

Remix tests

This tutorial covers basics of unit testing solidity smart contracts using [remix-tests](#)

Introduction

[remix-tests](#) by Iuri Matias is available as npm package & integrated in Remix IDE & Etheratom IDE.

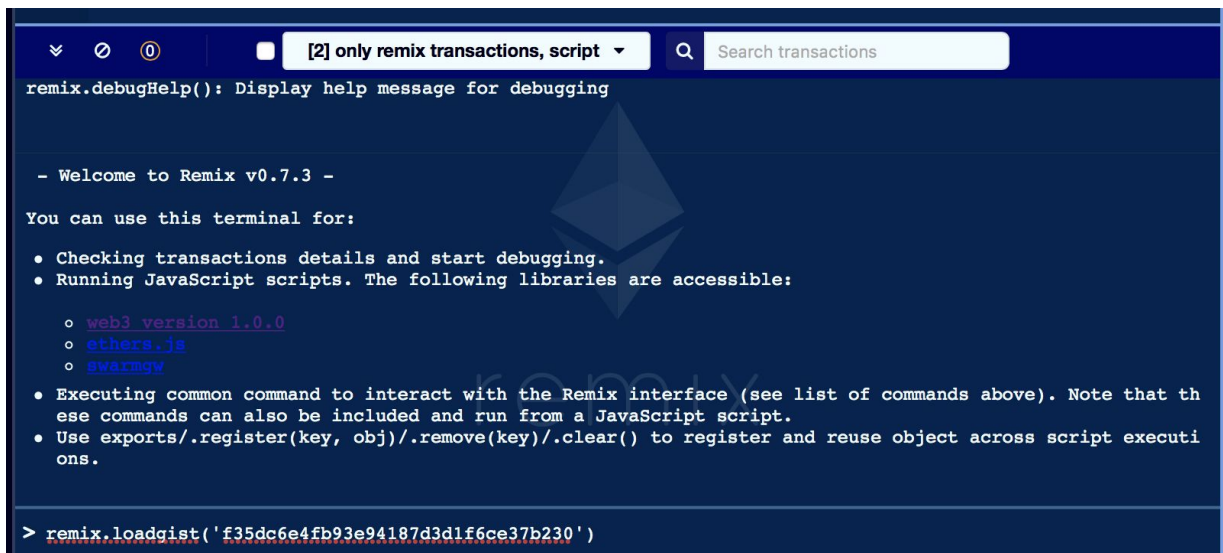
Why remix-tests ?

[remix-tests](#) lets developers' write unit testing in Solidity. It can be installed as a command line tool with ``npm -g install remix-tests``. It is also available in [Remix IDE](#) & [Etheratom](#) as a plugin.

Lets start to write Unit Tests for AwardToken

Load the AwardToken_tests by running the command

```
remix.loadgist('f35dc6e4fb93e94187d3d1f6ce37b230')
```



```
remix.debugHelp(): Display help message for debugging

- Welcome to Remix v0.7.3 -

You can use this terminal for:

• Checking transactions details and start debugging.
• Running JavaScript scripts. The following libraries are accessible:
  ◦ web3_version 1.0.0
  ◦ ethers.js
  ◦ swarmgw
• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.

> remix.loadgist('f35dc6e4fb93e94187d3d1f6ce37b230')
```

Writing unit tests

- Test file names should be suffixed by `_test.sol`
- Use ``import "remix_tests.sol";`` statement to import tests module.
- Use Assert library to check `Assert.equal()`, `Assert.notEqual()`, `Assert.greaterThan()`, `Assert.lessThan()`
- Use `beforeEach()`, `beforeAll()` functions to run tasks before running tests.

Running tests

Unit tests can be run from Tests tab by clicking Run tests button.

```
11 }
12 function addressShouldNotBe_0x000() public constant returns (bool) {
13     address zro = 0x0000000000000000000000000000000000000000000000000000000000000000;
14     return Assert.notEqual(address(awtkn), zro, "address should not be 0");
15 }
16
17 function totalSupplyShouldBe_0() public constant returns (bool) {
18     return Assert.equal(awtkn.totalSupply(), 0, "total supply should be 0");
19 }
20
21 function startRoundShouldReturn_true() public constant returns (bool) {
22     return Assert.equal(awtkn.startRound(), true, "startRound return is false");
23 }
24
```

For more details, see [How to test smart contracts guide](#) in our documentation.


Generate test file

☒ gist/AwardToken_test.sol

Run Tests

Test results

remix-tests results shows passing and failing tests.



The screenshot displays the test results for a Solidity contract named `gist/AwardToken_test.sol`. At the top, there is a blue header bar with a checkmark icon and the file name. Below this is a light blue button labeled "Run Tests". The main area shows two test suites. The first suite, `gist/AwardToken_test.sol (AwardTokenBalanceTest)`, contains one failing test: `✗ (Balance should be greater than 0)`, highlighted in a red box. The second suite, `gist/AwardToken_test.sol (AwardTokenTest)`, contains four passing tests, each in a green box: `✓ (Start round should return true)`, `✓ (Total supply should be 0)`, `✓ (Address should not be 0x000)`, and `✓ (Owner should be tester)`. At the bottom, a summary section shows `gist/AwardToken_test.sol` with `4 passing (4s)` and `1 failing`. The failing test is detailed as `AwardTokenBalanceTest - Balance should be greater than 0` with the message `function returned false` in red text.

gist/AwardToken_test.sol

Run Tests

gist/AwardToken_test.sol (AwardTokenBalanceTest)

✗ (Balance should be greater than 0)

gist/AwardToken_test.sol (AwardTokenTest)

✓ (Start round should return true)

✓ (Total supply should be 0)

✓ (Address should not be 0x000)

✓ (Owner should be tester)

gist/AwardToken_test.sol

4 passing (4s)

1 failing

AwardTokenBalanceTest - Balance should be greater than 0
function returned false

remix-tests CLI tool

```
[MacBook-Pro:~ 0mkar$ npm -g install remix-tests  
npm WARN deprecated babel-preset-es2017@6.24.1: 🙌 Thanks for using Babel:  
to update!  
npm WARN deprecated babel-preset-es2015@6.24.1: 🙌 Thanks for using Babel:
```

Install remix-tests with `npm -g install remix-tests`

```
[MacBook-Pro:~ 0mkar$ remix-tests -h  
Usage: remix-tests [options] [command]  
  
Options:  
  -V, --version          output the version number  
  -v, --verbose <level> run with verbosity  
  -h, --help             output usage information  
  
Commands:  
  version               output the version number  
  help                 output usage information  
MacBook-Pro:~ 0mkar$
```

remis-tests cli help

remix-tests CLI tool

```
MacBook-Pro:SafeMath 0mkar$ ls
SafeMath.sol      SafeMathProxy.sol      SafeMath_test.sol
MacBook-Pro:SafeMath 0mkar$ remix-tests SafeMath_test.sol
```

```
🔗:: Running remix-tests - Unit testing for solidity :: 🔗
```

```
[14:54:22] payload method is eth_accounts
[14:54:22] payload method is eth_estimateGas
[14:54:22] payload method is eth_gasPrice
[14:54:22] payload method is eth_sendTransaction
[14:54:22] payload method is eth_getTransactionReceipt
[14:54:22] payload method is eth_getCode
'creation of library remix_tests.sol:Assert pending...'
[14:54:22] payload method is eth_estimateGas
[14:54:22] payload method is eth_gasPrice
[14:54:22] payload method is eth_sendTransaction
[14:54:22] payload method is eth_getTransactionReceipt
[14:54:22] payload method is eth_getCode
[14:54:22] payload method is eth_estimateGas
[14:54:22] payload method is eth_gasPrice
[14:54:22] payload method is eth_sendTransaction
[14:54:22] payload method is eth_getTransactionReceipt
[14:54:22] payload method is eth_getCode
```

```
■ SafeMathTest
[14:54:22] payload method is eth_gasPrice
[14:54:22] payload method is eth_sendTransaction
[14:54:22] payload method is eth_getTransactionReceipt
[14:54:22] payload method is eth_call
✓ Unsafe multiplication should overflow
[14:54:22] payload method is eth_call
✓ Safe modulus should revert
[14:54:22] payload method is eth_call
✓ Safe subtract should revert
[14:54:22] payload method is eth_call
✓ Safe addition should revert
[14:54:22] payload method is eth_call
✓ Safe division by zero should revert
[14:54:23] payload method is eth_call
✓ Safe multiplication should revert
[14:54:23] payload method is eth_call
✓ Unsafe subtract should underflow
[14:54:23] payload method is eth_call
✓ Unsafe addition should overflow
```

```
8 passing (8s)
```

- remix-tests SafeMath_test.sol will run all the tests written in SafeMath_test.sol file
- remix-tests SafeMath/ will run all the files suffixed with _test.sol in SafeMath directory
- remix-tests cli tool list all the passing and failing tests and shows total number of passing & failing tests in the end

Debug test contracts

- Deploy YourContractTest from run tab
- Debug using remix-debugger

1. Select SafeMathTest from contracts dropdown

2. Deploy contract

3. Click Debug button to debug this transaction using remix-debugger

