Background submission for ZK DAO

Username: m00npapi

1. At the most fundamental level, a smart contract is a is a set of instructions, that can be carried out by somebody on the EVM. There are important features besides the instructions however, once deployed it cannot ever be changed, by the person who made the instructions in the first place or anyone else.

2. All things on the blockchain require some "energy" to change, gas is the unit used to refer to how much "energy" to carry out or compute something on the EVM. You can think about it like you would gas in a car, the amount of gas in your tank lets you know how far you can drive. Just like with a real car, if you run out of gas you will get "stuck". Gas is paid out in ether.

3. A hash is a different way to represent some information, let's say a message that I send to you. The message says, "Hey my favorite color is green." Let's say this might be some important information, or I want to be able to prove, without a doubt that it was me who sent the message to you. Well, I would jumble up this message but in a way that can be decoded later, by using a set of instructions, otherwise known as a key – technically you don't need the set of instructions but it would take you a really long time to figure out how to decode the message on your own. I would sign this message and let's say that when I sign this message I receive a hash of the message. At any point in time after this, ANYONE would be able to decode the hash as long as they have the instructions to decode it and verify that I was the person who sent the message as well as the message contents.

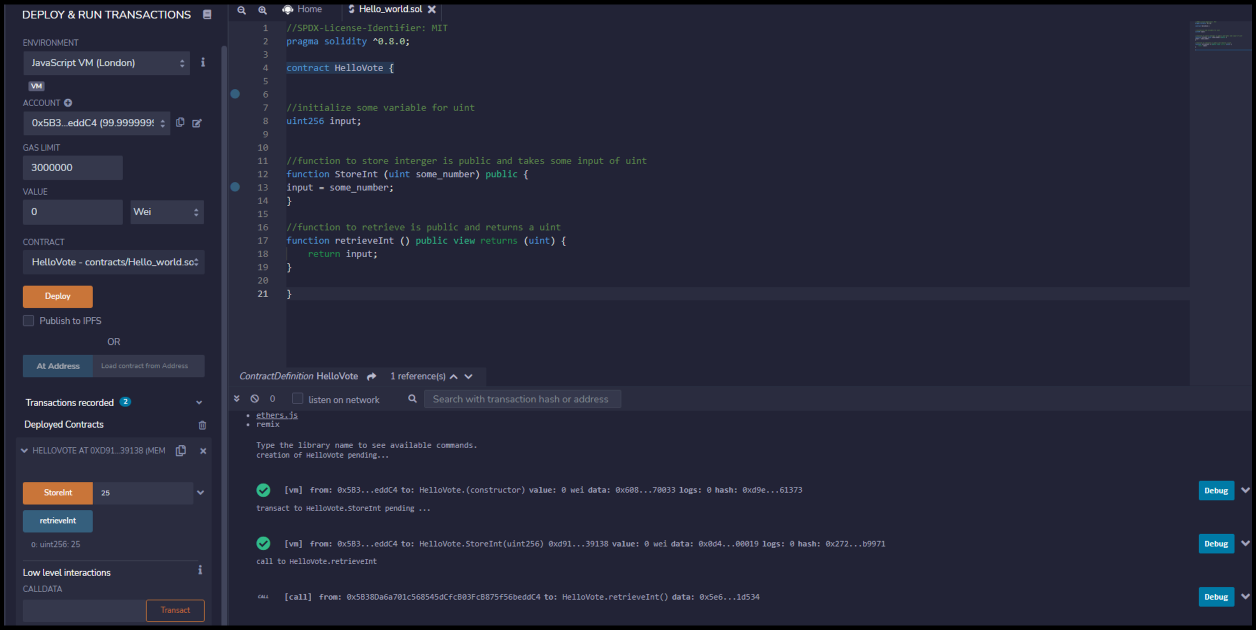
4. So if you had a colorblind friend (like me), who is red green colorblind, they would see red and green as the same color (we can also imagine that this friend can only see in black and white, doesn't really change anything). Let's say he grabs a color pencil, and these pencils are broken so the name of each color isn't visible anymore – the color he chose just so happens to be red – halfway through drawing the friend breaks that red color pencils tip, and we can't seem to find a pencil sharpener anywhere! So he looks around and searches to see if any of the other pencils around match the color of the one he broke, some are kind of close but then he finds one that matches, and he exclaims "Found one! This is the same color!" He's about to get back to his masterpiece before you stop him, and say "HEY! That pencil isn't the same color as the other one!" and he looks at you a little funny, he goes "I don't know what you're talking about. Look" *he holds the broken pencil next to the one he just picked up* "They're the same color." You pause for a second and realize your friend might be colorblind, so you look at him and say, "How about I prove it to you!" and he replies "I don't think I trust you, but ok prove it."

   "Here's what we are gonna do, you know which pencil is broken, and which pencil isn't, so I want you to take the pencils behind your back and switch them in your hands as many times as you like. You'll cover the ends of the pencils so I can't see which pencil is broken, but I will be able to see the other end of each pencil which are shaped the same. I'll be able to tell you exactly which pencil is in each hand, because each pencil has another property besides being broken/not broken – each pencil has a color, and I know that the broken pencil is what I'll call red, and the non-broken pencil is what I will call green."

Background submission for ZK DAO

Username: m00npapi

Hello_world – store/retrieve int example



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: JavaScript VM (London)

ACCOUNT: 0x5B3...eddC4 (99.999999!

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: HelloVote - contracts/Hello_world.so

Transactions recorded 2

Deployed Contracts

HELLOVOTE AT 0XD91...39138 (MEM

StoreInt: 25

retrieveInt

0: uint256: 25

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HelloVote {


//initialize some variable for uint
uint256 input;

//function to store interger is public and takes some input of uint
function StoreInt (uint some_number) public {
input = some_number;
}

//function to retrieve is public and returns a uint
function retrieveInt () public view returns (uint) {
    return input;
}

}
```

ContractDefinition HelloVote 1 reference(s)

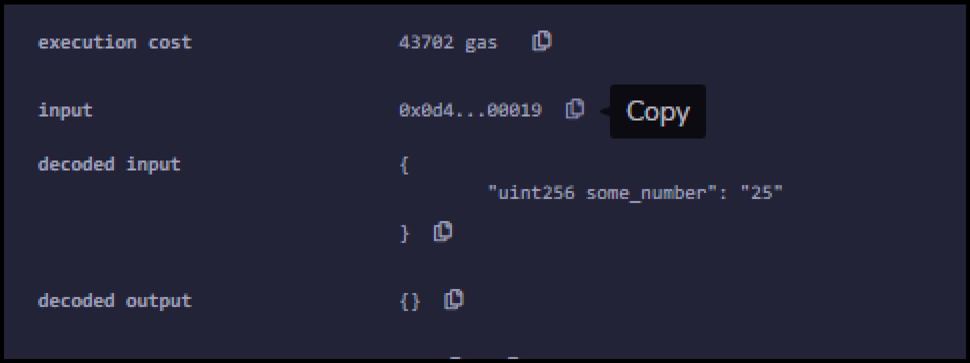
```
• ethers.js
• remix

Type the library name to see available commands.
creation of HelloVote pending...

[vm] from: 0x5B3...eddC4 to: HelloVote.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0xd9e...61373
transact to HelloVote.StoreInt pending ...

[vm] from: 0x5B3...eddC4 to: HelloVote.StoreInt(uint256) 0xd91...39138 value: 0 wei data: 0x0d4...00019 logs: 0 hash: 0x272...b9971
call to HelloVote.retrieveInt

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: HelloVote.retrieveInt() data: 0x5e6...1d534
```

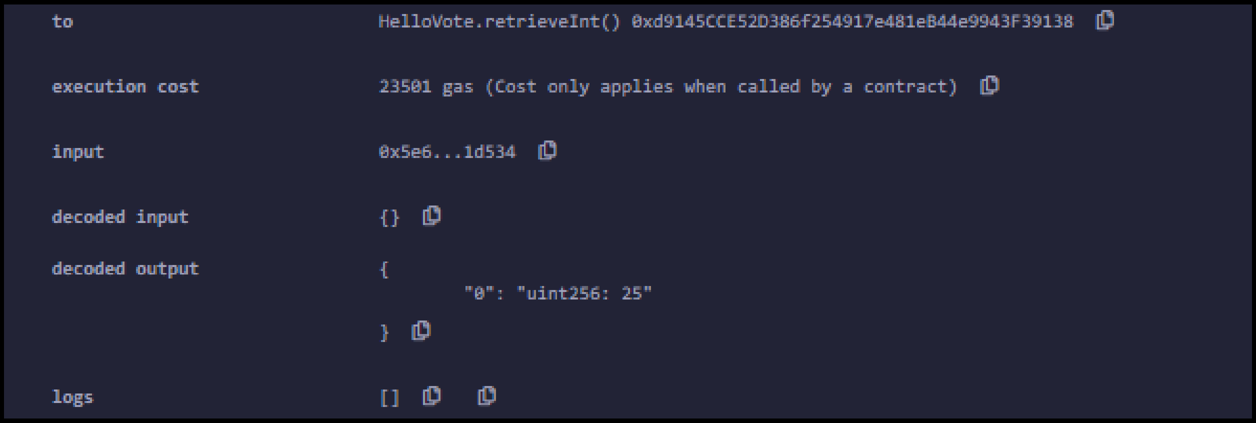


```
execution cost          43702 gas

input                   0x0d4...00019

decoded input           {
                                "uint256 some_number": "25"
                        }

decoded output          {}
```



```
to                  HelloVote.retrieveInt() 0xd9145CCE52D386f254917e481eB44e9943F39138

execution cost      23501 gas (Cost only applies when called by a contract)

input               0x5e6...1d534

decoded input       {}

decoded output      {
                            "0": "uint256: 25"
                    }

logs                []
```

Ballot Contract with 5 minute time limit/modifier

Background submission for ZK DAO

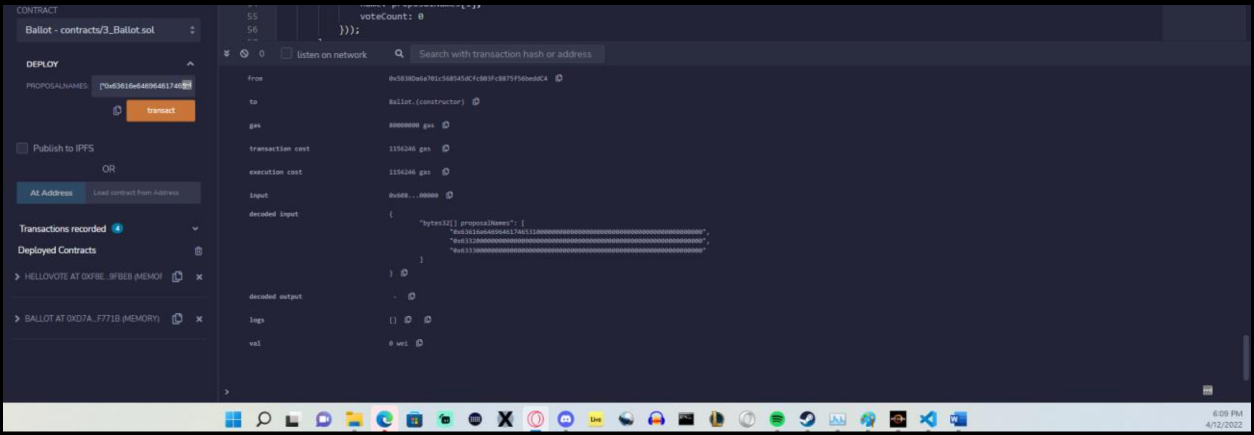
Username: m00npapi

Ballot input

[“candidate1”, “c2”, “c3”] ➔ bytes32[]

["0x63616e646964617465310000000000000000000000000000000000000000000000","0x6332000000000000000000000000000000000000000000000000000000000000","0x6333000000000000000000000000000000000000000000000000000000000000"]

Voting starting 6:09 pm est



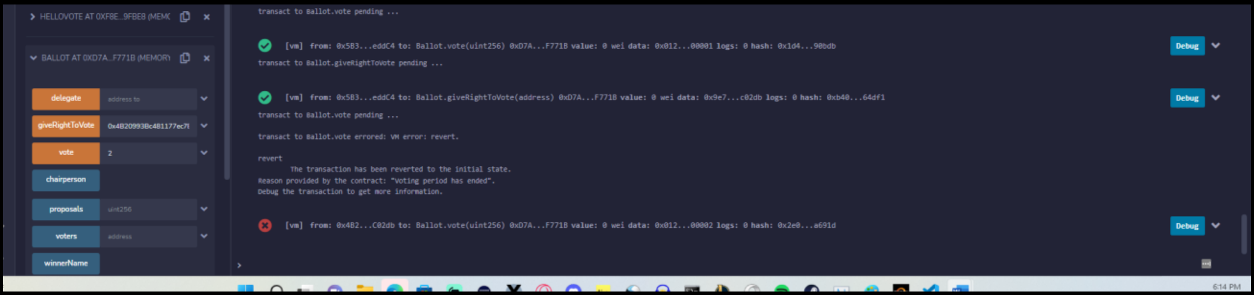
```
voteCount: 0
}));
```

CONTRACT
Ballot - contracts/3_Ballot.sol
DEPLOY
PROPOSALNAMES
transact
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 4
Deployed Contracts
HELLOVOTE AT 0XF8E...9FBE8 (MEMOR
BALLOT AT 0XD7A...F771B (MEMORY)
listen on network
Search with transaction hash or address
6:09 PM
4/12/2022

PROPOSALNAMES: ["0x63616e6469646174653

```
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Ballot.(constructor)
gas                 80000000 gas
transaction cost    1156246 gas
execution cost      1156246 gas
input               0x608...00000
decoded input       {
                        "bytes32[] proposalNames": [
                            "0x63616e646964617465310000000000000000000000000000000000000000000000",
                            "0x6332000000000000000000000000000000000000000000000000000000000000",
                            "0x6333000000000000000000000000000000000000000000000000000000000000"
                        ]
                    }
decoded output      -
logs                []
val                 0 wei
```

Voting ended: 6:14 pm est



delegate: address to
giveRightToVote: 0x4B20993Bc481177ec7E
vote: 2
proposals: uint256
voters: address

```
transact to Ballot.vote pending ...

[vm] from: 0x5B3...eddC4 to: Ballot.vote(uint256) 0xD7A...F771B value: 0 wei data: 0x012...00001 logs: 0 hash: 0x1d4...90bdb
transact to Ballot.giveRightToVote pending ...

[vm] from: 0x5B3...eddC4 to: Ballot.giveRightToVote(address) 0xD7A...F771B value: 0 wei data: 0x9e7...c02db logs: 0 hash: 0xb40...64df1
transact to Ballot.vote pending ...

transact to Ballot.vote errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Voting period has ended".
Debug the transaction to get more information.

[vm] from: 0x4B2...C02db to: Ballot.vote(uint256) 0xD7A...F771B value: 0 wei data: 0x012...00002 logs: 0 hash: 0x2e0...a691d
```

`commit a8400d554e365b77507fbb542e7bbd3d521ddc8e`

https://github.com/mugrebot/ZKstarter.git