

Presentation (record and replay contract deployment and interaction)

0. load gist with files from terminal

```
pragma solidity >0.4.0;
contract Ballot {
    struct Voter {
        uint weight;
        uint votedFor;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voterCount;
    }
    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }
    // Give right to voter with i to proposal i different proposals.
    // May only be called by $chairperson.
    function giveRightToVoter(address voter) public {
        chairperson = msg.sender;
        voters[voter].weight = 1;
        proposals.length = numProposals;
    }
    // Give right to voter with i to proposal i different proposals.
    // May only be called by $chairperson.
    function giveRightToVoted(address voter) public {
        chairperson = msg.sender;
        voters[voter].voted = true;
        voters[voter].weight = 1;
    }
    // Delegate your vote to the voter of to.
    function delegate(address to) public {
        Voter storage sender = voters[msg.sender]; // assigns reference
        while (voters[to].delegate != address(0) && voters[to].delegate != msg.sender)
            if (to == msg.sender) return;
            sender.voted = true;
            sender.delegate = to;
            Voter storage delegate = voters[to];
            if (delegate.voted) {
                proposals[delegate.vote].voteCount += sender.weight;
            } else {
                delegate.vote = to;
                proposals[delegate.vote].voteCount += sender.weight;
                delegate.weight += sender.weight;
            }
    }
    // Give a single vote to proposal i to proposal i.
    function giveSingleVote(uint8 i) public {
        Voter storage sender = voters[msg.sender];
        if (sender.voted || i > proposals.length) return;
        sender.voted = true;
        sender.vote = i;
        proposals[i].voteCount += sender.weight;
    }
    function winingProposal() public constant returns (uint8 _winingProposal) {
        uint256 winningVoteCount = 0;
        for (uint8 i = 0; i < proposals.length; i++) {
            if (proposals[i].voteCount > winningVoteCount) {
                winningVoteCount = proposals[i].voteCount;
                _winingProposal = i;
            }
        }
    }
}
```

1. open and compile Escrow.sol

```
pragma solidity >0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payOutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
    function contractBalance() public view returns (uint256){
        return address(this).balance;
    }
}
```

2. switch to run tab and check constructor

The screenshot shows the Remix IDE interface with the following details:

- Contract Definition:** Escrow
- Environment:** JavaScript VM (VM)
- Account:** 0xca3...a733c (100 ether)
- Gas limit:** 3000000
- Value:** 0 wei

Escrow contract details:

- Deploy address: buyer.address_arbiter.uint256_escrowAmount
- Or
- All Address Load contract from Address

Transactions recorded: 0

Deployed Contracts:

Currently you have no contract instances to interact with.

Bottom Bar:

- Only remix transactions, script
- Search transactions

Contract Code (browserEscrow.sol):

```
pragma solidity ^0.4.25;

contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;

    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        seller = _seller;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }

    function fund() payable public {
        require(msg.value == escrowAmount);
    }

    function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }

    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }

    function contractBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

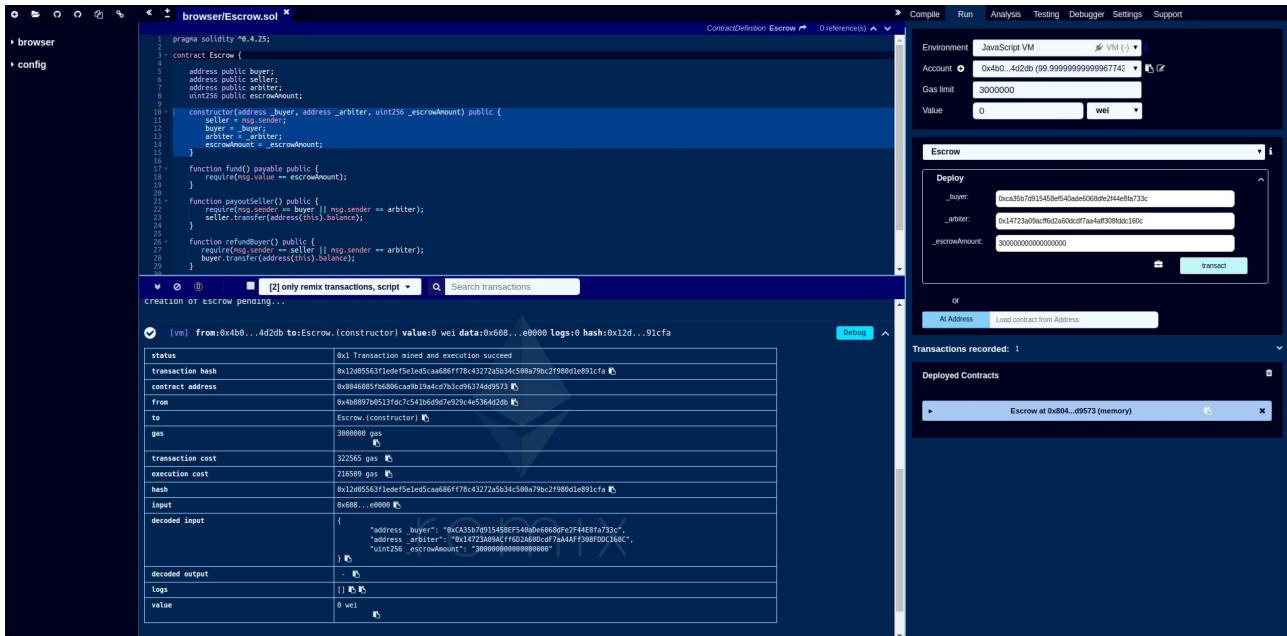
Bottom Notes:

- 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.

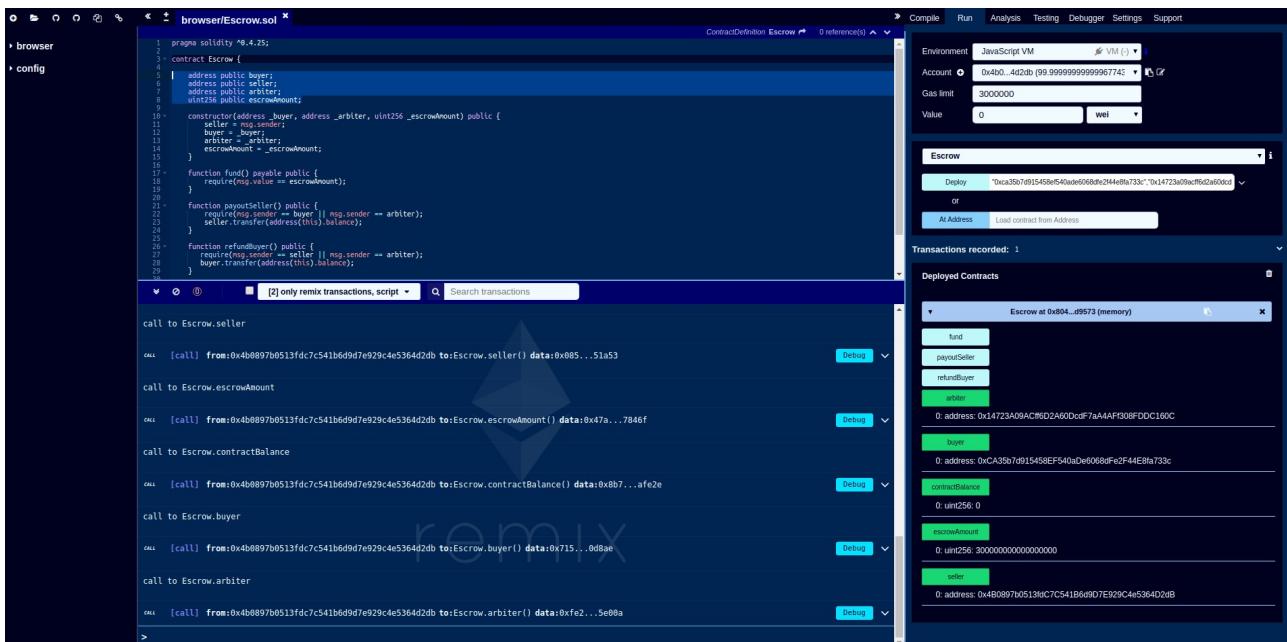
3. convert 0.3 ether to wei for escrowAmount parameter

 EtherConverter.online

4. drop down Escrow constructor form, fill out arguments and deploy (transact)



5. check some fields to see contractBalance is 0



6. switch to buyer address, set "Value" to 0.3 and select Ether, then click fund

- 0.3 Ether have been sent to the escrow contract instance
- After you click fund, the "Value" field will be reset to 0

The screenshot shows the Remix IDE interface. On the left, the Solidity code for the Escrow contract is visible. On the right, the Deployed Contracts section shows the Escrow contract at address 0x804...d9573. Below it, a transaction record for a fund transfer is shown:

status	0x1 Transaction mined and execution succeeded
transaction hash	0x14134685e915bfdbab15ebabb0f11881356fad99f6e9acdff63e58ef90567c5
from	0xc35b7d915458ef540ade6068dfef2f44e8fa733c
to	Escrow (fund)
gas	3000000
transaction cost	21710 gas
execution cost	438 gas
hash	0x14134685e915bfdbab15ebabb0f11881356fad99f6e9acdff63e58ef90567c5
input	0x66...d4288
decoded input	(1)
decoded output	(1)
logs	(1)
value	30000000000000000000 wei

7. click contractBalance to see it has changed from 0 to 30000000000000000000

The screenshot shows the Remix IDE interface. On the left, the Solidity code for the Escrow contract is visible. On the right, the Deployed Contracts section shows the Escrow contract at address 0x804...d9573. Below it, a transaction record for a fund transfer is shown:

transaction hash	0x14134685e915bfdbab15ebabb0f11881356fad99f6e9acdff63e58ef90567c5
from	0xc35b7d915458ef540ade6068dfef2f44e8fa733c
to	Escrow (fund)
gas	3000000
transaction cost	21710 gas
execution cost	438 gas
hash	0x14134685e915bfdbab15ebabb0f11881356fad99f6e9acdff63e58ef90567c5
input	0x66...d4288
decoded input	(1)
decoded output	(1)
logs	(1)
value	30000000000000000000 wei

Below the transaction details, a call to Escrow.contractBalance is shown:

```
call [call] from:0xc35b7d915458ef540ade6068dfef2f44e8fa733c to:Escrow.contractBalance() data:0xb07...afe2e
```

8. try to refundBuyer and see it fail

- Because the contract defines that you as the buyer cant refund

```

pragma solidity >=0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = msg.sender;
        seller = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.sender == buyer || msg.sender == arbiter);
    }
    function payoutSeller() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
}

```

9. switch to seller address and try to payoutSeller and see it fail

- Because the contract defines that you as a seller cant payout

```

[2] only remix transactions, script < Search transactions
hash 0x1431465f31910faabaa15e6bb8bf11881356fad99fe6ac0ff3c58e795687c5
Input 0x66...04288
decoded input () 
decoded output () 
Logs () 
value 3000000000000000000 wei

call to Escrow.contractBalance
out [call] from:0xca35b7d915458ef540ade6068dfc2f44e@fa733c to:Escrow.contractBalance() data:0x8b7...afe2e
transact to Escrow.refundBuyer pending ...
[vm] from:0xa3...a733c to:Escrow.refundBuyer() 0x804...d9573 value:0 wei data:0xe8a...61c8 logs:0 hash:0x479...79b07
transact to Escrow.refundBuyer errored: VM error: revert.
revert The transaction has been reverted to the initial state.
Note: The constructor should be payable if you send value. Debug the transaction to get more information.
>

```

10. switch to arbiter address and decide to cancel by refundBuyer

The screenshot shows the Remix IDE interface with the Escrow contract code in the left panel. The right panel displays the environment settings (JavaScript VM, Account 0x147...c160c, Gas limit 3000000, Value 0 ether), a transaction history section titled "Transactions recorded" (with one entry for refundBuyer), and a "Deployed Contracts" section showing the deployed Escrow contract at address 0x804...d9573.

```

pragma solidity ^0.4.25;
contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
}

```

11. check escrowAmount and see it has changed from 300000000000000000000000 to 0

This screenshot is similar to the previous one but shows a different transaction in the "Transactions recorded" section. The transaction details for refundBuyer() show the gas used and the value sent (0 wei).

Call to Escrow.contractBalance

```

call to Escrow.contractBalance
[call] from:0x14723a09acff6d2a60cd7aa4aff308fddc160c to:Escrow.contractBalance() data:0xb87...afe2e

```

12. Go to "Transactions recorded" and click the dropdown and "save icon"

```

pragma solidity ^0.4.25;

contract Escrow {
    address public buyer;
    address public seller;
    address public arbiter;
    uint256 public escrowAmount;
    constructor(address _buyer, address _arbiter, uint256 _escrowAmount) public {
        buyer = _buyer;
        arbiter = _arbiter;
        escrowAmount = _escrowAmount;
    }
    function fund() payable public {
        require(msg.value == escrowAmount);
    }
    function payoutSeller() public {
        require(msg.sender == buyer || msg.sender == arbiter);
        seller.transfer(address(this).balance);
    }
    function refundBuyer() public {
        require(msg.sender == seller || msg.sender == arbiter);
        buyer.transfer(address(this).balance);
    }
    function contractBalance() public view returns (uint256){
        return address(this).balance;
    }
}

```

13. choose file name and click ok

Transactions will be saved in a file under browser
scenario.json

14. see the saved recording json file

The screenshot shows the Remix IDE interface. On the left, there are two tabs: "browser/Escrow.sol" and "browser/scenario.json". The "browser/Escrow.sol" tab displays the Solidity code for the Escrow contract, which includes a constructor, a function `refundBuyer`, and a function `payoutSeller`. The "browser/scenario.json" tab shows a JSON object with various fields like accounts, timestamp, and parameters. On the right side, there's a "Deploy" button with the address "0xca35b7d915408e540ade6008d2944ed9a73c" and a "Gas limit" set to 3000000. Below the deploy button, there's a "Transactions recorded" section and a "Deployed Contracts" section showing the deployed contract at address "Escrow at 0xd0d...bcf7 (memory)".

15. for example change addresses and constructor params and click PLAY to replay

- see all 5 transactions repeat with the changed data
- see another instance of the newly deployed contract

This screenshot is similar to the previous one but shows changes made to the contract code. In the "browser/Escrow.sol" tab, the constructor parameters have been modified. In the "Deploy" section, the address has been changed to "0x147...c160c". The "Transactions recorded" and "Deployed Contracts" sections also reflect these changes, showing the new contract instance at the updated address.

16. check all read only methods and see values have the changed ones

- contractBalance is 0 because everything has already been refunded to the buyer by arbiter

The screenshot shows the Remix IDE interface with the following details:

- Contract Definition: Escrow**
- Value**: 0 Wei
- Transactions recorded**: 0
- Deployed Contracts**:
 - Escrow at 0x692...77f3a (memory)
 - Escrow at 0xd0d...46222 (memory)
 - fund
 - payable
 - refIdBuyer
 - seller
 - buyer
 - layer
 - contractBalance
 - 0: uint256: 0
 - escrowBalance
 - 0: uint256: 30000000000000000000000000000000
 - seller

The left panel displays the Solidity code for the Escrow contract, which includes functions for refunding buyers, escrowing amounts, and setting contract balances. The right panel shows the deployed contracts and their current states.