

CST4125: Blockchain Development

Week: 18

Title: Security & Unit Testing

Dr Ian Mitchell



2023

smerf.net

CST4125:L18

Winter 2023 1 / 45

Contact and Office Hours



Contact Details

- Name: Dr Ian Mitchell
- Room: TG10
- Address: Middlesex University, Computer Science, London, NW4 4BT
- email: smerf.net

Office Hours

- During term time only
- When: Autumn Term: Mondays 1100-1300hrs
- Please read notifications or emails
- There are occasions that these could be arranged online, e.g., due to industrial action or inclement weather

smerf.net

CST4125:L18

Winter 2023 2 / 45

Contact and Office Hours



Contact Details

- Name: Dr Ian Mitchell
- Room: TG10
- Address: Middlesex University, Computer Science, London, NW4 4BT
- email: smerf.net

smerf.net

CST4125:L18

Winter 2023 2 / 45

Lecture Objectives



- Private Networks
- Truffle, Ganache, node, geth
- Local remix
- Unit testing
- Functions
- Events & Logging
- Debugging
- Etherscan

smerf.net

CST4125:L18

February 8, 2023 4 / 45

Error Handling



- Inadvertent
- Robust contracts
- Run-time

smerf.net

CST4125:L18

Winter 2023 3 / 45

smerf.net

CST4125:L18

February 8, 2023 5 / 45

Error Handling



- Inadvertent
- Robust contracts
- Run-time
 - out-of-gas errors
 - divide by zero
 - data overflow
 - array errors
- throw is now obsolete
- no try ... catch
- require, revert, assert

smerf.net

CST4125:L18

February 8, 2023 5 / 45

require



- prerequisites
- conditions
- `require(< condition >)`
- evaluates as boolean
- declare conditions that are satisfied before executing code

smerf.net

CST4125:L18

February 8, 2023 6 / 45

require



```
1 uint32 constant max=8000;
2 uint32 constant min=1000;
3 event lessThan(string);
4 event greaterThan(string);
5 event inRange(string);
6 function lessThanX(uint32 _x) public {
7     require(_x <= max);
8     emit lessThan("Integer is less than");
9 }
10 function greaterThanX(uint32 _x) public {
11     require(_x >= min);
12     emit greaterThan("Integer is greater than");
13 }
14 function inRangeFn(uint32 _x) public {
15     require(_x>=min);
16     require(_x<=max);
17     emit inRange("Integer is in Range");
18 }
19 }
```

smerf.net

CST4125:L18

February 8, 2023 7 / 45

require with help



```
1 contract test{
2     uint32 constant max=8000;
3     uint32 constant min=1000;
4     event lessThan(string);
5     event greaterThan(string);
6     event inRange(string);
7     function lessThanX(uint32 _x) public {
8         require(_x <= max, "less than 8000");
9         emit lessThan("Integer is less than");
10    }
11    function greaterThanX(uint32 _x) public {
12        require(_x >= min, "greater than 1000");
13        emit greaterThan("Integer is greater than");
14    }
15    function inRangeFn(uint32 _x) public {
16        require(_x>=min, "> 1000");
17        require(_x<=max, "< 8000");
18        emit inRange("Integer is in Range");
19    }
20 }
```

smerf.net

CST4125:L18

February 8, 2023 8 / 45

Gate condition



- Make some access control rules
- inherit them
- include them as modifiers
- executed in order of modifier

smerf.net

CST4125:L18

February 8, 2023 9 / 45

Access Modifiers



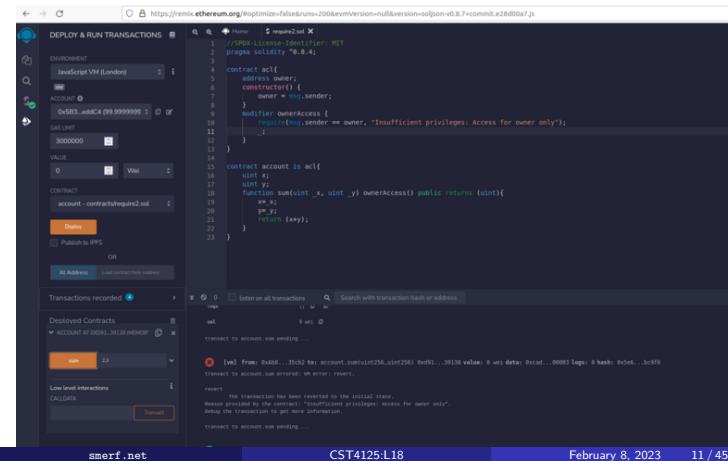
```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3 contract acl{
4     address owner;
5     constructor() {
6         owner = msg.sender;
7     }
8     modifier ownerAccess {
9         require(msg.sender == owner, "Insufficient privileges: Access for owner only");
10    }
11 }
12 contract account is acl{
13     uint x;
14     uint y;
15     function sum(uint _x, uint _y) ownerAccess() public returns (uint){
16         x=_x;
17         y=_y;
18         return (x+y);
19     }
20 }
```

smerf.net

CST4125:L18

February 8, 2023 10 / 45

Access Modifiers



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3
4 contract test{
5     uint32 constant max=8000;
6     uint32 constant min=1000;
7     event lessThan(string);
8     event greaterThan(string);
9     event inRange(string);
10    function sum(uint32 _x, uint32 _y) public returns (uint32){
11        if (inRangeFn(_x) && inRangeFn(_y)){
12            assert(_x+_y>=max);
13            return uint32(_x+_y);
14        }
15    }
16    function inRangeFn(uint32 _x) public returns (bool) {
17        require(_x>=min, " > 1000");
18        require(_x<=max, " < 8000");
19        return (true);
20    emit inRange("Integer is in Range");
21 }
22 function lessThanX(uint32 _x) public {
23     require(_x <= max, "less than 8000");
24     emit lessThan("Integer is less than");
25 }
26 function greaterThanX(uint32 _x) public {
27     require(_x >= min, "greater than 1000");
28     emit greaterThan("Integer is greater than");
29 }
30
31
32 }
```

assert

- same syntax as require statement
- does not accept an error message as a parameter (1 parameter)
- accepts a condition that either evaluates to true or false
- If it is true then executes code, else throws exception
- When to assert?
- require: values from outside
- assert: state and condition of the function
- current state has become inconsistent
- expected outcome is true

Assert Example



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3
4 contract test{
5     uint32 constant max=8000;
6     uint32 constant min=1000;
7     event lessThan(string);
8     event greaterThan(string);
9     event inRange(string);
10    function sum(uint32 _x, uint32 _y) public returns (uint32){
11        if (inRangeFn(_x) && inRangeFn(_y)){
12            assert(_x+_y>=max);
13            return uint32(_x+_y);
14        }
15    }
16    function inRangeFn(uint32 _x) public returns (bool) {
17        require(_x>=min, " > 1000");
18        require(_x<=max, " < 8000");
19        return (true);
20    emit inRange("Integer is in Range");
21 }
22 function lessThanX(uint32 _x) public {
23     require(_x <= max, "less than 8000");
24     emit lessThan("Integer is less than");
25 }
26 function greaterThanX(uint32 _x) public {
27     require(_x >= min, "greater than 1000");
28     emit greaterThan("Integer is greater than");
29 }
30
31
32 }
```

revert

- no parameters or conditions
- revert throws an exception
- returns unused gas
- reverts to original state
- similar to require without conditions

revert Example



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3
4 contract testRevert{
5     function checkUint8(uint16 _x) public pure returns(bool){
6         if( _x > 255){
7             revert();
8         }
9         return (true);
10    }
11 }
```

Testing

- Testing increases gas consumption
- right balance
- use of comments
- testnet err on the side of caution
- mainnet be frugal, but comment

Truffle

- Open terminal in VM
- Create a directory, and move there
- `mkdir truffleProject && cd truffleProject`
- `truffle init`
- Then open the project in VSCode
- `code .`
- Compile
- Migrate
- Test

smerf.net

CST4125:L18

February 8, 2023 17 / 45



Contract

```
1 // SPDX-License-Identifier: GPL3.0
2 pragma solidity ^0.8.0;
3 contract myFirstContract{
4     uint8 sum;
5     function add(uint8 _x, uint8 _y) public returns(uint8){
6         require(_x+_y<256);
7         return uint8(_x+_y);
8     }
9 }
```

Figure: Listing for `myFirstContract.sol`.

smerf.net

CST4125:L18

February 8, 2023 18 / 45

Configuration file

```
37 networks: {
38     // Useful for testing. The 'development' name is special - truffle uses it by default
39     // if it's defined here and no other network is specified at the command line.
40     // You should run a client (like ganache, geth, or parity) in a separate terminal
41     // tab if you use this network and you must also set the 'host', 'port' and '
42     // network_id'
43     // options below to some value.
44     development: {
45         host: "127.0.0.1",           // Localhost (default: none)
46         port: 7545,                 // Standard Ethereum port (default: none)
47         network_id: "*",           // Any network (default: none)
48     },
49 }
```

Figure: Partial Listing for `truffle-config.js`.

smerf.net

CST4125:L18

February 8, 2023 19 / 45



Contract Configuration

```
1 const myFirstContract = artifacts.require("myFirstContract");
2 module.exports = function (deployer) {
3     deployer.deploy(myFirstContract);
4 };
```

Figure: Listing for `2_myFirstContract.js`. Notice filenaming convention

smerf.net

CST4125:L18

February 8, 2023 20 / 45

Project Structure

```
1 .
2 |-- build
3 |   '-- contracts
4 |       |-- Migrations.json
5 |       '-- myFirstContract.json
6 |-- contracts
7 |   '-- Migrations.sol
8 |   '-- myFirstContract.sol
9 |-- migrations
10 |  '-- 1_initial_migration.js
11 |  '-- 2_myFirstContract.js
12 |-- test
13 |-- tree.lst
14 '-- truffle-config.js
15
16 5 directories, 8 files
```

Figure: Directory and File structure of an Ethereum project

smerf.net

CST4125:L18

February 8, 2023 21 / 45



Compile and Migrate

- ➊ `truffle compile`
- ➋ `truffle migrate`

smerf.net

CST4125:L18

February 8, 2023 22 / 45

Compile and Migrate



- ➊ truffle compile
- ➋ truffle migrate
- ➌ truffle test

smerf.net

CST4125:L18

February 8, 2023 22 / 45

Truffle migrate

Transaction

smerf.net

CST4125:L18

February 8, 2023 23 / 45

JSON config file



```
26 "networks": {
27   "5777": {
28     "events": {},
29     "links": {},
30     "address": "0x948231d3CeD48A6FCf0A925144Ca9be47a270dd9",
31     "transactionhash": "0
32       x60829cdcf064ad5e5efc896b87e308d5fef31ba0c42450ec396d888b2a30dea9"
33 }
```

Figure: JSON config file for truffle project, myFirstContract.json

smerf.net

CST4125:L18

February 8, 2023 24 / 45

Unit Testing



- ➊ test the functions and their results
- ➋ expected results and test for failures
- ➌ compare expected and actual for match

smerf.net

CST4125:L18

February 8, 2023 25 / 45

Truffle Test



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3 import "truffle/Assert.sol";
4 import "truffle/DeployedAddresses.sol";
5 import "../contracts/myFirstContract.sol";
6 contract testMyFirstContract{
7   myFirstContract mfc;
8   function beforeAll() public{
9     mfc = myFirstContract(DeployedAddresses.myFirstContract());
10  }
11  function testAdd() public {
12    uint8 expected = 200;
13    uint8 result = mfc.add(100,100);
14    Assert.equal(expected, result, "Should be equal");
15  }
16 }
```

Figure: Truffle test file for truffle project, test/testMyFirstContract.sol

smerf.net

CST4125:L18

February 8, 2023 26 / 45

Truffle Test Output



smerf.net

CST4125:L18

February 8, 2023 27 / 45

Truffle Test

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3 import "truffle/Assert.sol";
4 import "truffle/DeployedAddresses.sol";
5 import "../contracts/myFirstContract.sol";
6 contract testMyFirstContract{
7     myFirstContract mfc;
8     function beforeAll() public{
9         mfc = myFirstContract(DeployedAddresses.myFirstContract());
10    }
11    function testAdd() public {
12        uint8 expected = 200;
13        uint8 actual = mfc.add(100,100);
14        Assert.equal(expected, actual, "Should be equal");
15    }
16    function testRequire() public{
17        uint16 expected = 400;
18        uint8 actual = mfc.add(200,200);
19        Assert.equal(actual, expected, "200 + 200 > 400");
20    }
21 }
```

Figure: Truffle test file for truffle project, test/testMyFirstContract1.sol

smerf.net CST4125:L18 February 8, 2023 28 / 45

Truffle Test Output

```

12:11:18:57
truffle test
  testMyFirstContract1
    ✓ testAdd (1ms)
      Contract: myFirstContract
      1. myFirstContractIdentifier: MIT
      2. DeployedAddresses: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      3. assert: truffle/Test.sol:10
      4. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      5. Deployer: 0x724c75f56004fb0f5949ff80187575157c26
      6. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      7. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      8. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      9. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      10. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      11. Function testAdd() public {
      12.     uint8 expected = 200;
      13.     uint8 actual = mfc.add(100,100);
      14.     assert.equal(expected, actual, "Should be equal");
      15. }
      16. Function testRequire() public{
      17.     uint16 expected = 400;
      18.     uint8 actual = mfc.add(200,200);
      19.     assert.equal(actual, expected, "200 + 200 > 400");
      20. }
      21. }

  testMyFirstContract
    ✓ testAdd (1ms)
      Contract: myFirstContract
      1. myFirstContractIdentifier: MIT
      2. DeployedAddresses: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      3. assert: truffle/Test.sol:10
      4. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      5. Deployer: 0x724c75f56004fb0f5949ff80187575157c26
      6. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      7. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      8. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      9. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      10. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      11. Function testAdd() public {
      12.     uint8 expected = 200;
      13.     uint8 actual = mfc.add(100,100);
      14.     assert.equal(expected, actual, "Should be equal");
      15. }
      16. Function testRequire() public{
      17.     uint16 expected = 400;
      18.     uint8 actual = mfc.add(200,200);
      19.     assert.equal(actual, expected, "200 + 200 > 400");
      20. }
      21. }

  testMyFirstContract
    ✓ testAdd (1ms)
      Contract: myFirstContract
      1. myFirstContractIdentifier: MIT
      2. DeployedAddresses: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      3. assert: truffle/Test.sol:10
      4. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      5. Deployer: 0x724c75f56004fb0f5949ff80187575157c26
      6. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      7. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      8. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      9. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      10. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      11. Function testAdd() public {
      12.     uint8 expected = 200;
      13.     uint8 actual = mfc.add(100,100);
      14.     assert.equal(expected, actual, "Should be equal");
      15. }
      16. Function testRequire() public{
      17.     uint16 expected = 400;
      18.     uint8 actual = mfc.add(200,200);
      19.     assert.equal(actual, expected, "200 + 200 > 400");
      20. }
      21. }

  testMyFirstContract
    ✓ testAdd (1ms)
      Contract: myFirstContract
      1. myFirstContractIdentifier: MIT
      2. DeployedAddresses: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      3. assert: truffle/Test.sol:10
      4. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      5. Deployer: 0x724c75f56004fb0f5949ff80187575157c26
      6. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      7. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      8. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      9. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      10. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      11. Function testAdd() public {
      12.     uint8 expected = 200;
      13.     uint8 actual = mfc.add(100,100);
      14.     assert.equal(expected, actual, "Should be equal");
      15. }
      16. Function testRequire() public{
      17.     uint16 expected = 400;
      18.     uint8 actual = mfc.add(200,200);
      19.     assert.equal(actual, expected, "200 + 200 > 400");
      20. }
      21. }

  testMyFirstContract
    ✓ testAdd (1ms)
      Contract: myFirstContract
      1. myFirstContractIdentifier: MIT
      2. DeployedAddresses: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      3. assert: truffle/Test.sol:10
      4. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      5. Deployer: 0x724c75f56004fb0f5949ff80187575157c26
      6. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      7. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      8. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      9. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      10. myFirstContract: 0x5b30387d98071121b832cf0ab10fd9a9190682e5ea
      11. Function testAdd() public {
      12.     uint8 expected = 200;
      13.     uint8 actual = mfc.add(100,100);
      14.     assert.equal(expected, actual, "Should be equal");
      15. }
      16. Function testRequire() public{
      17.     uint16 expected = 400;
      18.     uint8 actual = mfc.add(200,200);
      19.     assert.equal(actual, expected, "200 + 200 > 400");
      20. }
      21. }
```

smerf.net CST4125:L18 February 8, 2023 29 / 45

Ganache Network with MetaMask

Address	Balance
0x5b30387d98071121b832cf0ab10fd9a9190682e5ea	100.00 ETH
0x724c75f56004fb0f5949ff80187575157c26	100.00 ETH
0x848e605e37cf56004fb0f5949ff80187575157c26	100.00 ETH
0x878130580712cf490de382f8a63267782c6e44fbfa	100.00 ETH
0x9a943e68acfb8b32cb89c4ca945983c8980da	100.00 ETH
0x412fe5b76cd231cfc75f8d668551ba8080bc084e	100.00 ETH
0x45d5aaef00d93346251a6510d378cf078e41b5e3	100.00 ETH
0x756baaef00d93346251a6510d378cf078e41b5e3	100.00 ETH
0x96cc949c1a8b1649228aa554eac8f5a58c882	100.00 ETH

Need help? Contact MetaMask Support

smerf.net CST4125:L18 February 8, 2023 30 / 45

Add Ganache Account to MetaMask

Settings

- General
- Advanced
- Contracts
- Security & Privacy
- Networks
- Alerts
- Networks
- Experimental
- About

Networks

- Ethereum Mainnet
- Ropsten Test Network
- Morden Test Network
- Workshop Test Network
- Kovan Test Network
- Localhost

New RPC URL: Check ID: Currency Symbol:

You have no accounts.

smerf.net CST4125:L18 February 8, 2023 31 / 45

MetaMask Account Added

Address	Balance
0x5b30387d98071121b832cf0ab10fd9a9190682e5ea	100.00 ETH
0x724c75f56004fb0f5949ff80187575157c26	100.00 ETH
0x848e605e37cf56004fb0f5949ff80187575157c26	100.00 ETH
0x878130580712cf490de382f8a63267782c6e44fbfa	100.00 ETH
0x9a943e68acfb8b32cb89c4ca945983c8980da	100.00 ETH
0x412fe5b76cd231cfc75f8d668551ba8080bc084e	100.00 ETH
0x45d5aaef00d93346251a6510d378cf078e41b5e3	100.00 ETH
0x756baaef00d93346251a6510d378cf078e41b5e3	100.00 ETH
0x96cc949c1a8b1649228aa554eac8f5a58c882	100.00 ETH
Account 1	100.00 ETH

Need help? Contact MetaMask Support

smerf.net CST4125:L18 February 8, 2023 32 / 45

Another MetaMask Account Added

Address	Balance
0x5b30387d98071121b832cf0ab10fd9a9190682e5ea	100.00 ETH
0x724c75f56004fb0f5949ff80187575157c26	100.00 ETH
0x848e605e37cf56004fb0f5949ff80187575157c26	100.00 ETH
0x878130580712cf490de382f8a63267782c6e44fbfa	100.00 ETH
0x9a943e68acfb8b32cb89c4ca945983c8980da	100.00 ETH
0x412fe5b76cd231cfc75f8d668551ba8080bc084e	100.00 ETH
0x45d5aaef00d93346251a6510d378cf078e41b5e3	100.00 ETH
0x756baaef00d93346251a6510d378cf078e41b5e3	100.00 ETH
Account 1	100.00 ETH
Account 2	100.00 ETH

You have no transactions.

Need help? Contact MetaMask Support

smerf.net CST4125:L18 February 8, 2023 33 / 45

Send Ether between Accounts



The screenshot shows the MetaMask wallet interface within a Firefox browser window. The top navigation bar includes 'ACTIVITIES', 'ACCOUNTS', 'BLOCKS', 'TRANSACTIONS', 'CONTRACTS', 'EVENTS', and 'LOGS'. Below this, the 'METAMASK' section displays several Ethereum addresses with their current ETH balances. A central 'Send' dialog box is open, prompting the user to enter the recipient's address ('0x5d5a...'), the amount ('10 ETH'), and the gas price ('20'). The total estimated fee is shown as '0.00040 0.00040 ETH'.

smurf.net

CST4125:L18

February 8, 2023 34 / 45

Confirm Ether sent between Accounts



This screenshot shows the MetaMask wallet interface after a transaction has been confirmed. The 'Send' dialog has been closed, and a confirmation message is displayed: 'Confirmed transaction Transaction 1 confirmed!'. The transaction details are listed, including the recipient address ('0x5d5a...'), amount ('10 ETH'), and total fee ('0.00040 0.00040 ETH').

smurf.net

CST4125:L18

February 8, 2023 35 / 45

Pending TX



The screenshot shows the MetaMask wallet interface with a pending transaction. The transaction details are identical to the previous screenshots: recipient address ('0x5d5a...'), amount ('10 ETH'), and total fee ('0.00040 0.00040 ETH'). The status is labeled 'Pending'.

smurf.net

CST4125:L18

February 8, 2023 36 / 45

TX confirmed player 1



This screenshot shows the MetaMask wallet interface after a transaction has been confirmed. The transaction details are identical to the previous screenshots. The status is labeled 'Confirmed transaction Transaction 1 confirmed!'.

smurf.net

CST4125:L18

February 8, 2023 37 / 45

TX confirmed



Account 1

The screenshot shows the MetaMask wallet interface for 'Account 1'. The account balance is listed as '119.998 ETH'. Below this, a message states 'You have no transactions'. A note at the bottom suggests contacting MetaMask Support.

smurf.net

CST4125:L18

February 8, 2023 38 / 45

Blocks



This screenshot shows the 'BLOCKS' section of the MetaMask wallet interface. It lists several blocks with their respective numbers, dates, and transaction counts. Block 6 was created on 2022-09-29 21:47:57 with 21000 transactions. Block 9 was created on 2022-09-29 20:54:31 with 8 transactions.

smurf.net

CST4125:L18

February 8, 2023 39 / 45

An individual block

Block 5



Block 5

MINER: 218900
SALT:MFT: 6721975
MHT:W: 2022-06-29 21:43:48
BLOCK NUMBER: 0xc259bead9934d8ecfcbae3798d0aed9aa918fc9a48a4ecf808e75a0e3ff0

TO: 894 ETH

FROM ADDRESS: 0x3274c280000f6baaa2d1ec12d4a1e5d770b129a8c7660d3368445bd6cad0cfe
TO CONTRACT ADDRESS: 0x4830109807112830c0a50f9910648215ea

MAX: 0x00
VALUE: 10000000000000000000

Activity

no transactions

Block MetaMask Support

smerf.net

CST4125:L18

February 8, 2023 40 / 45

An individual block

Block 6



Block 6

MINER: 218900
SALT:MFT: 6721975
MHT:W: 2022-06-29 21:47:57
BLOCK NUMBER: 0xe11397fe4466058858506107792b7573963aec253a21fd3efaf59e4ade3612

TO: 894 ETH

FROM ADDRESS: 0x1360b3fe32e448a726533087af770e9bb5d0d38eb099c3aa130f69d7364ab
TO CONTRACT ADDRESS: 0x5830109807112830c0a50f9910648215ea

MAX: 0x00
VALUE: 10000000000000000000

Activity

no transactions

Block MetaMask Support

smerf.net

CST4125:L18

February 8, 2023 41 / 45

Network Summary



- Ganache
- MetaMask Wallets
- Make TX
- Examine TX
- Network

smerf.net

CST4125:L18

February 8, 2023 42 / 45

Summary

- Network
- Accounts
- Exceptions
- Unit Testing
- Modifiers

smerf.net

CST4125:L18

February 8, 2023 43 / 45

Reading



- Chapter 8 in [1]

smerf.net

CST4125:L18

February 8, 2023 44 / 45

References I

- [1] Ritesh Modi. *Solidity Programming Essentials*. Packt, 2018. ISBN: 978-1-78883-138-3.

smerf.net

CST4125:L18

February 8, 2023 45 / 45