

# A Beginner's Intro to Remix

(version with all files in the GIST)

NINA BREZNIK

@ninabreznik

YANN LEVREAU

@ninabreznik

IURI MATIAS

@iurimatias

ROB STUPAY

@ryestew

ALEX PRAETORIUS

@SERAPATH

LIANA HUSIKYAN

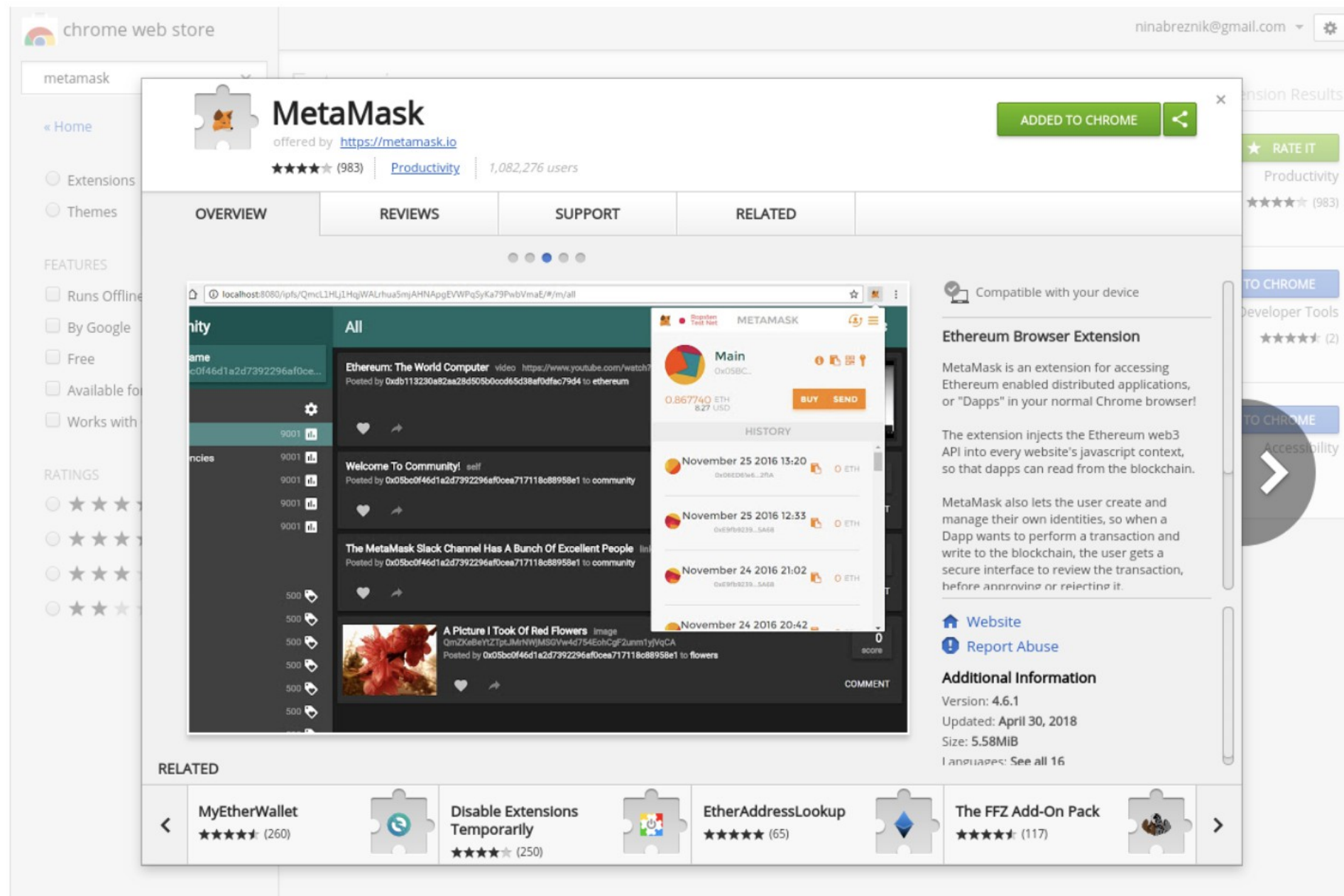
@LianaHus

# Part 1

Setup, Tour, Loading Contracts,  
& Compiling

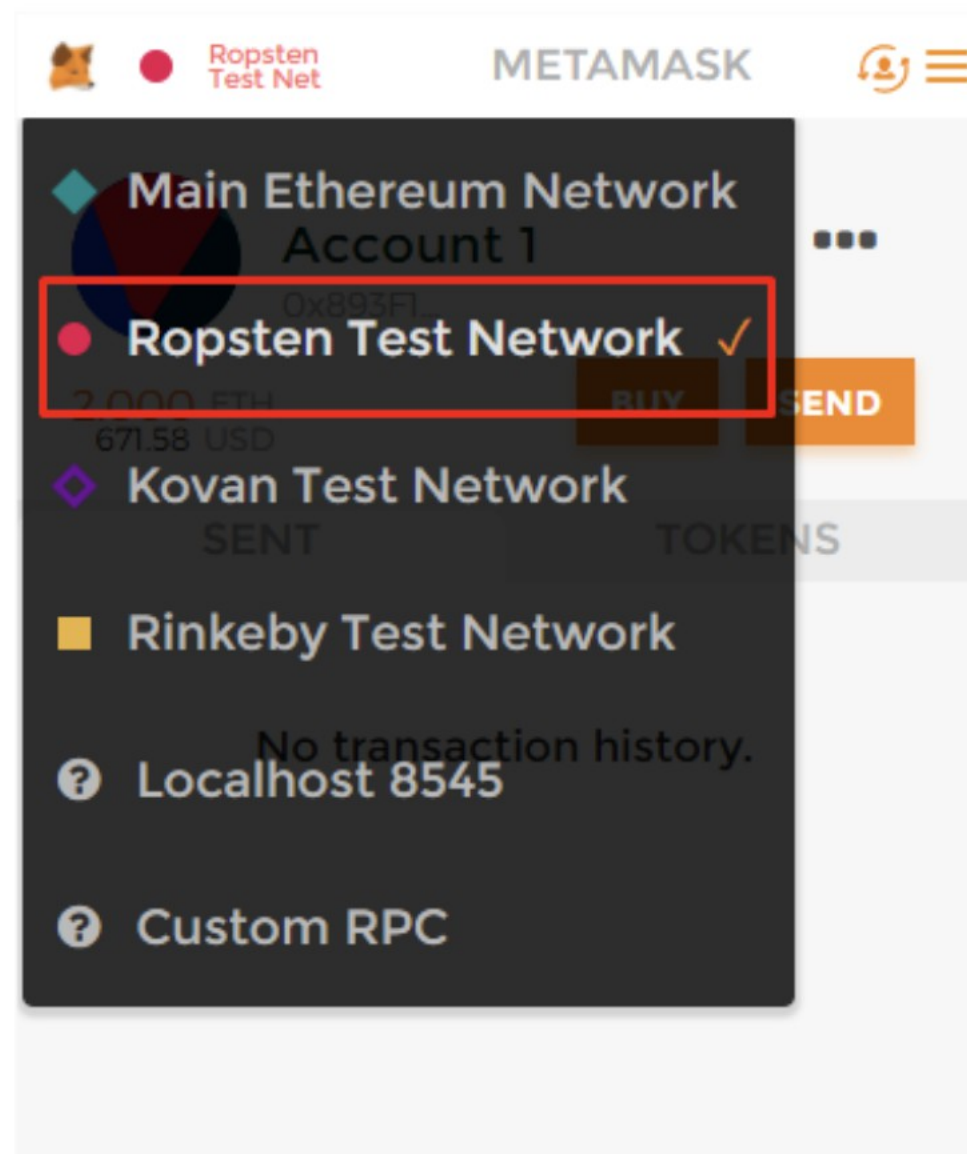
# Install Metamask

chrome.google.com/webstore



# Login to Metamask

And choose Ropsten Test Network



# Remix Tour

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

## File Explorer

## Compile Tab (active)



- ▼ browser
  - AwardToken.sol
  - Ballot2.sol
  - Ballot\_orig.sol
  - Donation.sol
  - README.md
  - multiSig2.sol
  - multisig.sol
  - multisig1.sol
  - scenario.json
  - setup.txt
- config

« + browser/Ballot\_orig.sol x

```
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
```

## Editor

» Compile Run Settings Analysis Debugger Support T

Start to compile Auto compile Hide warnings

Ballot

Details

Publish on Swarm

ABI

Bytecode

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

0 [2] only remix transactions, script Search transactions

```
- 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 p
rovider 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 gi
thub, swarm or ipfs.

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

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

remix.help(): Display this help message
```

## Terminal

>

## Console

# Run Tab

Universal DAPP

UI to the Contract

Compile

Run

Settings

Analysis



Debugger

Support




Test

Environment

Injected Web3

 Ropsten (3) 


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



# Remix Commands

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

The screenshot displays the Remix IDE interface. On the left, a file explorer shows a project named 'browser' containing files like 'AwardToken.sol', 'Ballot2.sol', 'Ballot\_orig.sol', 'Donation.sol', 'README.md', 'multiSig2.sol', 'multisig.sol', 'multisig1.sol', 'scenario.json', and 'setup.txt'. The main editor shows a Solidity contract named 'Ballot' with the following code:

```
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 }
```

On the right, the 'Compile' tab is active, showing a 'Start to compile' button, 'Auto compile' checked, and 'Hide warnings' unchecked. Below this, a dropdown menu shows 'Ballot' selected, with buttons for 'Details', 'Publish on Swarm', 'ABI', and 'Bytecode'. A warning message states: 'Static Analysis raised 2 warning(s) that requires your attention. Click here to show the warning(s)'. Below this, a specific warning is shown: 'browser/Ballot\_orig.sol:19:5: Warning: Defining function Ballot(uint8 \_numProposals) public ... ^ (Relevant source part starts here and spans ...'.

At the bottom, a terminal window 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 p
rovider 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 gi
thub, swarm or ipfs.

remix.setproviderurl(url): Change the current provider to Web3 provider and set the ur
l 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 displays the Remix IDE interface. On the left, the 'gist/AwardToken.sol' file is open, showing Solidity code for an ERC20 token contract. The right sidebar is active, showing the 'Run' tab. Within the 'Run' tab, the 'Environment' section is highlighted with a purple box. It shows 'Injected Web3' selected for the environment, with 'Ropsten (3)' as the network. Below this, the 'Account' is set to '0x667...d091d (1.453587112999635446)', the 'Gas limit' is '3000000', and the 'Value' is '0' in 'wei'. A dropdown menu shows 'AwardToken' as the selected contract. At the bottom, there is a 'Deploy' button and a section for 'Load contract from Address' with an 'At Address' button.

```
1 import "github.com/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        return prevWinners;
19    }
20
21    // either a name change or it works fine without it
22    // function approve(address spender, uint256 value) public returns (bool);
23    function startRound() onlyOwner canMint public returns (bool) {
24        // if this is the first minting then we should let this go immediately
25    }
```

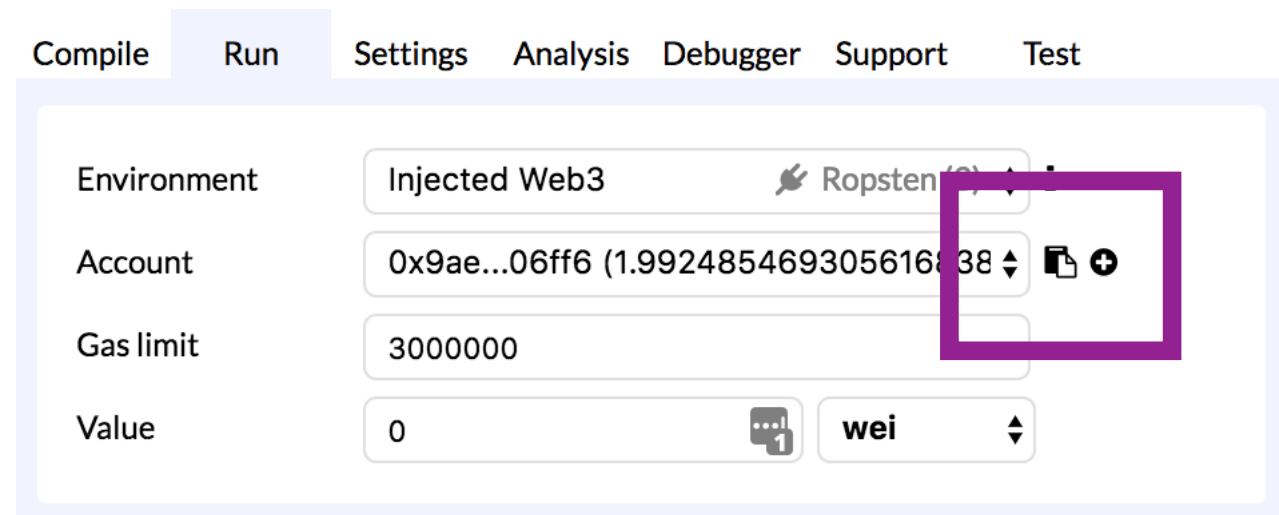


# Get some TEST ether

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

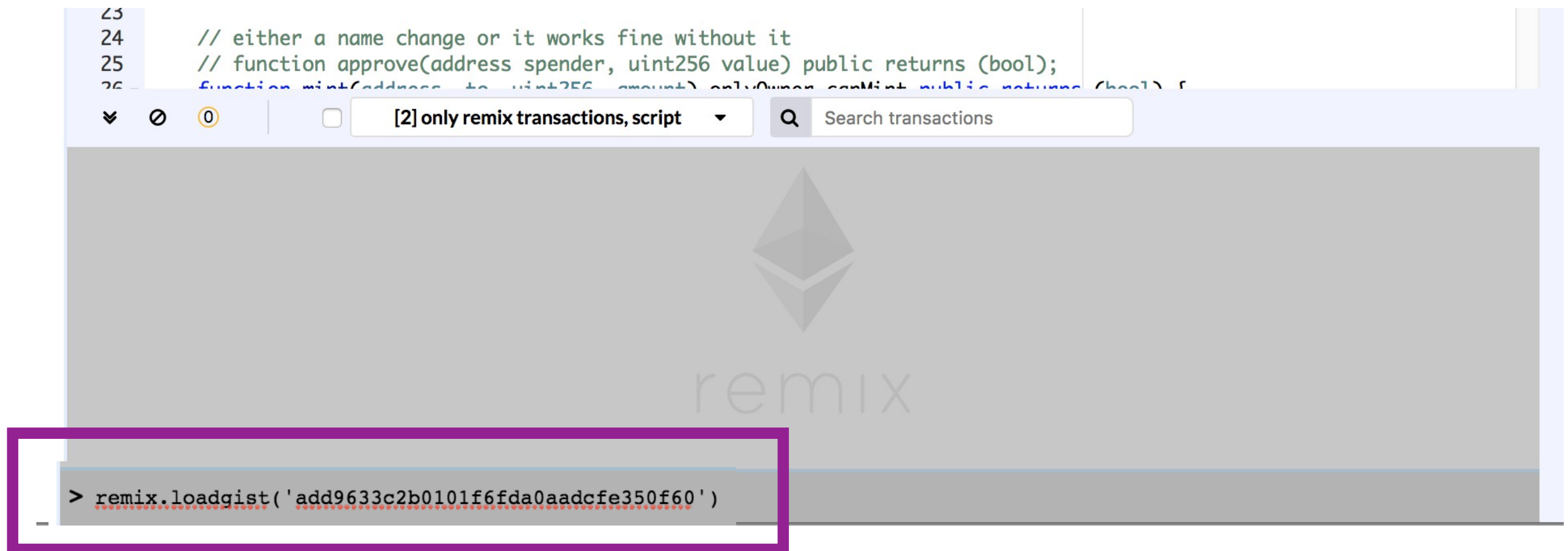
But FIRST:

Copy your address - here or in Metamask



# Load files to Remix

```
remix.loadgist('add9633c2b0101f6fda0aadcf350f60')
```



here in the console

# Open file

gist/AwardToken

The screenshot shows the Remix IDE interface. On the left, the file explorer is open, showing a list of files under the 'gist' folder. The file 'AwardToken.sol' is highlighted with a purple border. The main editor area displays the Solidity code for 'AwardToken.sol'. The code is as follows:

```
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        return prevWinners;
19    }
20
21    // either a name change or it works fine without it
22    // function approve(address spender, uint256 value) public returns (bool);
23    function startRound() onlyMinter public returns (bool) {
24        // if this is the first minting then we should let this go immediately
25        if (address(currBallot) == 0x0) {
26            currBallot = new Ballot(ballotPeriod);
27            newBallot(currBallot);
28        }
29    }
30}
```

The bottom status bar shows '[2] only remix transactions, script' and a search bar with the text 'Search transactions'.

# Compile the contract

Compile tab: Start to compile button

( when dependencies.js is the active file )

The screenshot displays the Remix IDE interface. On the left, the 'gist/dependencies.js' file is active, showing Solidity code for the 'AwardToken' contract. The code includes imports for 'MintableToken' and 'Ballot', and defines the 'AwardToken' contract with its constructor, 'getPreviousWinners' function, and 'startRound' function. The right sidebar shows the 'Compile' tab selected, with the 'Start to compile' button highlighted by a purple box. Below the button, the 'AwardToken' contract is listed, and the 'Static Analysis' section shows 38 warnings. The bottom status bar indicates '[2] only remix transactions, script'.

```
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        return prevWinners;
19    }
20
21    // either a name change or it works fine without it
22    // function approve(address spender, uint256 value) public returns (bool);
23    function startRound() onlyOwner canMint public returns (bool) {
24        // if this is the first minting then we should let this go immediately
25        if (address(currBallot) == 0x0) {
```

Compile Run Settings Analysis Debugger Support Test

Start to compile Auto compile Hide warnings

AwardToken

Details Publish on Swarm ABI Bytecode

Static Analysis raised 38 warning(s) that requires your attention. Click here to show the warning(s).

gist/Ballot.sol:26:5: Warning: Defining constructors as functions

```
function Ballot(uint duration) public {
^ (Relevant source part starts here and spans across multiple lines)
```

gist/AwardToken.sol:13:5: Warning: Defining constructors as functions

```
function AwardToken () {
^ (Relevant source part starts here and spans across multiple lines)
```

gist/AwardToken.sol:1:1: Warning: Source file does not specify the Solidity version

[2] only remix transactions, script Search transactions

# See compiled contracts

AwardToken + all dependencies



# Imported Contracts

