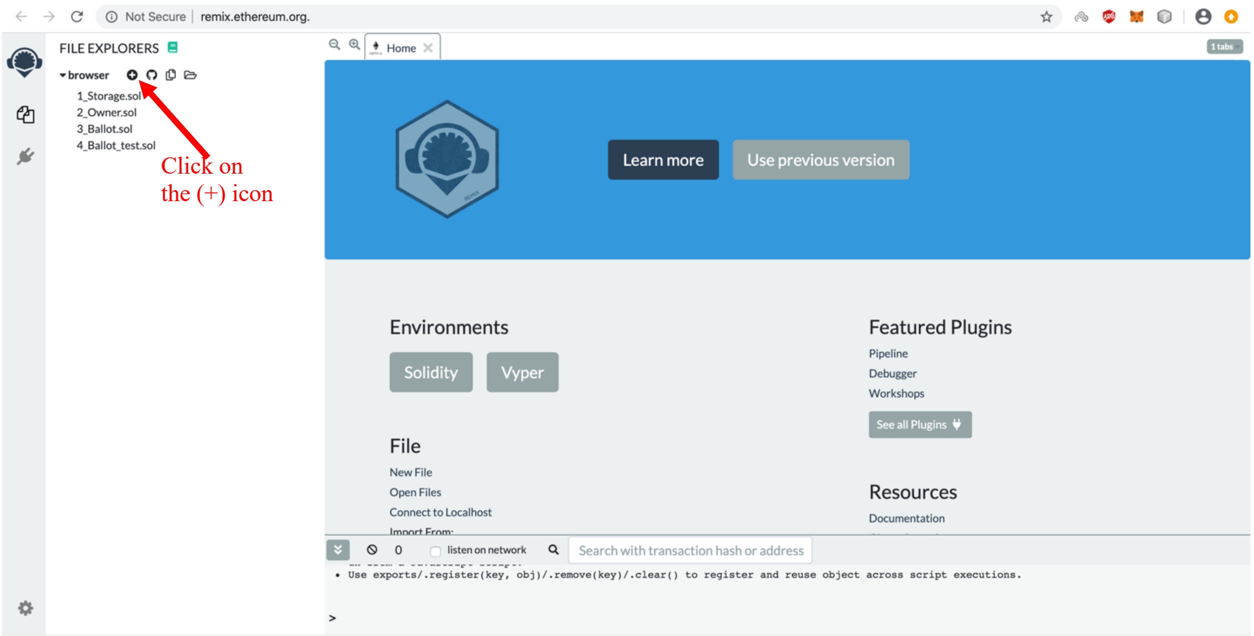
Ethereum Smart Contracts (Remix IDE)

Open URL: http://remix.ethereum.org./

Step 1:

Click on the (+) icon on the upper left portion of the editor to create a new smart contract. Call it "Coursetro.sol".



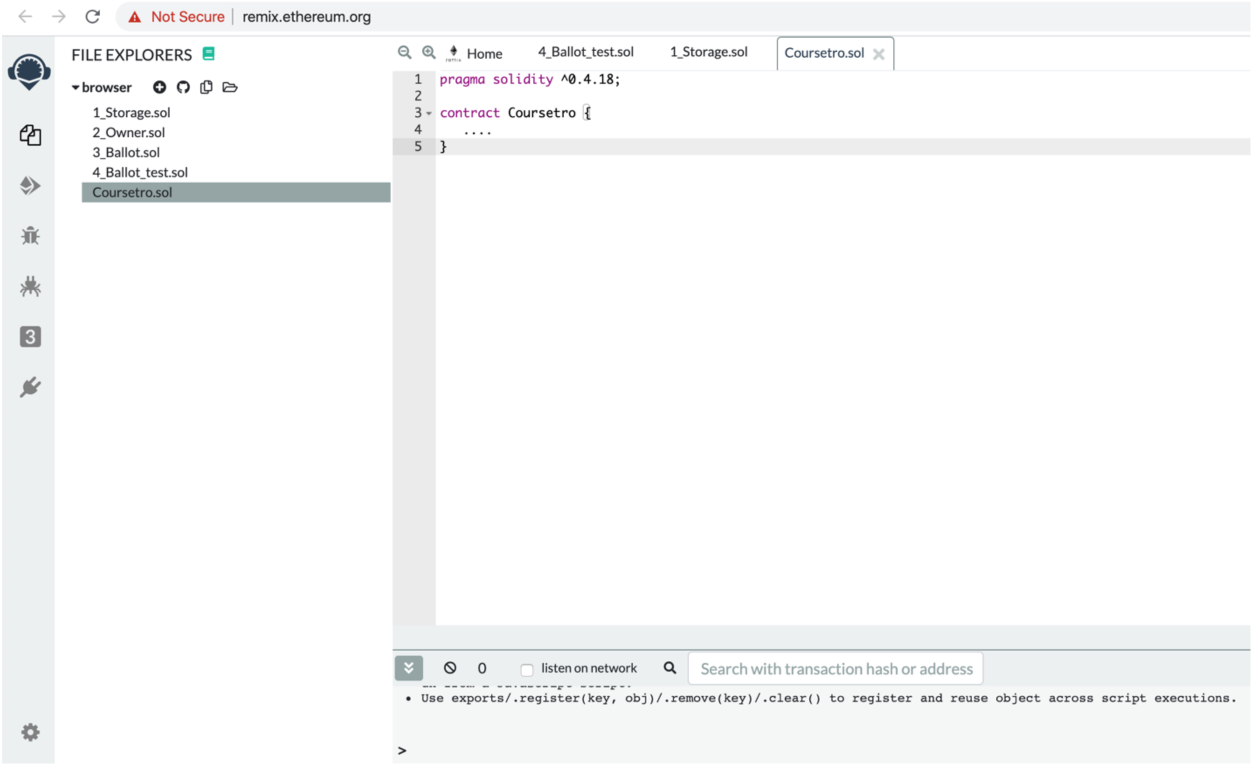
Click on the (+) icon

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
>
```

Step 2:

Start off by writing (or pasting) the following code:

```
// This line defines the version of solidity you're going to use.
pragma solidity ^0.4.18;

contract Coursetro {
   ....
}
```

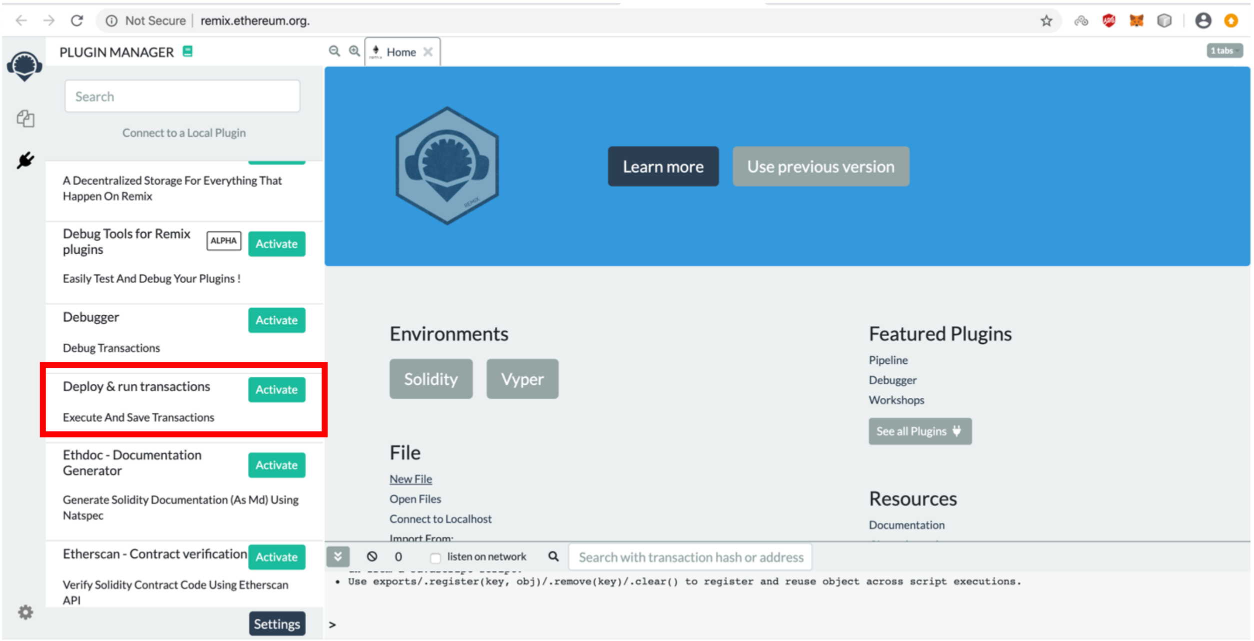


```
pragma solidity ^0.4.18;

contract Coursetro {
    ....
}
```

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.

>
```

Step 3:

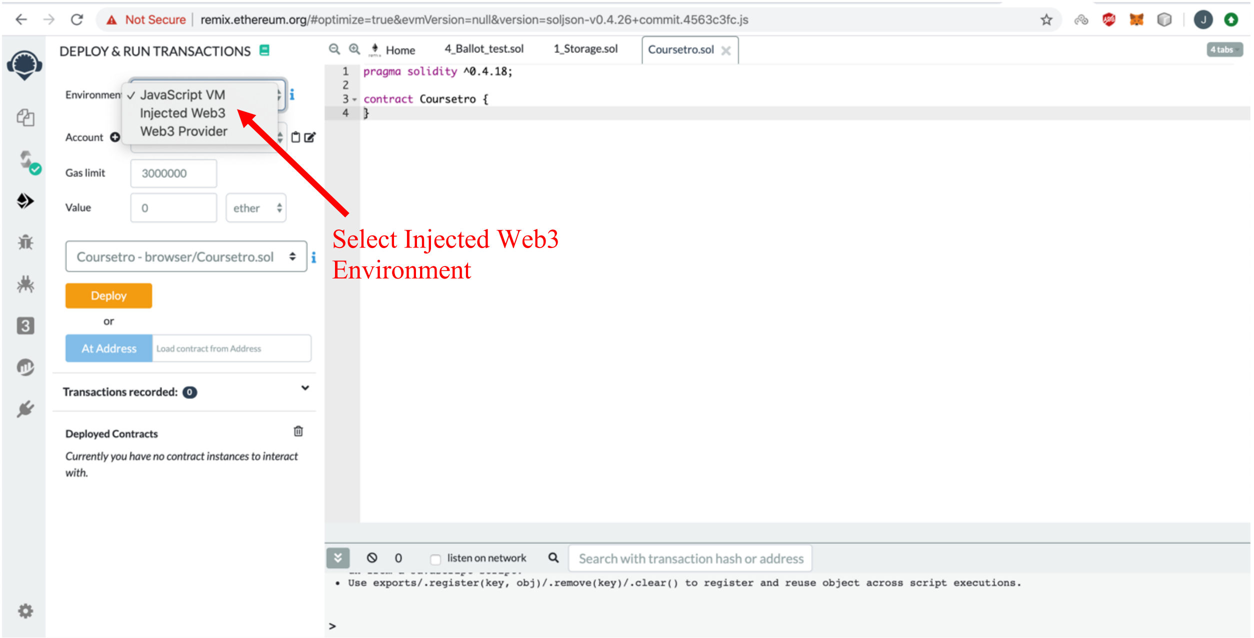
Click plugin manager **Activate Deploy & Run transactions**


Not Secure | remix.ethereum.org.
PLUGIN MANAGER
Search
Connect to a Local Plugin
A Decentralized Storage For Everything That Happen On Remix
Debug Tools for Remix plugins
ALPHA
Activate
Easily Test And Debug Your Plugins !
Debugger
Activate
Debug Transactions
Deploy & run transactions
Activate
Execute And Save Transactions
Ethdoc - Documentation Generator
Activate
Generate Solidity Documentation (As Md) Using Natspec
Etherscan - Contract verification
Activate
Verify Solidity Contract Code Using Etherscan API
Settings
Home
1 tabs
Learn more
Use previous version
Environments
Solidity
Vyper
File
New File
Open Files
Connect to Localhost
Import From:
Featured Plugins
Pipeline
Debugger
Workshops
See all Plugins
Resources
Documentation
0
listen on network
Search with transaction hash or address

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.

>
```

On the left side of the browser select **Deploy & Run transactions** you will see a Deploy button. Click on this.

You will notice that underneath the Deploy button shows a new section with the name of the contract and "at 0x..." (memory):



Not Secure | remix.ethereum.org/#optimize=true&evmVersion=null&version=soljson-v0.4.26+commit.4563c3fc.js
DEPLOY & RUN TRANSACTIONS
Environment
✓ JavaScript VM
Injected Web3
Web3 Provider
Account
Gas limit
3000000
Value
0
ether
Coursetro - browser/Coursetro.sol
Deploy
or
At Address
Load contract from Address
Transactions recorded: 0
Deployed Contracts
Home
4_Ballot_test.sol
1_Storage.sol
Coursetro.sol
4 tabs
listen on network
Search with transaction hash or address

Currently you have no contract instances to interact with.

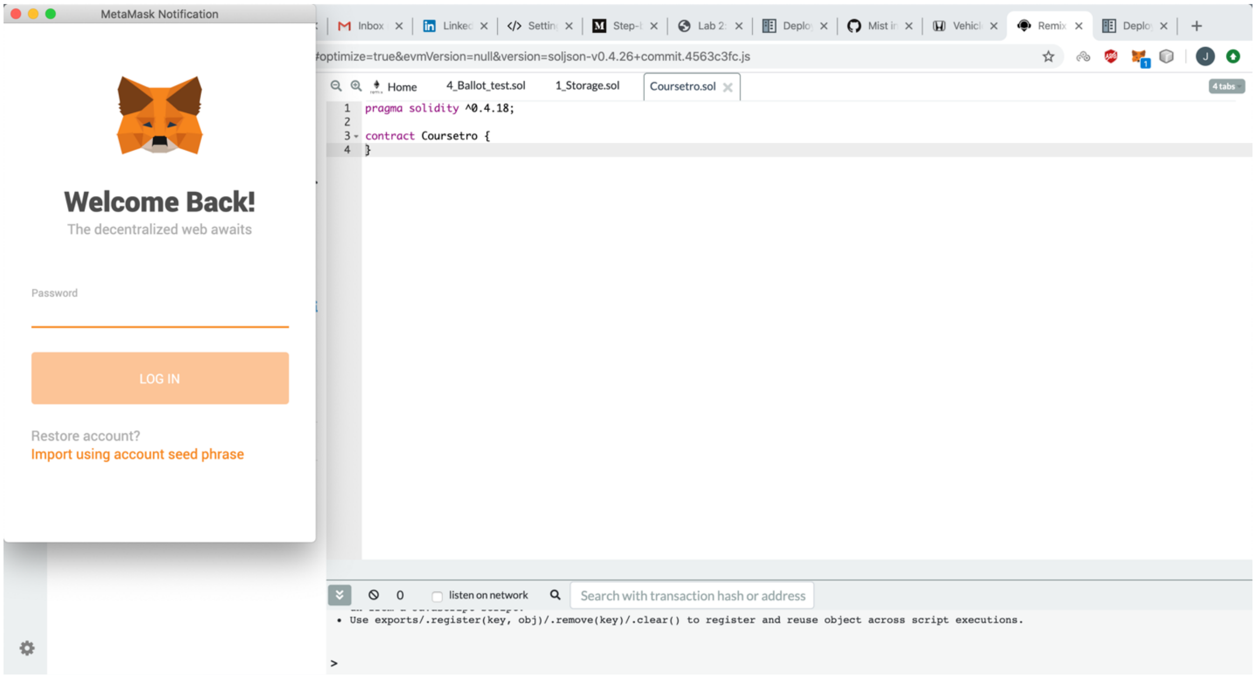
```
pragma solidity ^0.4.18;

contract Coursetro {
}
```

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
>
```

Select Injected Web3 Environment

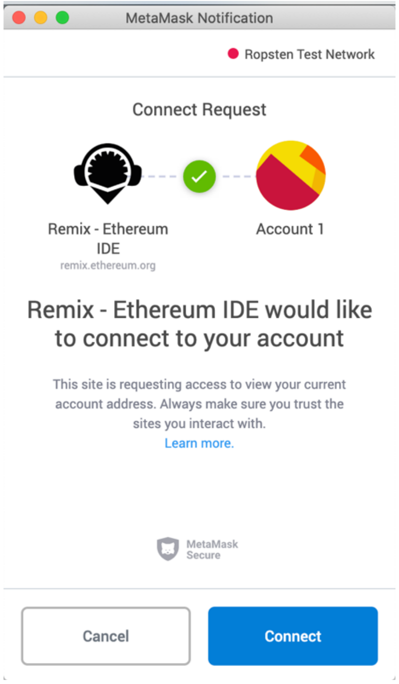
Step 4:

Create **MetaMask** Account and connect that account while deploying smart contract:



```
pragma solidity ^0.4.18;

contract Coursetro {
}
```

```
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
>
```



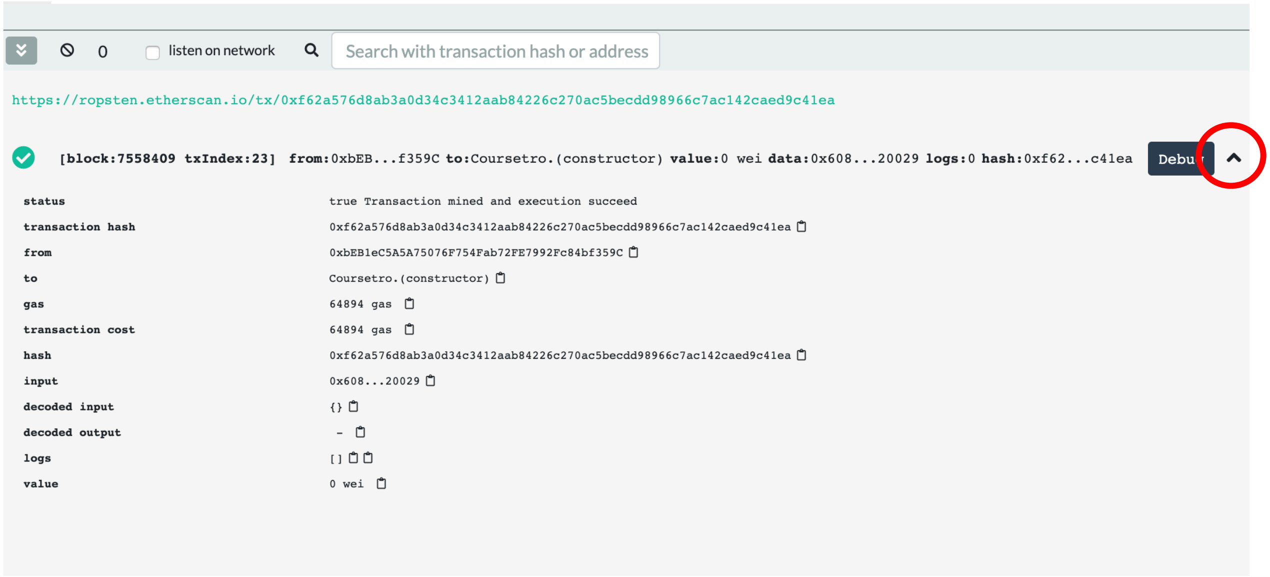
Connect Request

Remix - Ethereum IDE would like to connect to your account

This site is requesting access to view your current account address. Always make sure you trust the sites you interact with.
Learn more.

Step 5:

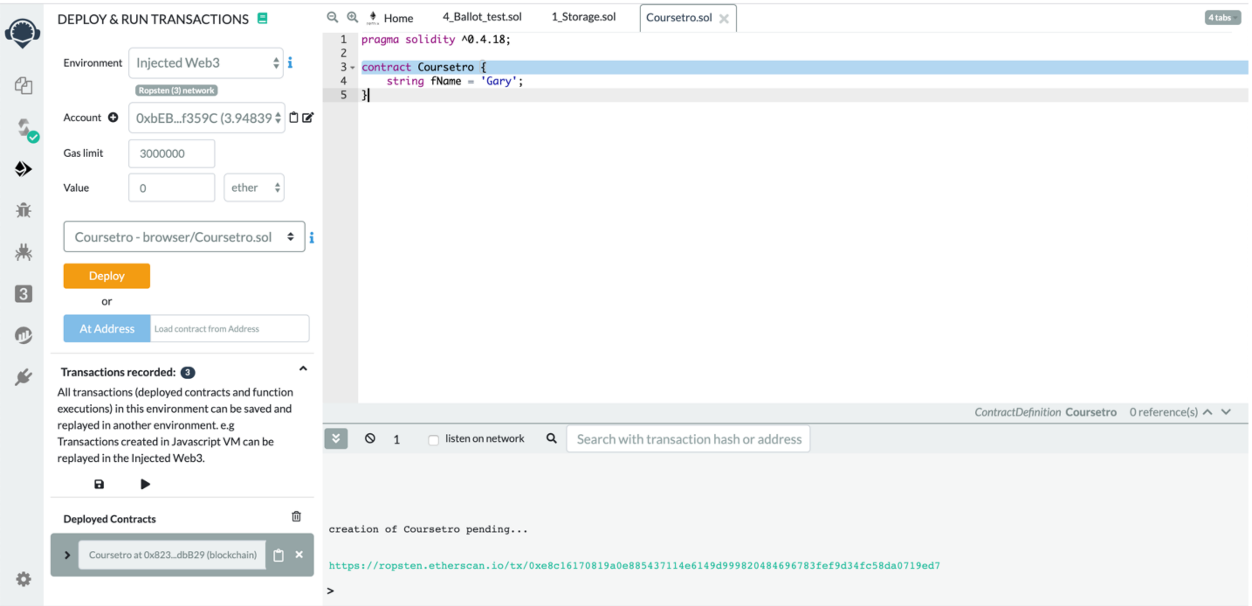
To see this full address along with other information, in the debugger if you click on the **down arrow** button, it will provide you with more information:



```
https://ropsten.etherscan.io/tx/0xf62a576d8ab3a0d34c3412aab84226c270ac5becdd98966c7ac142caed9c41ea

[block:7558409 txIndex:23] from:0xbEB...f359C to:Coursetro.(constructor) value:0 wei data:0x608...20029 logs:0 hash:0xf62...c41ea

 status                      true Transaction mined and execution succeed
 transaction hash            0xf62a576d8ab3a0d34c3412aab84226c270ac5becdd98966c7ac142caed9c41ea
 from                        0xbEB1eC5A5A75076F754Fab72FE7992Fc84bf359C
 to                          Coursetro.(constructor)
 gas                         64894 gas
 transaction cost            64894 gas
 hash                        0xf62a576d8ab3a0d34c3412aab84226c270ac5becdd98966c7ac142caed9c41ea
 input                       0x608...20029
 decoded input               {}
 decoded output               -
 logs                        []
 value                       0 wei
```

Step 6:

Define a string variable in our contract



DEPLOY & RUN TRANSACTIONS
Home
4_Ballot_test.sol
1_Storage.sol
Coursetro.sol
4 tabs
Deploy
or
At Address
Load contract from Address
Search with transaction hash or address
ContractDefinition Coursetro 0 reference(s)

Environment: Injected Web3 (Ropsten (3) network)

Account: 0xbEB...f359C (3.94839

Gas limit: 3000000

Value: 0 ether

Coursetro - browser/Coursetro.sol

Transactions recorded: 3

All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

Coursetro at 0x823...dbB29 (blockchain)

```
pragma solidity ^0.4.18;

contract Coursetro {
    string fName = 'Gary';
}
```

1 ☐ listen on network

```
creation of Coursetro pending...

https://ropsten.etherscan.io/tx/0xe8c16170819a0e885437114e6149d9998204846967 83fef9d34fc58da0719ed7
>
```

Step 7:

Paste Following code in the smart contract:

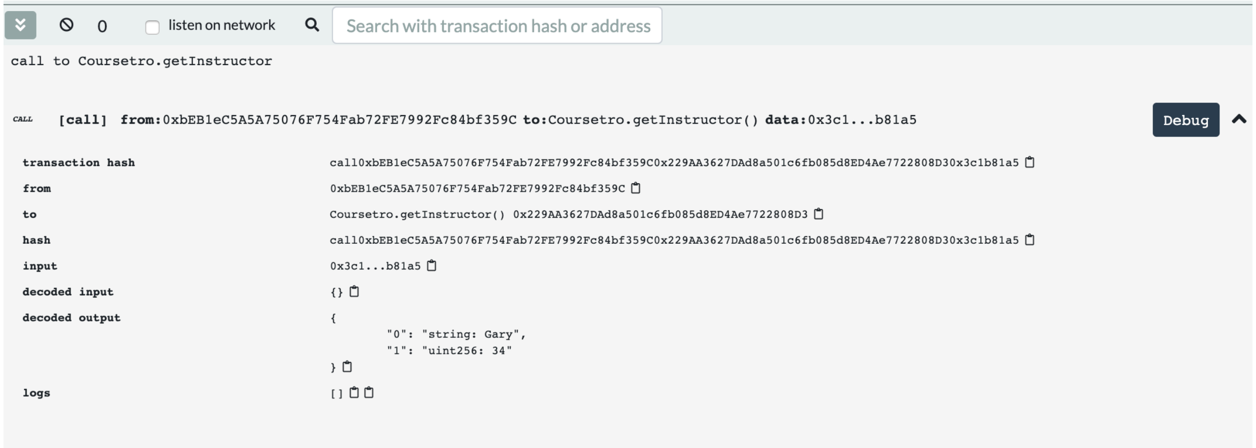
```
pragma solidity ^0.4.18;

contract Coursetro {

  string fName;
  uint age;

  function Coursetro() public {
    fName = 'Gary';
    age = 34;
  }

  // setInstructor accepts 2 parameters, _fName and _age.
  // Once called, we set our string fName to the returned
  // _fName, and same with age.
  function setInstructor(string _fName, uint _age) public {
     fName = _fName;
     age = _age;
  }
  // The getInstructor() function is defined as being
  // constant, and it returns a string and a uint.  This is where
  // we return the fName and age variable once it's called.
  function getInstructor() public constant returns (string, uint) {
      return (fName, age);
  }
}
```

Step 8:

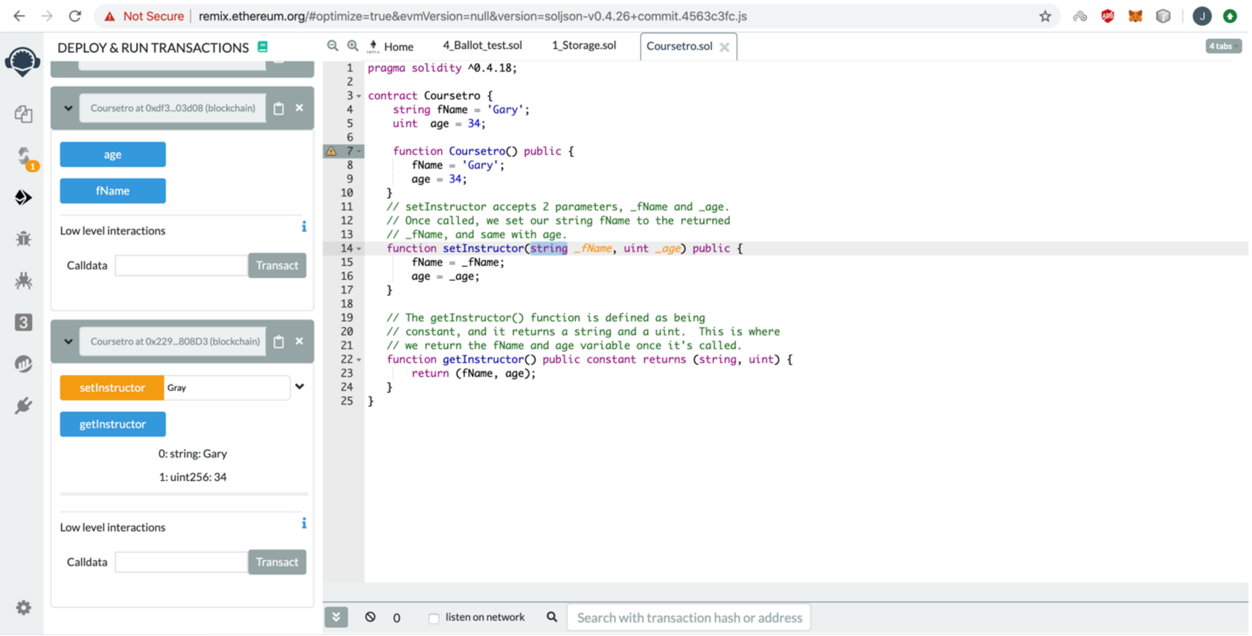
Click Deploy and

- Under the red set Instructor button, type in: "Gary", 34 and click the button.
- Next, click on the getInstructor() button and you will notice it now returns the inputed value!

This is how you set variables from user input in a smart contract.

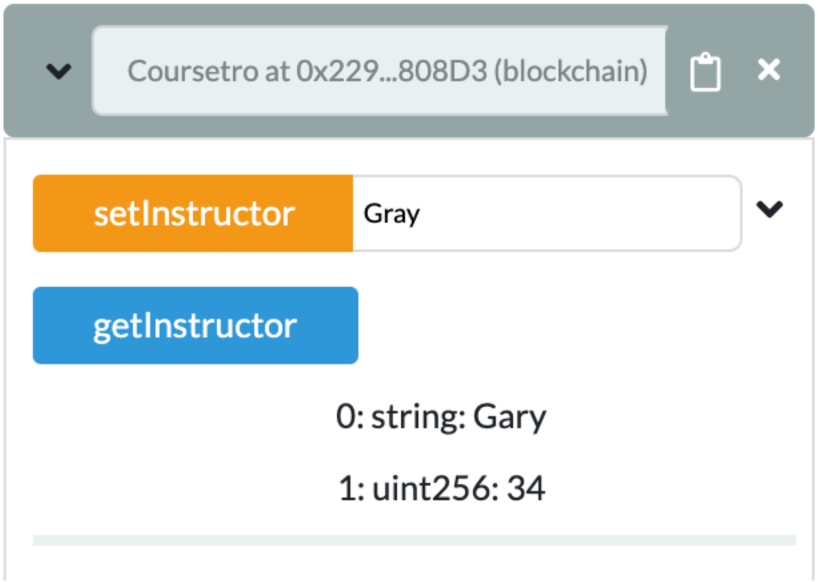


```
call to Coursetro.getInstructor

CALL   [call]  from:0xbEB1eC5A5A75076F754Fab72FE7992Fc84bf359C to:Coursetro.getInstructor() data:0x3c1...b81a5

transaction hash         call0xbEB1eC5A5A75076F754Fab72FE7992Fc84bf359C0x229AA3627DAd8a501c6fb085d8ED4Ae7722808D30x3c1b81a5
from                     0xbEB1eC5A5A75076F754Fab72FE7992Fc84bf359C
to                       Coursetro.getInstructor() 0x229AA3627DAd8a501c6fb085d8ED4Ae7722808D3
hash                     call0xbEB1eC5A5A75076F754Fab72FE7992Fc84bf359C0x229AA3627DAd8a501c6fb085d8ED4Ae7722808D30x3c1b81a5
input                    0x3c1...b81a5
decoded input            {}
decoded output           {
                                 "0": "string: Gary",
                                 "1": "uint256: 34"
                         }
logs                     []
```


Not Secure
remix.ethereum.org/#optimize=true&evmVersion=null&version=soljson-v0.4.26+commit.4563c3fc.js
DEPLOY & RUN TRANSACTIONS
Home
4_Ballot_test.sol
1_Storage.sol
Coursetro.sol
4 tabs
Coursetro at 0xdf3...03d08 (blockchain)
age
fName
Low level interactions
Calldata
Transact
Coursetro at 0x229...808D3 (blockchain)
setInstructor
Gray
getInstructor
Low level interactions
Calldata
Transact
0
listen on network
Search with transaction hash or address

```
pragma solidity ^0.4.18;

contract Coursetro {
    string fName = 'Gary';
    uint  age = 34;

    function Coursetro() public {
        fName = 'Gary';
        age = 34;
   }
   // setInstructor accepts 2 parameters, _fName and _age.
   // Once called, we set our string fName to the returned
   // _fName, and same with age.
   function setInstructor(string _fName, uint _age) public {
       fName = _fName;
       age = _age;
   }

   // The getInstructor() function is defined as being
   // constant, and it returns a string and a uint.  This is where
   // we return the fName and age variable once it's called.
   function getInstructor() public constant returns (string, uint) {
       return (fName, age);
   }
}
```

```
0: string: Gary
1: uint256: 34
```



```
0: string: Gary
1: uint256: 34
```