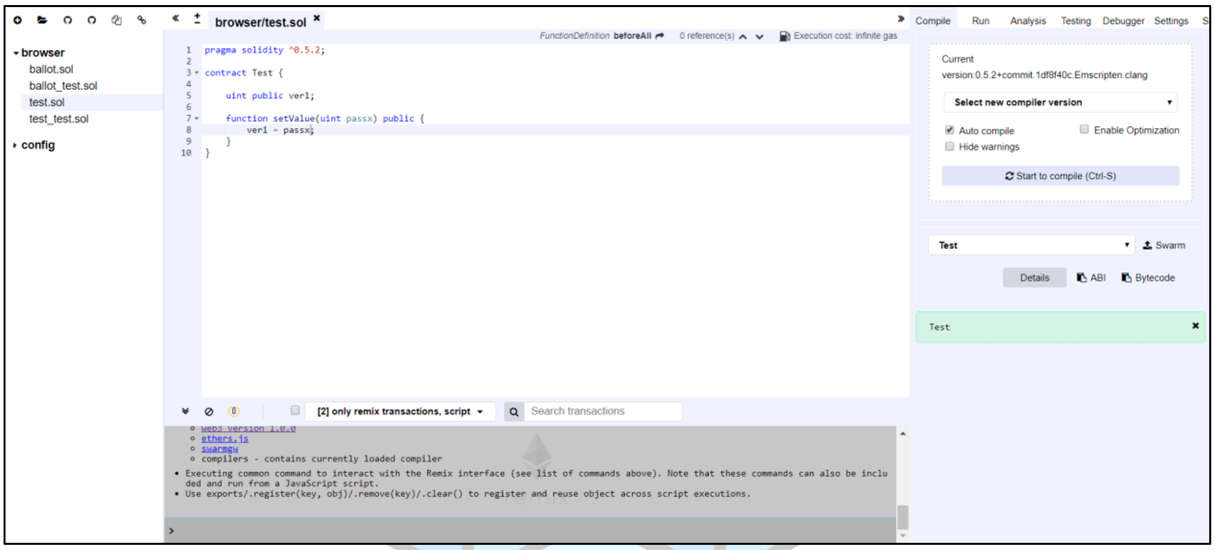
Remix IDE

Remix is a powerful, open source tool that helps you write Solidity contracts straight from the browser. Written in JavaScript, Remix supports both usage in the browser and locally.

Remix also supports testing, debugging and deploying of smart contracts and much more.



```
pragma solidity ^0.5.2;

contract Test {

    uint public ver1;

    function setValue(uint passx) public {
        ver1 = passx;
    }
}
```

Current
version:0.5.2+commit.1df8f40c.Emscripten.clang

✓ Auto compile
Enable Optimization
Hide warnings

```
  ○ web3 version 1.0.0
  ○ ethers.js
  ○ swarmgw
  ○ compilers - contains currently loaded compiler
• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be inclu
  ded and run from a JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
>
```

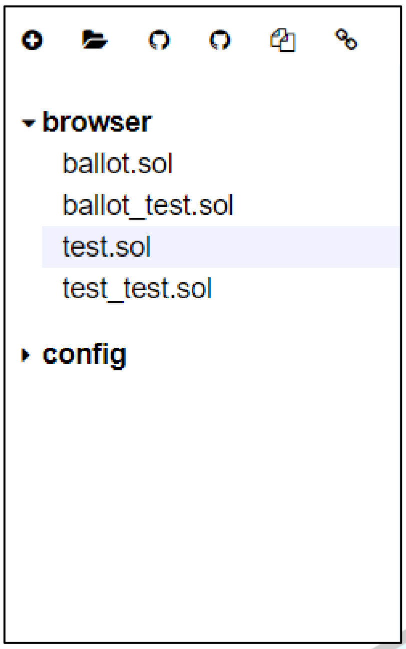
The remix IDE is a web based integrated development environment for developing, testing, and deploying smart contracts written in solidity language.

The remix ide has four sections:

1. Brower Explorer section
2. Editor
3. Log
4. Development Section

Remix IDE

1. Browser explore section



The left side of the remix ide contains the file manager section, which provides the user with the following features:

Create a new fie in the browser storage (which will be temporarily stored until the cookie data of the remix site is cleared)

Add a local fie to the browser storage

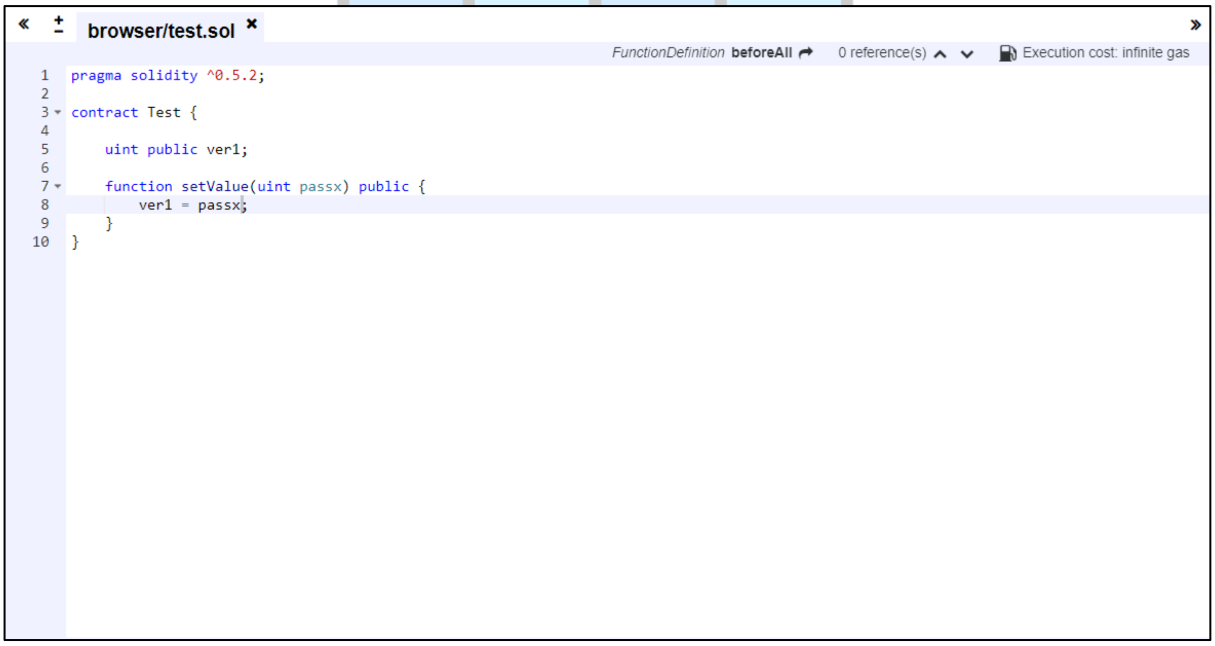
Push all the files in the browser explore to the connected Gist

Refresh the gist explorer to see changes.

Copy all the files from browser explorer of the remix ide to another instance of remix ide.

Link the remix ide to a local instance running on your system.

2. Editor



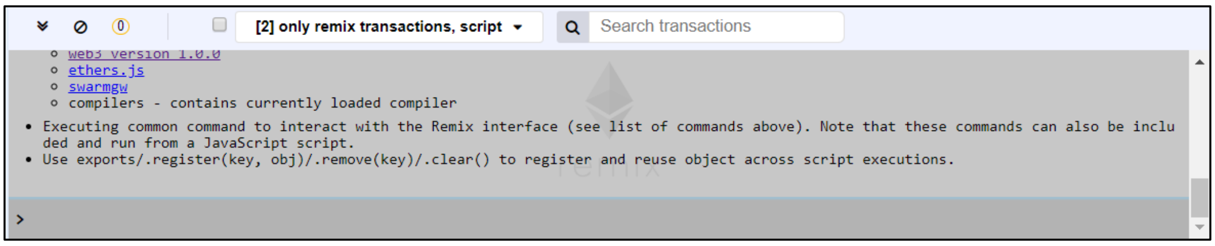
```
pragma solidity ^0.5.2;

contract Test {

    uint public ver1;

    function setValue(uint passx) public {
        ver1 = passx;
    }
}
```

The editor provides the developer with the following features:

- It displays al the opened files as tabs.
- Provides intellisense
- Displays syntax error
- Remix saves the current file continuously (5s after the last changes)

- +/- on the top left corner enable you to increase/decrease the font size of the editor

3. Log



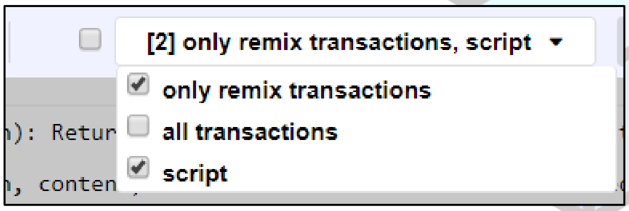
```
    o web3 version 1.0.0
    o ethers.js
    o swarmgw
    o compilers - contains currently loaded compiler
  • Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be inclu
    ded and run from a JavaScript script.
  • Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
>
```

This session of the remix ide displayed the log of the transactions, and some runtime virtual machine errors.

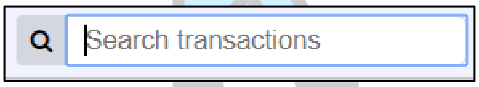
minimize the log windows.

clear the log.

show the number of pending transactions.


[2] only remix transactions, script
✓ only remix transactions
all transactions
✓ script

Provides the option to see only the transactions initiated from remix or all the mined transactions in the given environment.



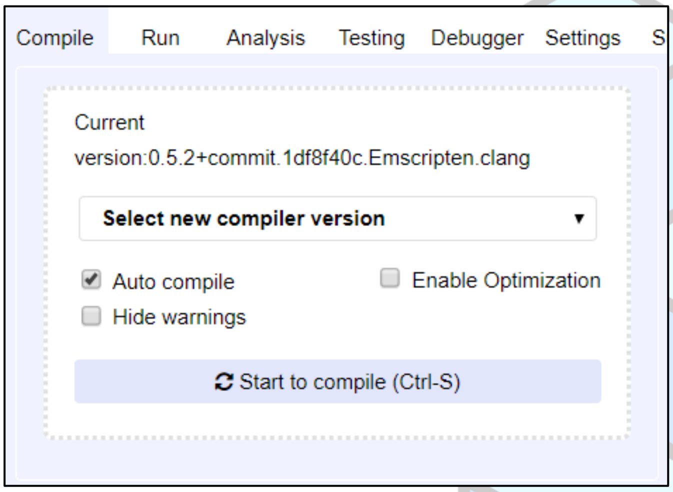
Search the logs for a specific transaction.

4. Development Section

The development section has 5 tabs which are:

4.1 Compile
4.2 Run
4.3 Analysis
4.4 Testing
4.5 Debugger

4.1 Compile



Current version:0.5.2+commit.1df8f40c.Emscripten.clang

Select new compiler version

✓ Auto compile

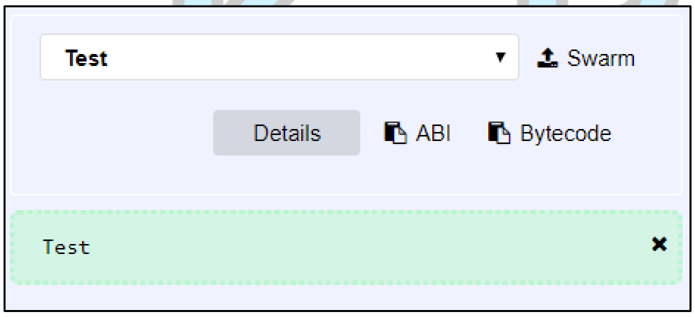
Enable Optimization

Hide warnings

The compiler tab shows the current compiler version loaded and provides you with options like:

- Auto compile
- Enable Optimization
- Hide warnings

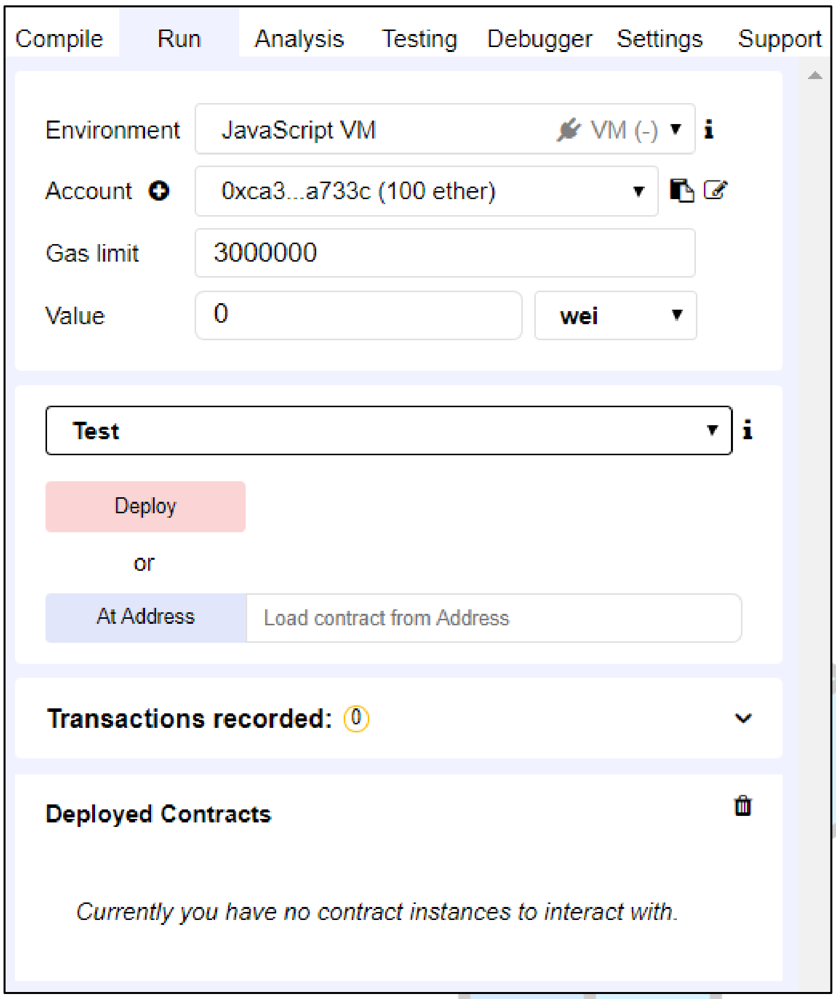
In the bottom of errors in the contract are shown.



```
Test
```

Once the contract is compiled successful, we can see the list of contracts compile in the drop down (by contracts I meant contracts in the same solidity file) and you can copy the **ABI** and **Bytecode** by clicking on the corresponding buttons.

4.2 Run



Environment: JavaScript VM — VM (-)
Account: 0xca3...a733c (100 ether)
Gas limit: 3000000
Value: 0 wei

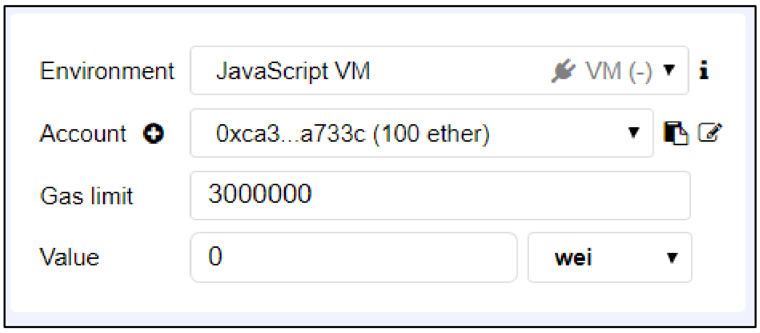
Test

Transactions recorded: 0

Deployed Contracts

Currently you have no contract instances to interact with.

The run tab allows the developer to deploy the smart contract to the targeted environment.



Environment: JavaScript VM — VM (-)
Account: 0xca3...a733c (100 ether)
Gas limit: 3000000
Value: 0 wei

The first section in the run tab consist of:

Environment: Allow the developer to connect to different blockchain environments.

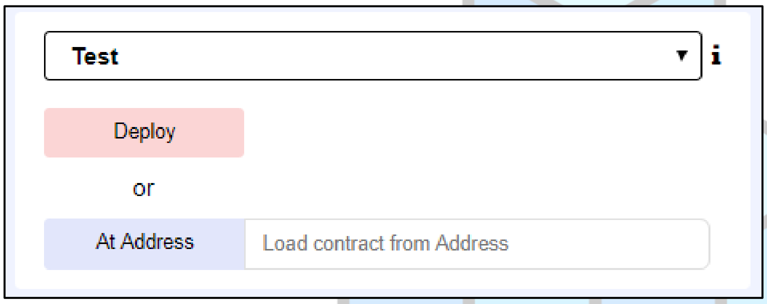
- JavaScript VM: Execution environment does not connect to any node, everything is local and in memory only. Provided by remix its self, a simulation of a blockchain node.

- Injected Web3: Execution environment has been provided by Metamask or similar provider.
- Web3 Provider: Execution environment connects to node at localhost (or via IPC if available), transactions will be sent to the network and can cause loss of money or worse. If this page is served via https and you access your node via http, it might not work. In this case, try cloning the repository and serving it via http.

Account: Shows the list of accounts available in the connected environment.

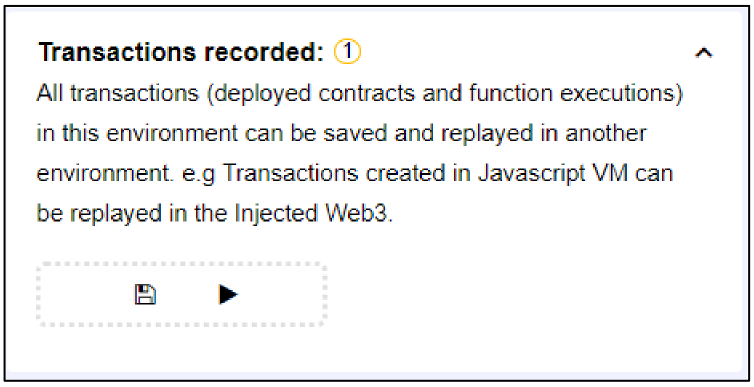
Gas Limit: Transaction Gas Limit.

Value: For transferring ether.



The dropdown list will show all the compiled list of smart contracts without, choice one and click the **Deploy** button, which will deploy the contract to the connected environment.

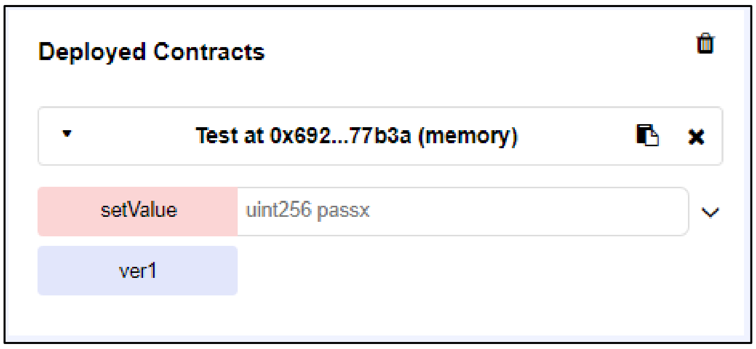
By passing the address of the contract and clicking the **At Address** you can load the contract that is already deployed in the connected environment.



Transactions recorded: 1

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.

The **Transaction recorded** section allows you to save all the transactions that you have done in the current environment and replay it in another environment.



Deployed contracts are displayed in the **Deployed Contracts** section from which developer can get the contract address and try out all the functions within the deployed contract.

Notice that **setValue** and **ver1** are shown in different colors. The red color indicates the function cost gas and the blue color indicates the function does not cause gas.