

Project: "E-voting using Blockchain Technology - Smart Contract"

Authors: Sereysathia Luy

Advisor: Omar Abuzaghleh

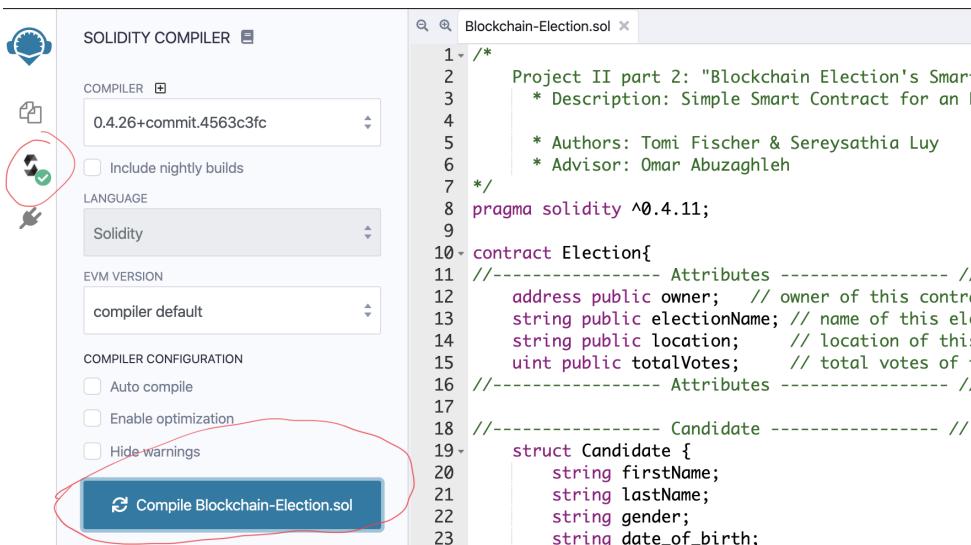
Description: Smart contract for a simple e-voting application using Blockchain Technology.

Programs/IDE used: Ganache, myEtherWallet, and Solidity

How to Run

1. Compile the code

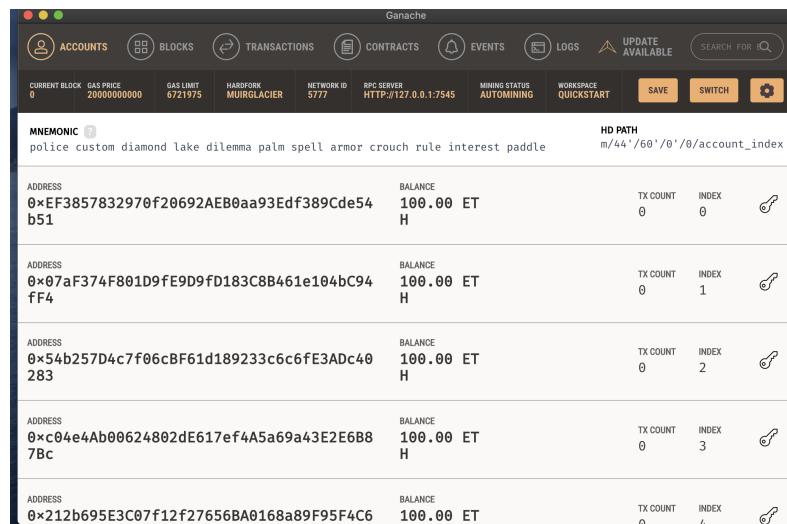
- Open Solidity IDE, compile the code



```
1- /*
2  Project II part 2: "Blockchain Election's Smart
3   * Description: Simple Smart Contract for an E
4
5   * Authors: Tomi Fischer & Sereysathia Luy
6   * Advisor: Omar Abuzaghleh
7 */
8 pragma solidity ^0.4.11;
9
10 contract Election{
11   //----- Attributes -----
12   address public owner; // owner of this contract
13   string public electionName; // name of this election
14   string public location; // location of this election
15   uint public totalVotes; // total votes of this election
16   //----- Attributes -----
17
18   //----- Candidate -----
19   struct Candidate {
20     string firstName;
21     string lastName;
22     string gender;
23     string date_of_birth;
```

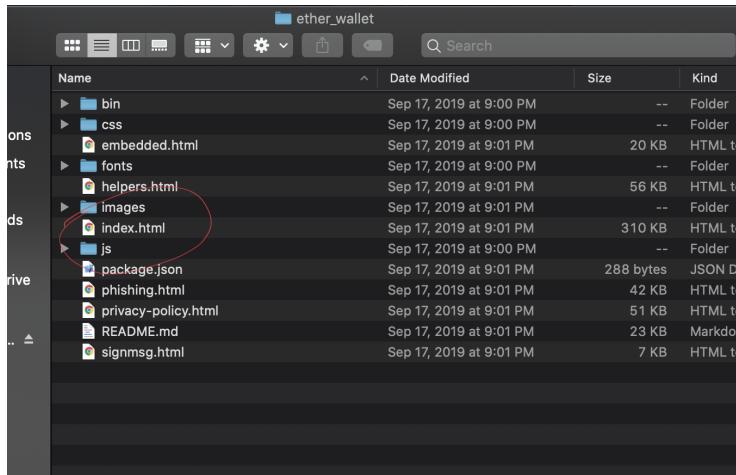
2. Open Ganache

- Open Ganache app -> Quick Start, and you should see this:



3. Open myEtherWallet

- o Open myEtherWallet
 - i. Using index.html file



A screenshot of the MyEtherWallet website. The top navigation bar includes links for 'New Wallet', 'Send Ether & Tokens', 'Swap', 'Send Offline', 'Contracts', 'ENS', 'DomainSale', 'Check TX Status', 'View Wallet Info', and 'Help'. The top right shows network status (3.40.0 English), gas price (41 Gwei), and network (ETH). A message indicates the network is full right now. The main content area is titled 'Create New Wallet' and contains a form for entering a password. A note says 'Do NOT forget to save this!' and a button labeled 'Create New Wallet'. A sidebar on the right lists other wallet options: Ledger / TREZOR / BitBox / Secalot, MetaMask Connect, Jaxx / imToken, and Mist / Geth / Parity. A note at the bottom states that this password encrypts your private key and you will need it to unlock your wallet. Navigation links 'How to Create a Wallet' and 'Getting Started' are at the bottom.

Try our new version here: <https://www.myetherwallet.com/>
Read more about it in our [Medium post](#)

3.40.0 English ▾ Gas Price: 41 Gwei ▾ Network ETH (myetherwallet.com) ▾
The network is really full right now. Check Eth Gas Station for gas price to use.

New Wallet Send Ether & Tokens Swap Send Offline Contracts ENS DomainSale Check TX Status View Wallet Info Help

Create New Wallet

Enter a password

Do NOT forget to save this!

Create New Wallet

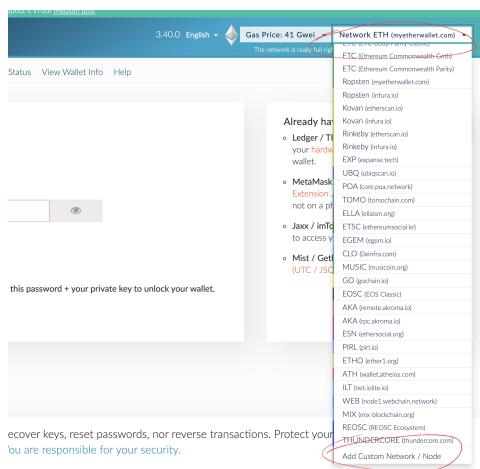
This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet.

How to Create a Wallet • Getting Started

Already have a wallet somewhere?

- o Ledger / TREZOR / BitBox / Secalot : Use your [hardware wallet](#). Your device *is* your wallet.
- o MetaMask Connect via your [MetaMask Extension](#). So easy! Keys stay in MetaMask, not on a phishing site! Try it today.
- o Jaxx / imToken Use your [Mnemonic Phrase](#) to access your account.
- o Mist / Geth / Parity: Use your [Keystore File \(UTC / JSON\)](#) to access your account.

- Connect it to Ganache
 - i. Click on the top-right drop-down menu “Network ETH”
 1. Then add Custom network/ node



- ii. Fill in the information, URL and Post can be found in Ganache

Set Up Your Custom Node

Instructions can be found here

Your node must be HTTPS in order to connect to it via MyEtherWallet.com. You can [download the MyEtherWallet repo & run it locally](#) to connect to any node. Or, get free SSL certificate via [Let's Encrypt](#)

Node Name
Election

URL
HTTP://127.0.0.1

Port
7545

HTTP Basic access authentication

ETH **ETC** **Ropsten** **Kovan** **Rinkeby** **Custom**

Cancel **Save & Use Custom Node**

Ganache

BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS UPDATE AVAILABLE

GAS PRICE 2000000000	GAS LIMIT 6721975	HARDFORK MUIRGLEACIER	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	WORKSPACE QUICKSTART	SAVE
-------------------------	----------------------	--------------------------	--------------------	-------------------------------------	-----------------------------	-------------------------	------

HD PATH
m/44'/60'/
stom diamond lake dilemma palm spell armor crouch rule interest paddle

7832970f20692AEB0aa93Edf389Cde54	BALANCE 100.00 ET H	TX COUNT 0
74F801D9fE9D9FD183C8B461e104bC94	BALANCE 100.00 ET H	TX COUNT 0

4. “Deploy the contract”

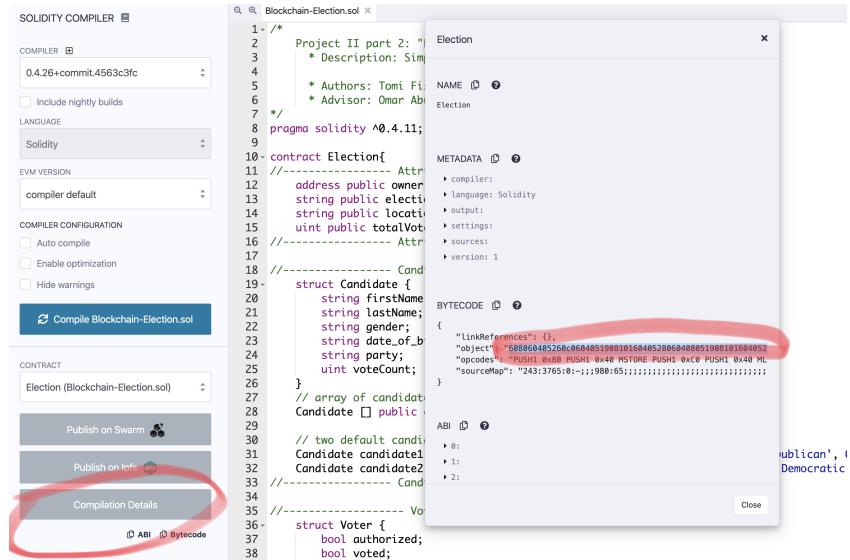
- o go to Contracts Tabs and then Deploy contract

The screenshot shows the MyEtherWallet interface. At the top, there's a navigation bar with links like 'New Wallet', 'Send Ether & Tokens', 'Swap', 'Send Offline', 'Contracts' (which is underlined and circled in red), 'ENS', 'DomainSale', 'Check TX Status', 'View Wallet Info', and 'Help'. Below the navigation bar, a large button labeled 'Interact with Contract or Deploy Contract' is visible, also circled in red.

- To Deploy a contract, Fill in the information for Byte code and Private key, Gas Limit is automatically filled in:

The screenshot shows the 'Deploy Contract' form. The 'Byte Code' field contains a very long hex string, which is circled in red. The 'Gas Limit' field is set to '1960274'. In the 'Paste Your Private Key' section, there is a note about the dangers of using MEWconnect and a link to protect oneself. A long hex string is pasted into the private key field, which is also circled in red. A blue 'Unlock' button is at the bottom of this section.

- Byte code - can be found in solidity -> compilation detail, note: copy only the data from “object”



The screenshot shows the Solidity Compiler interface with the following details:

- Compiler:** 0.4.26+commit.4563c3fc
- Language:** Solidity
- EVM Version:** compiler default
- Compiler Configuration:**
 - Auto compile
 - Enable optimization
 - Hide warnings
- Contract:** Election (Blockchain-Election.sol)
- Buttons:**
 - Publish on Swarm
 - Publish on Infura
 - Compilation Details (highlighted with a red circle)
 - ABI
 - Bytecode

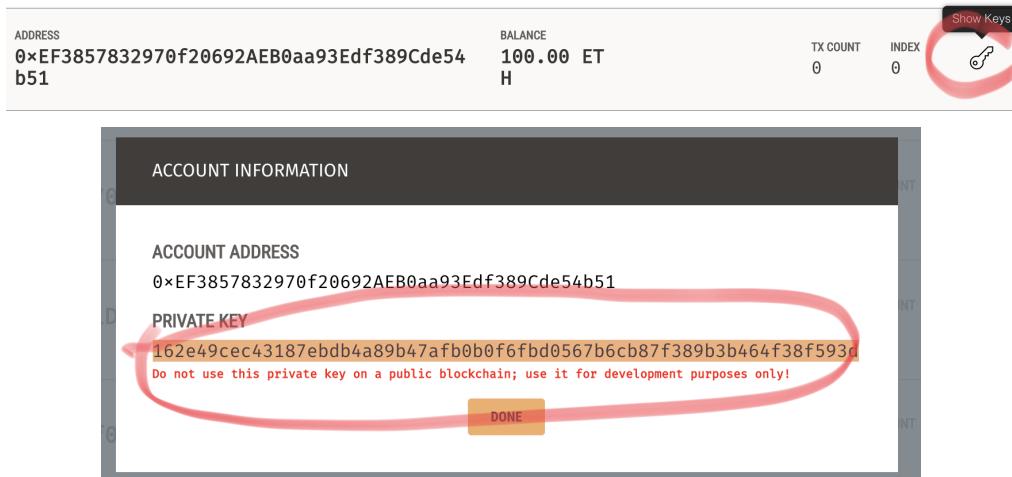
The right panel displays the Solidity source code for the 'Election' contract, including sections for **METADATA**, **BYTECODE**, and **ABI**. The **BYTECODE** section contains assembly-like code.

```

1- /*
2-  * Project II part 2: *
3-   * Description: Simple election contract
4-   * Authors: Tomi Finland
5-   * Advisor: Omar Alabdullah
6- */
7- pragma solidity ^0.4.11;
8-
9- contract Election{
10-     //----- ATTRIBUTES -----
11-     address public owner;
12-     string public electionName;
13-     string public location;
14-     uint public totalVotes;
15-     //----- STRUCT -----
16-     struct Candidate {
17-         string firstName;
18-         string lastName;
19-         string gender;
20-         string date_of_birth;
21-         string party;
22-         uint voteCount;
23-     }
24-     // array of candidates
25-     Candidate[] public candidates;
26-     // two default candidates
27-     Candidate candidate1;
28-     Candidate candidate2;
29-     //----- VOTING -----
30-     struct Voter {
31-         bool authorized;
32-         bool voted;
33-     }
34-     //----- FUNCTIONS -----
35-     function addCandidate(string _firstName, string _lastName, string _gender, string _date_of_birth, string _party) public {
36-         candidates.push(Candidate(_firstName, _lastName, _gender, _date_of_birth, _party));
37-     }
38- }

```

- Private key - can be found in ganache, choose an address, click on the key symbol and copy the private key



The screenshot shows the Ganache interface with the following details:

ADDRESS	BALANCE	TX COUNT	INDEX
0xEF3857832970f20692AEB0aa93Edf389Cde54b51	100.00 ET	0	0

ACCOUNT INFORMATION

ACCOUNT ADDRESS: 0xEF3857832970f20692AEB0aa93Edf389Cde54b51

PRIVATE KEY: 162e49cec43187ebdb4a89b47afb0b0f6fb0d0567b6cb87f389b3b464f38f593d
(Do not use this private key on a public blockchain; use it for development purposes only!)

DONE

- Click unlock and then sign Transaction and Deploy contract
-

- Result in Ganache:
 - A block is mined - this means that the smart contract is created

The screenshot shows two views of the Ganache interface. The top view is a summary bar with tabs for Accounts, Blocks, and Transactions. It displays the 'CURRENT BLOCK' as 1, 'GAS PRICE' as 20000000000, 'GAS LIMIT' as 6721975, 'HARDFORK' as MUIRGLEACIER, and 'NETWORK ID' as 5. The bottom view is a detailed table of blocks. It shows two rows: Block 1 (mined on 2020-04-26 15:58:04) and Block 0 (mined on 2020-04-26 15:57:17). Both rows have a green background.

CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER
1	20000000000	6721975	MUIRGLEACIER	5777	HTTP://127.0.0.1:7545
BLOCK 1	MINED ON 2020-04-26 15:58:04				
BLOCK 0	MINED ON 2020-04-26 15:57:17				

5. To “Interact with Contract”

- In myEther, go to Interact with Contract and fill in the information for Contract address and ABI

The screenshot shows the 'Interact with Contract or Deploy Contract' page of myEtherWallet. It has fields for 'Contract Address' (containing 0xD6234D74498FC50Eb5DA8614D739a5bbcd433009), 'Select Existing Contract' (with a dropdown menu showing 'Select a contract...'), and 'ABI / JSON Interface' (containing a JSON code block). A blue button labeled 'Access' is at the bottom left.

- Contract address - can be found in Ganache -> Transactions -> click on the first block we just mined or created, copy the address from “created contract address”

TX 0xf5ebce0b227c3ca78cdfa22ddf955aac38711713f795f8654040117d692c9881

SENDER ADDRESS	0x70665eB0ECCaB28dFa593Df8F9d23b965eF2705e	CREATED CONTRACT ADDRESS	0xD6234D74498FC50Eb5DA8614D739a5bbcd433009	CONTRACT CREATION	
VALUE	0.00 ETH	GAS USED	1060274	GAS PRICE	1.0000000000
				GAS LIMIT	1060274
				MINED IN BLOCK	1

- ABI - can be found in Solidity -> Compilation details -> ABI -> click on this to copy the value

```

6   | * Advisor: Omar Abi
7   */
8 pragma solidity ^0.4.11;
9
10 contract Election{
11     //----- Attr
12     address public owner;
13     string public electionName;
14     string public location;
15     uint public totalVotes;
16     //----- Attr
17
18     //----- Candidate
19     struct Candidate {
20         string firstName;
21         string lastName;
22         string gender;
23         string date_of_birth;
24         string party;
25         uint voteCount;
26     }
27     // array of candidates
28     Candidate[] public candidates;
29
30     // two default candidates
31     Candidate candidate1;
32     Candidate candidate2;
33     //----- Voting
34
35     //----- Voting
36
37 }
```

version: 1

BYTCODE

```
{
    "linkReferences": {},
    "object": "608060405260c060405",
    "opcodes": "PUSH1 0x80 PUSH1 0",
    "sourceMap": "243:3765:0:-;;98"
}
```

ABI

```

0:
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
11:
```

- Finally, click on Access - now we can interact with the contract

6. Interacting with Contract

- Setting election name and location - using `set_ElectionName_Location`

Read / Write Contract

0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

`set_ElectionName_Location`

`_name` string
UB Election

`_location` string
Bridgeport

WRITE

- Check result:

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbed433009

electionName ▾

↳ string
UB Election

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbed433009

location ▾

↳ string
Bridgeport

- Get candidates information using **candidates** - note: candidates are stored in an array, access them by inputting the **index**, meaning the first candidate is at index 0 and so on. Also when the contract is created, there will be two default candidates this is due to the constructor in the code
 -

- **Index 0:**

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

candidates ▾

uint256 0

↳ **firstName** string
Biff

↳ **lastName** string
Tannen

↳ **gender** string
male

↳ **date_of_birth** string
01/02/1955

↳ **party** string
Republican

↳ **voteCount** uint256
0

READ



- **Index 1:**

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

candidates ▾

uint256 1

↳ **firstName** string
Thomas

↳ **lastName** string
Whitmore

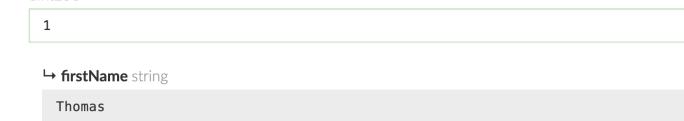
↳ **gender** string
male

↳ **date_of_birth** string
07/04/1970

↳ **party** string
Democratic

↳ **voteCount** uint256
0

READ



- Add Candidate by using **add_Candidate**,

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

add_Candidate ▾

_firstName string	Diana
_lastName string	Prince
_gender string	female
_dob string	02/14/1990
_party string	Independent

WRITE

- Check result by using the **candidates** and inputting the index it was created into, in this case, 2

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

candidates ▾

uint256	2
↳ firstName string	Diana
↳ lastName string	Prince
↳ gender string	female
↳ date_of_birth string	02/14/1990
↳ party string	Independent
↳ voteCount uint256	0

READ

-

- Authorize a Voter by using **authorized_Voter**: the difference between this and what we did in the app is that in this, the owner of this contract **authorized** the voter whereas in the app there is no authorization we just add the voter into the system or list to able to vote.

- Assume the **addresses** in ganache is the voter

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS

CURRENT BLOCK 3 GAS PRICE 2000000000 GAS LIMIT 6721975 HARDFORK MUIRGLACIER NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:7545 MINING ST AUTOMI

MNEMONIC ?
wage bird limb knee brave velvet pony city kit pig edit human

ADDRESS	BALANCE
0x70665eB0ECCaB28dFa593Df8F9d23b965eF2705e	99.91 ETH
0xEed2B03eA7816bbf1300fDa4Ca68D01C8F9C9ff7	100.00 ETH
0x357CCd8E107006c9EB65346026D65BB40688fA08	100.00 ETH
0xC79550Df04434E3645F27B76330851F9935eEA76	100.00 ETH
0xaE6Ef65b64c78bB23eD200050f4e559582f713C4	100.00 ETH
0x402d8c14F9D833B2c949061b257A4dE55F96C406	100.00 ETH
0xBde5ebA5D2d1D1FF938E93B85368048401E54b08	100.00 ETH

Type : constructor

Access

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

authorize_Voter

_voter address
0xEed2B03eA7816bbf1300fDa4Ca68D01C8F9C9ff7

WRITE

- Add Vote by using **add_Vote** - fill in the address we just authorized and input the candidate index - if voter attempted to vote again or is not authorized - there will be an **error**

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbed433009

add_Vote ▾

_voter address
0xEed2B03eA7816bbf1300fDa4Ca68D01C8F9C9ff7

_voteIndex uint256
1

*→ vote for candidate at index 1
Thomas Whitmore*

WRITE

- Check result

Read / Write Contract
0xD6234D74498FC50Eb5DA8614D739a5bbed433009

candidates ▾

uint256
1

↳ **firstName** string
Thomas

↳ **lastName** string
Whitmore

↳ **gender** string
male

↳ **date_of_birth** string
07/04/1970

↳ **party** string
Democratic

↳ **voteCount** uint256
1

(The 'voteCount' field is circled in red)

-

- Get Voter information using **voters**- put in the address we just authorized and used to vote:

Read / Write Contract

0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

voters ▾

address

0xEed2B03eA7816bbf1300fDa4Ca68D01C8F9C9ff7

↳ **authorized** bool
 TRUE

↳ **voted** bool
 TRUE

↳ **vote** uint256

voted for candidate at index 1

READ

A voter that has not authorized and voted yet:

Read / Write Contract

0xD6234D74498FC50Eb5DA8614D739a5bbcd433009

voters ▾

address

0x357CCd8E107006c9EB65346026D65BB40688fA08

↳ **authorized** bool
 FALSE

↳ **voted** bool
 FALSE

↳ **vote** uint256

READ

-

- Get the winner using **winnerName**

Read / Write Contract

0xD6234D74498FC50Eb5DA8614D739a5bbed433009

winnerName ▾

↳ _firstName string

Thomas

↳ _lastName string

Whitmore

↳ _gender string

male

↳ _dob string

07/04/1970

↳ _party string

Democratic

↳ _votes uint256

3

↳ _totalVotes uint256

5

votes for this candidate

votes

for this

in this

total votes in this election

