Lecture 04:
Testing and Tokens

Nick Zoghb

BLOCKCHAIN
AT BERKELEY

LECTURE OUTLINE

1. TESTING.JS
2. TESTING SMART CONTRACTS
3. BEST PRACTICES WHEN WRITING TESTS
4. TOKENS
5. THE ERC20 STANDARD

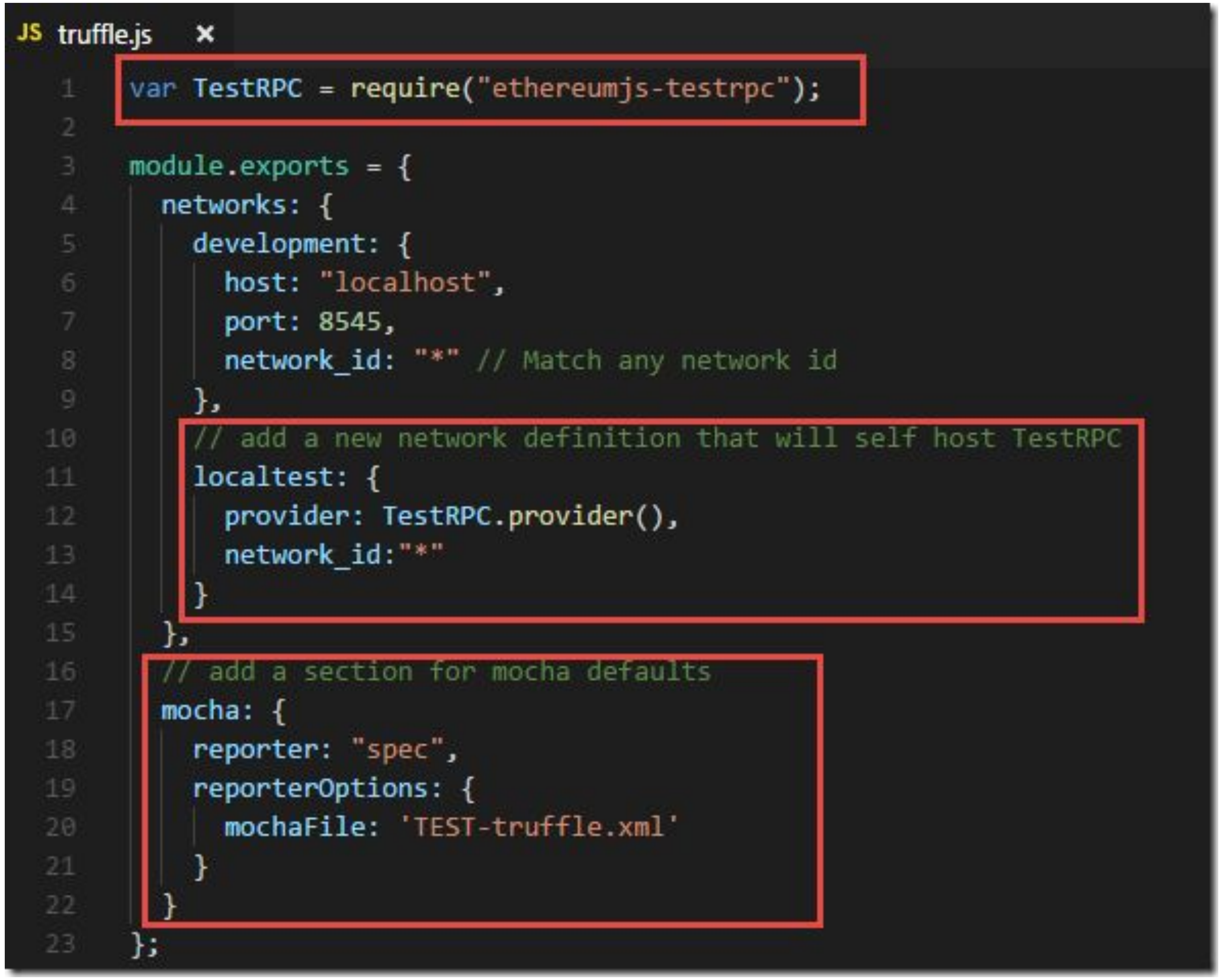
1

TESTING.JS

1.1 JAVASCRIPT.JS.JS

TRUFFLE.JS

IT’S BEEN THERE ALL ALONG



```
var TestRPC = require("ethereumjs-testrpc");

module.exports = {
  networks: {
    development: {
      host: "localhost",
      port: 8545,
      network_id: "*" // Match any network id
    },
    // add a new network definition that will self host TestRPC
    localtest: {
      provider: TestRPC.provider(),
      network_id:"*"
    }
  },
  // add a section for mocha defaults
  mocha: {
    reporter: "spec",
    reporterOptions: {
      mochaFile: 'TEST-truffle.xml'
    }
  }
};
```

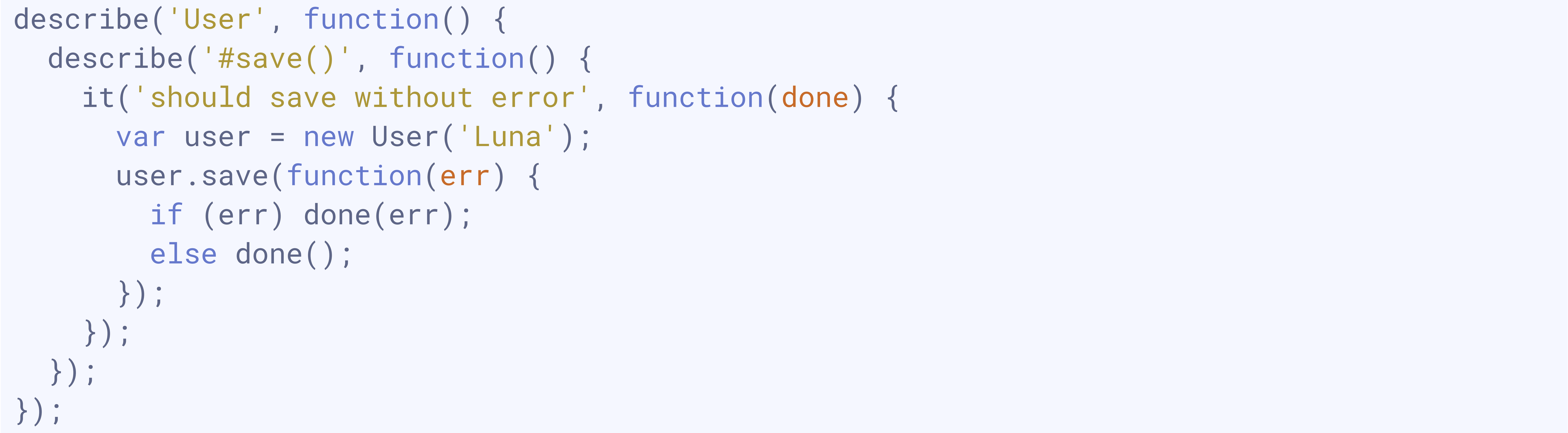
TESTING WITH MOCHA

HOLD THE SUGAR

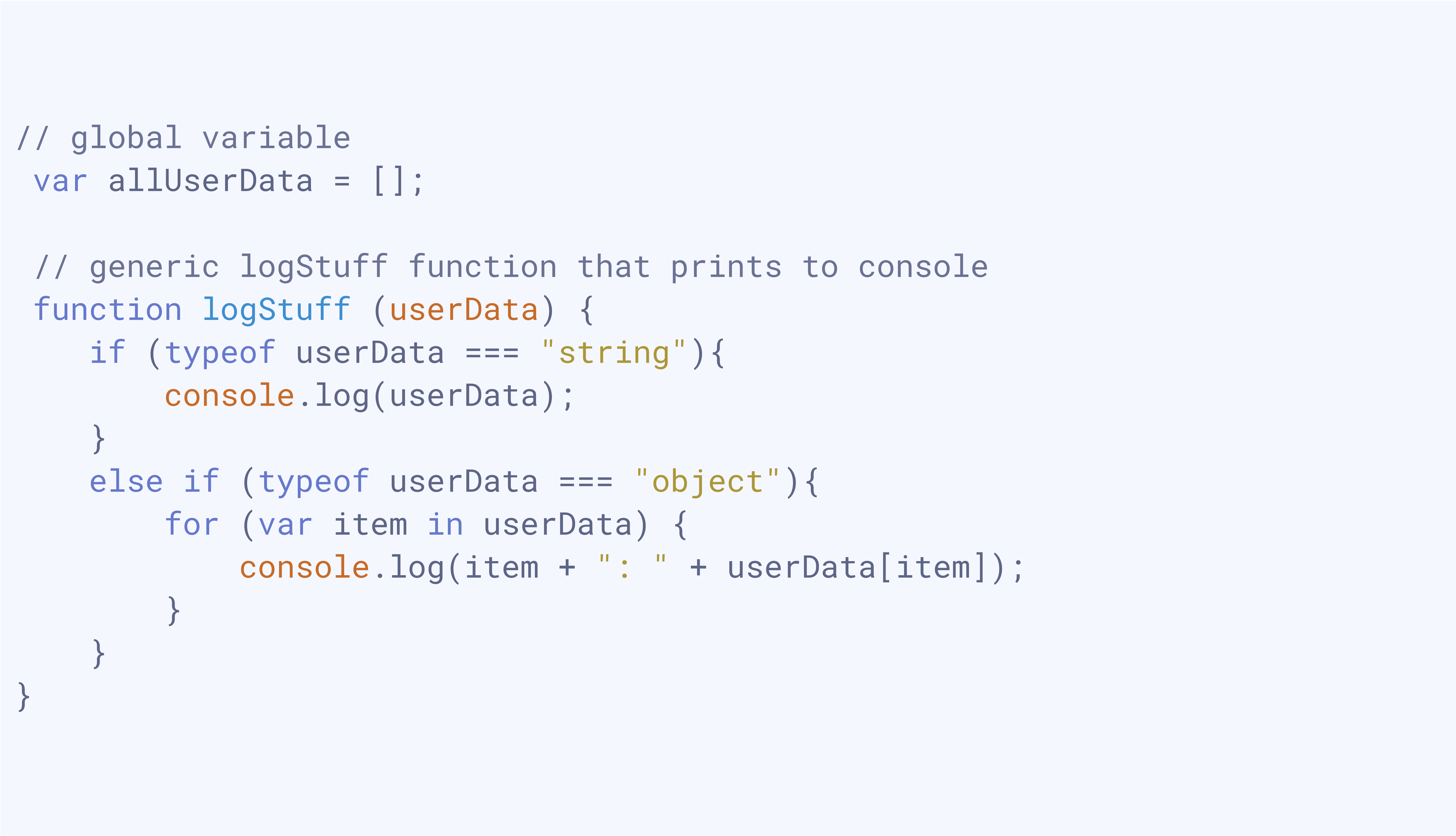
JAVASCRIPT SYNTAX

WHY ARE WE DOING THIS

- How to print? Use `console.log(‘...’)`
- What is `'use strict';`? Enables StrictMode which detects accidental things you may be doing, throws exceptions. Why would you do this? “Fail fast and fail loudly”
- What are callbacks?



```
describe('User', function() {
  describe('#save()', function() {
    it('should save without error', function(done) {
      var user = new User('Luna');
      user.save(function(err) {
        if (err) done(err);
        else done();
      });
    });
  });
});
```



```
// global variable
 var allUserData = [];

 // generic logStuff function that prints to console
 function logStuff (userData) {
    if (typeof userData === "string"){
        console.log(userData);
    }
    else if (typeof userData === "object"){
        for (var item in userData) {
            console.log(item + ": " + userData[item]);
        }
    }
}
```

```
// A function that takes two parameters, the last one a callback
function
 function getInput (options, callback) {
    allUserData.push (options);
    callback (options);
}

 // When we call the getInput function, we pass logStuff as a parameter.
 // So logStuff will be the function that will called back (or executed)
// inside the getInput function
getInput ({name:"Rich", speciality:"JavaScript"}, logStuff);

// name: Rich
 // speciality: JavaScript
```

JAVASCRIPT SYNTAX

WHY

What are keywords?

- **`let`** vs. **`var`**
    - Difference is that **`var`** allows variable access to global scope
    - **`let`** is limited to local scope
- **`const`**
    - Variable cannot change

```
function foo () {
  typeof(bar);
  let bar = “baz”;
  var jumbo = “tron”;
}

foo();

// ReferenceError: can't access
// lexical declaration `bar`
// before initialization

console.log(jumbo);
```

```
const x = 5;
x = 6;

// ERROR - read only
```

MOCHA SYNTAX

ACTUALLY PRETTY NICE

What are keywords?

- **`describe`**, **`it`**
  - describe is used to group tests together by some criteria
  - it is used to define a test case
- **`before`**, **`beforeEach`**, **`after`**, **`afterEach`**
  - These are hooks to run before/after first/each it or describe

(Source)

MOCHA SYNTAX

AN EXAMPLE

```
// Run once before the first test case
describe('earth', function(){
  beforeEach(function(){
    console.log('hello, World!')
  });
  it('sky', function(){
    /** ... */
  });
  describe('sea', function() {
    /** ... */
  });
});
```

```
// beforeEach() can be applied to describe()
describe('earth', function(){
  beforeEach(function(){
    console.log('hello, World!')
  });
  describe('sky', function(){
    it('birds should soar', function(){ /** ... */ });
  });
  describe('sea', function(){
    it('fish should swim', function(){ /** ... */ });
  });
});
```

What's going on here?

a) **`beforeEach()`** runs on each **`it()`** block only

b) **`beforeEach()`** runs on each **`describe()`** block only

c) **`beforeEach()`** runs on both

MOCHA SYNTAX

AN EXAMPLE

```
// Run once before the first test case
describe('earth', function(){
  beforeEach(function(){
    console.log('hello, World!')
  });
  it('sky', function(){
    /** ... */
  });
  describe('sea', function() {
    /** ... */
  });
});
```

```
// beforeEach() can be applied to describe()
describe('earth', function(){
  beforeEach(function(){
    console.log('hello, World!')
  });
  describe('sky', function(){
    it('birds should soar', function(){ /** ... */ });
  });
  describe('sea', function(){
    it('fish should swim', function(){ /** ... */ });
  });
});
```

What's going on here?

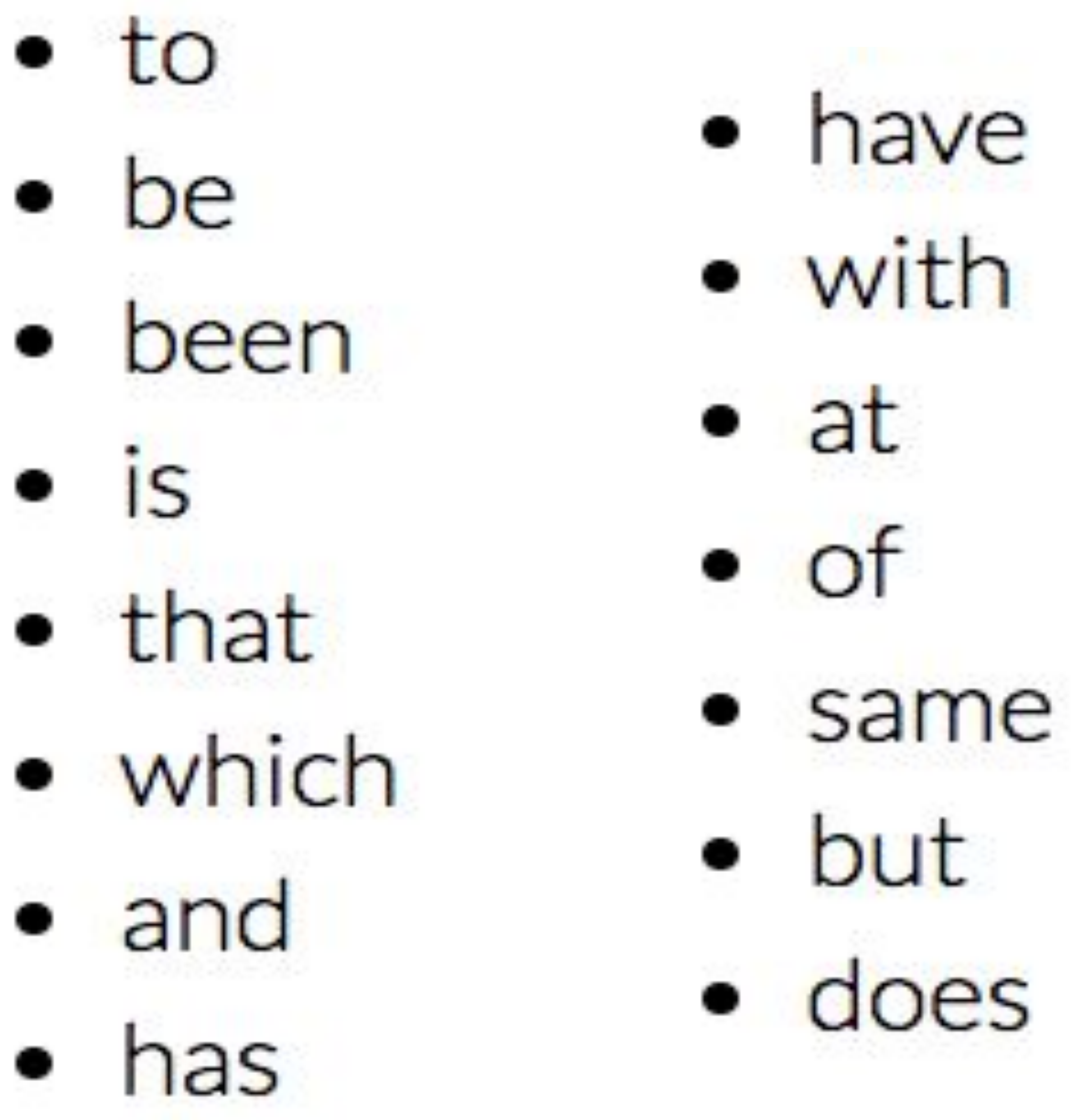
a) **`beforeEach()`** runs on each **`it()`** block only (even nested ones)

b) **`beforeEach()`** runs on each **`describe()`** block only

c) **`beforeEach()`** runs on both

CHAI ASSERTIONS

SO.MANY.CHAINED.GETTERS

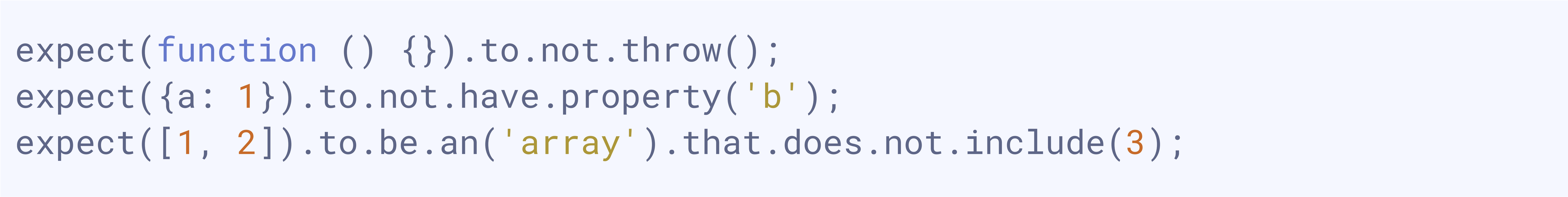


- to
- be
- been
- is
- that
- which
- and
- has
- have
- with
- at
- of
- same
- but
- does

CHAI ASSERTIONS

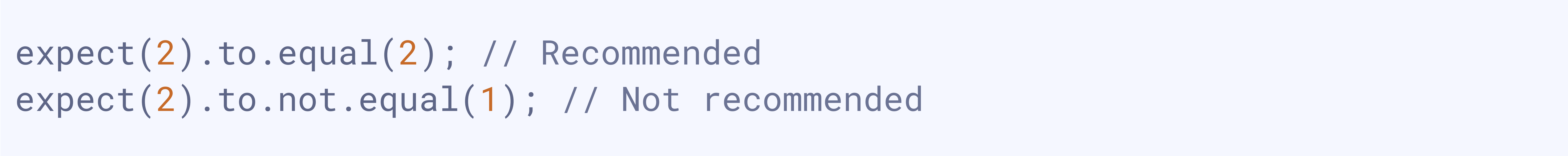
AN EXAMPLE

Take the example of **`.not`**:



```
expect(function () {}).to.not.throw();
expect({a: 1}).to.not.have.property('b');
expect([1, 2]).to.be.an('array').that.does.not.include(3);
```

Please abide by best practices (see more here):

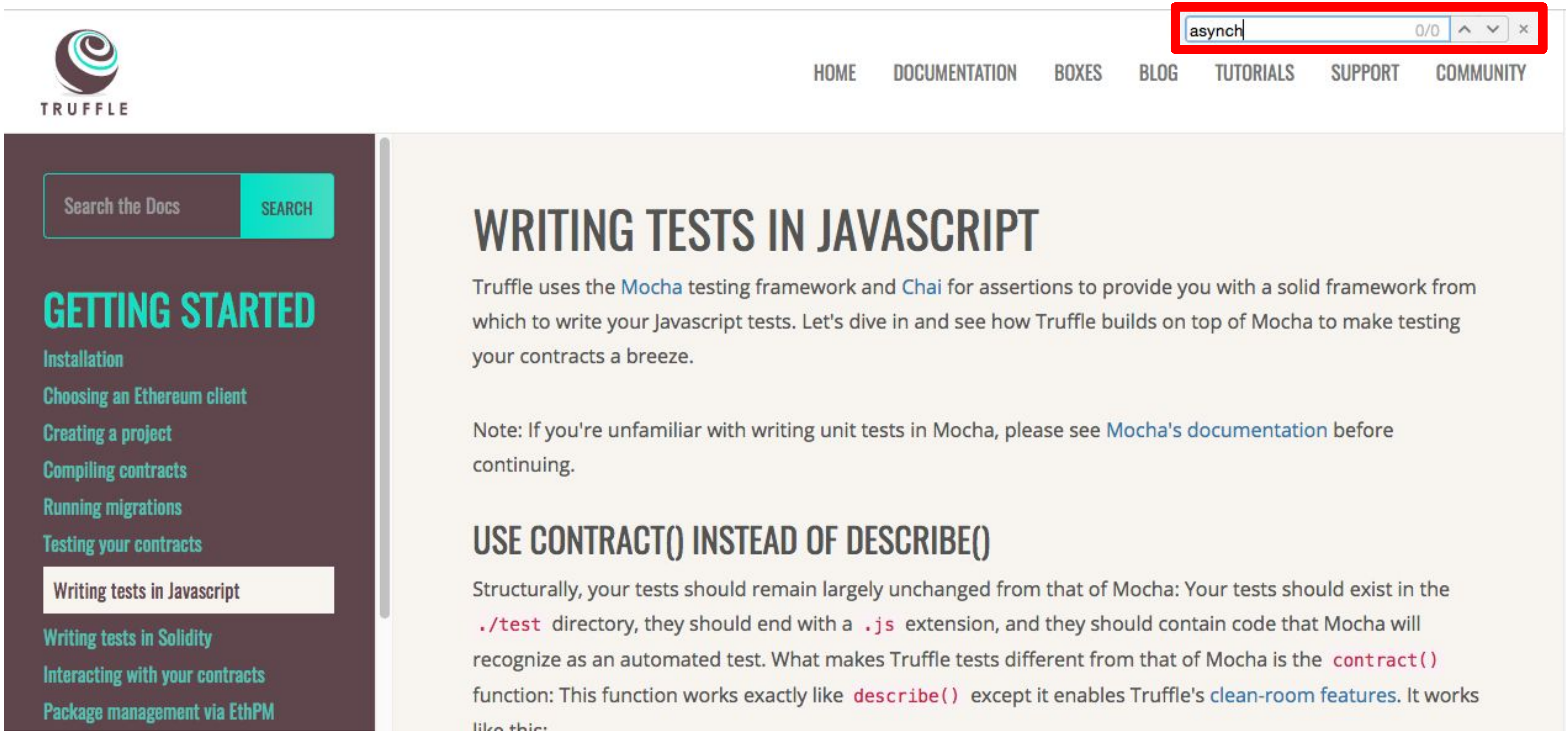


```
expect(2).to.equal(2); // Recommended
expect(2).to.not.equal(1); // Not recommended
```

1.2 ASYNCHRONOUS PROGRAMMING

INTRO TO ASYNCH

DOCS? WHAT DOCS?



WRITING TESTS IN JAVASCRIPT

Truffle uses the Mocha testing framework and Chai for assertions to provide you with a solid framework from which to write your Javascript tests. Let's dive in and see how Truffle builds on top of Mocha to make testing your contracts a breeze.

Note: If you're unfamiliar with writing unit tests in Mocha, please see Mocha's documentation before continuing.

USE CONTRACT() INSTEAD OF DESCRIBE()

Structurally, your tests should remain largely unchanged from that of Mocha: Your tests should exist in the `./test` directory, they should end with a `.js` extension, and they should contain code that Mocha will recognize as an automated test. What makes Truffle tests different from that of Mocha is the `contract()` function: This function works exactly like `describe()` except it enables Truffle's clean-room features. It works like this:

WHAT IS A PROMISE?

WE’LL GET BACK TO YOU ON THAT

(Source)

AUTHOR: NICK ZOGHB

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables: –

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables: n = 5

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

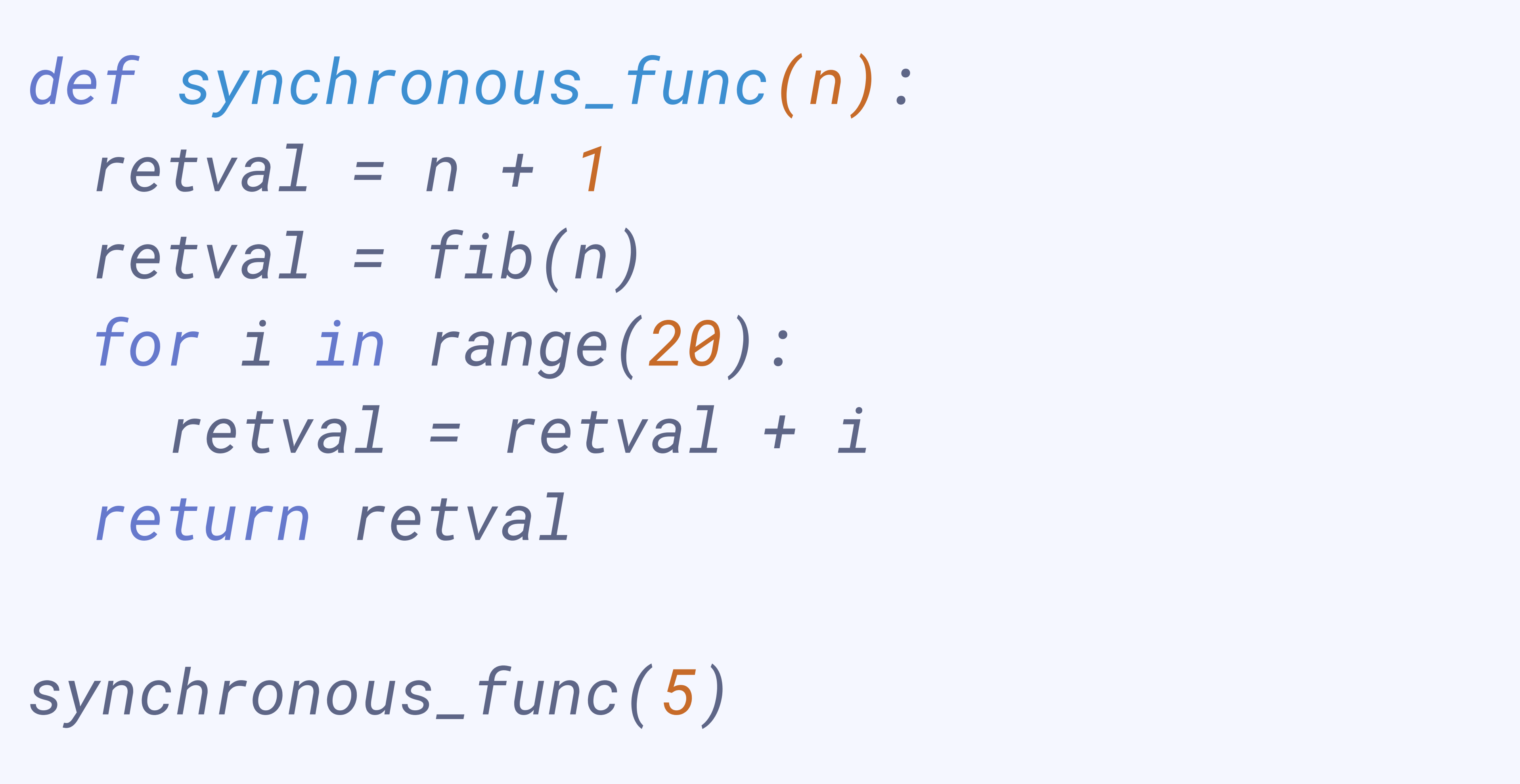
Variables:

```
  n      = 5
retval  = 6
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:



```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

```
 n       = 5
retval  = ...
(10ms)
```

BLOCKCHAIN AT BERKELEY

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

```
n       = 5
retval  = ...
(20ms)
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

BLOCKING!

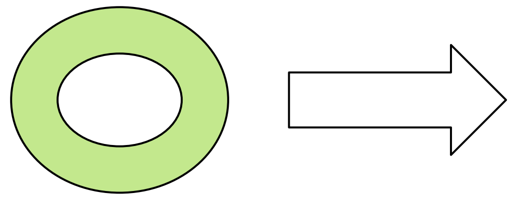
Variables:

```
 n        = 5
retval   = ...
(30ms)
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:



```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

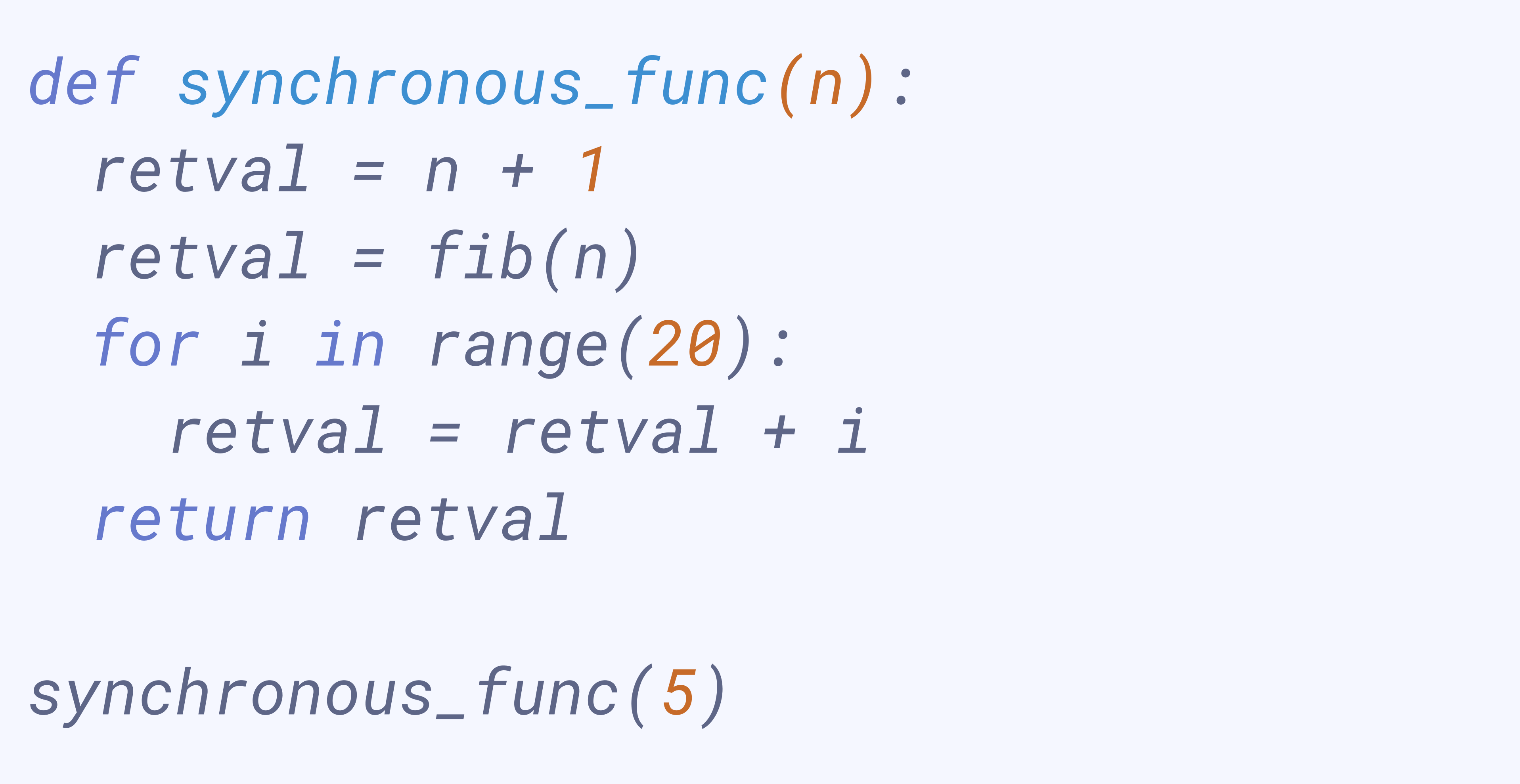
Variables:

```
n      = 5
retval = 8
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:



```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

```
 n      = 5
retval  = 13
i       = 0
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

n	=	5
retval	=	13
i	=	0

WHAT IS SYNCHRONOUS CODE

A SYNCHRONOUS EXAMPLE

Python Code:

```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

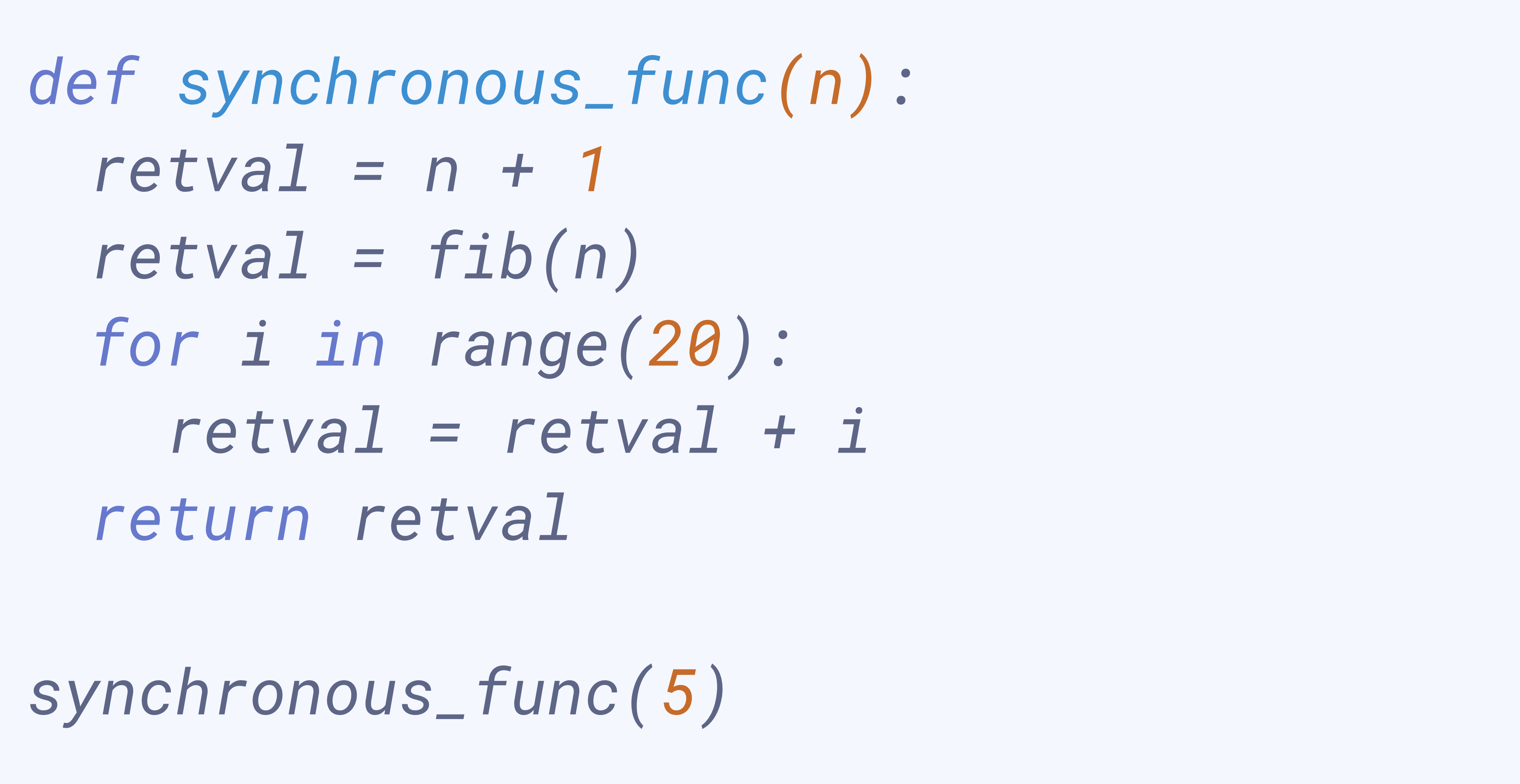
```
 n        = 5
retval   = 14
i        = 1
```

AUTHOR: NICK ZOGHB

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:



```
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

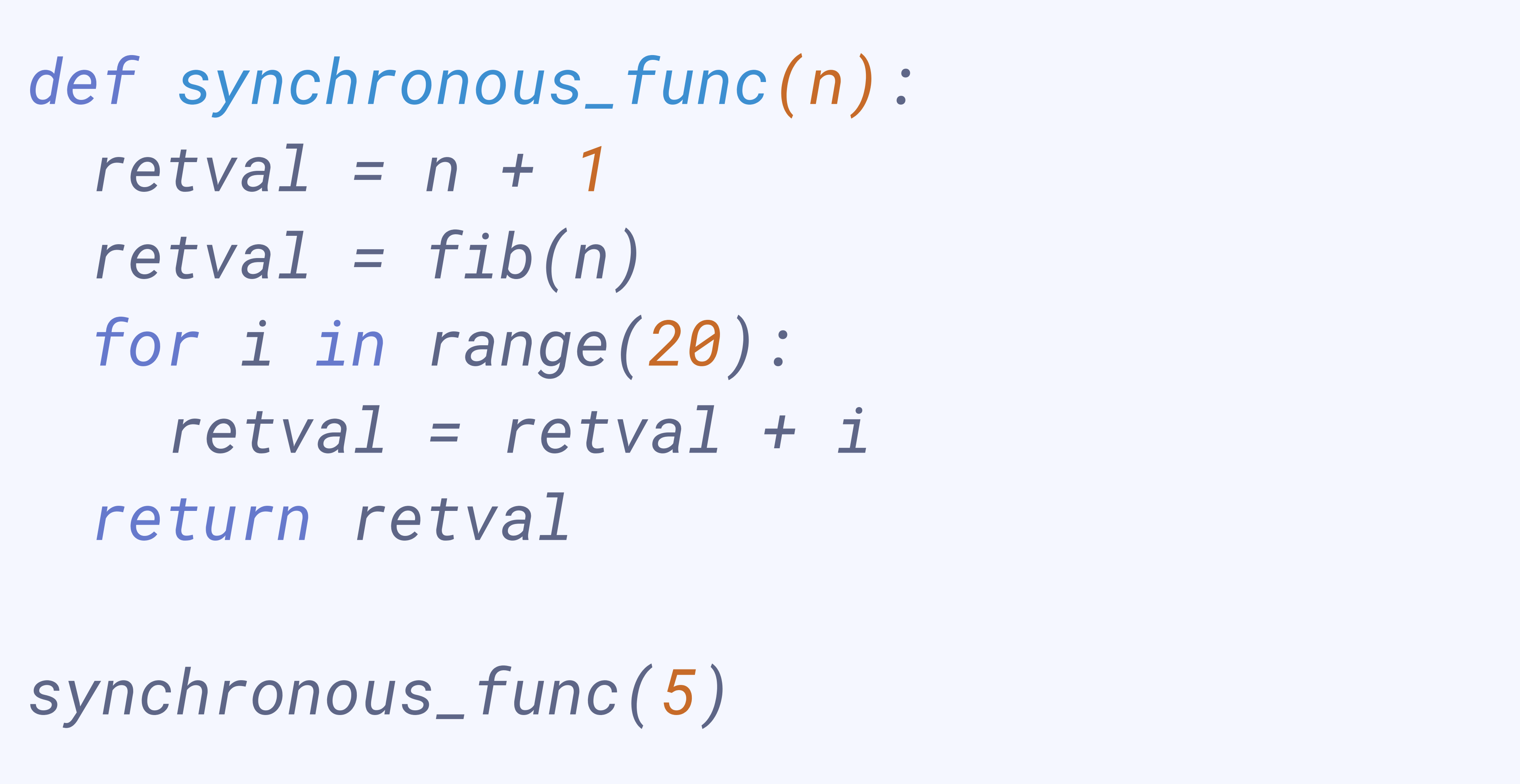
Variables:

```
 n       = 5
retval  = 16
i       = 2
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:



```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

```
 n       = 5
retval  = ...
i       = ...
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

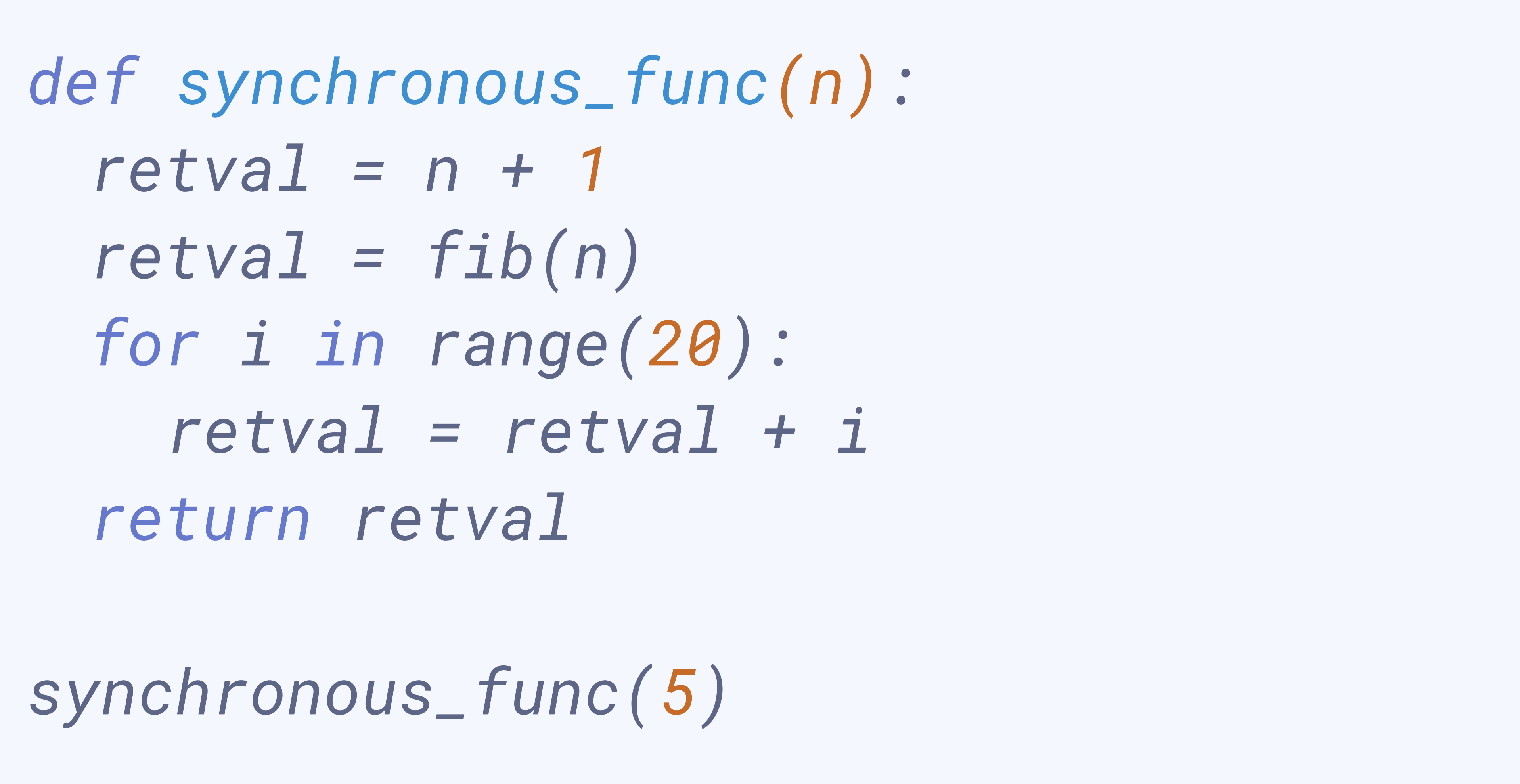
Variables:

```
 n       = 5
retval   = 203
i        = 19
```

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:



```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

```
n       = 5
retval  = 203
i       = 19
```

Returns: 203

WHAT IS SYNCHRONOUS CODE?

A SYNCHRONOUS EXAMPLE

Python Code:

```python
def synchronous_func(n):
  retval = n + 1
  retval = fib(n)
  for i in range(20):
    retval = retval + i
  return retval

synchronous_func(5)
```

Variables:

```
n      = 5
retval = 203
i      = 19
```

Returns: 203

Synchronous programming: functions are blocking. In other words, if you call a function foo it will not relinquish control till it has completed its execution. Read more here.

WHAT IS A PROMISE?

OK, SO HERE IT IS

(Source)

WHAT IS A PROMISE?

OK, SO HERE IT IS

A *promise* is a 'black box'. It could be in one of three states

- Pending: The asynchronous operation is still running
- Resolved: The asynchronous operation has completed successfully
- Rejected: The asynchronous operation is complete but is unsuccessful, probably some error/exception was thrown

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

Variables: –

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

Variables: n = 5

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

Variables:

```
     n      = 5
retval    = 6
```

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

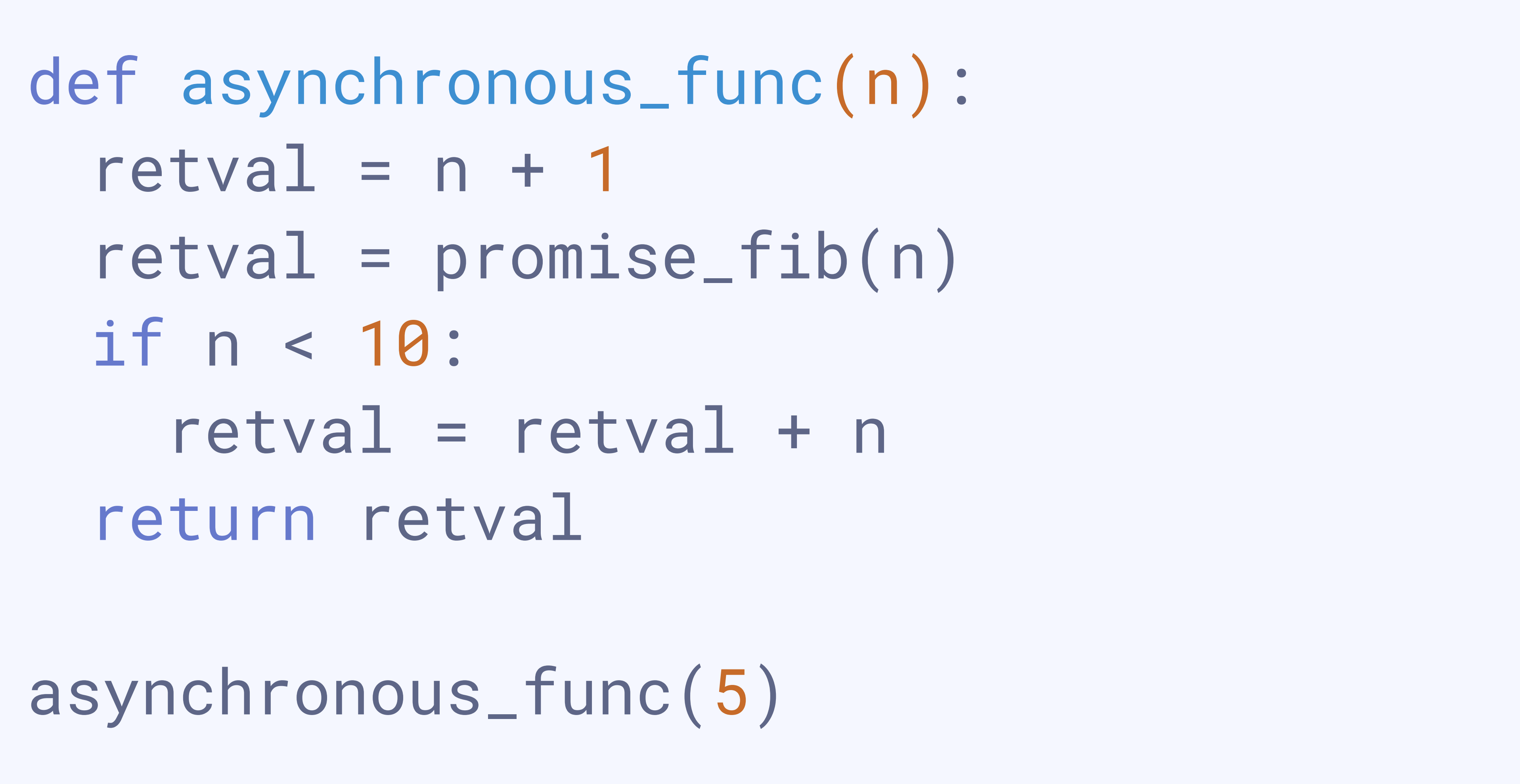
Variables: n = 5

retval = 🎁 <Promise>

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:



```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

Variables:

```
n       = 5
retval  = 🎁
<Promise>
```

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

error ⇨

```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

Variables:

```
 n        = 5
retval   = 🎁
<Promise>
```

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```
def asynchronous_func(n):
  retval = n + 1
  retval = promise_fib(n)
  if n < 10:
    retval = retval + n
  return retval

asynchronous_func(5)
```

error ⇨

Variables:

```
    n      = 5
retval    = 🎁
<Promise>
```

Cannot conduct outside operations on *promises*. So let’s backtrack…

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

NEW

```
def asynchronous_func(n):
  retval = n + 1
  pretval = promise_fib(n)
  temp = 0
  if n < 10:
    temp = n
  return pretval.ret_prom() + temp

asynchronous_func(5)
```

Variables:

```
      n      = 5
pretval      = 🎁
  <Promise>
   temp      = 5
```

Added one extra line to maintain functionality.

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```
def asynchronous_func(n):
  retval = n + 1
  pretval = promise_fib(n)
  temp = 0
  if n < 10:
    temp = n
  return pretval.resolve() + temp

asynchronous_func(5)
```

```
Variables:      n          = 5
           pretval   = 8
        <ResolvedPromise>
             temp      = 5

Returns:        13
```

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:

```python
def asynchronous_func(n):
  retval = n + 1
  pretval = promise_fib(n)
  temp = 0
  if n < 10:
    temp = n
  return pretval.resolve() + temp

asynchronous_func(5)
```

Variables:

```
n        = 5
pretval  = 8
<ResolvedPromise>
temp     = 5
```

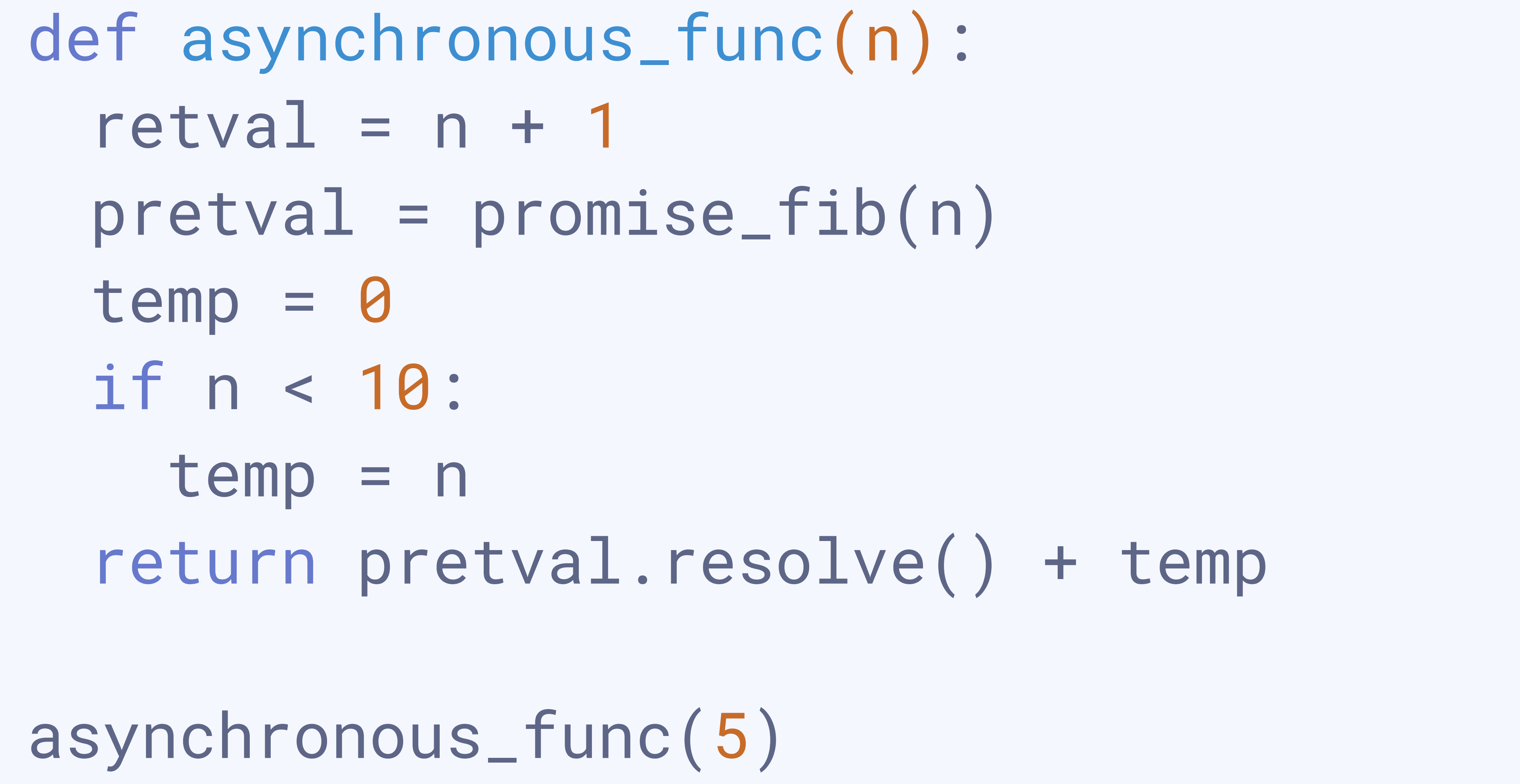
Returns: 13

The promise is forced to return at the end with **`pretval.resolve()`**

WHAT IS A PROMISE?

OK, SO HERE IT IS

Python Code:



```python
def asynchronous_func(n):
  retval = n + 1
  pretval = promise_fib(n)
  temp = 0
  if n < 10:
    temp = n
  return pretval.resolve() + temp

asynchronous_func(5)
```

Variables: n = 5
pretval = 8
<ResolvedPromise>
temp = 5

Returns: 13
(0ms blocked)

The promise is forced to return at the end with **`pretval.resolve()`**

BLOCKCHAIN
AT BERKELEY

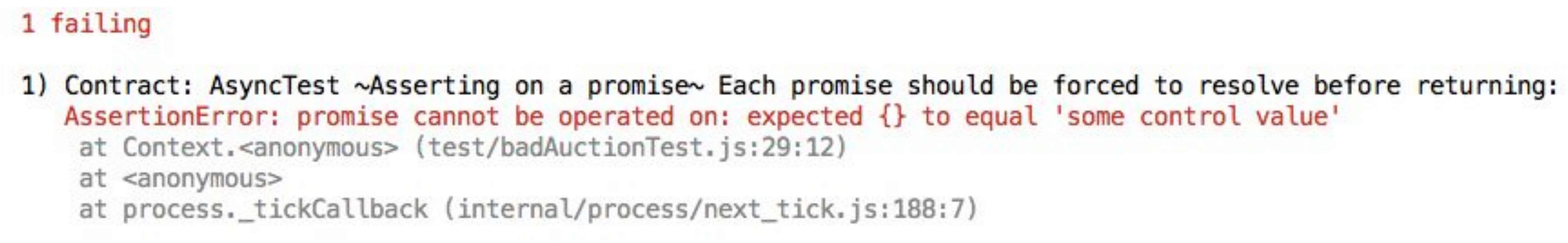
WHAT IS A PROMISE?

OK, SO HERE IT IS

```
def promise_fib(x):
    ...
    ...
    return retval
```

WHAT IS A PROMISE?

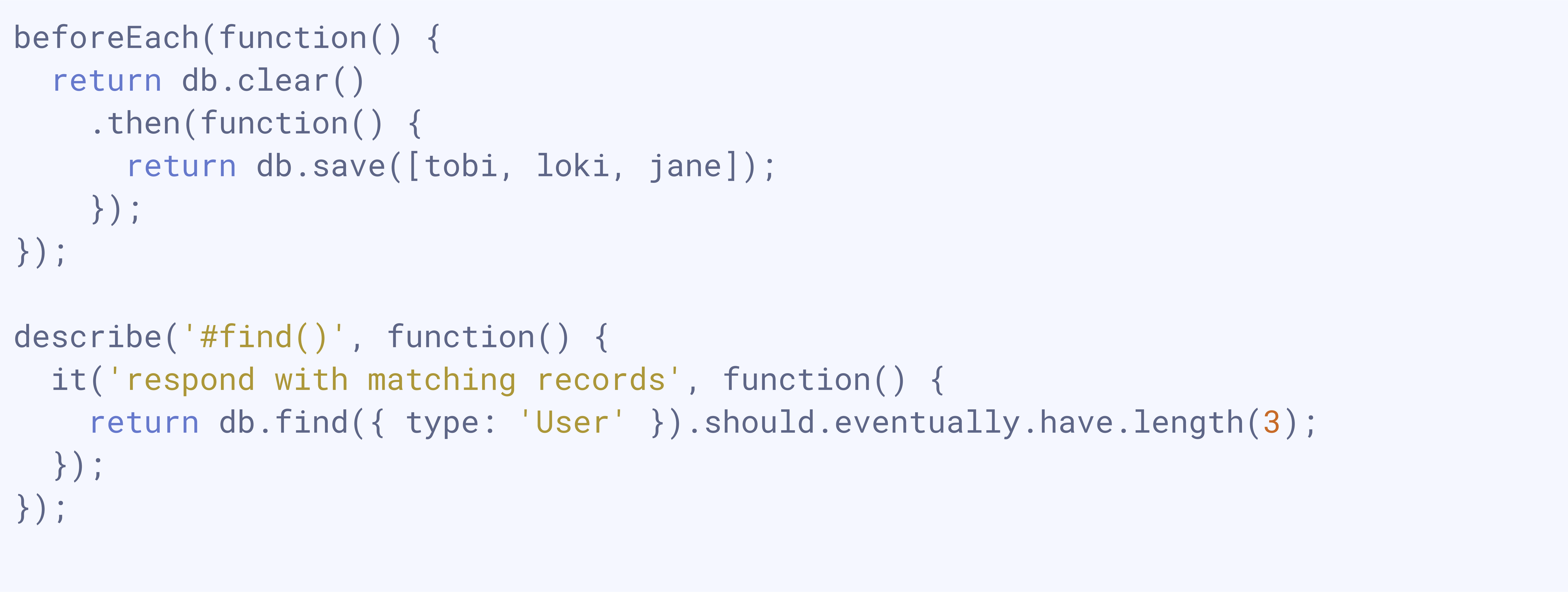
ERRORS MAY COME UP



```
1 failing

1) Contract: AsyncTest ~Asserting on a promise~ Each promise should be forced to resolve before returning:
   AssertionError: promise cannot be operated on: expected {} to equal 'some control value'
    at Context.<anonymous> (test/badAuctionTest.js:29:12)
    at <anonymous>
    at process._tickCallback (internal/process/next_tick.js:188:7)
```

WHAT IS A PROMISE?

.then(...)

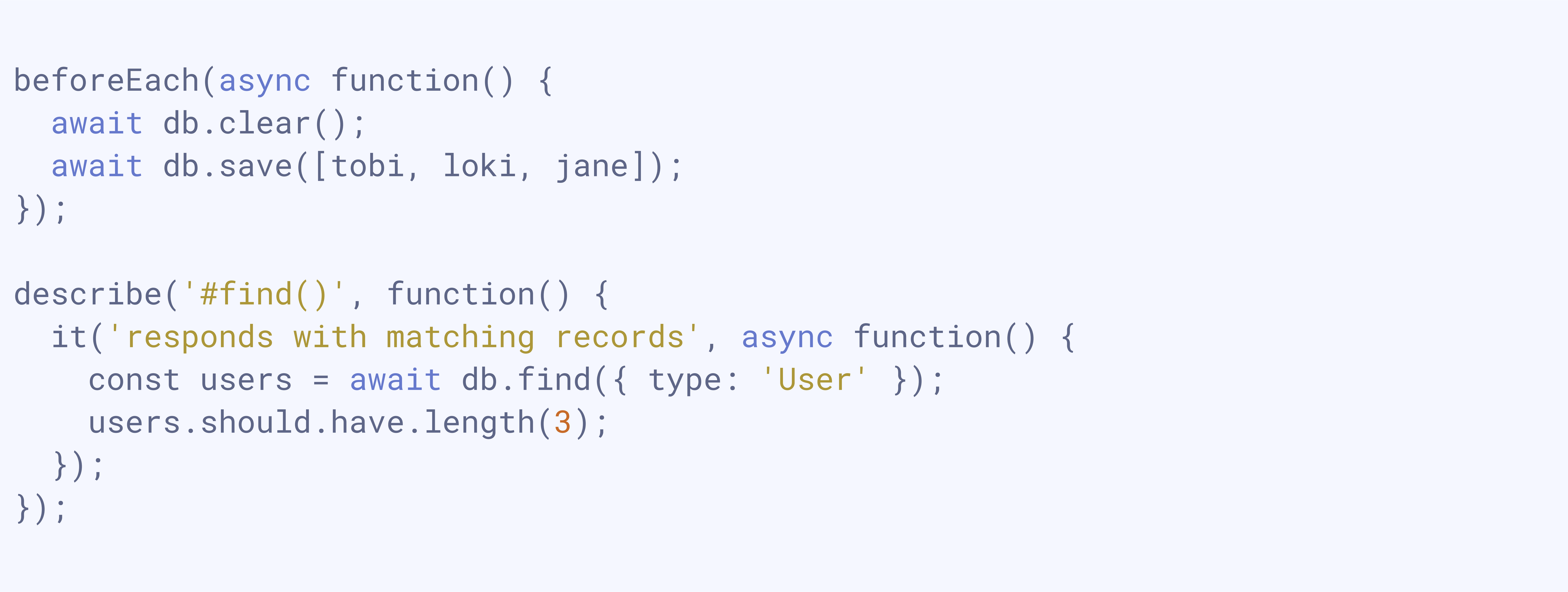


```
beforeEach(function() {
  return db.clear()
    .then(function() {
      return db.save([tobi, loki, jane]);
    });
});

describe('#find()', function() {
  it('respond with matching records', function() {
    return db.find({ type: 'User' }).should.eventually.have.length(3);
  });
});
```

(Source)

WHAT IS A PROMISE?

async/await

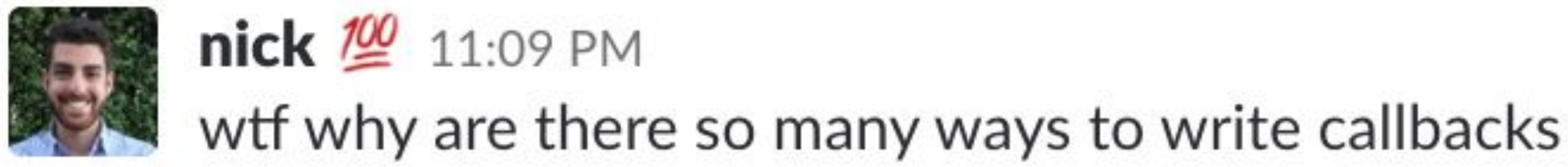


```
beforeEach(async function() {
  await db.clear();
  await db.save([tobi, loki, jane]);
});

describe('#find()', function() {
  it('responds with matching records', async function() {
    const users = await db.find({ type: 'User' });
    users.should.have.length(3);
  });
});
```

(Source)

WHAT IS A PROMISE?

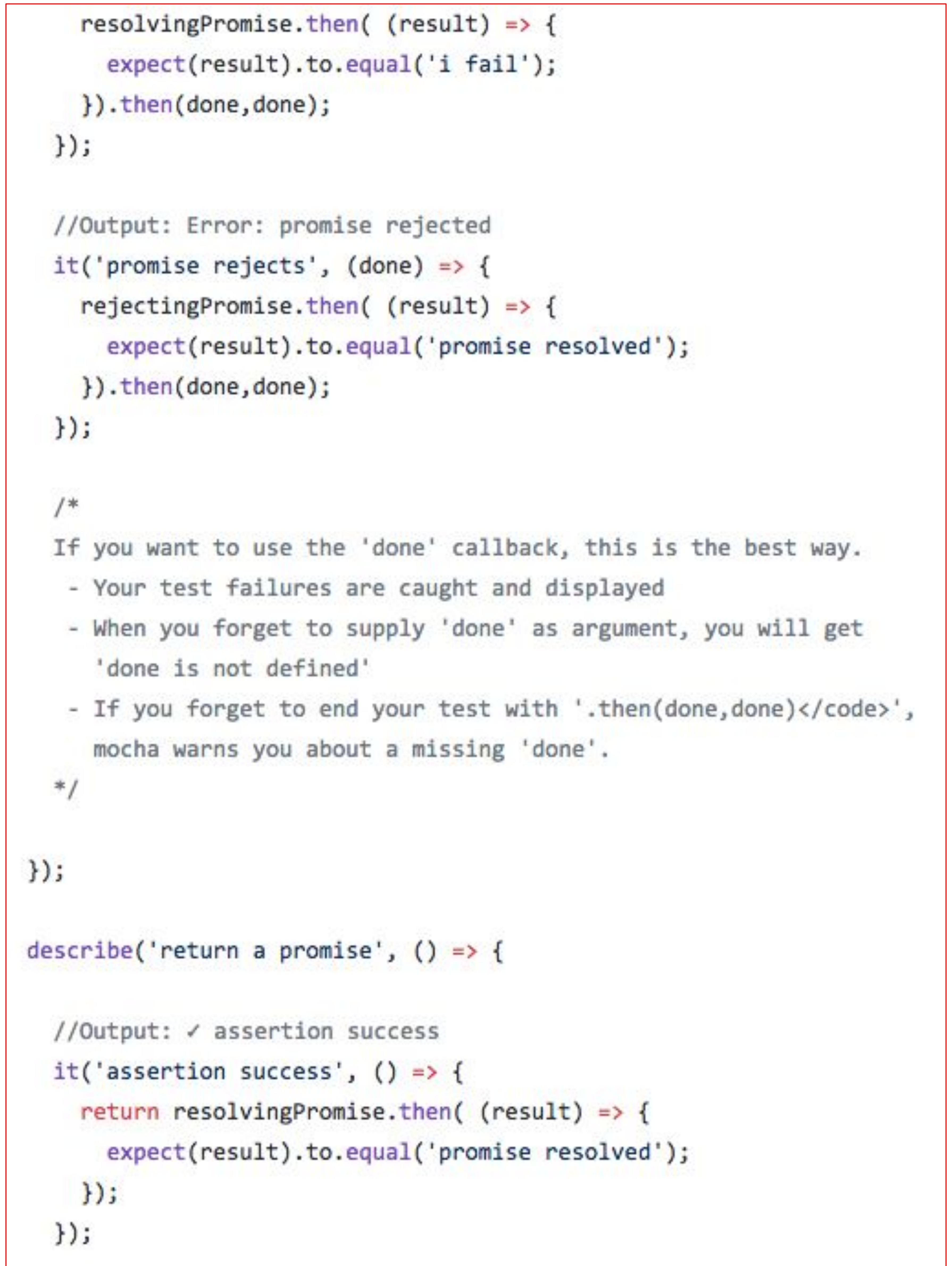
TOO MANY WAYS



nick 💯 11:09 PM

wtf why are there so many ways to write callbacks

Read more here



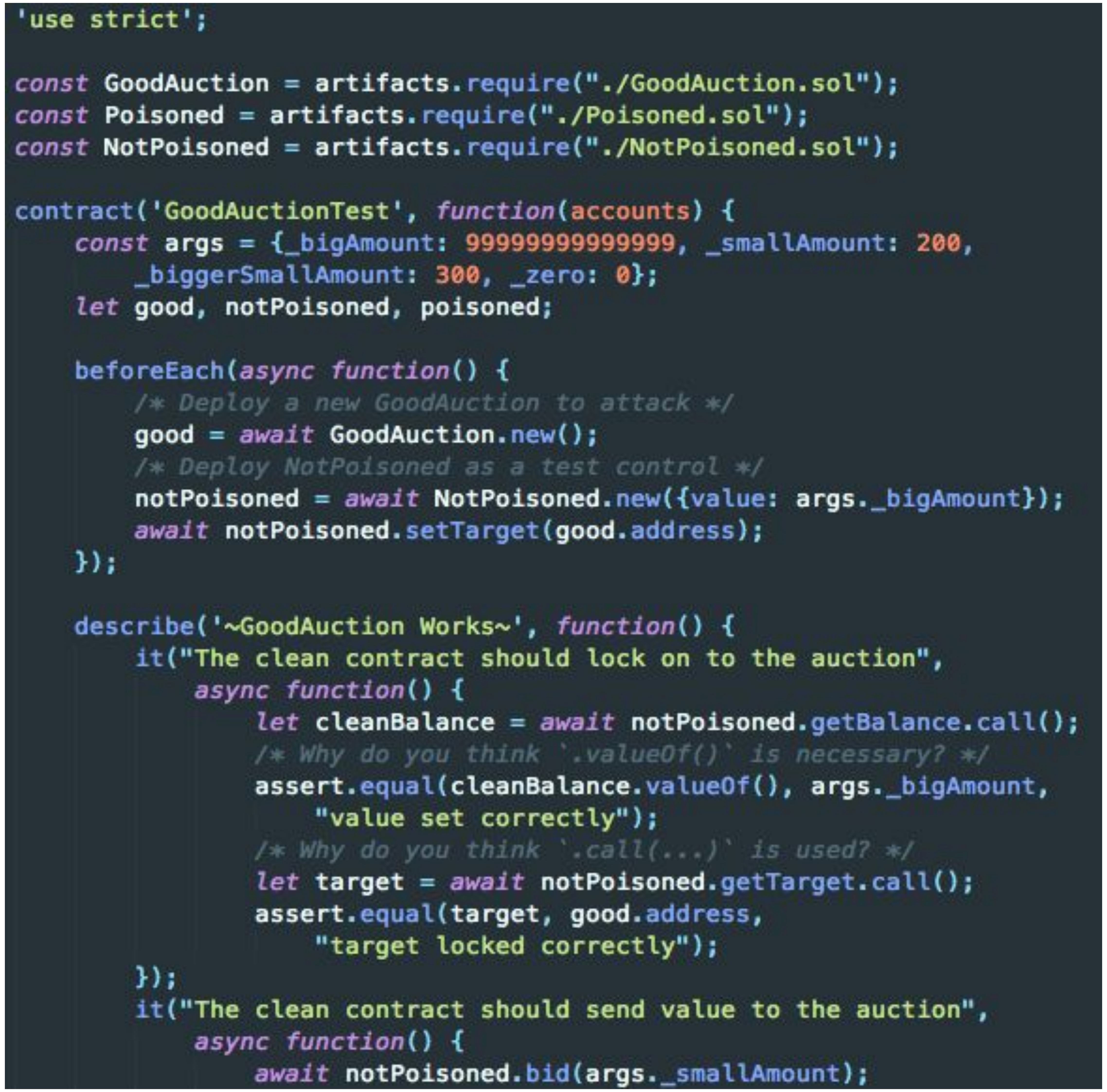
```
    resolvingPromise.then( (result) => {
      expect(result).to.equal('i fail');
    }).then(done,done);
  });

  //Output: Error: promise rejected
  it('promise rejects', (done) => {
    rejectingPromise.then( (result) => {
      expect(result).to.equal('promise resolved');
    }).then(done,done);
  });

  /*
  If you want to use the 'done' callback, this is the best way.
   - Your test failures are caught and displayed
   - When you forget to supply 'done' as argument, you will get
     'done is not defined'
   - If you forget to end your test with '.then(done,done)</code>',
     mocha warns you about a missing 'done'.
  */

});

describe('return a promise', () => {

  //Output: ✓ assertion success
  it('assertion success', () => {
    return resolvingPromise.then( (result) => {
      expect(result).to.equal('promise resolved');
    });
  });
```

2 TESTING SMART CONTRACTS

2.1 …IN JAVASCRIPT

A SNAPSHOT

DO THIS!



```
'use strict';

const GoodAuction = artifacts.require("./GoodAuction.sol");
const Poisoned = artifacts.require("./Poisoned.sol");
const NotPoisoned = artifacts.require("./NotPoisoned.sol");

contract('GoodAuctionTest', function(accounts) {
    const args = {_bigAmount: 999999999999999, _smallAmount: 200,
        _biggerSmallAmount: 300, _zero: 0};
    let good, notPoisoned, poisoned;

    beforeEach(async function() {
        /* Deploy a new GoodAuction to attack */
        good = await GoodAuction.new();
        /* Deploy NotPoisoned as a test control */
        notPoisoned = await NotPoisoned.new({value: args._bigAmount});
        await notPoisoned.setTarget(good.address);
    });

    describe('~GoodAuction Works~', function() {
        it("The clean contract should lock on to the auction",
            async function() {
                let cleanBalance = await notPoisoned.getBalance.call();
                /* Why do you think `.valueOf()` is necessary? */
                assert.equal(cleanBalance.valueOf(), args._bigAmount,
                    "value set correctly");
                /* Why do you think `.call(...)` is used? */
                let target = await notPoisoned.getTarget.call();
                assert.equal(target, good.address,
                    "target locked correctly");
        });
        it("The clean contract should send value to the auction",
            async function() {
                await notPoisoned.bid(args._smallAmount);
```

MOCHA SYNTAX

REVISITED

What are the new keywords?

- `contract(‘NameOfSuite’, function(accounts)) {...}`
  - Before each contract function is run, your contracts are redeployed to the running Ethereum client so the tests within it run with a clean contract state
  - The contract function provides a list of accounts made available by your Ethereum client which you can use to write tests
- `artifacts.require("./contract.sol");`
  - Because Truffle has no way of detecting which contracts you'll need to interact with within your tests, you'll need to ask for those contracts explicitly
- `web3.eth.getBalance`

(Source)

MOCHA SYNTAX

REVISITED

What are the new keywords?

- `.call(...)`
    - Used to specify that a method is explicitly NOT a transaction, e.g. a getter method
    - Transactions will not execute if `.call(...)` is used
- **function(accounts)**
    - This is used to reference *Ganache* accounts. By default, `accounts[0]` is what is used for methods
- `contract.address`
    - Default way to get a contract's address on the network
- `.valueOf()`
    - Used to retrieve number from a contract balance

MOCHA SYNTAX

REVISITED

What are the new keywords?

- `{from: someOther.address, value: someAmount}`
    - `from`
        - Usually executed from EOA’s for testing
        - Only non-transaction calls work for contract addresses
    - `value`
        - Specify some amount of *wei* to send from an account’s balance
        - Does not work if the account does not have enough value

DEPLOYING WITHIN TESTS

CONTRACTS ON CONTRACTS ON CONTRACTS

```
contract('MetaCoin', function(accounts) {
  it("should put 10000 MetaCoin in the first account", function() {
    return MetaCoin.deployed().then(function(instance) {
      return instance.getBalance.call(accounts[0]);
    }).then(function(balance) {
      assert.equal(balance.valueOf(), 10000, "10000 wasn't in the first account");
    });
  });
});
```

v.s.

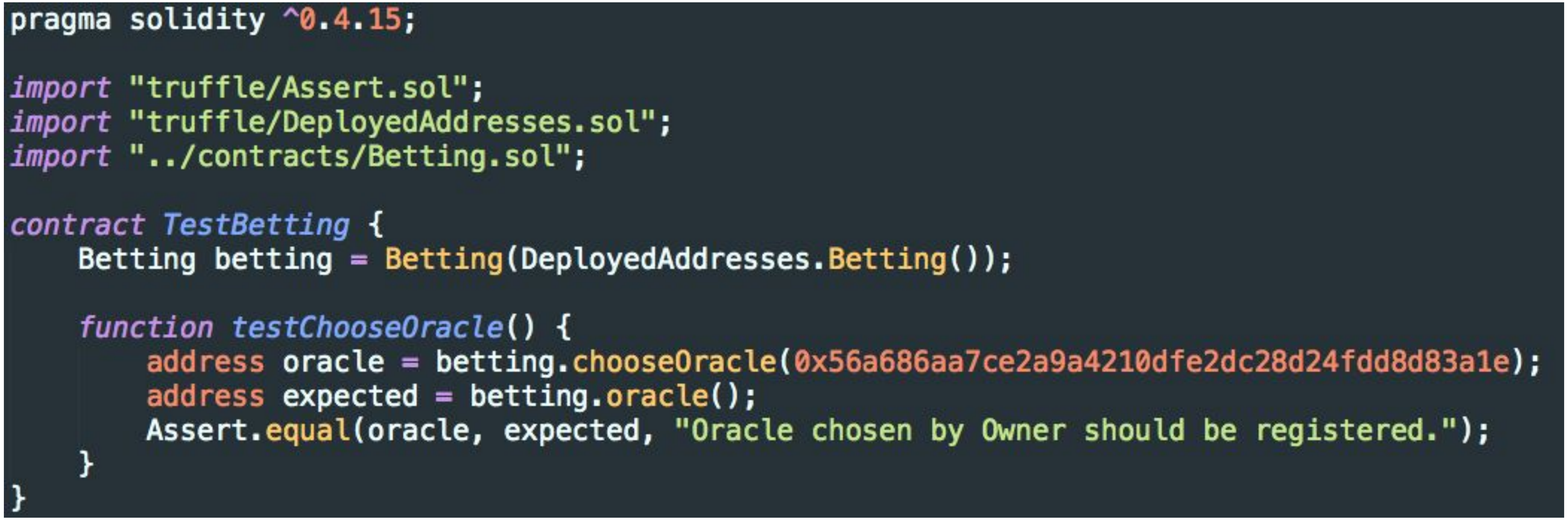
```
let contract = await MetaCoin.new();
let balance = await MetaCoin.getBalance.call(accounts[0]);
assert.equal(balance.valueOf(), 10000, "10000 wasn't in the first account");
```

Read more here

2.2 …IN SOLIDITY

A SNAPSHOT

DON'T DO THIS!



```
pragma solidity ^0.4.15;

import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/Betting.sol";

contract TestBetting {
    Betting betting = Betting(DeployedAddresses.Betting());

    function testChooseOracle() {
        address oracle = betting.chooseOracle(0x56a686aa7ce2a9a4210dfe2dc28d24fdd8d83a1e);
        address expected = betting.oracle();
        Assert.equal(oracle, expected, "Oracle chosen by Owner should be registered.");
    }
}
```

(DeployedAddresses.sol is dynamically created at test time)

THOUGHTS

THINGS TO NOTE

- Truffle Solidity tests can be used to cover a small piece of code, basically providing the ability to test every single function in contracts in an isolated way.
- Truffle JavaScript tests (Mocha) demonstrate that different pieces of the system work together. Allows for testing complex scenarios with multiple calls and transactions.

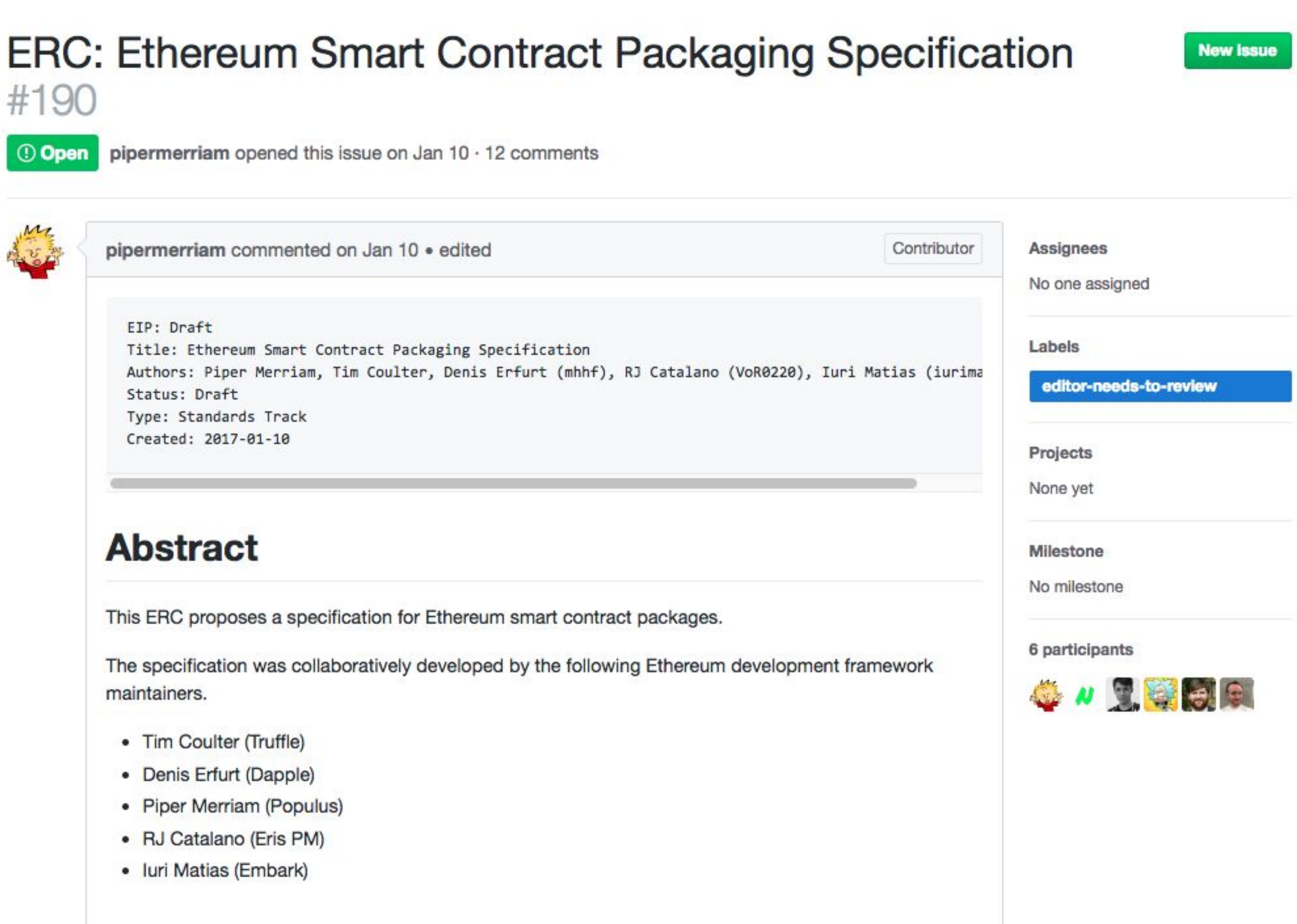
Read more here

3

BEST PRACTICES WHEN WRITING TESTS

ERC190

ETHEREUM SMART CONTRACT PACKAGING SPECIFICATION



ERC: Ethereum Smart Contract Packaging Specification #190

pipermerriam opened this issue on Jan 10 · 12 comments

pipermerriam commented on Jan 10 • edited

```
EIP: Draft
Title: Ethereum Smart Contract Packaging Specification
Authors: Piper Merriam, Tim Coulter, Denis Erfurt (mhhf), RJ Catalano (VoR0220), Iuri Matias (iurima
Status: Draft
Type: Standards Track
Created: 2017-01-10
```

Abstract

This ERC proposes a specification for Ethereum smart contract packages.

The specification was collaboratively developed by the following Ethereum development framework maintainers.

- Tim Coulter (Truffle)
- Denis Erfurt (Dapple)
- Piper Merriam (Populus)
- RJ Catalano (Eris PM)
- Iuri Matias (Embark)

Assignees
No one assigned

Labels
editor-needs-to-review

Projects
None yet

Milestone
No milestone

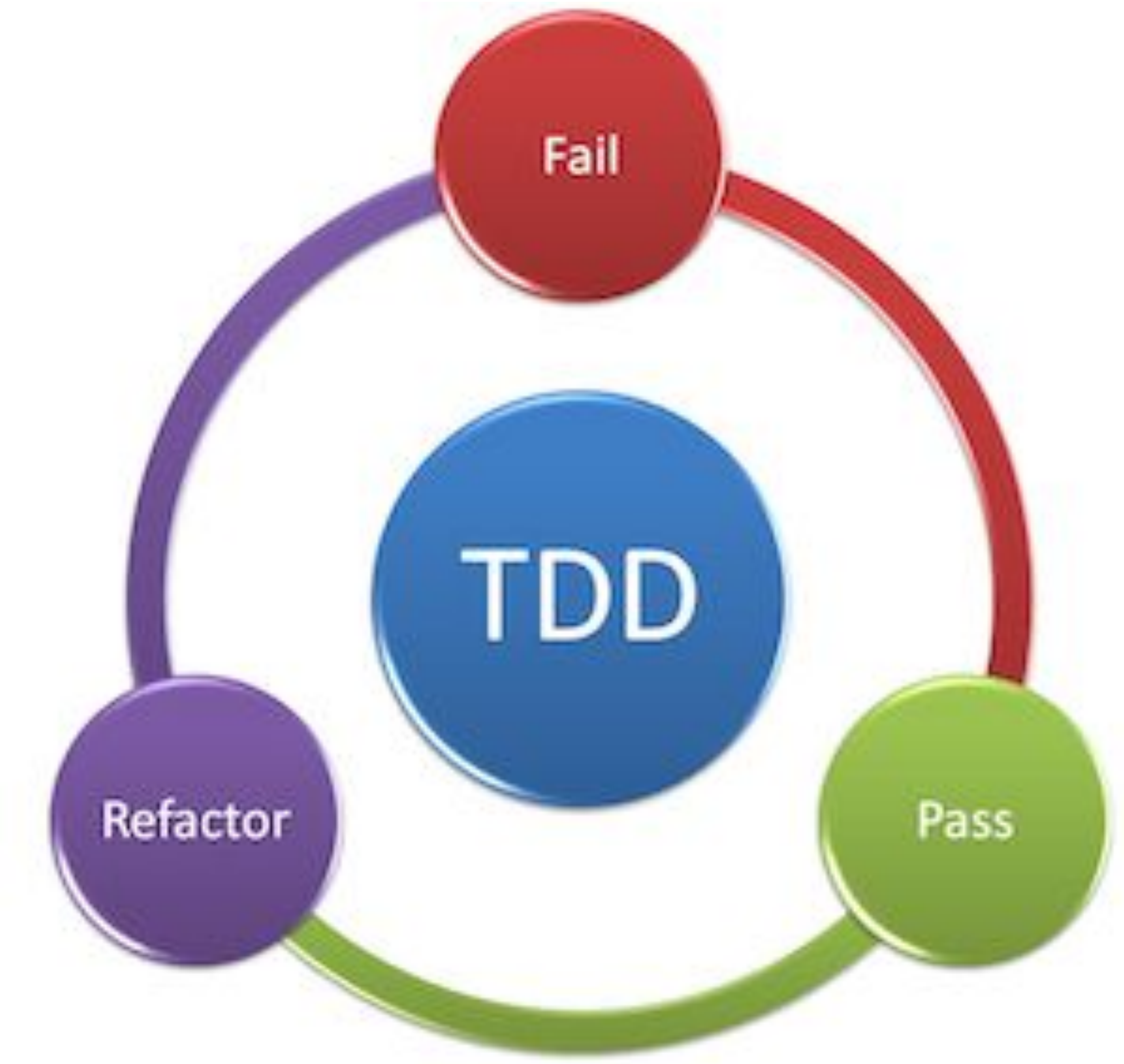
6 participants

(Source)

3.1 TEST-DRIVEN DEVELOPMENT

THE CYCLE DIAGRAM

IT'S ACTUALLY MORE INTUITIVE THAN IT SEEMS



AUTHOR: NICK ZOGHB

THE MAIN TAKEAWAY

YOU'LL BE FINE IF YOU KNOW THIS

- Write very specific test cases
  - Write a lot of them!
  - Cover 💯% of application functionality
- Make new changes only after passing old tests
- Make tests run as fast as possible as possible

AUTHOR: NICK ZOGHB

(Source)

3.2 WHAT TO TEST

LECTURE 03: SMART CONTRACT SECURITY

WOAH, SO META!

Lecture 04:
Smart Contract Security

Akash Khosla
Nick Zoghb

UNIT AND INTEGRATION TESTING

YOU NEED BOTH

- A unit test is a test written by the programmer to verify that a relatively small piece of code is doing what it is intended to do. They are narrow in scope, they should be easy to write and execute
  - Unit tests shouldn't have dependencies on outside systems
  - They test internal consistency as opposed to proving that they play nicely with some outside system.
- An integration test is done to demonstrate that different pieces of the system work together
  - Integration tests cover whole applications
  - Example cases: proving interoperability between two smart contracts, querying external databases, etc.

(Source)

UNIT AND INTEGRATION TESTING

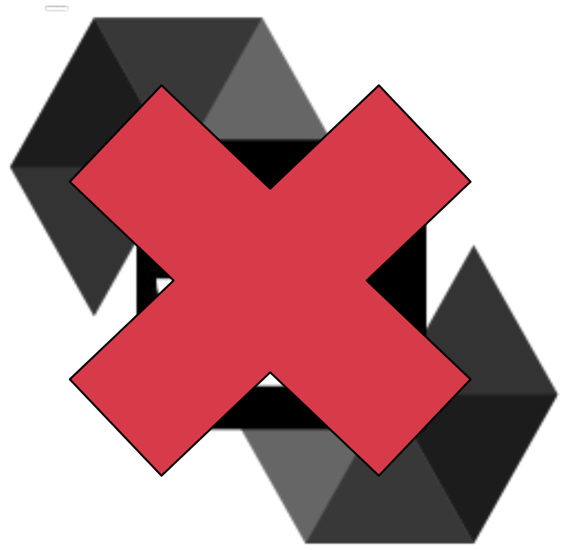
WHICH LANGUAGE IS BEST?

UNIT TESTS ⇝ Solidity

INTEGRATION TESTS → JavaScript

UNIT AND INTEGRATION TESTING

ACTUALLY YOU ONLY NEED ONE



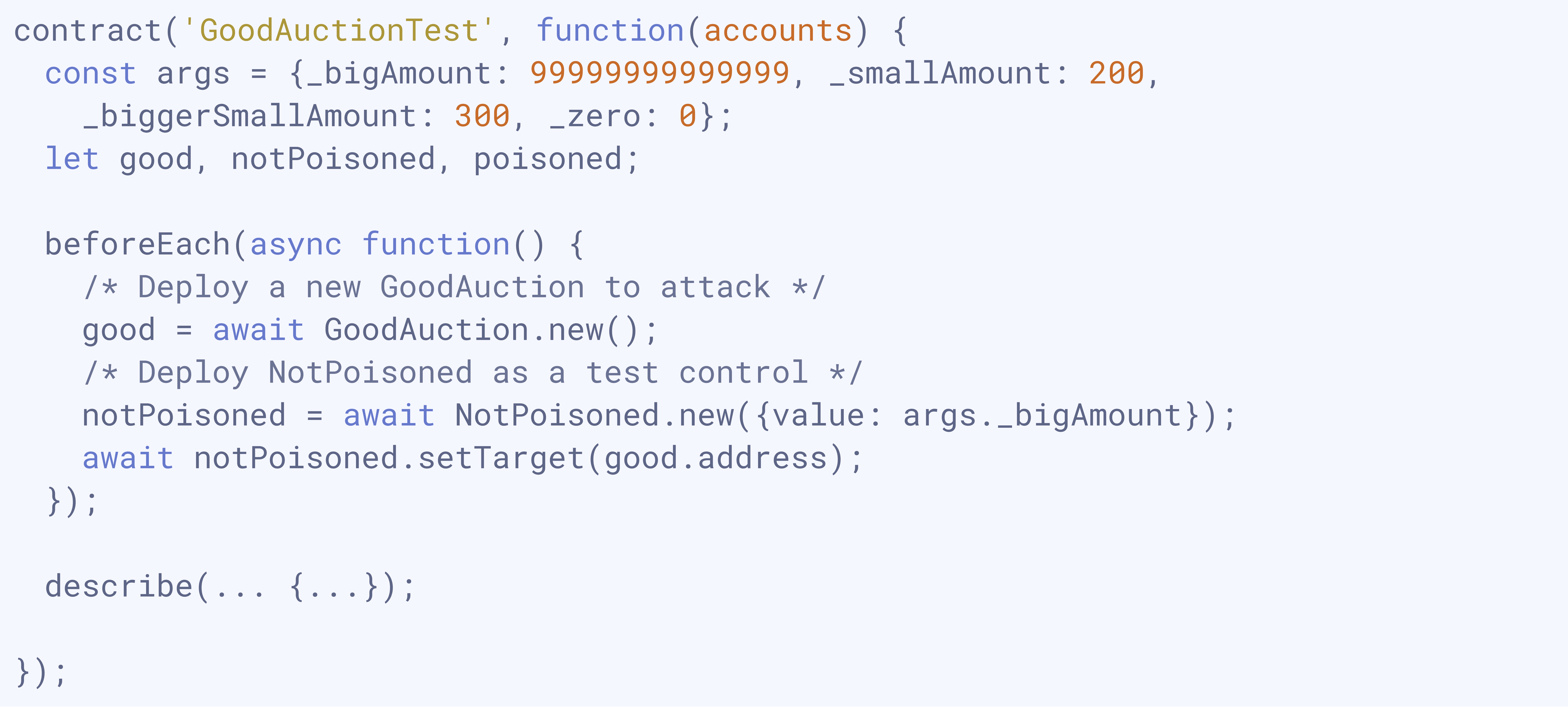
UNIT TESTS → JavaScript

INTEGRATION TESTS → JavaScript

3.3 NEED FOR SPEED

DECLARATIONS, SETUP, TEARDOWN

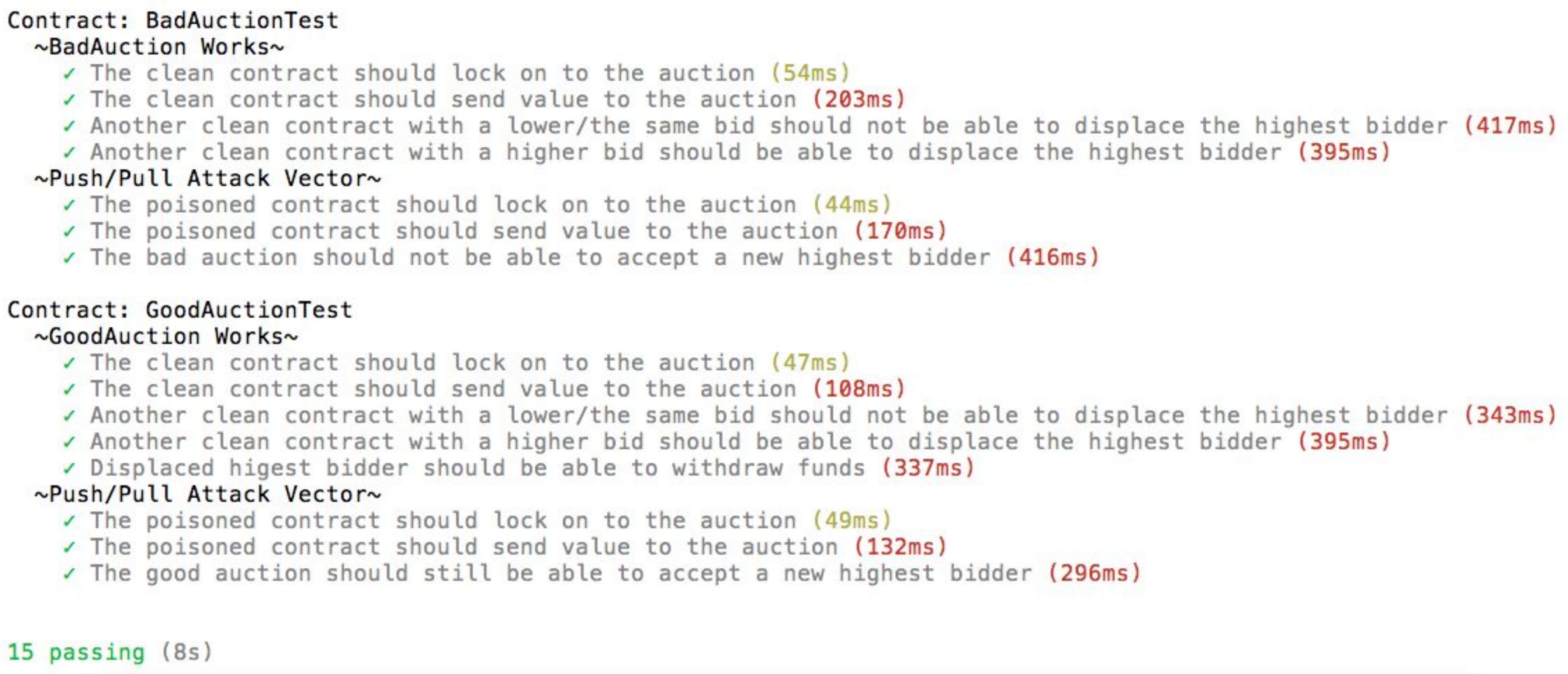
PUT ON YOUR SEATBELTS FOR THIS



```
contract('GoodAuctionTest', function(accounts) {
  const args = {_bigAmount: 999999999999999, _smallAmount: 200,
    _biggerSmallAmount: 300, _zero: 0};
  let good, notPoisoned, poisoned;

  beforeEach(async function() {
    /* Deploy a new GoodAuction to attack */
    good = await GoodAuction.new();
    /* Deploy NotPoisoned as a test control */
    notPoisoned = await NotPoisoned.new({value: args._bigAmount});
    await notPoisoned.setTarget(good.address);
  });

  describe(... {...});

});
```

READABILITY HELPS

MAKE IT EASY FOR YOURSELF



```
Contract: BadAuctionTest
  ~BadAuction Works~
    ✓ The clean contract should lock on to the auction (54ms)
    ✓ The clean contract should send value to the auction (203ms)
    ✓ Another clean contract with a lower/the same bid should not be able to displace the highest bidder (417ms)
    ✓ Another clean contract with a higher bid should be able to displace the highest bidder (395ms)
  ~Push/Pull Attack Vector~
    ✓ The poisoned contract should lock on to the auction (44ms)
    ✓ The poisoned contract should send value to the auction (170ms)
    ✓ The bad auction should not be able to accept a new highest bidder (416ms)

Contract: GoodAuctionTest
  ~GoodAuction Works~
    ✓ The clean contract should lock on to the auction (47ms)
    ✓ The clean contract should send value to the auction (108ms)
    ✓ Another clean contract with a lower/the same bid should not be able to displace the highest bidder (343ms)
    ✓ Another clean contract with a higher bid should be able to displace the highest bidder (395ms)
    ✓ Displaced higest bidder should be able to withdraw funds (337ms)
  ~Push/Pull Attack Vector~
    ✓ The poisoned contract should lock on to the auction (49ms)
    ✓ The poisoned contract should send value to the auction (132ms)
    ✓ The good auction should still be able to accept a new highest bidder (296ms)


15 passing (8s)
```

3.4 EXTRA TOOLS

SOLIDITY-COVERAGE

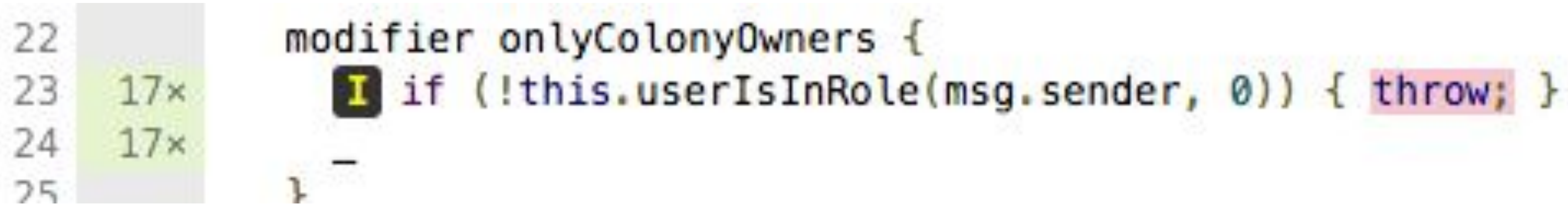
GOING FOR 💯

What is SolCover?

- Code coverage for Solidity testing

How do I run it / how does it work?

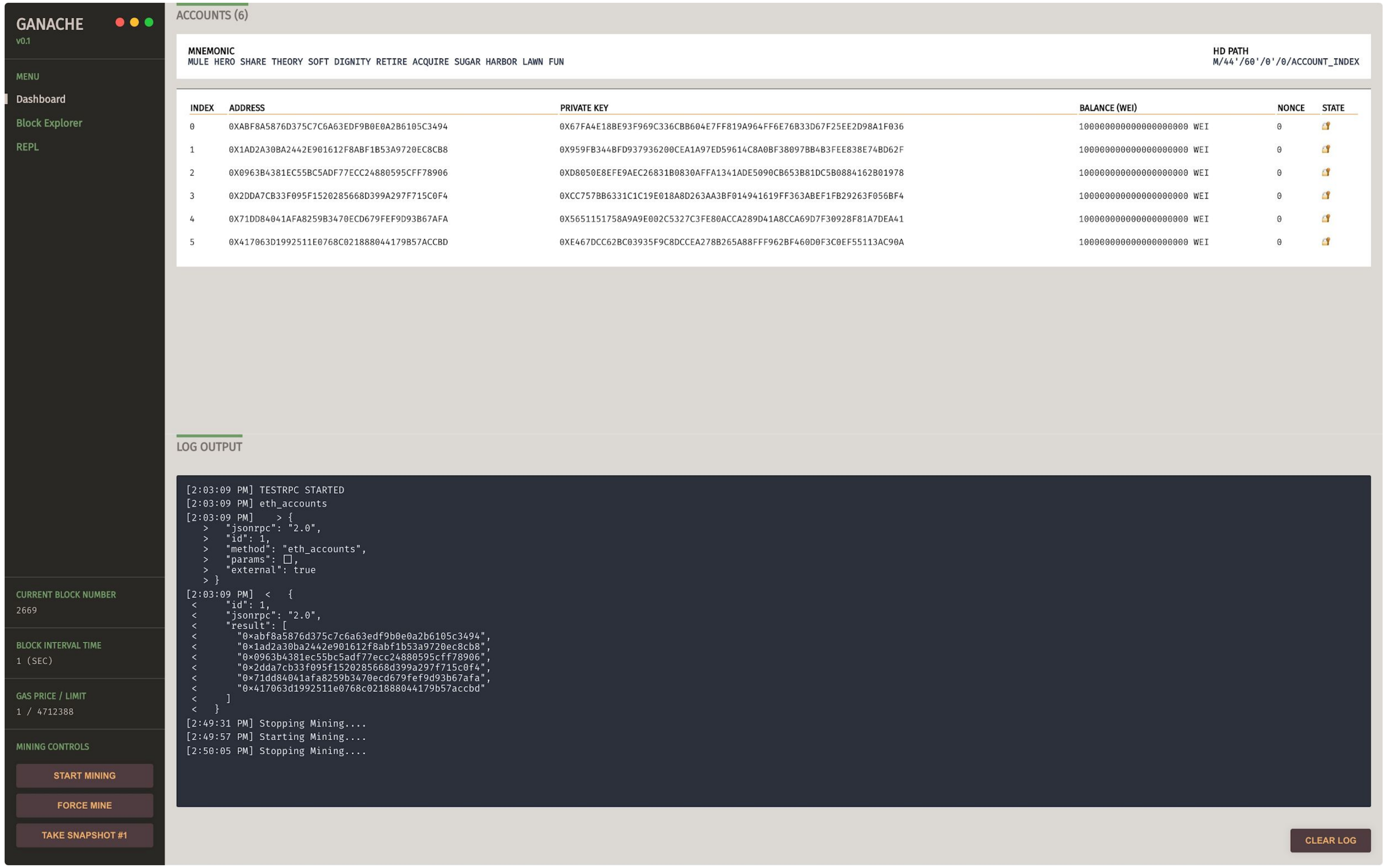
- Github: https://github.com/sc-forks/solidity-coverage
- Article: https://blog.colony.io/code-coverage-for-solidity-eecfa88668c2



```
         modifier onlyColonyOwners {
  17×      I if (!this.userIsInRole(msg.sender, 0)) { throw; }
  17×       _
         }
```

GANACHE

OUT WITH THE OLD...



CURRENT BLOCK NUMBER
2669

BLOCK INTERVAL TIME
1 (SEC)

GAS PRICE / LIMIT
1 / 4712388

ACCOUNTS (6)

MNEMONIC
MULE HERO SHARE THEORY SOFT DIGNITY RETIRE ACQUIRE SUGAR HARBOR LAWN FUN

HD PATH
M/44'/60'/0'/0/ACCOUNT_INDEX

INDEX	ADDRESS	PRIVATE KEY	BALANCE (WEI)	NONCE	STATE
0	0XABF8A5876D375C7C6A63EDF9B0E0A2B6105C3494	0X67FA4E18BE93F969C336CBB604E7FF819A964FF6E76B33D67F25EE2D98A1F036	100000000000000000000 WEI	0	🔓
1	0X1AD2A30BA2442E901612F8ABF1B53A9720EC8CB8	0X959FB344BFD937936200CEA1A97ED59614C8A0BF38097BB4B3FEE838E74BD62F	100000000000000000000 WEI	0	🔓
2	0X0963B4381EC55BC5ADF77ECC24880595CFF78906	0XD8050E8EFE9AEC26831B0830AFFA1341ADE5090CB653B81DC5B0884162B01978	100000000000000000000 WEI	0	🔓
3	0X2DDA7CB33F095F1520285668D399A297F715C0F4	0XCC757BB6331C1C19E018A8D263AA3BF014941619FF363ABEF1FB29263F056BF4	100000000000000000000 WEI	0	🔓
4	0X71DD84041AFA8259B3470ECD679FEF9D93B67AFA	0X5651151758A9A9E002C5327C3FE80ACCA289D41A8CCA69D7F30928F81A7DEA41	100000000000000000000 WEI	0	🔓
5	0X417063D1992511E0768C021888044179B57ACCBD	0XE467DCC62BC03935F9C8DCCEA278B265A88FFF962BF460D0F3C0EF55113AC90A	100000000000000000000 WEI	0	🔓

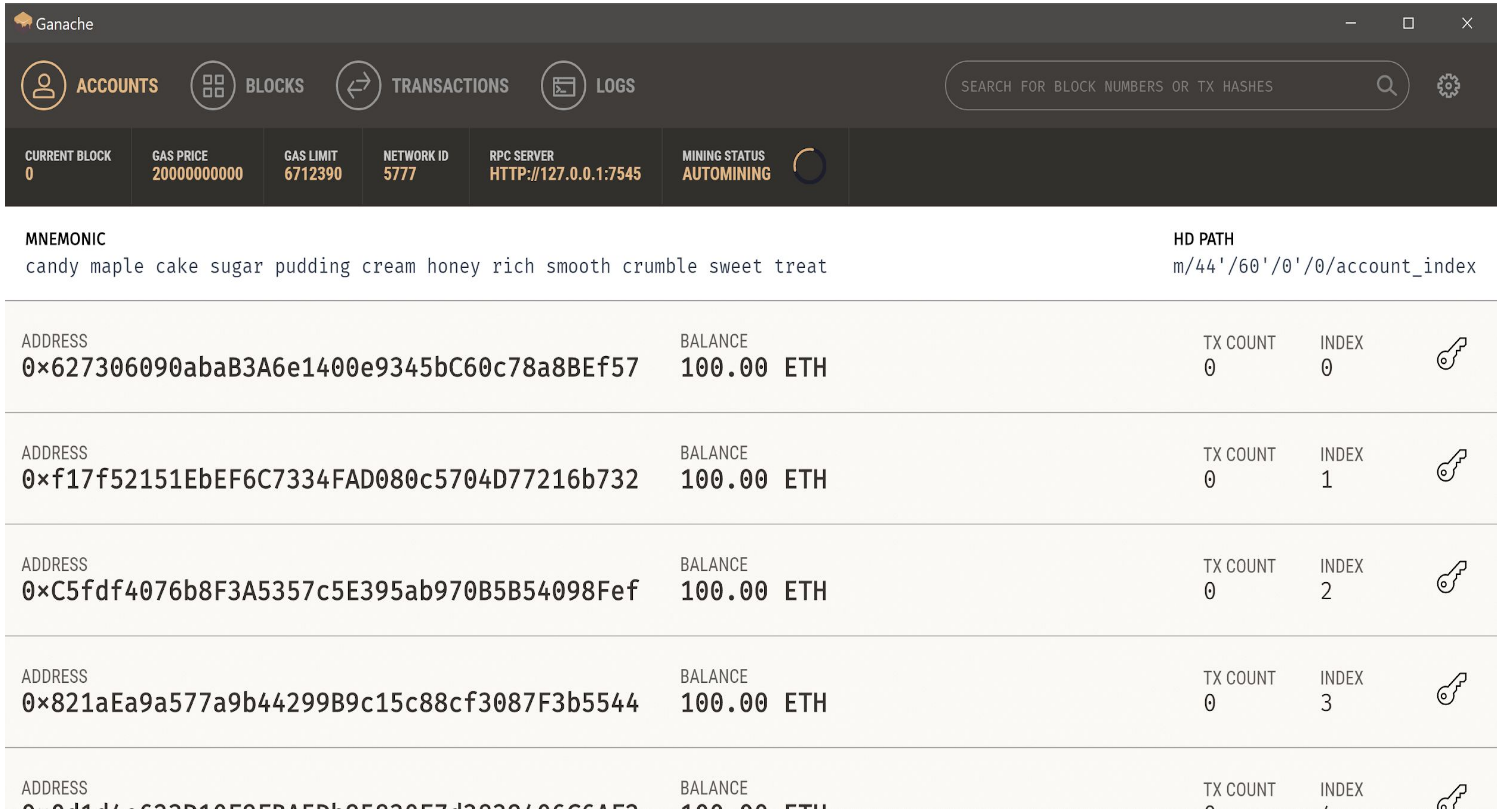
LOG OUTPUT

```
[2:03:09 PM] TESTRPC STARTED
[2:03:09 PM] eth_accounts
[2:03:09 PM]    > {
   >   "jsonrpc": "2.0",
   >   "id": 1,
   >   "method": "eth_accounts",
   >   "params": □,
   >   "external": true
   > }
[2:03:09 PM]  <   {
 <     "id": 1,
 <     "jsonrpc": "2.0",
 <     "result": [
 <       "0×abf8a5876d375c7c6a63edf9b0e0a2b6105c3494",
 <       "0×1ad2a30ba2442e901612f8abf1b53a9720ec8cb8",
 <       "0×0963b4381ec55bc5adf77ecc24880595cff78906",
 <       "0×2dda7cb33f095f1520285668d399a297f715c0f4",
 <       "0×71dd84041afa8259b3470ecd679fef9d93b67afa",
 <       "0×417063d1992511e0768c021888044179b57accbd"
 <     ]
 <   }
[2:49:31 PM] Stopping Mining....
[2:49:57 PM] Starting Mining....
[2:50:05 PM] Stopping Mining....
```

AUTHOR: NICK ZOGHB

GANACHE

...IN WITH THE NEW



CURRENT BLOCK	GAS PRICE	GAS LIMIT	NETWORK ID	RPC SERVER	MINING STATUS
0	20000000000	6712390	5777	HTTP://127.0.0.1:7545	AUTOMINING

MNEMONIC
candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

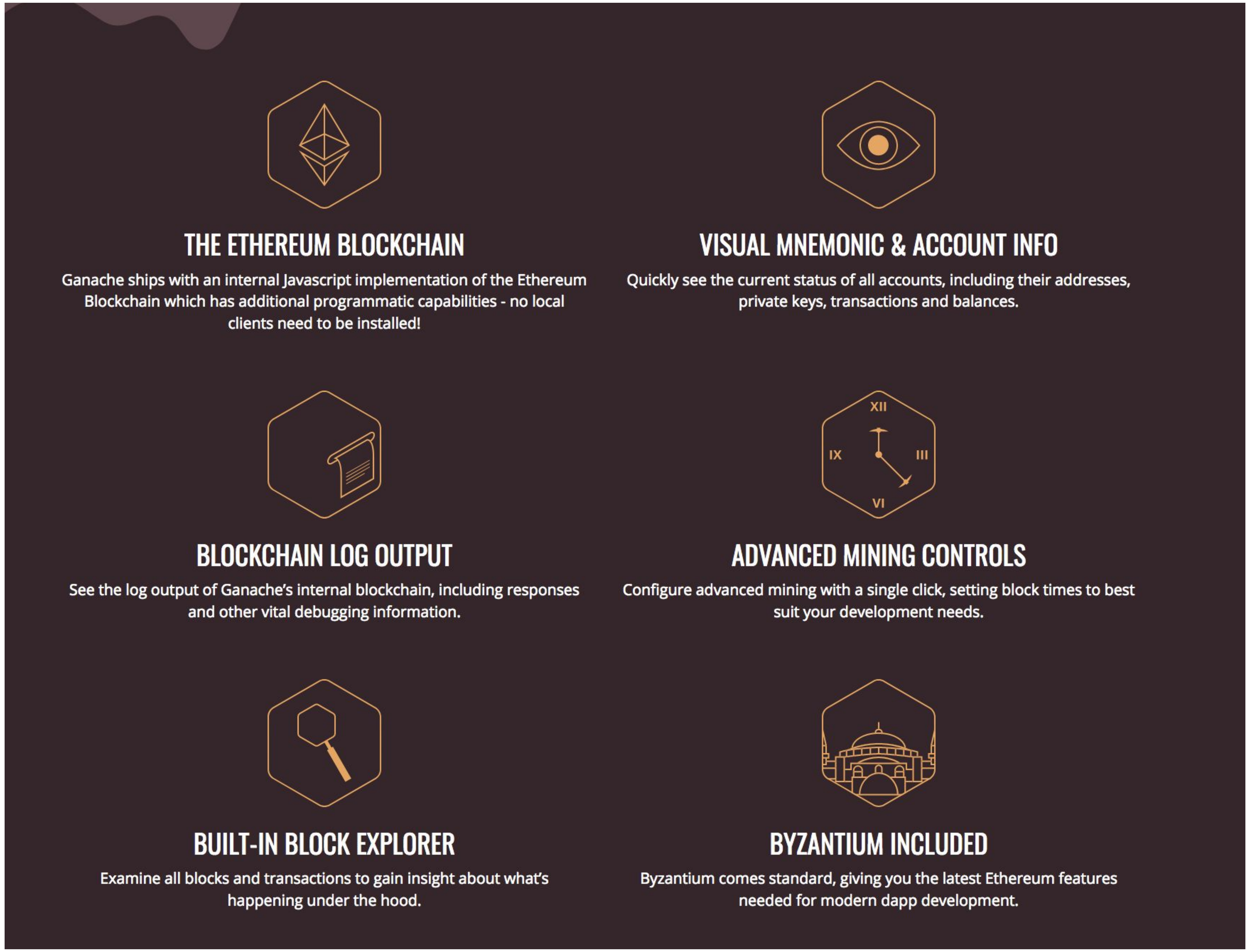
HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×627306090abaB3A6e1400e9345bC60c78a8BEf57	100.00 ETH	0	0
0×f17f52151EbEF6C7334FAD080c5704D77216b732	100.00 ETH	0	1
0×C5fdf4076b8F3A5357c5E395ab970B5B54098Fef	100.00 ETH	0	2
0×821aEa9a577a9b44299B9c15c88cf3087F3b5544	100.00 ETH	0	3

AUTHOR: NICK ZOGHB

GANACHE

...IN WITH THE NEW



THE ETHEREUM BLOCKCHAIN

Ganache ships with an internal Javascript implementation of the Ethereum Blockchain which has additional programmatic capabilities - no local clients need to be installed!

VISUAL MNEMONIC & ACCOUNT INFO

Quickly see the current status of all accounts, including their addresses, private keys, transactions and balances.

BLOCKCHAIN LOG OUTPUT

See the log output of Ganache's internal blockchain, including responses and other vital debugging information.

ADVANCED MINING CONTROLS

Configure advanced mining with a single click, setting block times to best suit your development needs.

BUILT-IN BLOCK EXPLORER

Examine all blocks and transactions to gain insight about what's happening under the hood.

BYZANTIUM INCLUDED

Byzantium comes standard, giving you the latest Ethereum features needed for modern dapp development.

AUTHOR: NICK ZOGHB

ESPRESSO

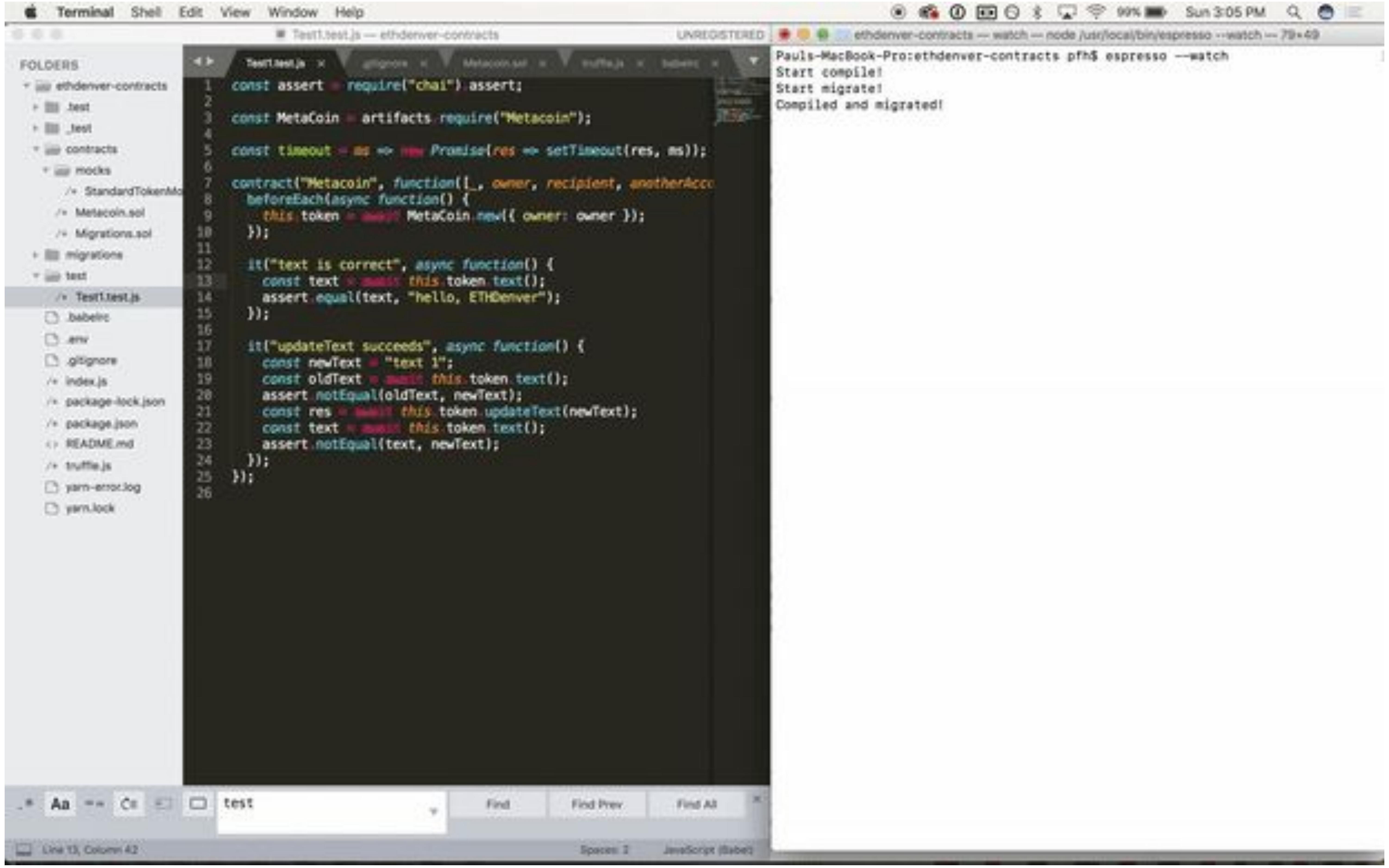
A BETTER FUTURE?

***espresso* is a testing framework for Solidity smart contracts, written in Javascript. Features include:**

- ✓ Test parallelization
- ✓ Hot-reloading and running of tests (with a **`--watch`** flag)
- ✓ Isolated test RPC, so you don't have to have an RPC like *Ganache* running or muddy your development RPC
- ✓ Backwards compatibility with truffle test

ESPRESSO

A BETTER FUTURE?



```javascript
const assert = require("chai").assert;

const MetaCoin = artifacts.require("Metacoin");

const timeout = ms => new Promise(res => setTimeout(res, ms));

contract("Metacoin", function([_, owner, recipient, anotherAcc
  beforeEach(async function() {
    this.token = await MetaCoin.new({ owner: owner });
  });

  it("text is correct", async function() {
    const text = await this.token.text();
    assert.equal(text, "hello, ETHDenver");
  });

  it("updateText succeeds", async function() {
    const newText = "text 1";
    const oldText = await this.token.text();
    assert.notEqual(oldText, newText);
    const res = await this.token.updateText(newText);
    const text = await this.token.text();
    assert.notEqual(text, newText);
  });
});
```

```
Pauls-MacBook-Pro:ethdenver-contracts pfh$ espresso --watch
Start compile!
Start migrate!
Compiled and migrated!
```

AUTHOR: NICK ZOGHB

4 TOKENS

HOW DID WE GET HERE

FROM BITCOIN TO ERC20

- Colored coins
  - Layered on top of Bitcoin, creating a new set of information about coins being exchanged
  - Bitcoins "colored" with specific attributes
- Colored coins => Smart assets
  - Tie ownership to real-world assets (i.e. gold on blockchain)

(Source)

AUTHOR: NICK ZOGHB

APP-COIN VS PROTOCOL TOKEN

FROM BITCOIN TO ERC20

“Open platforms have proved difficult to create because it has been historically difficult to monetize them … Now, however, the developers of a cloud storage service can incorporate a scarce access-token, an appcoin, into the design, distribute that token to users, retain some amount of the token for themselves, and if the platform proves popular, the token (alongside the holdings of the developers) will grow in value and remunerate the developers for providing a public good. This new model challenges the concept of equity as traditionally understood, and carries entirely different risks and rewards.”

(Source)

APP-COIN VS PROTOCOL TOKEN

FROM BITCOIN TO ERC20

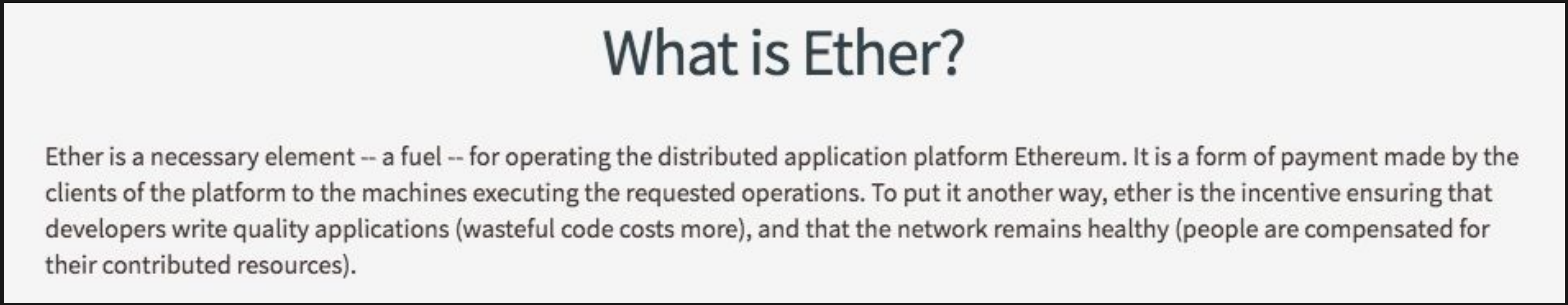
Augur (REP) is an example of an application built on top of protocols:

- Decentralized Oracle Protocol
  - *REP* provides a financial incentive for a network of nodes to arrive at a consensus around real-world happenings
- Exchange Protocol

Protocol tokens are decoupled from specific use-cases (even unrelated to prediction markets) and nodes active on either protocol may not be the same.

ALIGNING INCENTIVES IN THE NETWORK

BACK TO BASICS



What is Ether?

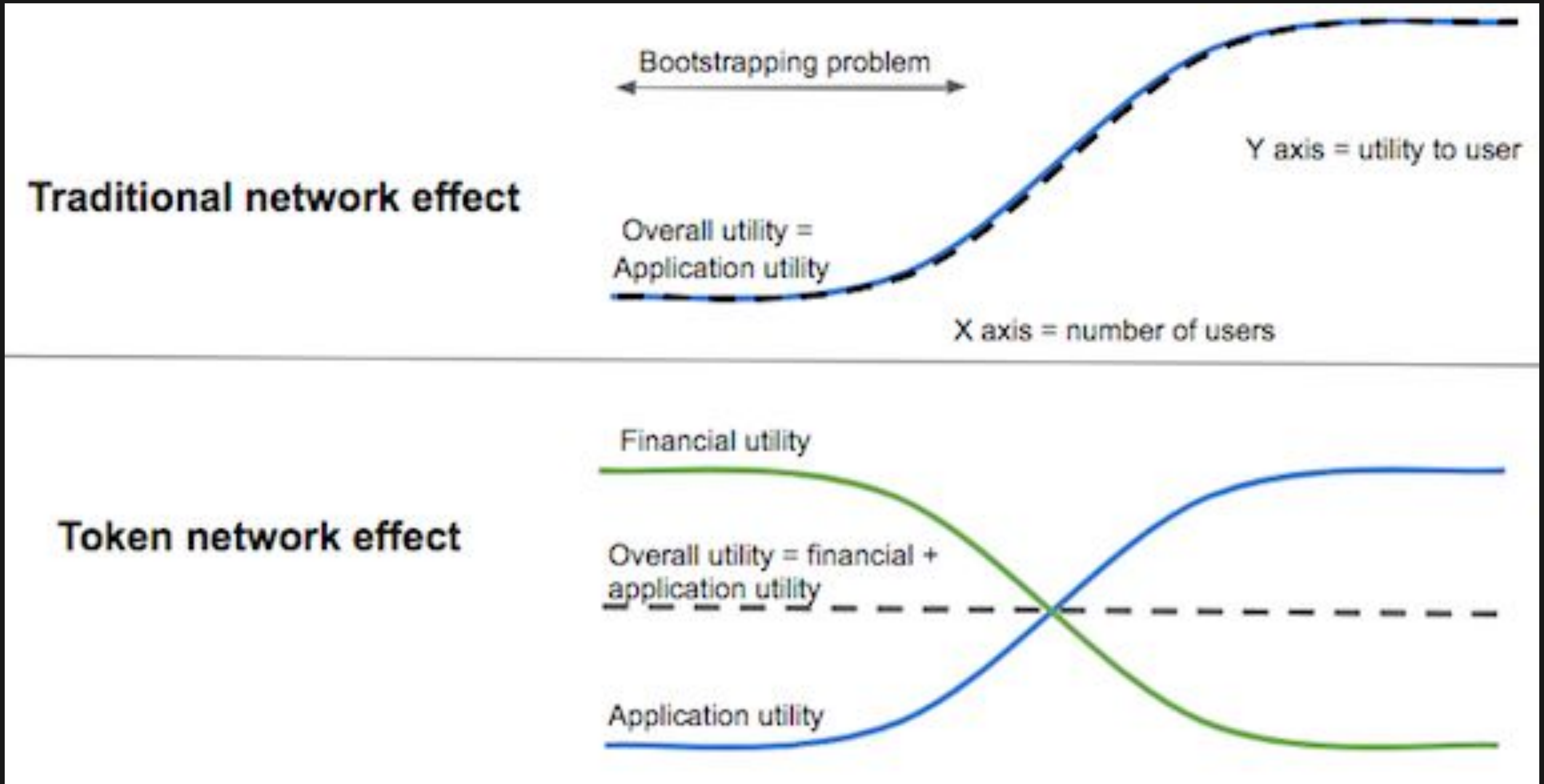
Ether is a necessary element -- a fuel -- for operating the distributed application platform Ethereum. It is a form of payment made by the clients of the platform to the machines executing the requested operations. To put it another way, ether is the incentive ensuring that developers write quality applications (wasteful code costs more), and that the network remains healthy (people are compensated for their contributed resources).

(Source)

AUTHOR: NICK ZOGHB

ALIGNING INCENTIVES IN THE NETWORK

BACK TO BASICS



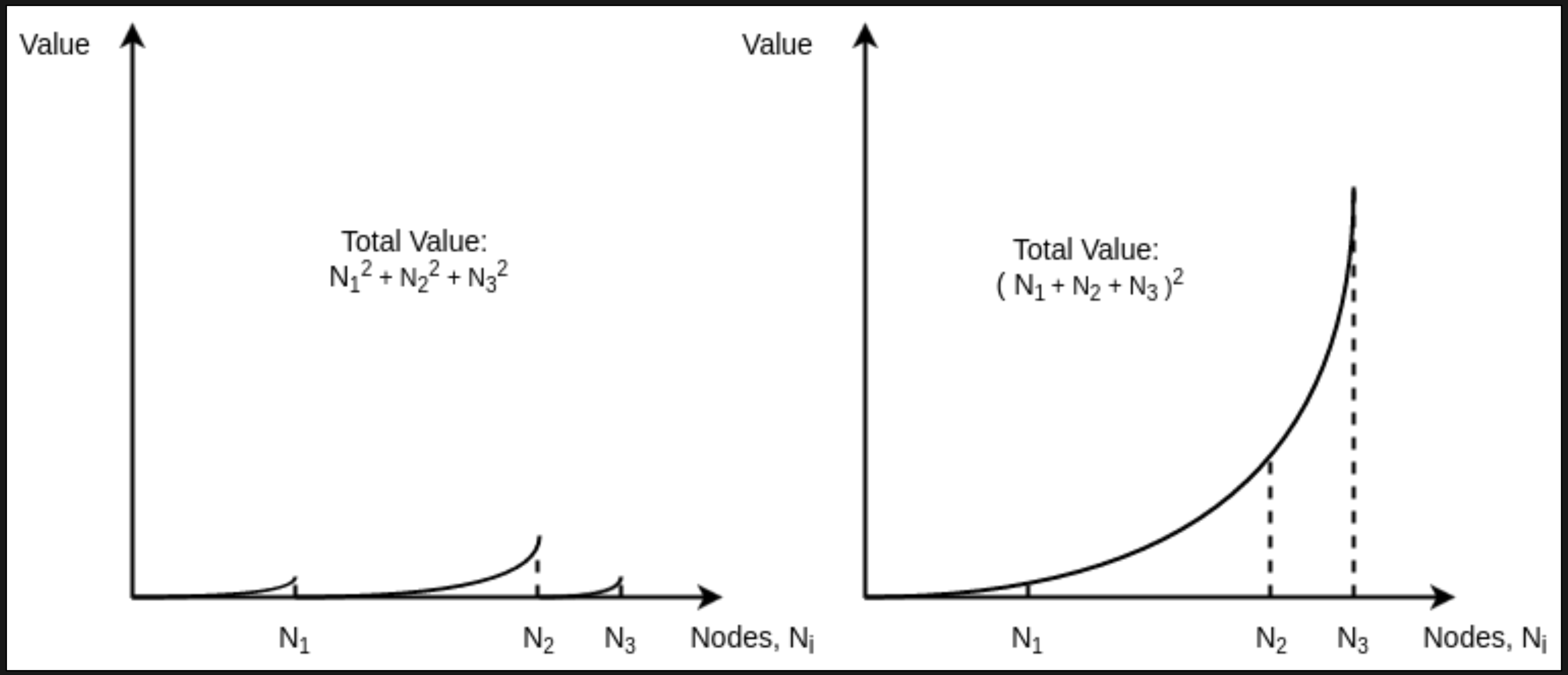
(Source)

ALIGNING INCENTIVES IN THE NETWORK

NOT A WELCOME INNOVATION

- Protocol tokens provide the financial incentives needed to drive a cryptoeconomic protocol which may or may not be implemented within an Ethereum smart contract
- DApps act as access points into protocols but have no cryptoeconomic backbone; App coins align dApp developer and investor incentives
- Redundancy introduces unneeded costs for end users and causes a splintering

(Source)



Metcalfe's Law: $V \propto N^2$

$(\sum N_i)^2 \geq \sum(N_i^2)$

AUTHOR: NICK ZOGHB

BLOCKCHAIN AT BERKELEY

A WORD ON ETHICS

FROM BITCOIN TO ERC20

golem

Filecoin

- UNREGISTERED SECURITIES REGULATION

- Examples of interesting projects:
  - Filecoin
  - Golem

Bancor

- Examples of red flags:
  - Buzzwords
  - Lack of code base
  - Mishandled ICO
    - Ethereum network congestion
  - Drama (here)

QUESTIONS?

SEE YOU NEXT TIME

ERC, Token Standards

ERC20

Token Examples

Other EIPs

Frameworks and Tools

Launching an ICO