

Building Smart Contracts with Remix

NINA BREZNIK

@ninabreznik

YANN LEVREAU

@ninabreznik

IURI MATIAS

@iurimatias

ROB STUPAY

@ryestew

ALEX PRAETORIUS

@SERAPATH

LIANA HUSIKYAN

@LianaHus

Ballot Dapp Workshop

bit.ly/remix-workshop-repository

The screenshot displays the Ballot Dapp Workshop interface. At the top right is a "CREATE NEW PROPOSAL" button. Below it are four rounded rectangular boxes labeled "ROUND: 1", "ROUND: 2", "ROUND: 3", and "ROUND: 4". A central call-to-action bar reads "Vote for proposal and help us reward the projects that benefit the community!". The main content area is titled "PROPOSALS" and contains a "PROPOSAL TITLE/DESCRIPTION" section with the instruction "Choose only one". It lists two proposals: "PROPOSAL 1" (with a placeholder text about Lorem ipsum dolor sit amet) and "PROPOSAL 3" (with a placeholder text about Neque porro quisquam est, qui dolorem ipsum quia ...). Each proposal has a small circular icon to its right.

CREATE NEW PROPOSAL

ROUND: 1 ROUND: 2 ROUND: 3 ROUND: 4

Vote for proposal and help us reward the projects that benefit the community!

PROPOSALS

PROPOSAL TITLE/DESCRIPTION

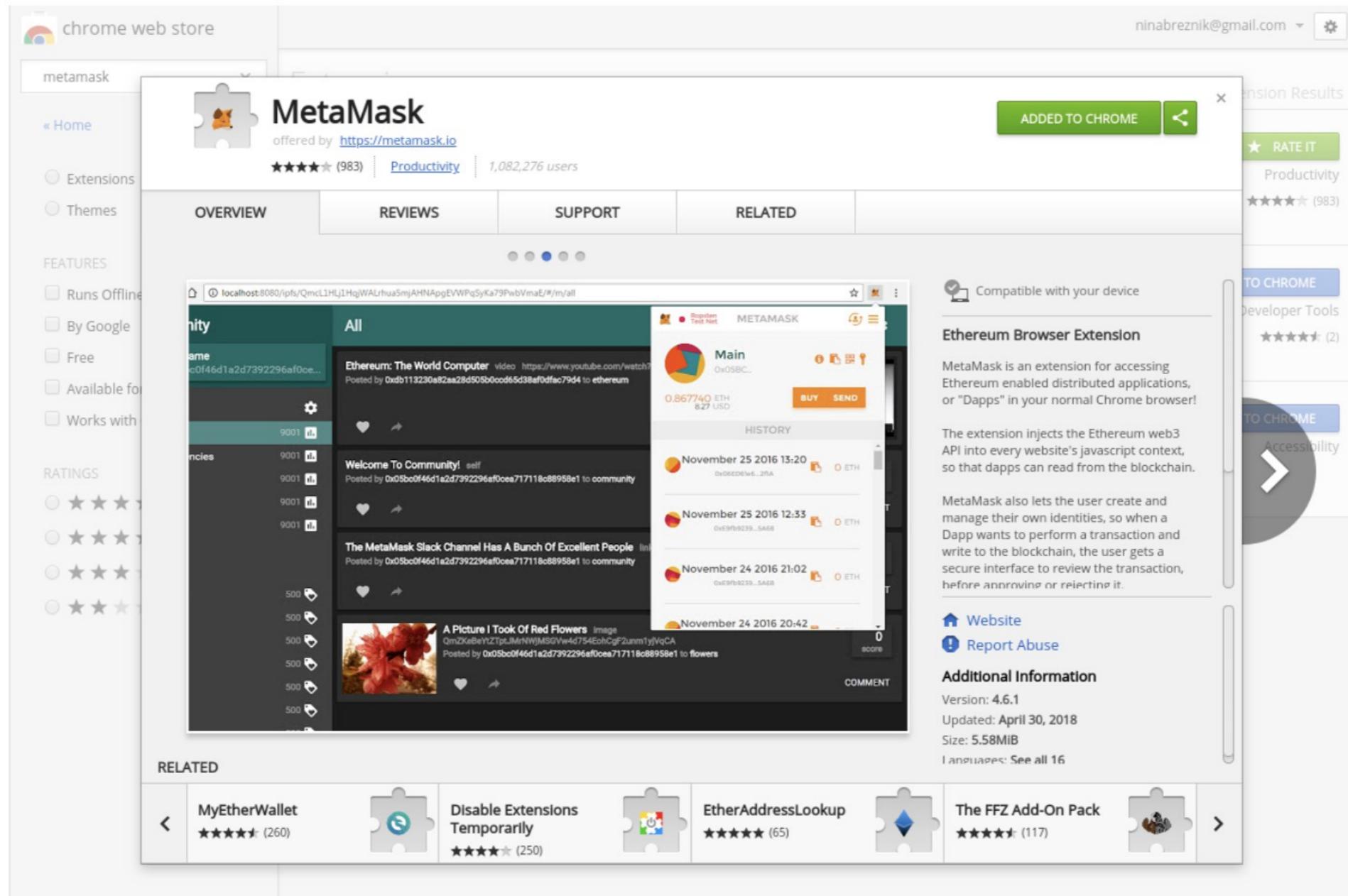
Choose only one

» PROPOSAL 1
Lorem ipsum dolor sit amet, consectetur adipiscin...
1 vote(s)

» PROPOSAL 3
Neque porro quisquam est, qui dolorem ipsum quia ...
0 vote(s)

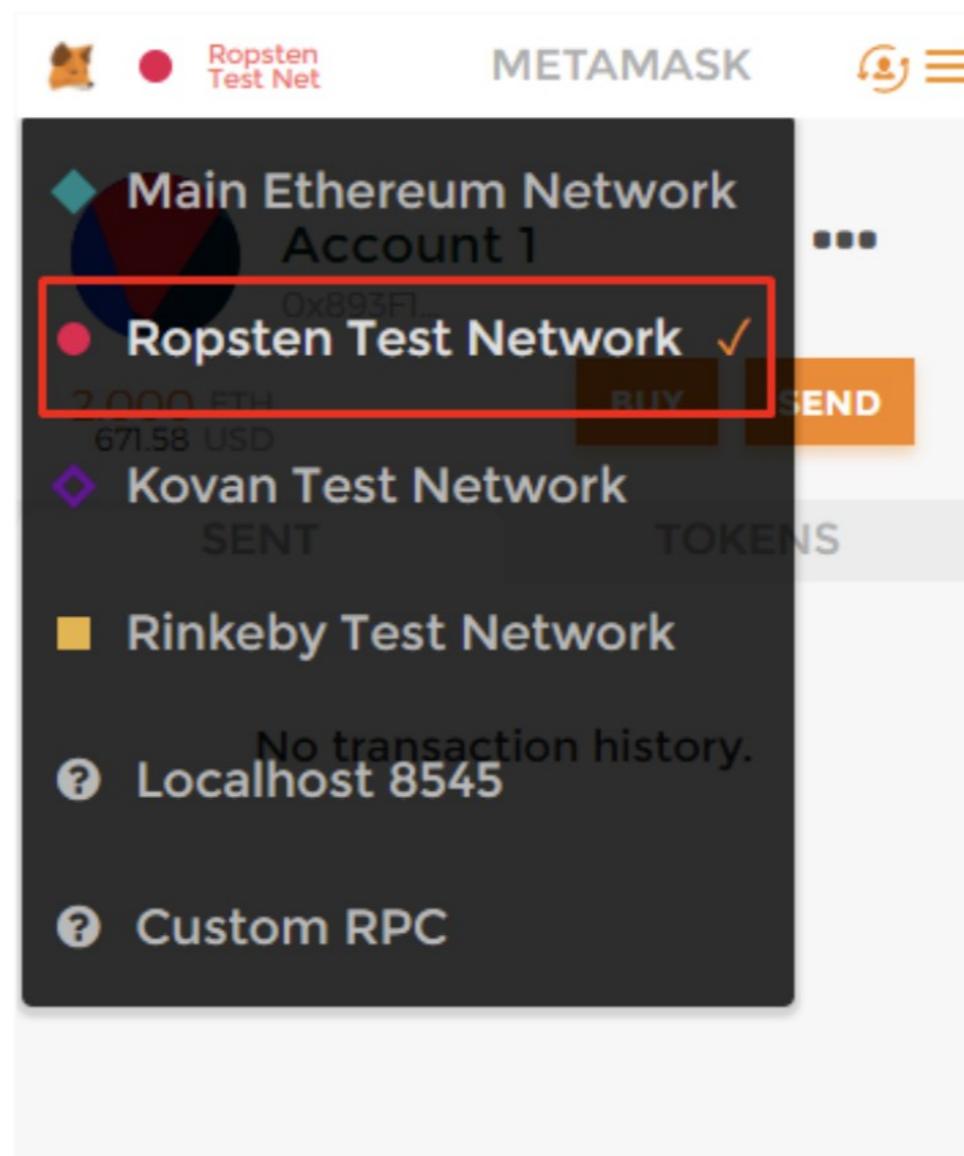
Install Metamask

chrome.google.com/webstore



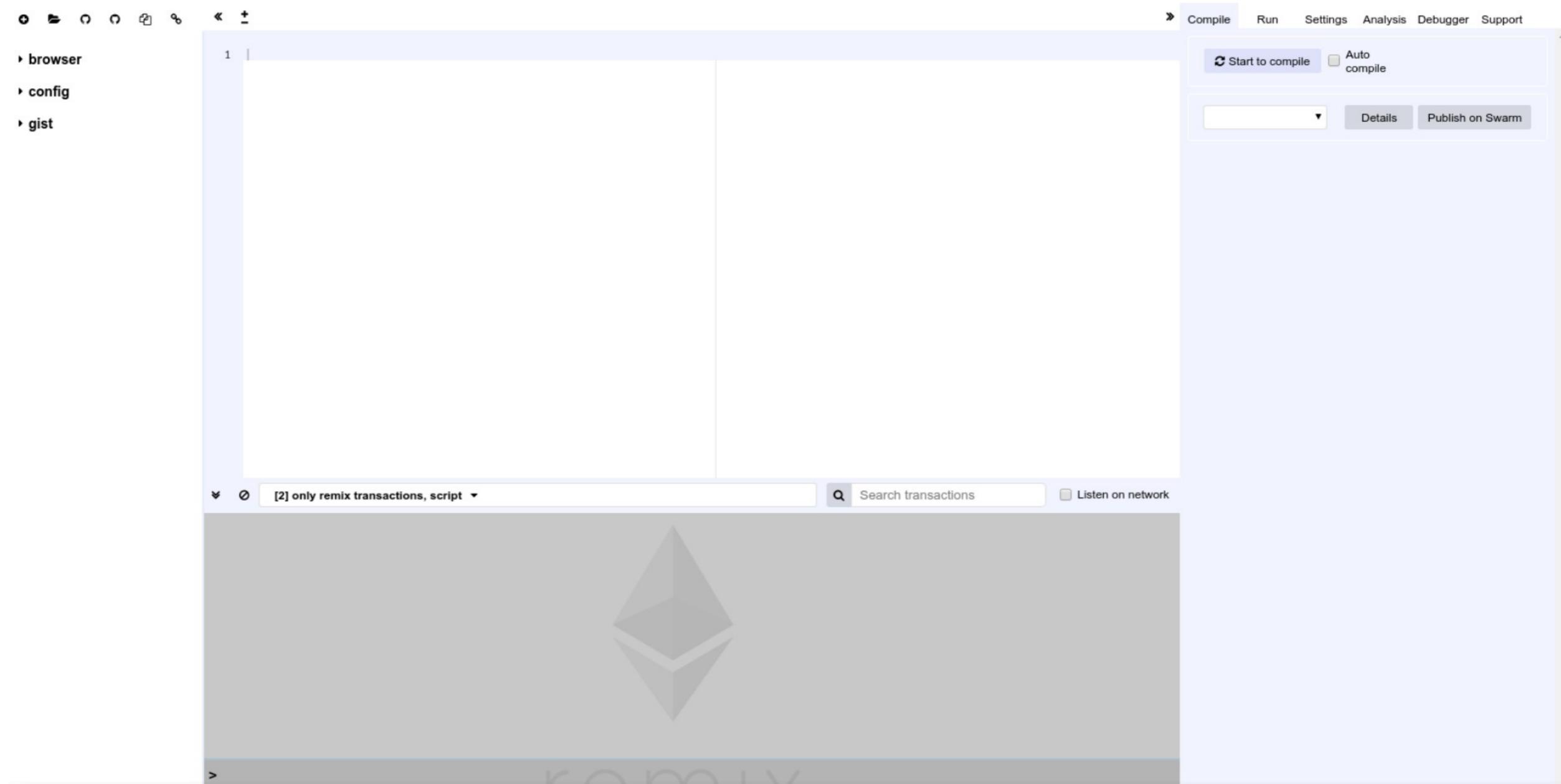
Login to Metamask

Robsten Test Network



Let's get started

<https://bit.ly/remix-workshop-repository>



Remix Tour

<https://remix-alpha.ethereum.org>

File Explorer



browser
AwardToken.sol
Ballot2.sol
Ballot_orig.sol
Donation.sol
README.md
multiSig2.sol
multisig.sol
multisig1.sol
scenario.json
setup.txt

config

Editor

```
1 pragma solidity ^0.4.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17 }
```

[2] only remix transactions, script ▾ Search transactions

Terminal

- Welcome to Remix v0.6.4 -

You can use this terminal for:

- Checking transactions details and start debugging.
- Running JavaScript scripts.
- Running JavaScript scripts involving web3 if the current environment is injected provider or Web3 provider.
- Executing common command to interact with the Remix interface (see list of commands below). Note that these command can also be included in a JavaScript script.

remix.debug(hash): Start debugging a transaction.

remix.loadgist(id): Load a gist in the file explorer.

remix.loadurl(url): Load the given url in the file explorer. The url can be of type git, hub, swarm or ipfs.

remix.setproviderurl(url): Change the current provider to Web3 provider and set the url endpoint.

remix.exeCurrent(): Run the script currently displayed in the editor

remix.help(): Display this help message

Console

Compile Tab (active)

» Compile Run Settings Analysis Debugger Support T...

Start to compile Auto compile Hide warnings

Ballot

Details Publish on Swarm

Static Analysis raised 2 warning(s) that requires your attention. ×
Click here to show the warning(s).

browser/Ballot_orig.sol:19:5: Warning: Defining ×
function Ballot(uint8 _numProposals) public
^ (Relevant source part starts here and spans

Run Tab

Compile Run Settings Analysis Debugger Support Test

Environment Injected Web3 Ropsten (3) i

Account 0x9ae...06ff6 (1.992485469305616838) +

Gas limit 3000000

Value 0 wei

AwardToken

Deploy

Load contract from Address At Address

Transactions recorded: 4

Deployed Contracts

AwardToken at 0x574...40360 (blockchain)

approve address _spender, uint256 _value

closeRound

closeRoundEarly

decreaseApproval address _spender, uint256 _subtractedValue

finishMinting

increaseApproval address _spender, uint256 _addedValue

mint address _to, uint256 _amount

renounceOwnership

Universal DAPP

UI to the Contract

The screenshot shows the Truffle UI's Run tab interface. At the top, tabs for Compile, Run, Settings, Analysis, Debugger, Support, and Test are visible, with Run selected. Below the tabs are configuration fields for Environment (Injected Web3, Ropsten), Account (0x9ae...06ff6), Gas limit (3000000), and Value (0 wei). A dropdown menu shows 'AwardToken' selected. A 'Deploy' button is present. Below this, options for 'Load contract from Address' and 'At Address' are shown. A section titled 'Transactions recorded: 4' lists recent transactions. The main area displays the 'Deployed Contracts' section for 'AwardToken at 0x574...40360 (blockchain)'. It lists several functions: approve (address _spender, uint256 _value), closeRound, closeRoundEarly, decreaseApproval (address _spender, uint256 _subtractedValue), finishMinting, increaseApproval (address _spender, uint256 _addedValue), mint (address _to, uint256 _amount), and renounceOwnership. Each function has a corresponding input field below it. The bottom left contains the text 'Universal DAPP' and 'UI to the Contract'.

Remix Commands

<https://remix-alpha.ethereum.org>

The screenshot shows the Remix IDE interface. On the left, a sidebar lists project files: browser/AwardToken.sol, browser/Ballot2.sol, browser/Ballot_orig.sol (selected), browser/Donation.sol, README.md, multiSig2.sol, multisig.sol, multisig1.sol, scenario.json, and setup.txt. Below this is a 'config' section.

The main area displays the Solidity code for the Ballot contract:

```
1 pragma solidity ^0.4.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17 }
```

The code editor has tabs for 'Compile', 'Run', 'Settings', 'Analysis', 'Debugger', and 'Support'. A status bar at the bottom indicates 'browser/Ballot_orig.sol:19:5: Warning: Defining function Ballot(uint8 _numProposals) public + ^ (Relevant source part starts here and spans)

On the right, the 'Ballot' contract details are shown, including 'Details', 'Publish on Swarm', 'ABI', and 'Bytecode' buttons. A message box states: 'Static Analysis raised 2 warning(s) that requires your attention. Click here to show the warning(s.)' A yellow box highlights the warning message.

The bottom section is a terminal window with the title '[2] only remix transactions, script'. It displays the following text:

```
- Welcome to Remix v0.6.4 -
You can use this terminal for:
- Checking transactions details and start debugging.
- Running JavaScript scripts.
- Running JavaScript scripts involving web3 if the current environment is injected provider or Web3 provider.
- Executing common command to interact with the Remix interface (see list of commands below). Note that these command can also be included in a JavaScript script.
```

A purple box highlights the command list:

```
remix.debug(hash): Start debugging a transaction.
remix.loadgist(id): Load a gist in the file explorer.
remix.loadurl(url): Load the given url in the file explorer. The url can be of type git, swarm or ipfs.
remix.setproviderurl(url): Change the current provider to Web3 provider and set the url endpoint.
remix.exeCurrent(): Run the script currently displayed in the editor
remix.help(): Display this help message
```

Set environment

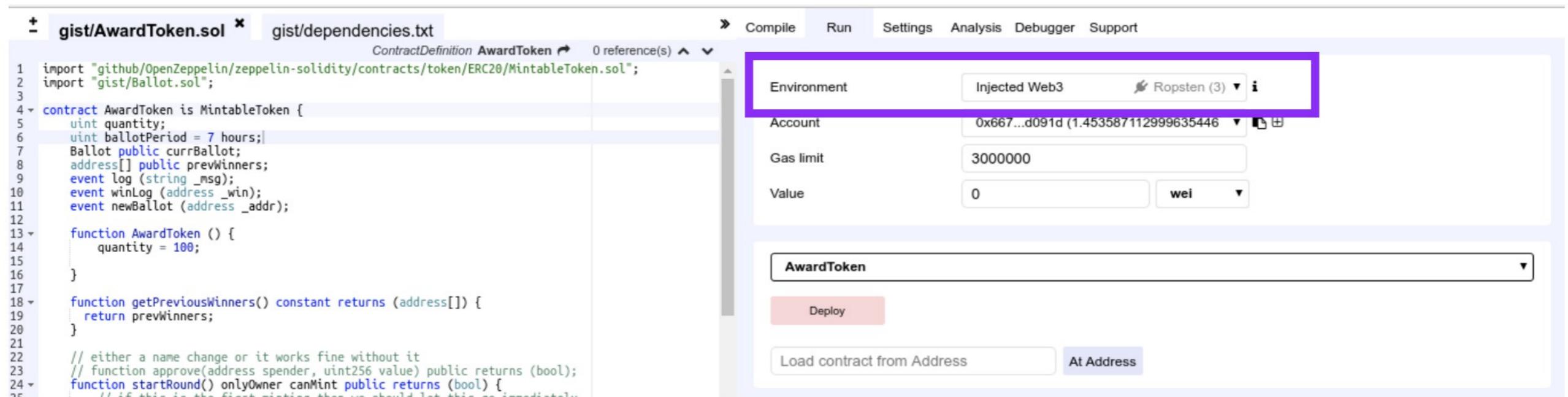
Run tab-> Environment:

Javascript VM - a simple and quick environment – only visible to your machine

Injected web3 (Ropsten) – or whatever testnet you have setup in metamask

Web3 Provider

Choose: **Injected web3 (Ropsten)**



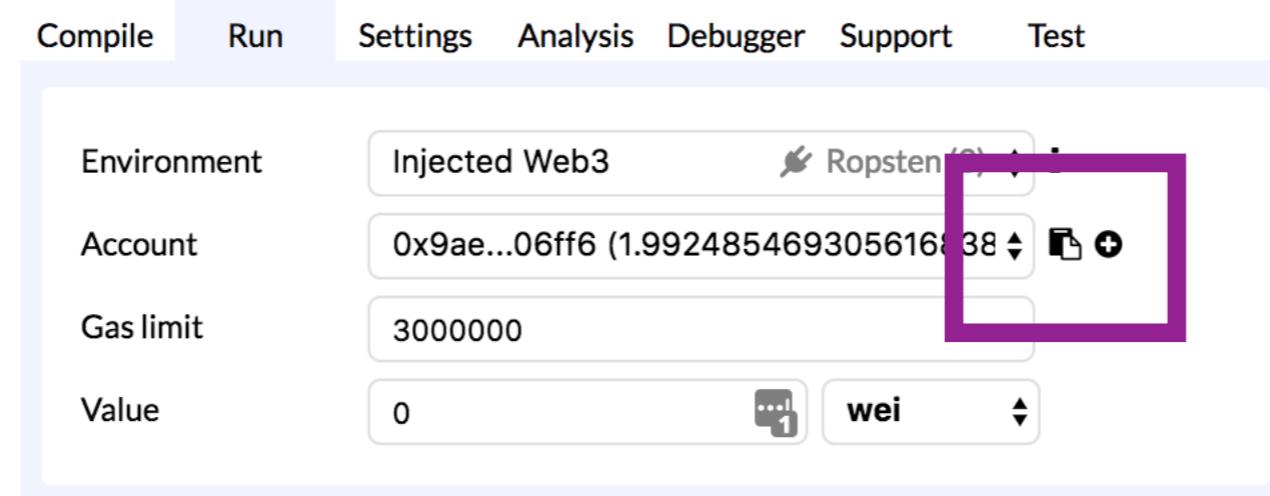
The screenshot shows the Truffle UI interface with the following details:

- Left Panel:** Displays the Solidity code for the `AwardToken.sol` contract. The code includes imports from OpenZeppelin and a custom `Ballot.sol`, defines a `AwardToken` contract that inherits from `MintableToken`, and includes functions for minting tokens and getting previous winners.
- Top Bar:** Shows tabs for `ContractDefinition AwardToken`, `0 reference(s)`, and various menu options: `Compile`, `Run`, `Settings`, `Analysis`, `Debugger`, and `Support`.
- Right Panel:** Contains the `Environment` configuration section, which is highlighted with a purple border.
 - Account:** Set to `Injected Web3` (Ropsten (3))
 - Gas limit:** Set to `3000000`
 - Value:** Set to `0` with unit `wei`
- Contract Selection:** A dropdown menu is open, showing the `AwardToken` contract.
- Bottom Buttons:** Includes a `Deploy` button and a `Load contract from Address` input field with a `At Address` button.

Get some TEST ether

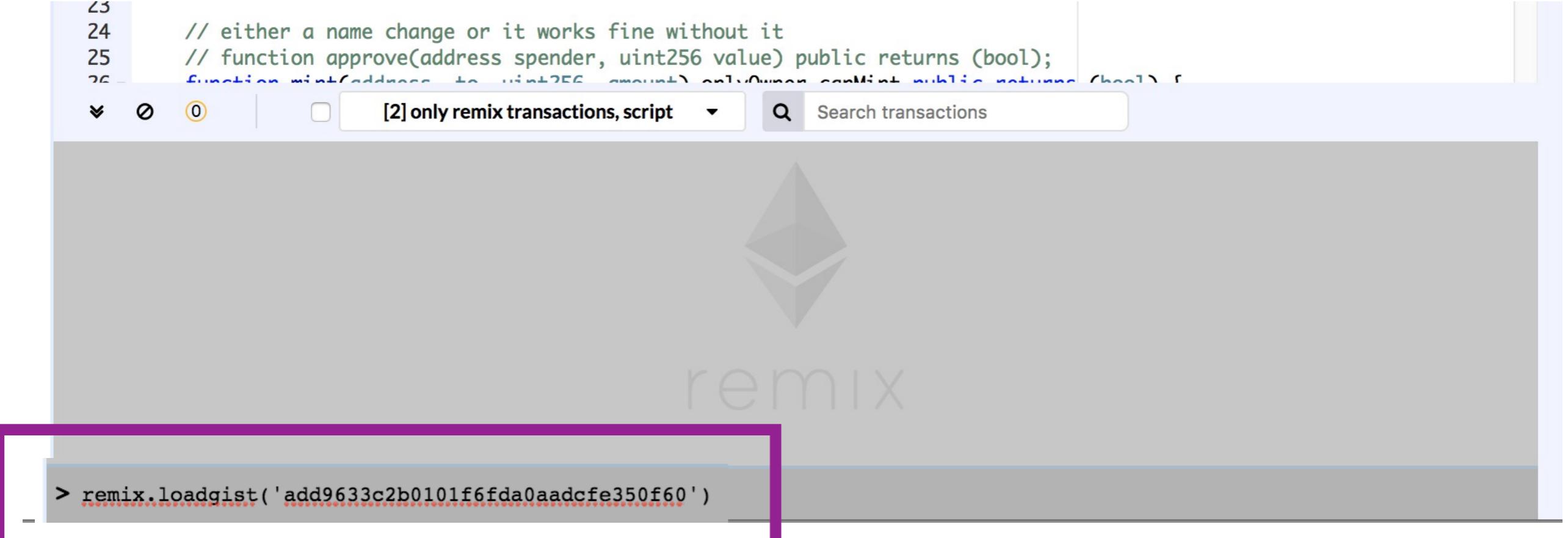
<http://faucet.ropsten.be:3001/>

But FIRST:
Copy your address - here or in Metamask



Load files to Remix

```
remix.loadgist('add9633c2b0101f6fda0aadcfe350f60')
```



The screenshot shows the Remix IDE interface. At the top, there is a code editor window displaying Solidity code. Below it is a transaction history section with a dropdown menu set to "[2] only remix transactions, script". A search bar labeled "Search transactions" is also present. The main area of the interface features the Ethereum logo and the word "remix". At the bottom, there is a purple-bordered terminal-like window containing the command: `> remix.loadgist('add9633c2b0101f6fda0aadcfe350f60')`. This command is highlighted with a red dotted underline.

here in the console

Open file

gist/AwardToken

The screenshot shows the Remix IDE interface. The left sidebar has a purple box around the 'gist' folder, which contains 'AwardToken.sol'. The main editor area shows the Solidity code for 'AwardToken.sol'.

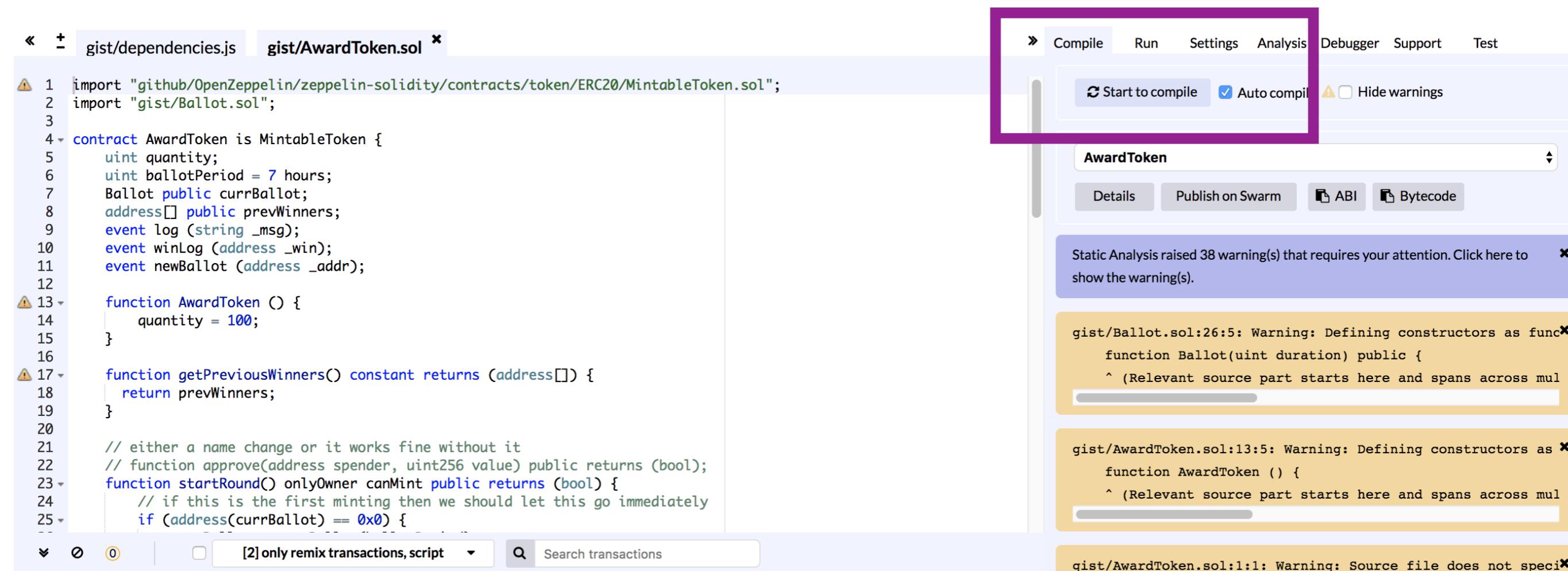
```
1 import "./ERC20Mintable.sol";
2 import "./Ballot.sol";
3
4 contract AwardToken is ERC20Mintable {
5     uint quantity;
6     uint ballotPeriod = 7 hours;
7     Ballot public currBallot;
8     address[] public prevWinners;
9     event log (string _msg);
10    event winLog (address _win);
11    event newBallot (address _addr);
12
13    function AwardToken () {
14        quantity = 100;
15    }
16
17    function getPreviousWinners() constant returns (address[])
18    {
19        return prevWinners;
20    }
21
22    // either a name change or it works fine without it
23    // function approve(address spender, uint256 value) public returns (bool);
24    function startRound() onlyMinter public returns (bool) {
25        // if this is the first minting then we should let this go immediately
26        if (address(currBallot) == 0x0) {
27            currBallot = new Ballot(ballotPeriod);
28            prevWinners.push(_addr);
29        }
30    }
31}
```

The bottom navigation bar shows '2 only remix transactions, script' and a search bar 'Search transactions'.

Compile the contract

Compile tab: Start to compile button

(when dependencies.js is the active file)



The screenshot shows the Remix IDE interface. On the left, there are two tabs: 'gist/dependencies.js' (active) and 'gist/AwardToken.sol'. The code editor displays Solidity code for a token contract. On the right, the 'Compile' tab is active, featuring a toolbar with 'Compile', 'Run', 'Settings', 'Analysis', 'Debugger', 'Support', and 'Test' buttons. A purple box highlights the 'Start to compile' button, which is currently grayed out. Below it is a checkbox for 'Auto compil'. To the right of the toolbar is a status bar with 'AwardToken' and buttons for 'Details', 'Publish on Swarm', 'ABI', and 'Bytecode'. A message box indicates 'Static Analysis raised 38 warning(s) that requires your attention. Click here to show the warning(s.)'. Two examples of warnings are shown in yellow boxes: one from 'gist/Ballot.sol' and one from 'gist/AwardToken.sol'. At the bottom, a transaction history shows '[2] only remix transactions, script' and a search bar.

```
1 import "github/OpenZeppelin/zeppelin-solidity/contracts/token/ERC20/MintableToken.sol";
2 import "gist/Ballot.sol";
3
4 contract AwardToken is MintableToken {
5     uint quantity;
6     uint ballotPeriod = 7 hours;
7     Ballot public currBallot;
8     address[] public prevWinners;
9     event log (string _msg);
10    event winLog (address _win);
11    event newBallot (address _addr);
12
13    function AwardToken () {
14        quantity = 100;
15    }
16
17    function getPreviousWinners() constant returns (address[])
18    {
19        return prevWinners;
20    }
21
22    // either a name change or it works fine without it
23    // function approve(address spender, uint256 value) public returns (bool);
24    function startRound() onlyOwner canMint public returns (bool) {
25        // if this is the first minting then we should let this go immediately
26        if (address(currBallot) == 0x0) {
```

See compiled contracts

AwardToken + all dependencies



Imported Contracts

The screenshot shows the Truffle UI interface. At the top, there is a navigation bar with tabs: Compile, Run (which is selected), Settings, Analysis, Debugger, Support, and Test. Below the navigation bar, there are several configuration fields:

- Environment: Injected Web3 (Ropsten (3))
- Account: 0x9ae...06ff6 (1.992485469305616838)
- Gas limit: 3000000
- Value: 0 wei

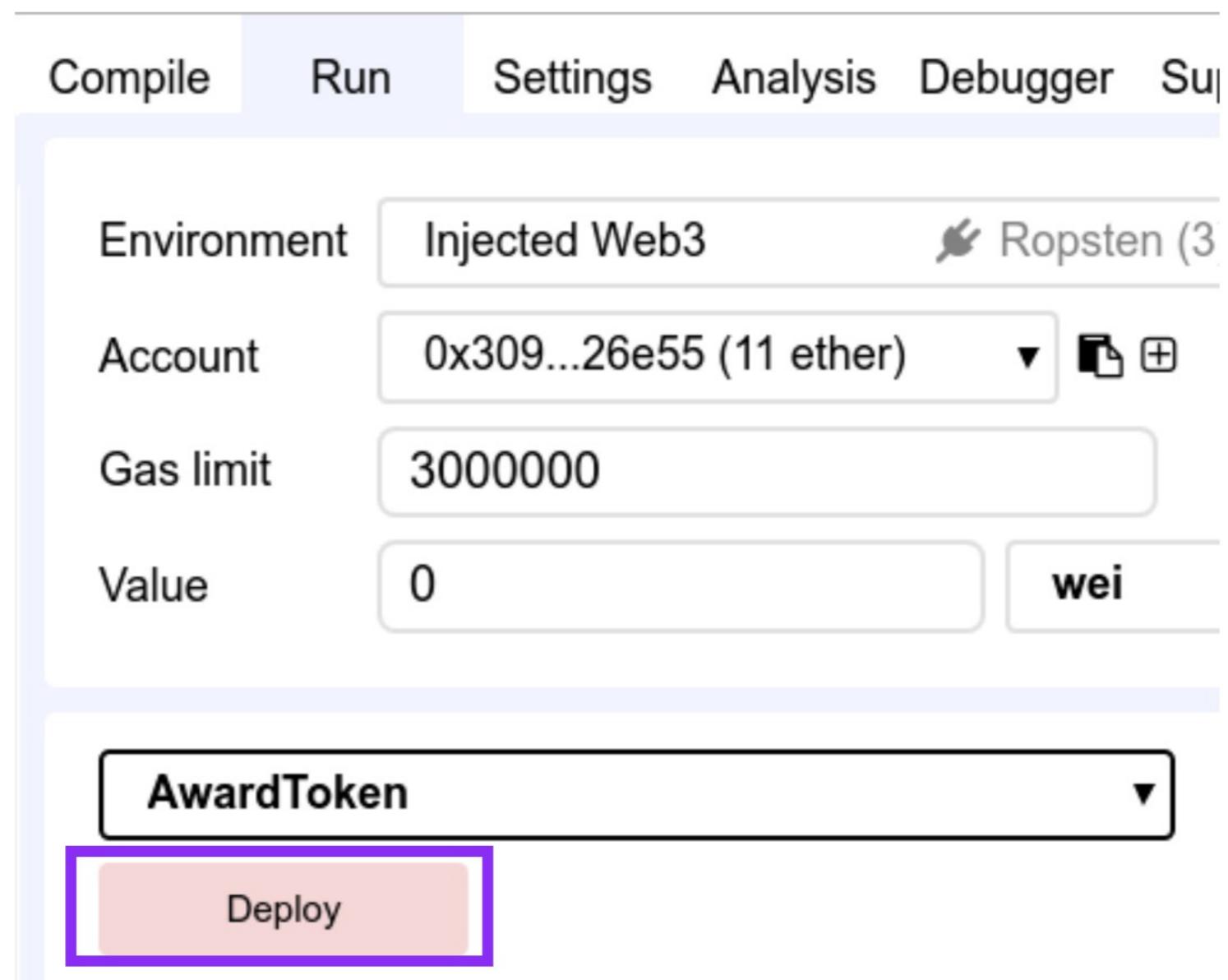
In the center, a dropdown menu is open, showing a list of imported contracts:

- AwardToken (selected)
- Ballot
- SafeMath
- Ownable
- BasicToken
- ERC20
- ERC20Basic
- MintableToken
- StandardToken

At the bottom left, there is a section labeled "Deployed Contracts" with a trash can icon.

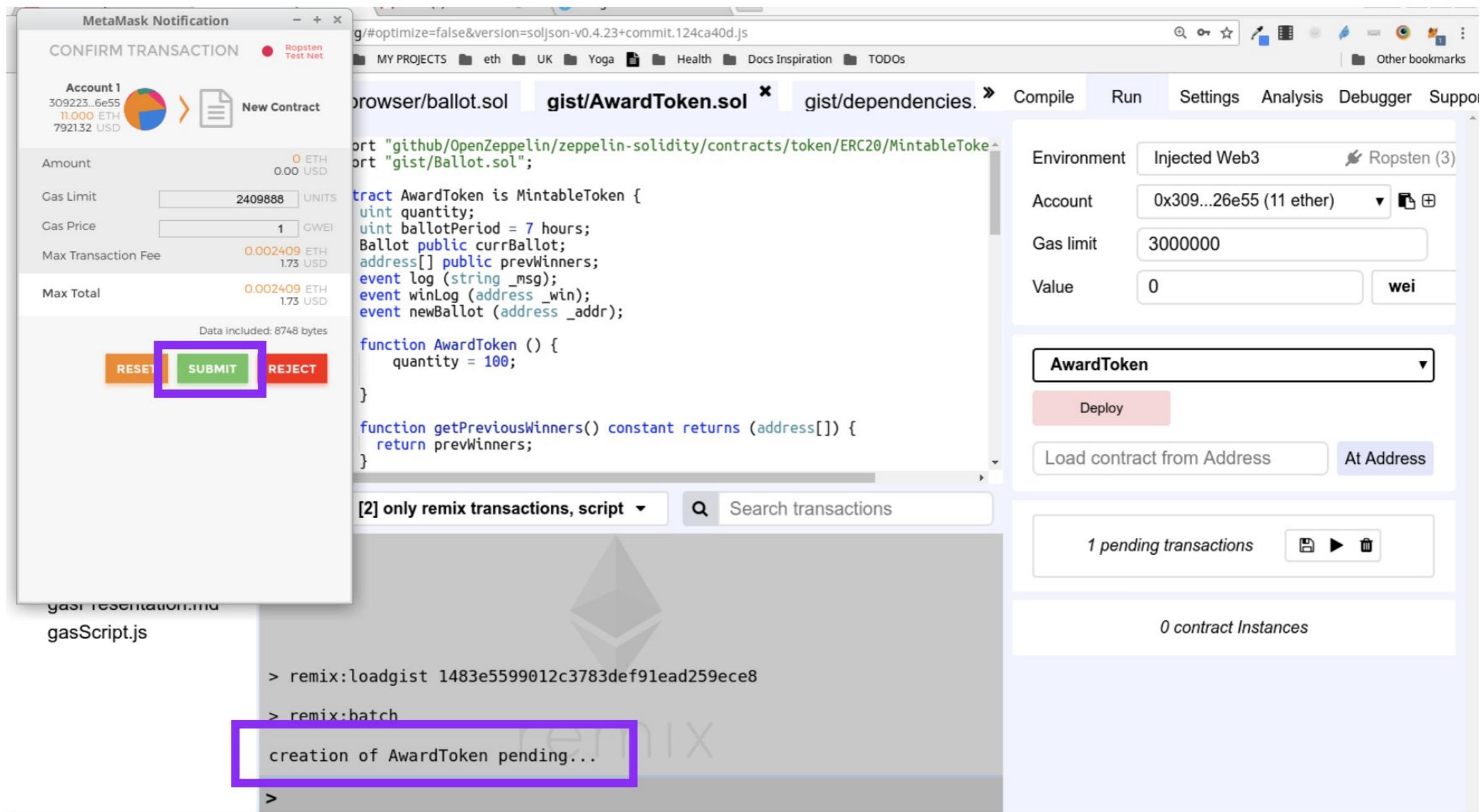
Deploy the contract

Run tab: Deploy button



Confirm the transaction

Submit button
But make sure you put in a gas price!



Check if tx is mined

Terminal logs in Remix

creation of AwardToken pending...

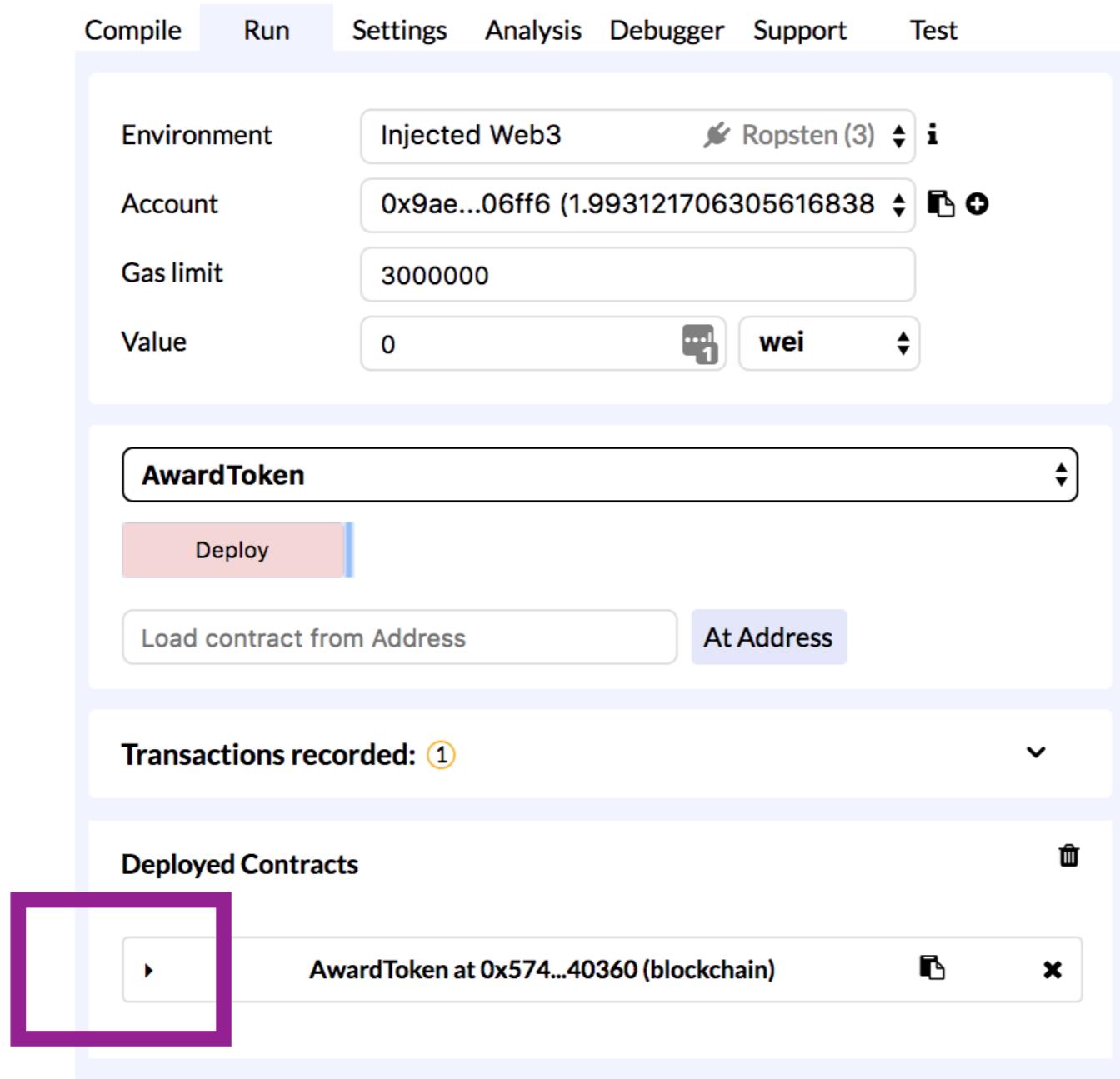
<https://ropsten.etherscan.io/tx/0x404a4445ebb3a969b15257a586a61582afa07dcf02b1b2617f77519b30378be8>

▶ [block:3159099 txIndex:2] from:0x309...26e55
to:AwardToken.(constructor) value:0 wei data:0x608...70029
logs:0 hash:0x404...78be8

Debug

Click to see the contract's UI

On the deployed contract



Behold!

The Interactive UI for AwardToken.sol contract

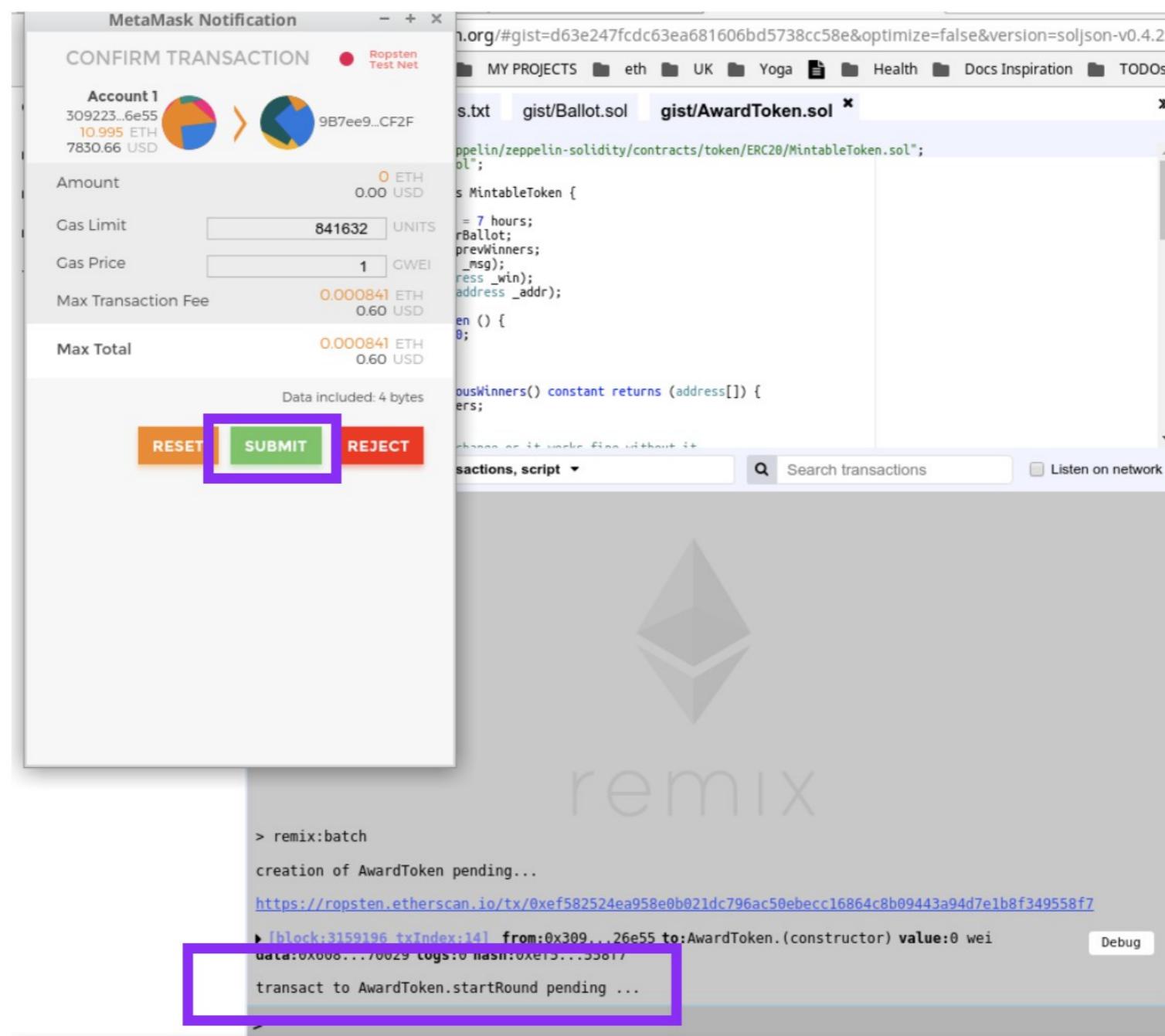


Execute startRound

Its a payable function
(as opposed to a call function - which is free)



Confirm the transaction



Check if tx is mined

In the terminal logs in Remix

```
transact to AwardToken.startRound pending ...
```

```
https://ropsten.etherscan.io/tx/0x5a97b4946979f52dfb6dc8ab2fecebb8fd43515ff4e25597ecb9d0a88472c8b2
```

```
▶ [block:3159300 txIndex:12] from:0x309...26e55 to:AwardToken.startRound() 0x9b7...0cf2f  
value:0 wei data:0x55e...3f086 logs:1 hash:0x5a9...2c8b2
```

Debug

Expand tx log

to see the logs

 [block:3665523 txIndex:4] from:0x9ae...06ff6 to:AwardToken.startRound() 0x574...40360 value:0 wei data:0x55e...3f086 logs:1 hash:0x16c...0a81c	Debug ^
status	0x1 Transaction mined and execution succeed
transaction hash	0x16c8af5a3fd0e5bcacd8858ab42d4f8eff39fc33bb98290740c03eeb4880a81c 
from	0x9ae59af2e33480caa48f2dc6f6cede7ffab06ff6 
to	AwardToken.startRound() 0x574d270dc04e89c5d65e24e19f1deb9e17240360 
gas	613643 gas 
transaction cost	613643 gas 
hash	0x16c8af5a3fd0e5bcacd8858ab42d4f8eff39fc33bb98290740c03eeb4880a81c 
input	0x55e...3f086 
decoded input	{ } 
decoded output	-
logs	[{ "from": "0x574d270dc04e89c5d65e24e19f1deb9e17240360", "topic": "0x65f35fb257c91daed794331bfd2ad0f4439d49319d52a5b3bfb04c8496 9fdbeb", "event": "newBallot", "args": { "0": "0xD6052C85A3D26eE9EeC8262d462bfDC672B80D93", "_addr": "0xD6052C85A3D26eE9EeC8262d462bfDC672B80D93", "length": 1 } }]
value	0 wei 

Take a look at the startRound function in the editor

– just to see what it is doing...

```
// SPDX-License-Identifier: MIT
// pragma solidity ^0.8.0;
// function approve(address spender, uint256 value) public returns (bool);
function startRound() onlyOwner canMint public returns (bool) {
    // if this is the first minting then we should let this go immediately
    if (address(currBallot) == 0x0) {
        currBallot = new Ballot(ballotPeriod);
        newBallot(currBallot);
    } else {
        revert();
    }
}

function closeRoundEarly() onlyOwner {
    if (address(currBallot) != 0x0 && !currBallot.timeOut()) {
        currBallot.finish();
    } else revert();
}

function closeRound() onlyOwner {
    // this can only be done by the owner of the contract

    if (address(currBallot) != 0x0 && currBallot.timeOut()) {
        // get winner
        address winner = currBallot.winningProposal();
        winLog(winner);
        // send to winner - but first make sure the address is valid
        if (winner == 0x0) {
            revert("Winner must be a valid address");
        }
        ...
    }
}
```

Get ballot's address

Execute currBallot call

The screenshot shows a list of functions for the contract 'AwardToken'. The 'currBallot' function is highlighted with a purple rectangular border. The other functions listed are: approve, closeRound, decreaseApproval, finishMinting, increaseApproval, mint, startRound, transfer, transferFrom, transferOwnership, allowance, balanceOf, getPreviousWinners, mintingFinished, owner, prevWinners, and totalSupply.

- approve address _spender, uint256 _value
- closeRound
- decreaseApproval address _spender, uint256 _subtractedValue
- finishMinting
- increaseApproval address _spender, uint256 _addedValue
- mint address _to, uint256 _amount
- startRound
- transfer address _to, uint256 _value
- transferFrom address _from, address _to, uint256 _value
- transferOwnership address newOwner
- allowance address _owner, address _spender
- balanceOf address _owner
- currBallot
- getPreviousWinners
- mintingFinished
- owner
- prevWinners uint256
- totalSupply

Copy ballot's address

currBallot output

The screenshot shows a blockchain interface for the `AwardToken` contract at address `0x9b7...0cf2f`. The interface lists various functions and their parameters. The `currBallot` function is highlighted with a purple rectangle around its return value. The return value is a string starting with `: address:` followed by the hex address `0xE7bF60cee009DCDb2Ad8D045c19e76597bbF3c6`.

currBallot	
: address: 0xE7bF60cee009DCDb2Ad8D045c19e76597bbF3c6	

Other listed functions include `approve`, `closeRound`, `decreaseApproval`, `finishMinting`, `increaseApproval`, `mint`, `startRound`, `transfer`, `transferFrom`, `transferOwnership`, `allowance`, `balanceOf`, `getPreviousWinners`, `mintingFinished`, `owner`, `prevWinners`, and `totalSupply`.

Switch to Ballot

Run tab: dropdown

Compile Run Settings Analysis Debugger Support

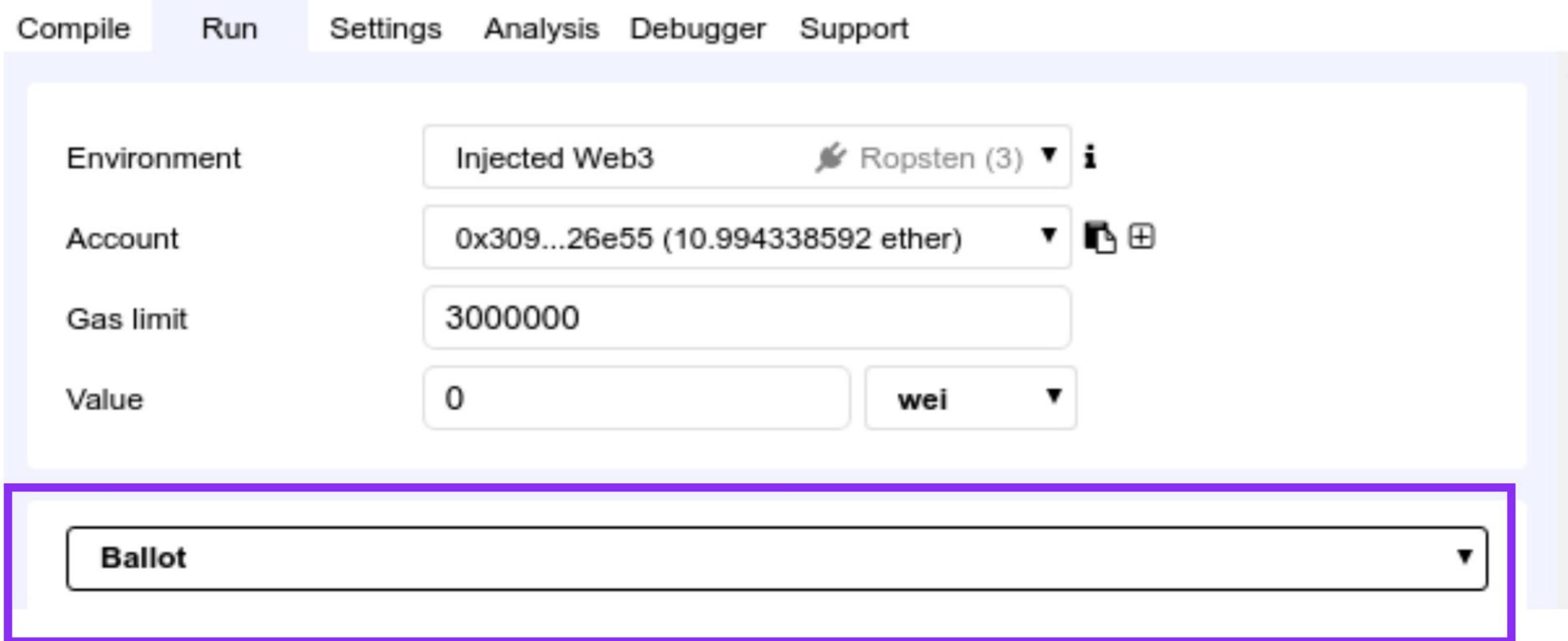
Environment Injected Web3 🌐 Ropsten (3) ▾ i

Account 0x309...26e55 (10.994338592 ether) ▾  

Gas limit 3000000

Value 0 wei ▾

Ballot ▾



Access Ballot contract

Paste address + click At Address

The screenshot shows a user interface for interacting with a Ethereum smart contract, specifically a Ballot contract. The top navigation bar includes links for Compile, Run, Settings, Analysis, Debugger, and Support. The Settings tab is currently selected.

The configuration section contains the following fields:

- Environment:** Set to "Injected Web3" with a dropdown showing "Ropsten (3)" and an information icon.
- Account:** Set to "0x309...26e55 (10.994338592 ether)" with a dropdown, a copy icon, and a plus icon.
- Gas limit:** Set to "3000000".
- Value:** Set to "0" with a dropdown menu showing "wei".

A dropdown menu at the bottom left is set to "Ballot". Below it, there are two buttons: "Deploy" (highlighted in pink) and "uint256 duration". At the bottom, there are two input fields: one containing the address "0xE7bF60cee009DCDb2Ad8D045c19" and another labeled "At Address". Both of these bottom fields are highlighted with a purple rectangular border.

Access Ballot contract

Paste address + click At Address

The screenshot shows a user interface for interacting with a Ethereum smart contract, specifically a Ballot contract. The top navigation bar includes links for Compile, Run, Settings, Analysis, Debugger, and Support. The Settings tab is currently selected.

The configuration section contains the following fields:

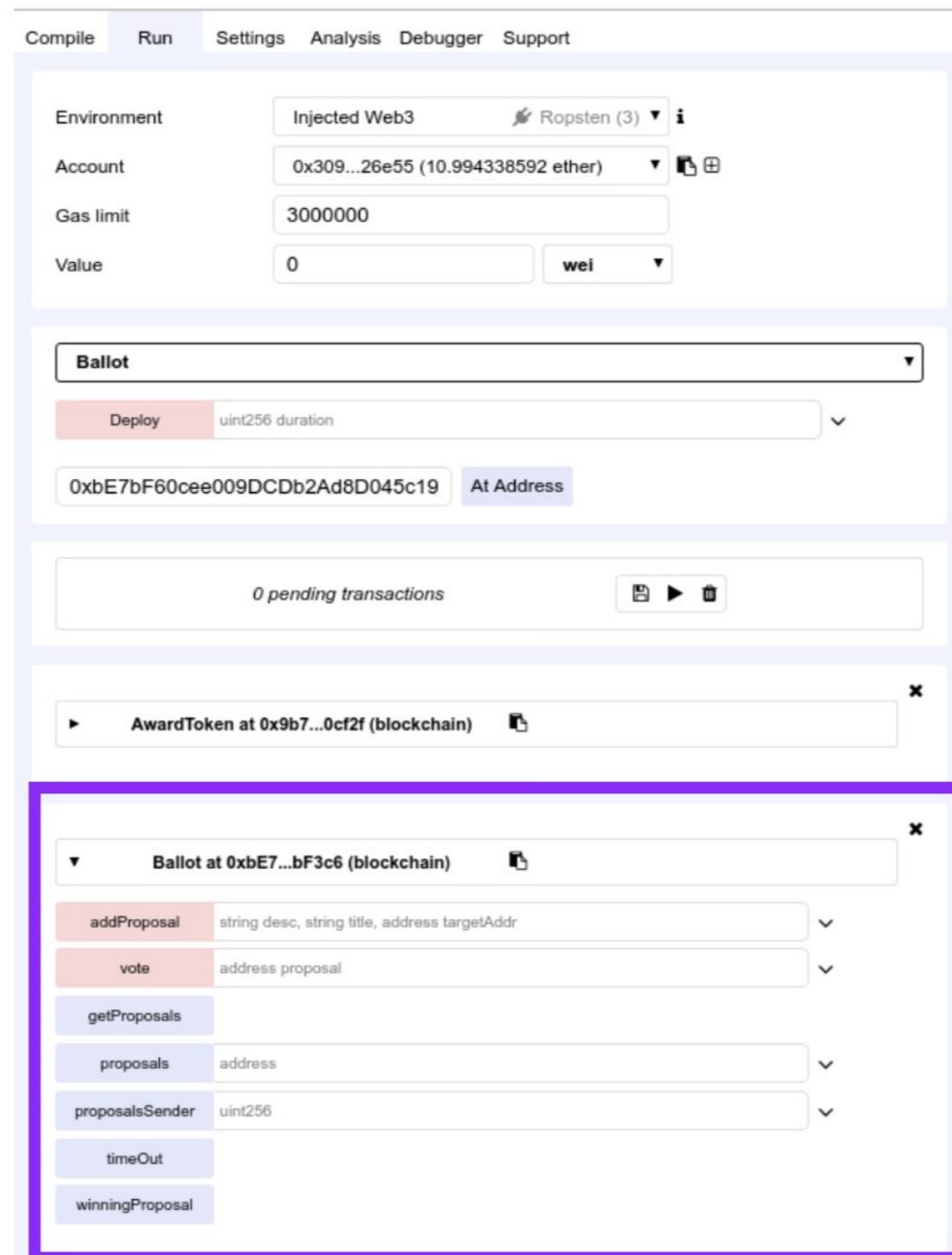
- Environment:** Set to "Injected Web3" with a dropdown showing "Ropsten (3)" and an information icon.
- Account:** Set to "0x309...26e55 (10.994338592 ether)" with a dropdown, a copy icon, and a plus icon.
- Gas limit:** Set to "3000000".
- Value:** Set to "0" with a dropdown menu showing "wei".

A dropdown menu at the bottom left is set to "Ballot". Below it, there are two buttons: "Deploy" (highlighted in pink) and "uint256 duration". At the bottom, there are two input fields: one containing the address "0xE7bF60cee009DCDb2Ad8D045c19" and another labeled "At Address". Both of these bottom fields are highlighted with a purple rectangular border.

See autogenerated UI

(you might need to scroll down)

- Interact with all the functions in the contract
- And all the functions in the inherited contracts



Add a new proposal

Expand addProposal function

▼ **Ballot at 0xbE7...bF3c6 (blockchain)** 

×

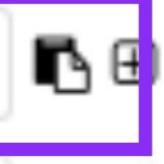
addProposal	string desc, string title, address targetAddr	
vote	address proposal	
getProposals		
proposals	address	
proposalsSender	uint256	
timeOut		
winningProposal		

Copy your address

Run tab: Account

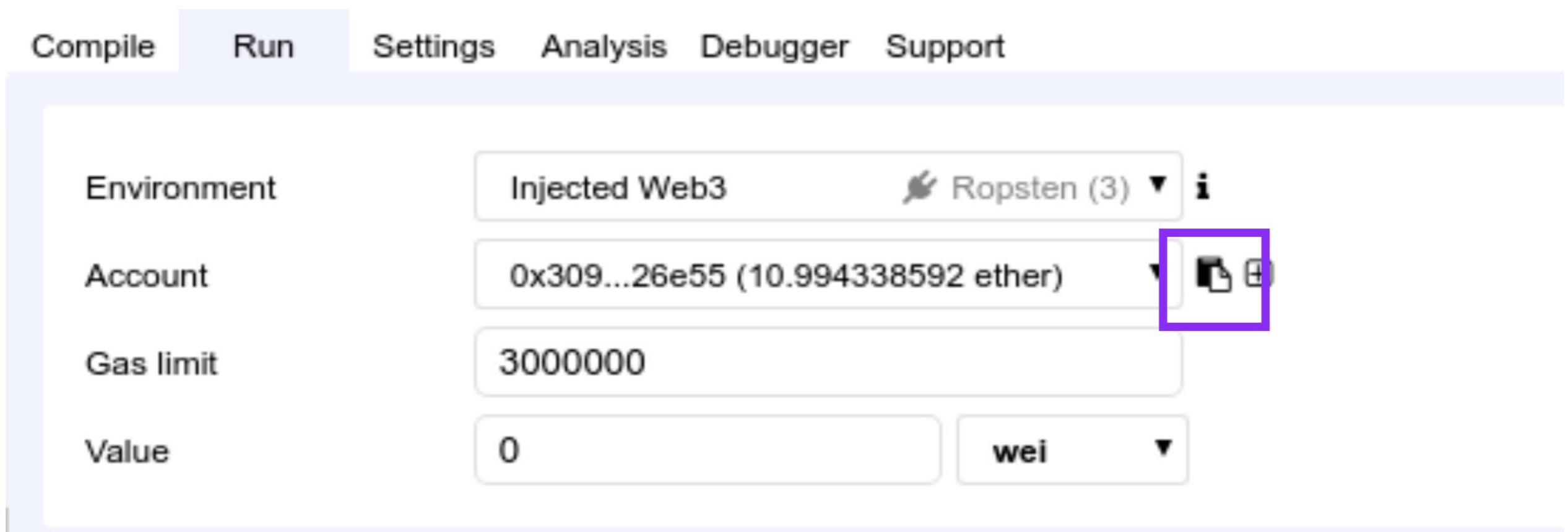
Compile Run Settings Analysis Debugger Support

Environment Injected Web3 Ropsten (3) ▾ i

Account 0x309...26e55 (10.994338592 ether) ▾ 

Gas limit 3000000

Value 0 wei ▾



Type a proposal

Run tab: Account

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

- Contract Tab:** Ballot at 0xbE7...bF3c6 (blockchain)
- Code View:** The code for the Ballot contract is visible.
- Storage View:** The storage view shows the state of the ballot contract.
- Transactions View:** The transactions view shows recent transactions.
- Logs View:** The logs view shows recent logs.
- Gas View:** The gas view shows the current gas usage.
- Run Tab:** The "Account" tab is selected. A proposal is being typed into the "addProposal" field:
 - desc:** "I think you could add a new feature to Remix that does..."
 - title:** "This is my Remix improvements proposal"
- Target Address:** An input field for the target address where the proposal will be sent.
- Buttons:** A blue "call" button and a pink "transact" button.

Add your address

Paste the address

▼ Ballot at 0xbE7...bF3c6 (blockchain) ✖

addProposal ^

desc: "I think you could add a new feature to Remix that does..."

title: "This is my Remix improvements proposal"

targetAddr: "0x3092232fb25e6b359a9fead9ed07ad752ff26e55"

transact

Execute addProposal

Hit the transact button

▼ Ballot at 0xbE7...bF3c6 (blockchain)  X

addProposal ^

desc: "I think you could add a new feature to Remix that does..."

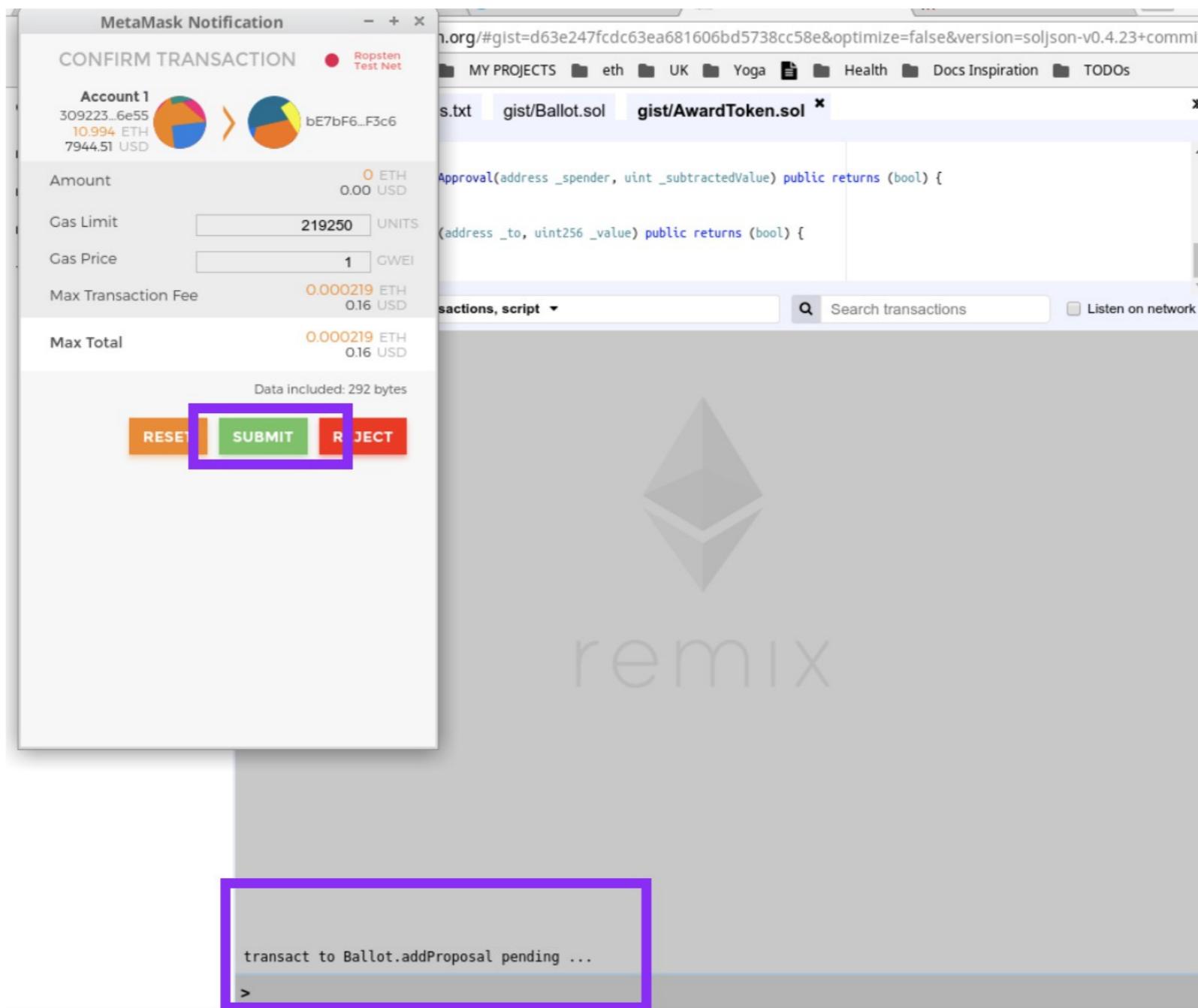
title: "This is my Remix improvements proposal"

targetAddr: "0x3092232fb25e6b359a9fead9ed07ad752ff26e55"



Confirm the transaction

Submit button



Check if tx succeeded

Terminal logs in Remix

Execute getProposals

getProposals call

Ballot at 0xbE7...bF3c6 (blockchain) 

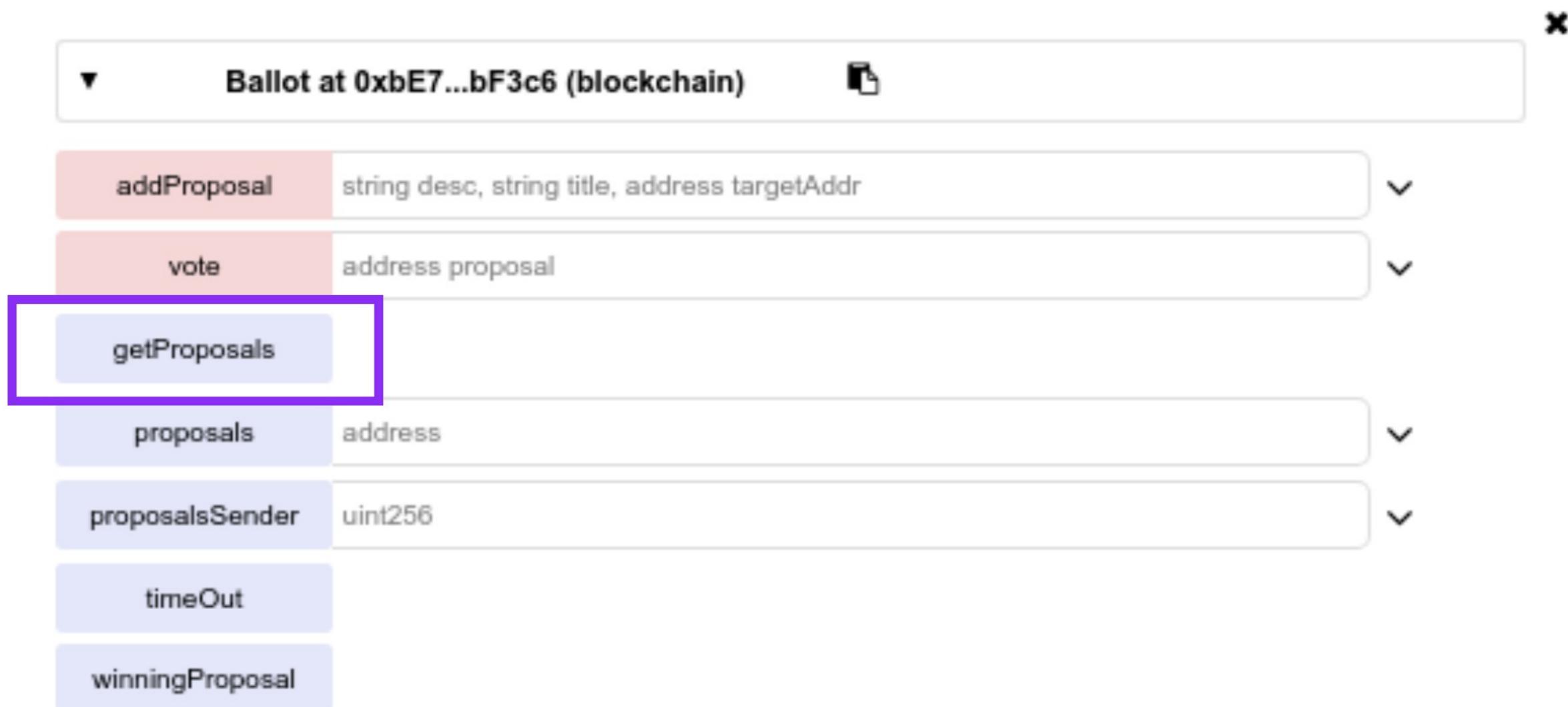
getProposals

proposals address

proposalsSender uint256

timeOut

winningProposal



See Proposals Addresses

well in so far there will only be 1 address

call to Ballot.getProposals

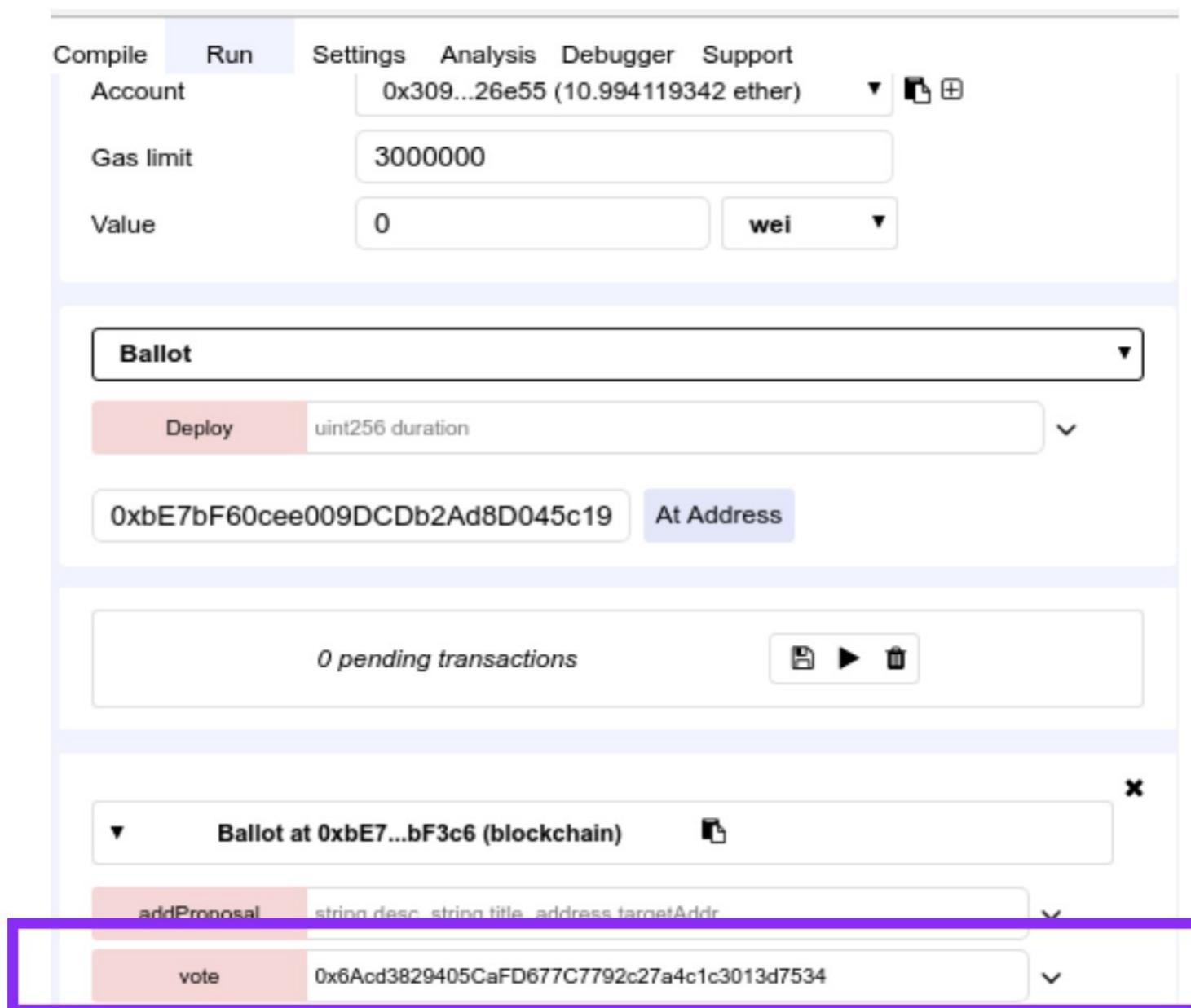
▼ [call] from:0x3092232fb25e6b359a9fead9ed07ad752ff26e55 to:Ballot.getProposals()
data:0x625...64c48

Debug

from	0x3092232fb25e6b359a9fead9ed07ad752ff26e55
to	Ballot.getProposals() 0xbE7bF60cee009DCDb2Ad8D045c19e76597bbF3c6
input	0x62564c48
decoded input	{}
decoded output	{ "0": "address[]: 0x3092232FB25e6b359a9fEad9eD07Ad752Ff26e55,0xFd0f51afb6 85Cd8735AfE7685D21355589602b8c,0x6Acd3829405CaFD677C7792c27a4c1c3013d7534" }
logs	[]

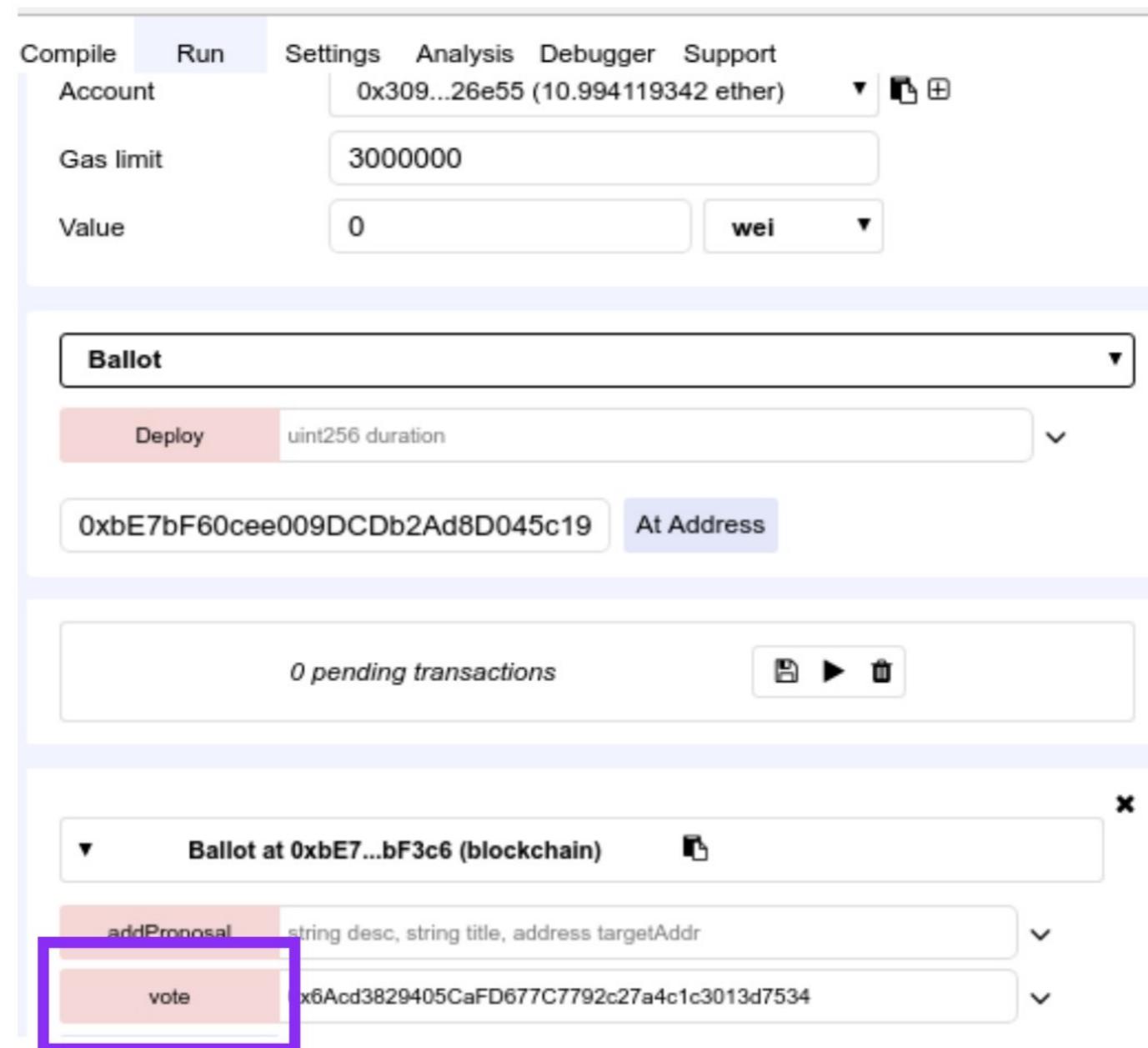
Vote for one Proposal

Paste Proposal Address you want to vote for



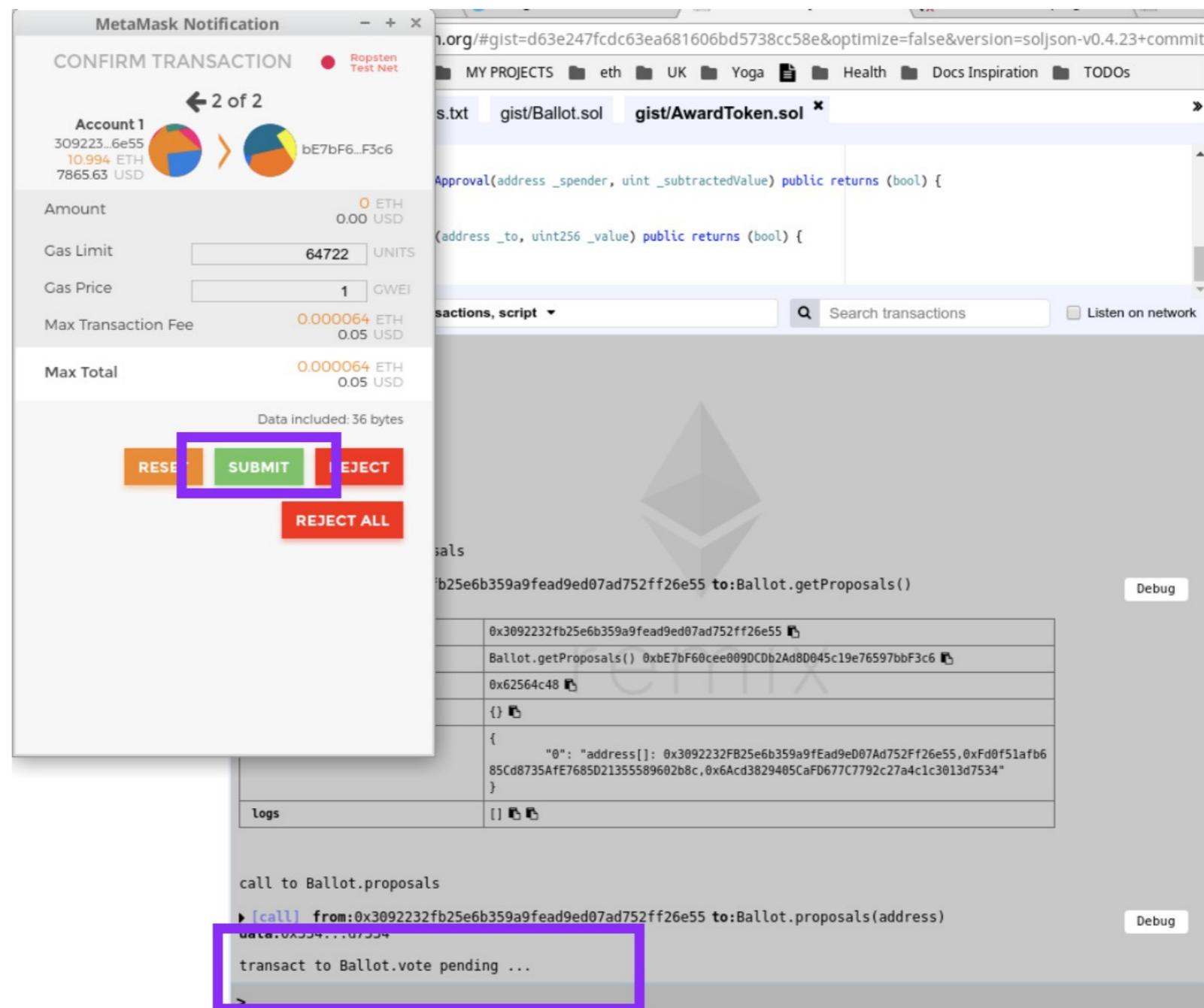
Execute vote transaction

vote button



Confirm the transaction

Submit button



Check if tx succeeded

Terminal logs in Remix

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

- File Explorer:** Shows files in the 'gist' folder: dependencies.txt, Ballot.sol, and AwardToken.sol.
- Code Editor:** Displays the Solidity code for AwardToken.sol, specifically the `decreaseApproval` and `transfer` functions.
- Terminal Logs:** The logs pane shows two transactions:
 - Call to Ballot.getProposals:** A call from address 0x3092232fb25e6b359a9fead9ed07ad752ff26e55 to Ballot.getProposals(). The data is 0x625...64c48. The response shows the decoded output: "0": "address[]: 0x3092232FB25e6b359a9fEad9eD07Ad752Ff26e55,0xFd0f51afb685Cd8735AfE7685D21355589602b8c,0x6Acd3829405CaFD677C7792c27a4c1c3013d7534".
 - Call to Ballot.proposals:** A call from address 0x3092232fb25e6b359a9fead9ed07ad752ff26e55 to Ballot.proposals(address). The data is 0x334...d7534. The response shows the transaction hash: 0x3159861, txIndex: 27, value: 0 wei.

Now let's try it out connecting a frontend

<http://bit.ly/remix-voting>

(when dependencies.js is the active file)

To access our Award Token from this frontend -
you need the address of the Award Token.

Go to ethereum/remix-workshop to access the award token I just deployed

```
contract Ballot {  
  
    uint _duration;  
    uint _startTime;  
    struct Proposal {  
        string description;  
        string title;  
        uint voteCount;  
    }  
    Proposal[] proposals;  
}
```



```
contract AwardToken is MintableToken {  
    uint quantity;  
    uint ballotPeriod = 7 hours;  
    Ballot public currBallot;  
    address[] public prevWinners;
```

The image shows a user interface for a proposal voting platform. At the top, there are four rounded rectangular boxes labeled "ROUND: 1", "ROUND: 2", "ROUND: 3", and "ROUND: 4". To the right of these boxes is a button labeled "CREATE NEW PROPOSAL". Below the boxes is a white bar containing the text "Vote for proposal and help us reward the projects that benefit the community!". Underneath this bar is a section titled "PROPOSALS". It contains a form field labeled "PROPOSAL TITLE/DESCRIPTION" with placeholder text "Choose only one". Two proposal items are listed: "PROPOSAL 1" and "PROPOSAL 3". Each proposal item has a small icon, a title, a truncated description, and a radio button for voting.

CREATE NEW PROPOSAL

ROUND: 1 ROUND: 2 ROUND: 3 ROUND: 4

Vote for proposal and help us reward the projects that benefit the community!

PROPOSALS

PROPOSAL TITLE/DESCRIPTION
Choose only one

PROPOSAL 1
Lorem ipsum dolor sit amet, consectetur adipiscing...
1 vote(s)

PROPOSAL 3
Neque porro quisquam est, qui dolorem ipsum quia ...
0 vote(s)

Let's check results

<http://bit.ly/remix-voting>

Check the state of the contract

Ballot at 0x712...0Aa64 (blockchain)

The interface shows a list of proposals with their corresponding addresses:

- addProposal: string desc, string title, address targetAddr
- finish
- vote: address proposal
- getProposals

Below the list, it shows the addresses of the proposals:

O: address[]:
0x9Ae59aF2E33480cAa48f2DC6F6CeDe7FFAb06Ff6, 0xdc7b1AaC1D13d58C
EcEEc58434C1E32Fe2A1297f

2 proposals have been added

Thank you

@ninabreznik @ryestew @yann300
@serapath @LianaHus @iurimatias

<http://bit.ly/remix-workshop-repository>