

Unit testing of smart contracts



Why do we need to pay even higher attention of unit testing our smart contracts ?

- lots of money involved
- bugs affects accounts holding real money
- once a bug is introduced, not easy to fix (if it is possible at all)
- soft / hard forks



What do we need to cover with unit tests ?

- positive test cases (what should happen)
- negative test cases (what should NOT happen)



When do we need to execute unit tests ?

- updating old functionality
- adding new features
- before each deployment



Available development / testing frameworks

- **Truffle: Javascript & Solidity (Mocha/Chai)**
- **Embark: Javascript (Mocha)**
- **Etherlime: Javascript (Mocha/Chai asserts)**
- **Dapple: Solidity**
- **Populus: Python**
- **Go-Ethereum: Go**



Writing sample unit test with Etherlime

- unit tests location
- basic unit test structure
 - `require('etherlime')`
 - `.describe()`
 - `.it()`
 - `asserts`
- running the tests
 - etherlime test



Hooks

- `.before()`
- `.after()`
- `.beforeEach()`
- `.afterEach()`



Pending unit tests



Exclusive unit tests

- `.skip()`
- `.xdescribe()`
- `.xit()`



Inclusive unit tests

- `.only()`



Events

- `utils.hasEvent(txReceipt, contract, eventName)`
- `utils.parseLogs(txReceipt, contract, eventName)`



Exceptions

- `require`
- `function modifiers`
- `assert.revert(promiseOfFailingTransaction)`



Time travel

- `utils.timeTravel(provider, seconds)`
- `utils.setTimeTo(provider, timestamp)`



Code coverage

- `etherlime coverage`



Best practices & guidelines

- readable code & comments
- positive, negative cases, events
- overflows / underflows
- for each function
 - who can execute it
 - who can NOT execute it
 - when could be executed
- write as many unit tests as possible



Best practices & tools

- use automated code review tools
 - significantly decreases chances of mistakes
 - developers save up to 30-50% of the reviewing time
- external audit
- code coverage
 - highest coverage, the best (> 90%)



Further reading

- Mocha framework documentation

<https://mochajs.org/>

- etherlime testing & coverage

<https://etherlime.readthedocs.io/en/latest/cli/test.html>



Homework

- add (as much as you can) unit tests to your crypto cars app
- get at least 90% code coverage



Q & A



Thanks !

vlado@limechain.tech