Lab: Events and Return Variables

Prerequisites

1. Chrome or Firefox browser.
2. An Internet connection
3. Open Remix with the following Smart Contract:

```
pragma solidity ^0.5.11;

contract EventExample {

    mapping(address => uint) public tokenBalance;

    constructor() public {
        tokenBalance[msg.sender] = 100;
    }

    function sendToken(address _to, uint _amount) public returns(bool) {

        require(tokenBalance[msg.sender] >= _amount, "Not enough tokens");
        assert(tokenBalance[_to] + _amount >= tokenBalance[_to]);
        assert(tokenBalance[msg.sender] - _amount <= tokenBalance[msg.sender])
;
        tokenBalance[msg.sender] -= _amount;
        tokenBalance[_to] += _amount;

        return true;
    }

}
```

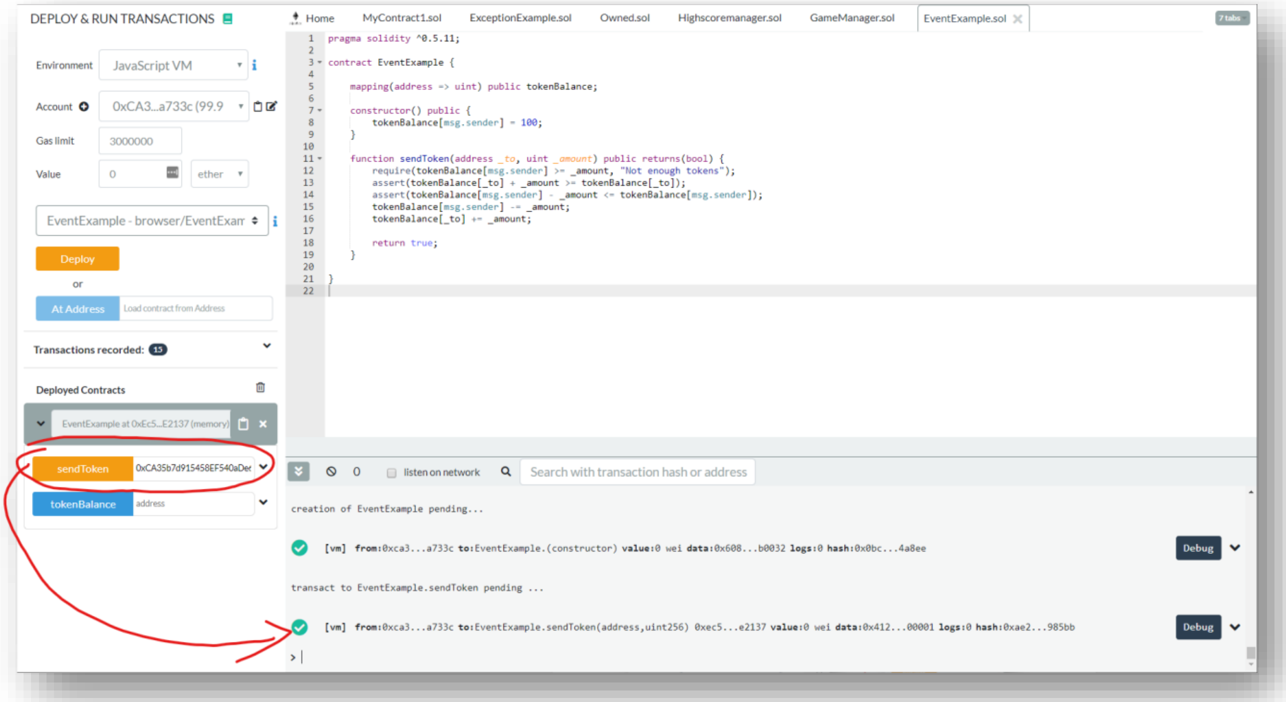
Step by Step Instruction

Deploy the Smart Contract in the JavaScript VM

Open the “Deploy and Run Transactions” view in Remix with the smart contract. Deploy the Smart Contract!

Send 1 Token to any account

Send 1 token and observe what happens with the return value:



Environment: JavaScript VM

Account: 0xCA3...a733c (99.9

Gas limit: 3000000

Value: 0 ether

EventExample - browser/EventExam

Transactions recorded: 15

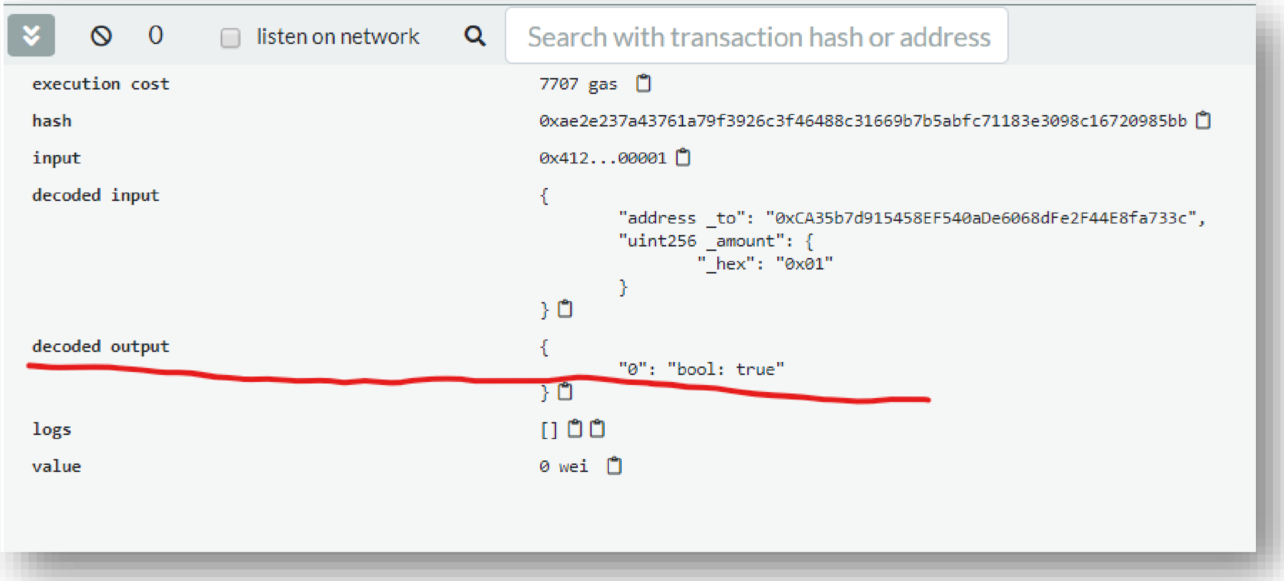
EventExample at 0xEc5...E2137 (memory)

sendToken: 0xCA35b7d915458EF540aDe

```
pragma solidity ^0.5.11;

contract EventExample {

    mapping(address => uint) public tokenBalance;

    constructor() public {
        tokenBalance[msg.sender] = 100;
    }

    function sendToken(address _to, uint _amount) public returns(bool) {
        require(tokenBalance[msg.sender] >= _amount, "Not enough tokens");
        assert(tokenBalance[_to] + _amount >= tokenBalance[_to]);
        assert(tokenBalance[msg.sender] - _amount <= tokenBalance[msg.sender]);
        tokenBalance[msg.sender] -= _amount;
        tokenBalance[_to] += _amount;

        return true;
    }

}
```

```
creation of EventExample pending...

[vm] from:0xca3...a733c to:EventExample.(constructor) value:0 wei data:0x608...b0032 logs:0 hash:0x0bc...4a8ee

transact to EventExample.sendToken pending ...

[vm] from:0xca3...a733c to:EventExample.sendToken(address,uint256) 0xec5...e2137 value:0 wei data:0x412...00001 logs:0 hash:0xae2...985bb
>
```

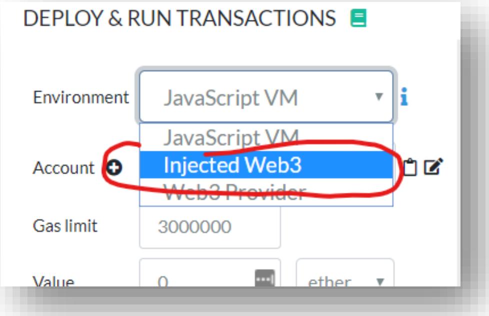
If you open the details, you can see the return value.



```
execution cost        7707 gas
hash                  0xae2e237a43761a79f3926c3f46488c31669b7b5abfc71183e3098c16720985bb
input                 0x412...00001
decoded input         {
                              "address _to": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                              "uint256 _amount": {
                                      "_hex": "0x01"
                              }
                      }
decoded output        {
                              "0": "bool: true"
                      }
logs                  []
value                 0 wei
```

Unfortunately, this doesn't work on real blockchains! Try to deploy the same using MetaMask and your previously acquired Test-Ether

Deploy on Görli or Ropsten Test-Network



DEPLOY & RUN TRANSACTIONS

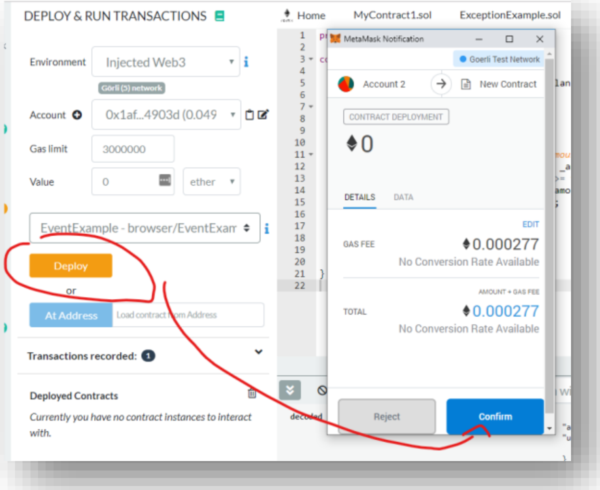
Environment: JavaScript VM

- JavaScript VM
- Injected Web3
- Web3 Provider

Account

Gas limit: 3000000

Value: 0 ether



DEPLOY & RUN TRANSACTIONS

Environment: Injected Web3

Görli (5) network

Account: 0x1af...4903d (0.049

Gas limit: 3000000

Value: 0 ether

EventExample - browser/EventExam

Deploy

or

At Address: Load contract from Address

Transactions recorded: 1

Deployed Contracts

Currently you have no contract instances to interact with.

Home | MyContract1.sol | ExceptionExample.sol

MetaMask Notification

Goerli Test Network

Account 2 → New Contract

CONTRACT DEPLOYMENT

0

DETAILS | DATA

EDIT

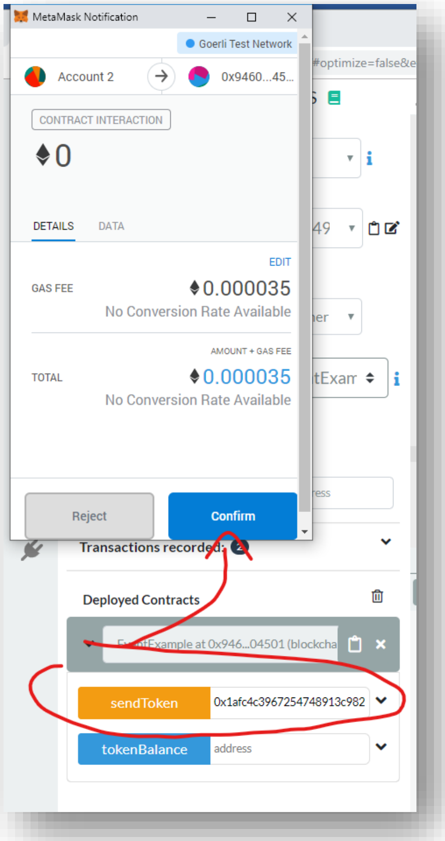
GAS FEE: 0.000277
No Conversion Rate Available

AMOUNT + GAS FEE

TOTAL: 0.000277
No Conversion Rate Available

Reject | Confirm

Wait until the transaction is mined… Then try again to send 1 Token.



MetaMask Notification

Goerli Test Network

Account 2 → 0x9460...45...

CONTRACT INTERACTION

0

DETAILS | DATA

EDIT

GAS FEE: 0.000035
No Conversion Rate Available

AMOUNT + GAS FEE

TOTAL: 0.000035
No Conversion Rate Available

Reject | Confirm

Transactions recorded: 2

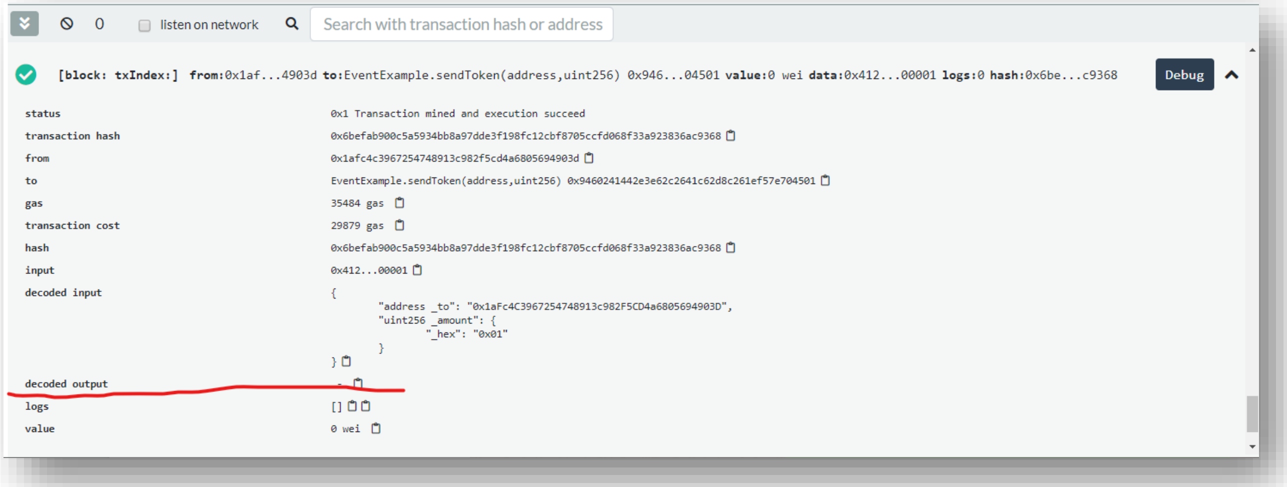
Deployed Contracts

EventExample at 0x946...04501 (blockcha

sendToken: 0x1afc4c3967254748913c982

tokenBalance: address

As soon as the transaction is mined you will not see an output, it’s empty.



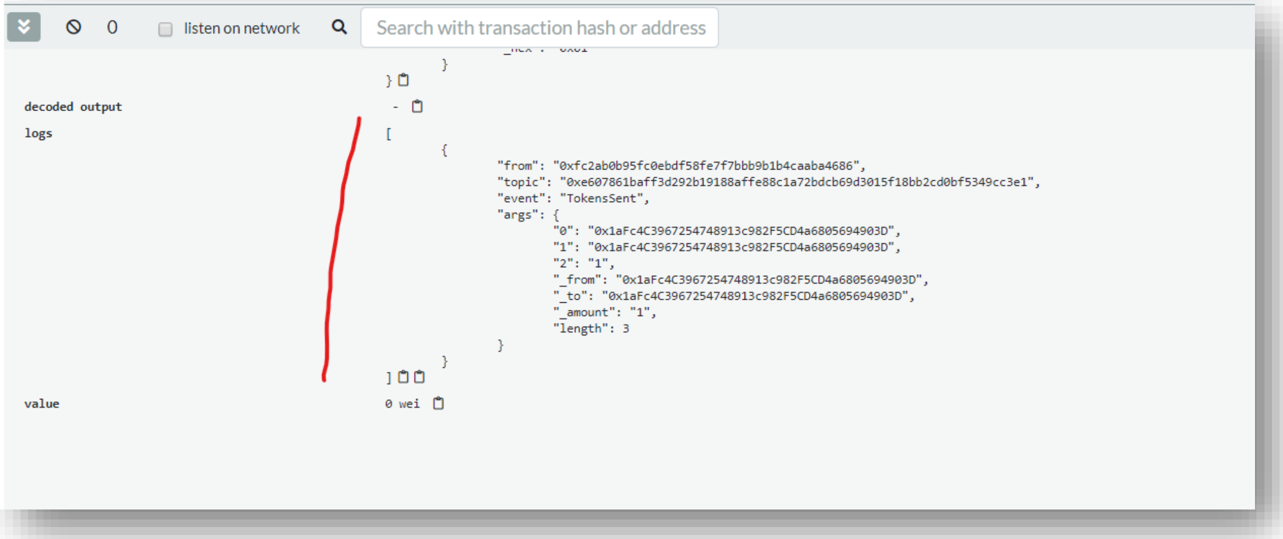
```
[block: txIndex:]  from:0x1af...4903d to:EventExample.sendToken(address,uint256) 0x946...04501 value:0 wei data:0x412...00001 logs:0 hash:0x6be...c9368

status                          0x1 Transaction mined and execution succeed
transaction hash                0x6befab900c5a5934bb8a97dde3f198fc12cbf8705ccfd068f33a923836ac9368
from                            0x1afc4c3967254748913c982f5cd4a6805694903d
to                              EventExample.sendToken(address,uint256) 0x9460241442e3e62c2641c62d8c261ef57e704501
gas                             35484 gas
transaction cost                29879 gas
hash                            0x6befab900c5a5934bb8a97dde3f198fc12cbf8705ccfd068f33a923836ac9368
input                           0x412...00001
decoded input                   {
                                        "address _to": "0x1aFc4C3967254748913c982F5CD4a6805694903D",
                                        "uint256 _amount": {
                                                "_hex": "0x01"
                                        }
                                }
decoded output                   -
logs                            []
value                           0 wei
```

Return values are meant for inter-smart-contract communication, they won’t return anything to the transaction origin. This is why we have to use Events for outputting anything in writing transactions.

Add an Event to the Smart Contract

```
pragma solidity ^0.5.11;

contract EventExample {

    mapping(address => uint) public tokenBalance;

    event TokensSent(address _from, address _to, uint _amount);

    constructor() public {
        tokenBalance[msg.sender] = 100;
    }

    function sendToken(address _to, uint _amount) public returns(bool) {
        require(tokenBalance[msg.sender] >= _amount, "Not enough tokens");
        assert(tokenBalance[_to] + _amount >= tokenBalance[_to]);
        assert(tokenBalance[msg.sender] - _amount <= tokenBalance[msg.sender])
;
        tokenBalance[msg.sender] -= _amount;
        tokenBalance[_to] += _amount;

        emit TokensSent(msg.sender, _to, _amount);

        return true;
    }

}
```

Re-Deploy to Görli or any other Test-Net and see if you can see the event in the output



```
                                                 "_hex": "0x01"
                                         }
                                 }
decoded output                    -
logs                             [
                                         {
                                                 "from": "0xfc2ab0b95fc0ebdf58fe7f7bbb9b1b4caaba4686",
                                                 "topic": "0xe607861baff3d292b19188affe88c1a72bdcb69d3015f18bb2cd0bf5349cc3e1",
                                                 "event": "TokensSent",
                                                 "args": {
                                                         "0": "0x1aFc4C3967254748913c982F5CD4a6805694903D",
                                                         "1": "0x1aFc4C3967254748913c982F5CD4a6805694903D",
                                                         "2": "1",
                                                         "_from": "0x1aFc4C3967254748913c982F5CD4a6805694903D",
                                                         "_to": "0x1aFc4C3967254748913c982F5CD4a6805694903D",
                                                         "_amount": "1",
                                                         "length": 3
                                                 }
                                         }
                                 ]
value                            0 wei
```

Congratulations, LAB is completed