){
3           this.transactions = [];
4       }
5
6       updateOrAddTransaction(transaction){
7           let transactionWithId = this.transactions.find(t => t.id === transaction.id);
8
9           if(transactionWithId) {
10              this.transactions[this.transactions.indexOf(transactionWithId)] = transaction;
11          } else {
12              this.transactions.push(transaction);
13          }
14      }
15  }
16
17  module.exports = TransactionPool;
```

BlockChain

Test the Transaction Pool:

wallet > JS transaction-pool.test.js > ...

```
const TransactionPool = require('./transaction-pool');
const Transaction = require('./transaction');
const Wallet = require('./index');

describe('TransactionPool', ()=>{
    let tp, wallet, transaction;

    beforeEach(() =>{
        tp = new TransactionPool();
        wallet = new Wallet();
        transaction = Transaction.newTransaction(wallet, 'r4nd-4dr355', 30);
        tp.updateOrAddTransaction(transaction);
    });

    it('adds a transaction to the pool', () =>{
        exportAllDeclaration(tp.transactions.find(t=>t.id === transaction.id)).toEqual(transaction);
    });

    it('updates a transaction inthe pool', ()=>{
        const oldTransaction = JSON.stringify(transaction);
        const newTransaction = transaction.update(wallet, 'foo-4ddr355', 40);
        tp.updateOrAddTransaction(newTransaction);

        expect(JSON.stringify(tp.transactions.find(t => t.id === newTransaction.id)))
        .not.toEqual(oldTransaction);

    });

});
```

Create Transactions with the Wallet:

JS index.js > ...

```
createTransaction(recipient, amount, TransactionPool){
    if(amount > this.balance){
        console.log(`Amount: ${amount} exceeds current balance: ${this.balance}`);
        return;
    }

    let transaction = TransactionPool.existingTransaction(this.publicKey);

    if(transaction){
        transaction.update(this, recipient, amount);
    }else{
        transaction= Transaction.newTransaction(this, recipient, amount);
        TransactionPool.updateOrAddTransaction(transaction);
    }
    return transaction;

}
```

BlockChain

wallet > JS transaction-pool.js > TransactionPool > existingTransaction

```
15
16      existingTransaction(address){
17          return this.transactions.find(t => t.input.address === address);
18
19      }
20  }
```

Get Transactions:

wallet > JS index.js > Transaction

```
2   const Transaction = require('./transactions');
```

app > JS index.js > app.get('/transactions') callback

```
8   const Wallet = require('../wallet');
9   const TransactionPool = require('../wallet/transaction-pool');
10
```

app > JS index.js > app.get('/transactions') callback

```
35
36  app.get('/transactions', (req, res) =>{
37      res.json(tp.transactions);
38
39  })
```

Post Transactions:

JS transaction-pool.js | JS index.js app | JS index.js wallet | JS transaction-pool.test.js

app > JS index.js > ...

```
41  app.post('/transact', (req, res) => {
42      const { recipient, amount} =req.body;
43      const transaction = wallet.createTransaction(recipient, amount, tp);
44      res.redirect('/transactions');
45  });
```

BlockChain

Add the Transaction Pool to the Peer to peer Server:

app > JS p2p-server.js > P2pServer

```
55    sendChains(socket){
56        socket.send(JSON.stringify(this.blockchain.chain));
57    }
58    sendTransaction(socket, transaction){
59    socket.send(JSON.stringify({type : MESSAGE_TYPES.transaction, transaction}));
60    }
```

app > JS p2p-server.js > P2pServer

```
5    const MESSAGE_TYPES = {
6        chain: 'CHAIN',
7        transaction: 'TRANSACTION'
8    };
```

app > JS p2p-server.js > P2pServer

```
11   class P2pServer {
12       constructor(blockchain, transactionPool){
13           this.blockchain = blockchain;
14           this.transactionPool = transactionPool;
15           this.sockets = [];
16
17       }
```

```
14   const app = express();
15   const bc = new Blockchain();
16   const wallet = new Wallet();
17   const tp = new TransactionPool();
```

BlockChain

Handle Transaction Messages in the Peer to peer Server:

app > JS p2p-server.js > P2pServer > messageHandler > socket.on('message') callback

```
47      messageHandler(socket){
48          socket.on('message', message =>{
49              const data = JSON.parse(message);
50              switch(data.type){
51                  case MESSAGE_TYPES.chain:
52                      this.blockchain.replaceChain(data.chain);
53                      break;
54                  case MESSAGE_TYPES.transaction:
55                      this.transactionPool.updateOrAddTransaction(data.transaction);
56                      break;
57              }
58
59          } );
60      }
```

app > JS index.js > app.post('/transact') callback

```
40
41  app.post('/transact', (req, res) => {
42      const { recipient, amount} =req.body;
43      const transaction = wallet.createTransaction(recipient, amount, tp);
44      p2pServer.broadcastTransaction(transaction);
45      res.redirect('/transactions');
46  });
```

Public Key Endpoint:

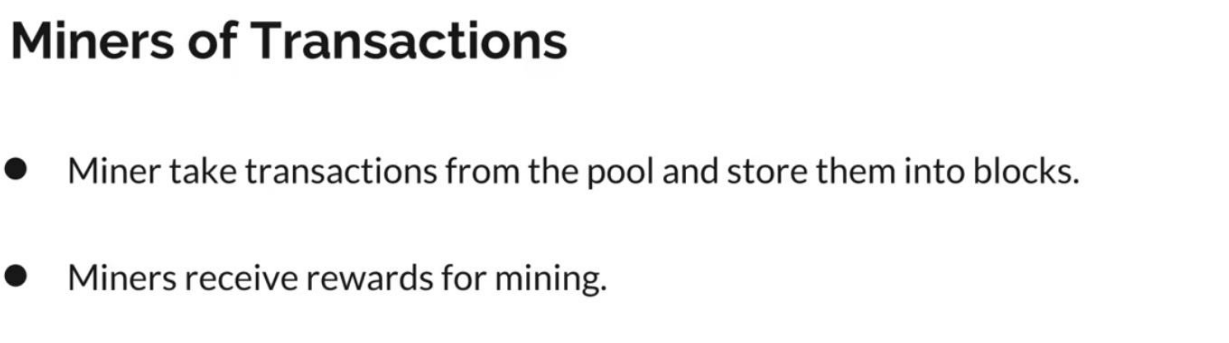
app > JS index.js > app.get('/public-key') callback

```
47
48  app.get('/public-key', (req, res) =>{
49      res.json({publicKey: wallet.publicKey});
50  });
```

BlockChain

Mine Transactions in a Block:

Miners of Transactions:



Create the Miner Class:

app > JS miner.js > ...

```
1   class Miner {
2       constructor(blockchain, transactionPool, wallet, p2pServer){
3           this.blockchain = blockchain;
4           this.transactionPool = transactionPool;
5           this.wallet = wallet;
6           this.p2pServer = p2pServer;
7       }
8
9       mine(){
10          const validTransactions = this.transactionPool.validTransactions();
11          // include a reward for the miner
12
13          //create a block consisting of the valid transactions
14          //synchronize chains in the peer-to-peer server
15          //clear the transaction pool
16          //broadcast to every miner to clear their transaction pools
17      }
18  }
19
20  module.exports(Miner);
```

BlockChain

Grab Valid Transactions:

wallet > JS transaction-pool.js > Transaction

```
23      validTransactions(){
24          return this.transactions.filter(transaction =>{
25              const outputTotal = transaction.outputs.reduce((total, output) =>{
26                  return total + output.amount;
27              }, 0);
28
29              if(transaction.input.amount !== outputTotal){
30                  console.log(`Invalid transaction from ${transaction.input.address}.`);
31                  return;
32              }
33
34              if(!Transaction.verifyTransaction(transaction)){
35                  console.log(`Invalid signature from ${transaction.input.address}.`);
36                  return;
37              }
38
39              return transaction;
40          });
41      }
```

Test Valid Transactions:

wallet > JS index.js > Wallet > blockchainWallet

```
40
41      static blockchainWallet(){
42          const blockchainWallet = new this();
43          blockchainWallet.address = 'blockchain-wallet';
44          return blockchainWallet;
45      }
46  }
```

wallet > JS transaction-pool.js > TransactionPool

```
7
8       updateOrAddTransaction(transaction){
9           let transactionWithId = this.transactions.find(t => t.id === transaction.id);
10
11          if(transactionWithId) {
12              this.transactions[this.transactions.indexOf(transactionWithId)] = transaction
13          } else {
14              this.transactions.push(transaction);
15          }
16      }
```

BlockChain

Reward Transactions:

config.js > ...

```
1   const DIFFICULTY = 4;
2
3   const MINE_RATE = 3000;
4   const INITIAL_BALANCE = 500;
5   const MINING_REWARD = 50;
6
```

Test Reward Transactions:

wallet > transaction.test.js > ...

```
77      describe('creating a reward transaction', () => {
78          beforeEach(() =>{
79              transaction = Transaction.rewardTransaction(wallet, Wallet.blockchainWallet());
80          });
81
82          it(`reward the miner's wallet`, () =>{
83              expect(transaction.outputs.find(output => output.address === wallet.publicKey).amount)
84              .toEqual(MINING_REWARD);
85          });
86
87      });
88  });
```

wallet > transaction.test.js > ...

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

BlockChain

Reward Valid and Clear Transactions:

wallet > JS transaction.test.js > describe('Transcation') callback > beforeEach() callback

```
77      describe('creating a reward transaction', () => {
78          beforeEach(() =>{
79              transaction = Transaction.rewardTransaction(wallet, Wallet.blockchainWallet());
80          });
81
82          it(`reward the miner's wallet`, () =>{
83              expect(transaction.outputs.find(output => output.address === wallet.publicKey).amount)
84              .toEqual(MINING_REWARD);
85          });
86
87      });
88  });
```

Broadcast Clear Transactions:

app > JS index.js > app.get('/mine-transactions') callback

```
50  app.get('/mine-transactions', (req, res) =>{
51      const block = miner.mine();
52      console.log(`New block added: ${block.toString()}`);
53      res.redirect('/blocks');
54  })
```

app > JS index.js > app.get('/mine-transactions') callback

```
21  const  miner = new Miner(bc, tp, wallet, p2pServer);
22
```

app > JS index.js > app.get('/mine-transactions') callback

```
10  const Miner = require('./miner');
11
```

Mine Transactions Endpoint:

BlockChain

app > JS miner.js > Miner > mine

```
10
11      mine(){
12          const validTransactions = this.transactionPool.validTransactions();
13          // include a reward for the miner
14
15          //create a block consisting of the valid transactions
16          //synchronize chains in the peer-to-peer server
17          //clear the transaction pool
18          //broadcast to every miner to clear their transaction pools
19
20          validTransactions.push(
21              Transaction.rewardTransaction(this.wallet, Wallet.blockchainWallet())
22          );
23          const block = this.blockchain.addBlock(validTransactions);
24          this.p2pServer.syncChains();
25          this.transactionPool.clear();
26          this.p2pServer.broadcastClearTransactions();
27
28          return block;
29      }
30  }
```

app > JS index.js > app.get('/mine-transactions') callback

```
42
43  app.post('/transact', (req, res) => {
44      const { recipient, amount} =req.body;
45      const transaction = wallet.createTransaction(recipient, amount, tp);
46      p2pServer.broadcastTransaction(transaction);
47      res.redirect('/transactions');
48  });
49
```

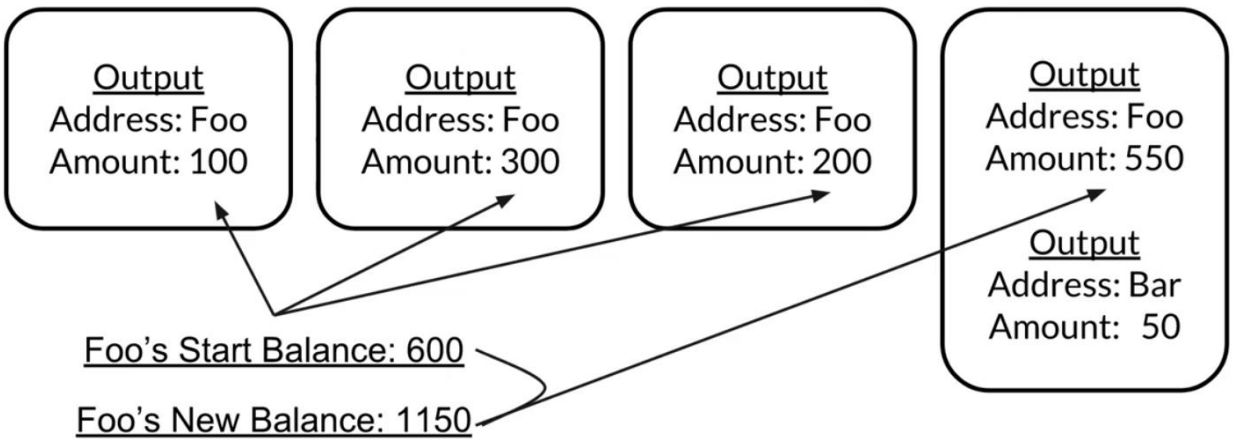
BlockChain

The Nuance of Wallet Balance:

The Nuance of Wallet Balance

- Balance = all output amounts that belong to a user.
- Update the balance at least before each transaction.
- All outputs *since* a user's most recent transaction.
- Possible double counting of transactions within the blockchain history.

Foo wants to send 50 to Bar



BlockChain

Calculate the Wallet Balance:

index.js

wallet > index.js > Wallet > calculateBalance > transactions.forEach() callback

```
41      calculateBalance(blockchain){
42          let balance = this.balance;
43          let transactions = [];
44          blockchain.chain.forEach(block => blockchain.data.forEach(transaction => {
45              transaction.push(transaction);
46          }));
47
48          const walletInputs = transactions
49          .filter(transaction => transactions.input.address === this.publicKey);
50
51          let startTime = 0;
52              if(walletInputs.length > 0){
53          const recentInputT = walletInputs.reduce(
54              (prev, current) => prev.input.timestamp > current.input.timestamp ? prev : current
55
56          );
57
58          balance = recentInputT.outputs.find(output => output.address === this.publicKey).amount;
59          startTime = recentInputT.input.timestamp;
60      }
61
62      transactions.forEach(transaction => {
63          if(transactions.input.timestamp > startTime){
64              transaction.outputs.find(output =>{
65                  if(output.address === this.publicKey){
66                      balance += output.amount;
67                  }
68              });
69          }
    });
}
```

BlockChain

Calculate the Balance during each Transaction:

wallet > JS index.test.js > describe('Wallet') callback > describe('creating a transaction') callback > describe('and doing the same transaction') callback

```
1  const Wallet = require('./index');
2  const TransactionPool = require('./transaction-pool');
3  const Blockchain = require('../blockchain');
4
5  describe('Wallet', ()=> {
6      let wallet, tp, bc;
7
8      beforeEach(() =>{
9          wallet = new Wallet;
10         tp = new TransactionPool();
11         bc =new Blockchain();
12     });
13
14     describe('creating a transaction', ()=>{
15         let transaction, sendAmount, recipient;
16
17         beforeEach(() => {
18             sendAmount = 50;
19             recipient = ' r4nd0n-4ddr355';
20             transaction = wallet.createTransaction(recipient, sendAmount,bc, tp);
21         });
22
23         describe('and doing the same transaction', () =>{
24             beforeEach(() =>{
25                 wallet.createTransaction(recipient, sendAmount, bc, tp);
26             });
27             it('doubles the sendAmount subtrancted from the wallet balance', ()=>{
28
29             });
```


wallet > JS transaction-pool.test.js > describe('TransactionPool') callback

```
31
32     it('clears transactions', ()=>{
33         tp.clear();
34         expect((tp.transactions).toEqual([]));
35     });
36
37     describe('mixing valid and corrupt transactions', () =>{
38         let validTransactions;
39
40         beforeEach(() =>{
41             validTransactions = [...tp.transactions];
42             for (let i = 0; i<6 ; i++){
43                 wallet = new Wallet();
44                 transaction = wallet.createTransaction('r4nd-4ddr355', 30,bc, tp);
45                 if(i%2 ==0){
46                     transaction.input.amount = 99999;
47                 } else {
48                     validTransactions.push(transaction);
49                 }
50             }
51
52         });
53     });
```

Test Balance Calculation:

wallet > JS transaction-pool.test.js > Transaction

```
31
32      it('clears transactions', ()=>{
33          tp.clear();
34          expect((tp.transactions).toEqual([]));
35      });
36
37      describe('mixing valid and corrupt transactions', () =>{
38          let validTransactions;
39
40          beforeEach(() =>{
41              validTransactions = [...tp.transactions];
42              for (let i = 0; i<6 ; i++){
43                  wallet = new Wallet();
44                  transaction = wallet.createTransaction('r4nd-4ddr355', 30,bc, tp);
45                  if(i%2 ==0){
46                      transaction.input.amount = 99999;
47                  } else {
48                      validTransactions.push(transaction);
49                  }
50              }
51
52          });
53      });
```

wallet > JS transaction-pool.test.js > Transaction

```
const Transaction = require('./transactions');
const Wallet = require('./index');
const Blockchain = require('../Blockchain');

describe('TransactionPool', ()=>{
    let tp, wallet, transaction, bc;

    beforeEach(() =>{
        tp = new TransactionPool();
        wallet = new Wallet();
        bc = new Blockchain();
        transaction = Transaction.newTransaction(wallet, 'r4nd-4dr355', 30);
        tp.updateOrAddTransaction(transaction);

    });

    it('adds a transaction to the pool', () =>{
        expect(tp.transactions.find(t=>t.id === transaction.id)).toEqual(transaction);
    });

    it('updates a transaction inthe pool', ()=>{
        const oldTransaction = JSON.stringify(transaction);
        const newTransaction = transaction.update(wallet, 'foo-4ddr355', 40);
        tp.updateOrAddTransaction(newTransaction);

        expect(JSON.stringify(tp.transactions.find(t => t.id === newTransaction.id)))
        .not.toEqual(oldTransaction);

    });
```

The Cryptocurrency in Action: