UDACITY

Back to Blockchain Developer

# Private Blockchain

REVIEW

CODE REVIEW 7

HISTORY

## Requires Changes

1 SPECIFICATION REQUIRES CHANGES

## Very close!

Excellent start! Your code is well organized and I liked how you separate it into different files according to their functions. Very neat! You just have to fix a few things in your validateChain routine. Please see the rubric item and the code review for details and possible solutions. I highly recommend that you test it out first before resubmitting. You can use this snippet in your Node REPL to corrupt a block and reinsert it in the same position:

```
bc.getBlock(2).then((block) => {     // where bc is a Blockchain instance
 and you've added at least 2 blocks
    block.body = "error";
    leveldb.addBlock(2, JSON.stringify(block));
```

```
})
```

If you run validateChain(), it should show an error in Block #2. Do this for multiple blocks and examine results. That's it. All the best for the next submission! You're almost done!

## Configure LevelDB to persist dataset

**SimpleChain.js includes the Node.js level library and configured to persist data within the project directory.**

## Modify simpleChain.js functions to persist data with LevelDB

**addBlock(newBlock) includes a method to store newBlock within LevelDB**

Works great!

```
> bc.addBlock(new Block("test 1"));
> Previous block hash: 0c60dc81ba536d02dbe692c2a64c99ef682aedeaeeb50354
f8cb9db2c7a8627e
New block hash: 4a27b73f0855bfb2b58fd548272f1db2196adbaea84e03570283e1e
3b3b6600f
Added block: 1
```

**Genesis block persist as the first block in the blockchain using LevelDB.**

**Additionally, when adding a new block to the chain, code checks if a Genesis block already exists. If not, one is created before adding the a block.**

Persisted and added only once!

```
> let bc = new Blockchain();
```

```
> New block hash: 0c60dc81ba536d02dbe692c2a64c99ef682aedeaeeb50354f8cb9
db2c7a8627e
Added block: 0
Genesis block added
```

## Modify validate functions

**validateBlock() function to validate a block stored within levelDB**

Detecting valid and invalid blocks!

```
> bc.validateBlock(2).then(value => console.log(value));
> Block 2 hash invalid:
0c4c672d0d1338778a20e5f1ab6683c1e827580de4ae795556726c05a27c7551<>2ed14
f8a72becebba0a9f669cb4a6276bb954f35288b3ecb9099660e0edeca0b
false
```

**validateChain() function to validate blockchain stored within levelDB**

There are some problems with the implementation. Here is the scenario I ran into when I have a chain height = 4 with blocks 2 and 4 corrupted:

```
> bc.validateChain();
> No errors detected                // the errorLog contents are checke
d immediately. But it should be the last step!
No errors detected
No errors detected
No errors detected
No errors detected                   // errorLog contents is checked 4
 times. But it should only be one time.
Block 0 validation confirmed             // the validateBlock() calls r
esolved before the errorLog is populated
```

```
Block 1 validation confirmed
Block 2 hash invalid:
effd81d74364c4bc481b0ef84c9c754f40bbe949316626eaaa108e12170a82ef<>0824b
d1d7a1a563b532df1e01b18a9cda469f23d61205fd4376eb16b6d69a573
Block 3 validation confirmed
Block 4 hash invalid:
61aee58f7157c431ef083eba7e907c878b2081c3d9e5cedaa9948aa76215672c<>6c5c1
f3c5063c45b9d06f12df6579f185d10c927fd901c5a45aeb40f5ddbad0f
```

Please see code review for tips to get around these issues.

## Modify getBlock() function

**getBlock() function retrieves a block by block heigh within the LevelDB chain.**

Gets blocks as JSON:

```
> bc.getBlock(2).then(value => console.log(value));
> { hash:
   '0c4c672d0d1338778a20e5f1ab6683c1e827580de4ae795556726c05a27c7551',
  height: 2,
  body: 'error',
  time: '1537039620',
  previousBlockHash:
   '4a27b73f0855bfb2b58fd548272f1db2196adbaea84e03570283e1e3b3b6600f' }
```

## Modify getBlockHeight() function

**getBlockHeight() function retrieves current block height within the LevelDB chain.**

Returning the correct height

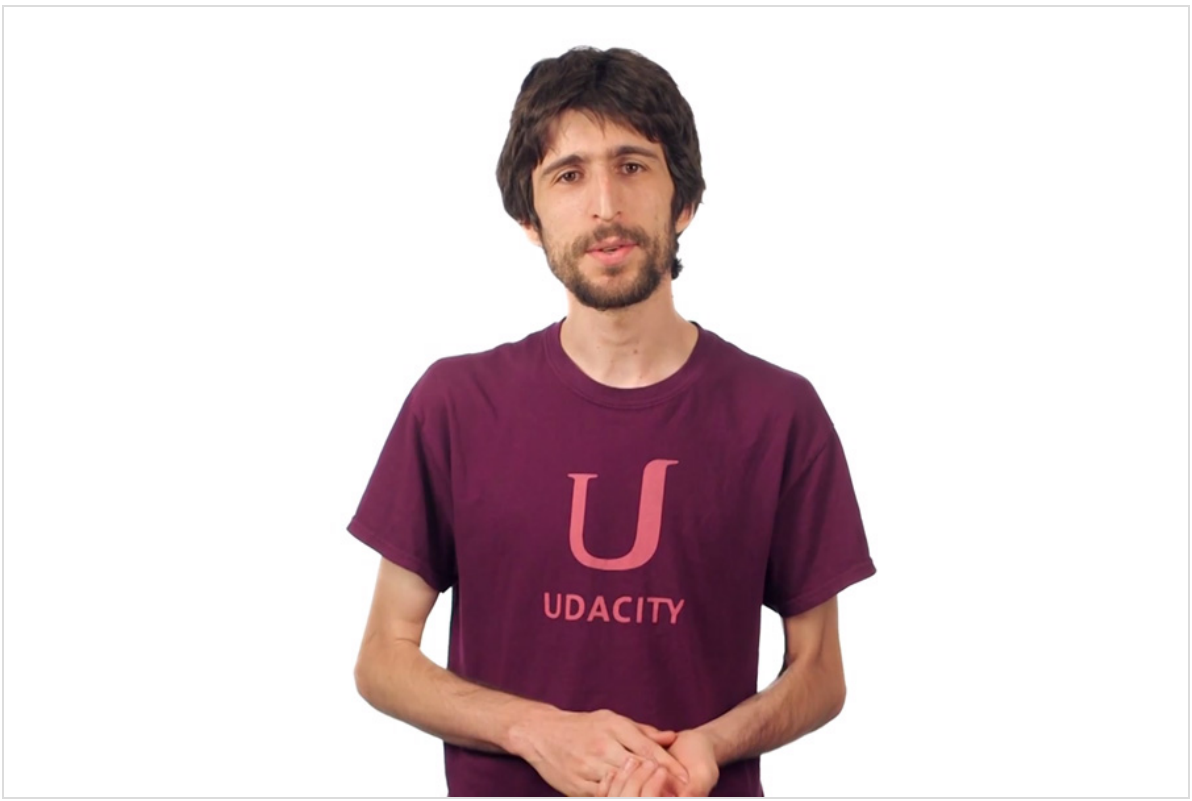
```
> bc.getBlockHeight().then(value => console.log(value));
```

> 4

RESUBMIT

DOWNLOAD PROJECT

7 CODE REVIEW COMMENTS ›




## Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review