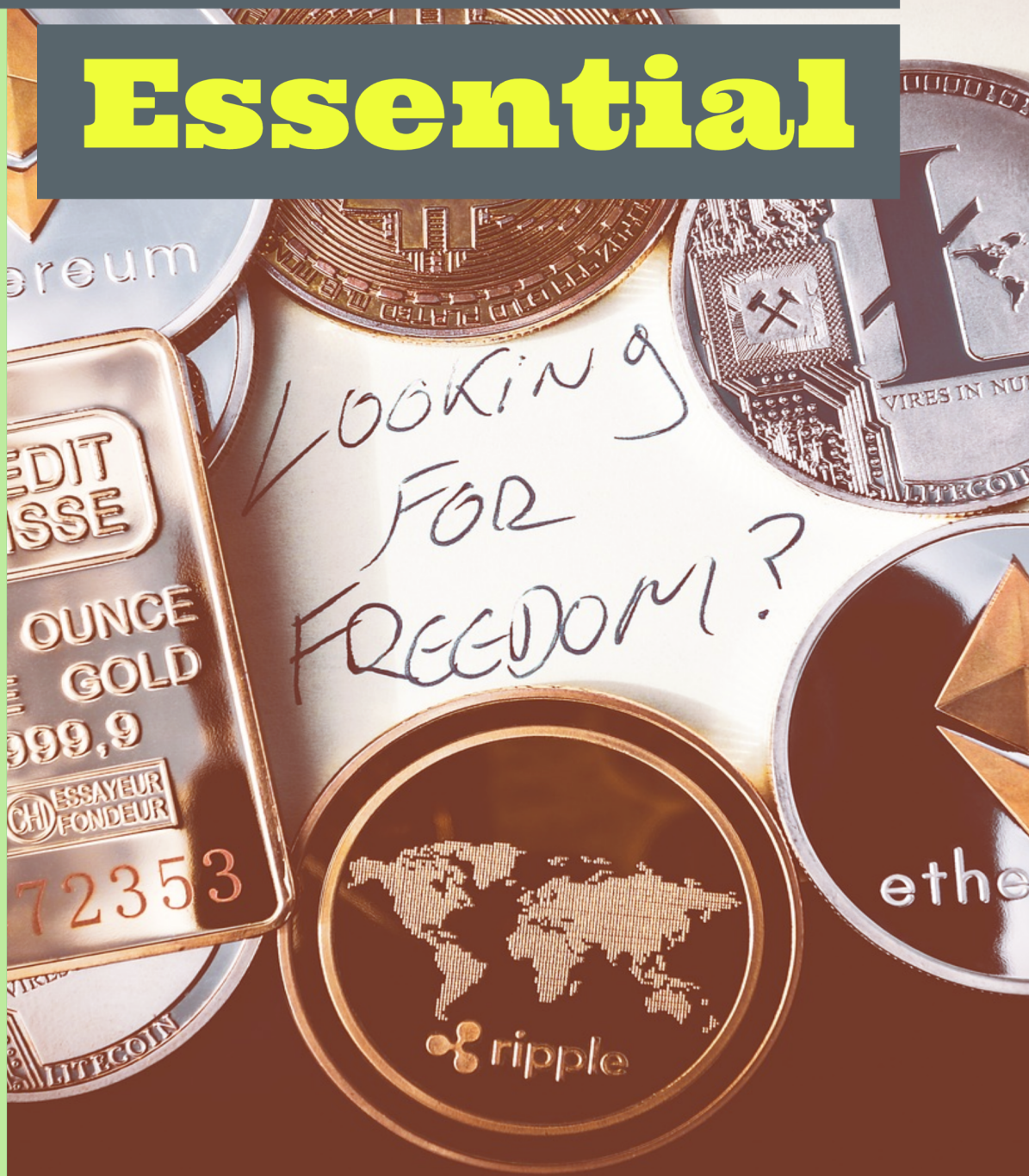


SOLIDITY

Programming

Essential

Author:
Anish Nath



Solidity Programming Essentials

Anish Nath

This book is for sale at <http://leanpub.com/solidity>

This version was published on 2021-07-05



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2021 Anish Nath

Tweet This Book!

Please help Anish Nath by spreading the word about this book on [Twitter](#)!

The suggested hashtag for this book is [#solidity blockchain ethereum](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

[#solidity blockchain ethereum](#)

Contents

Remix Environment	1
HelloWorld with Remix	1

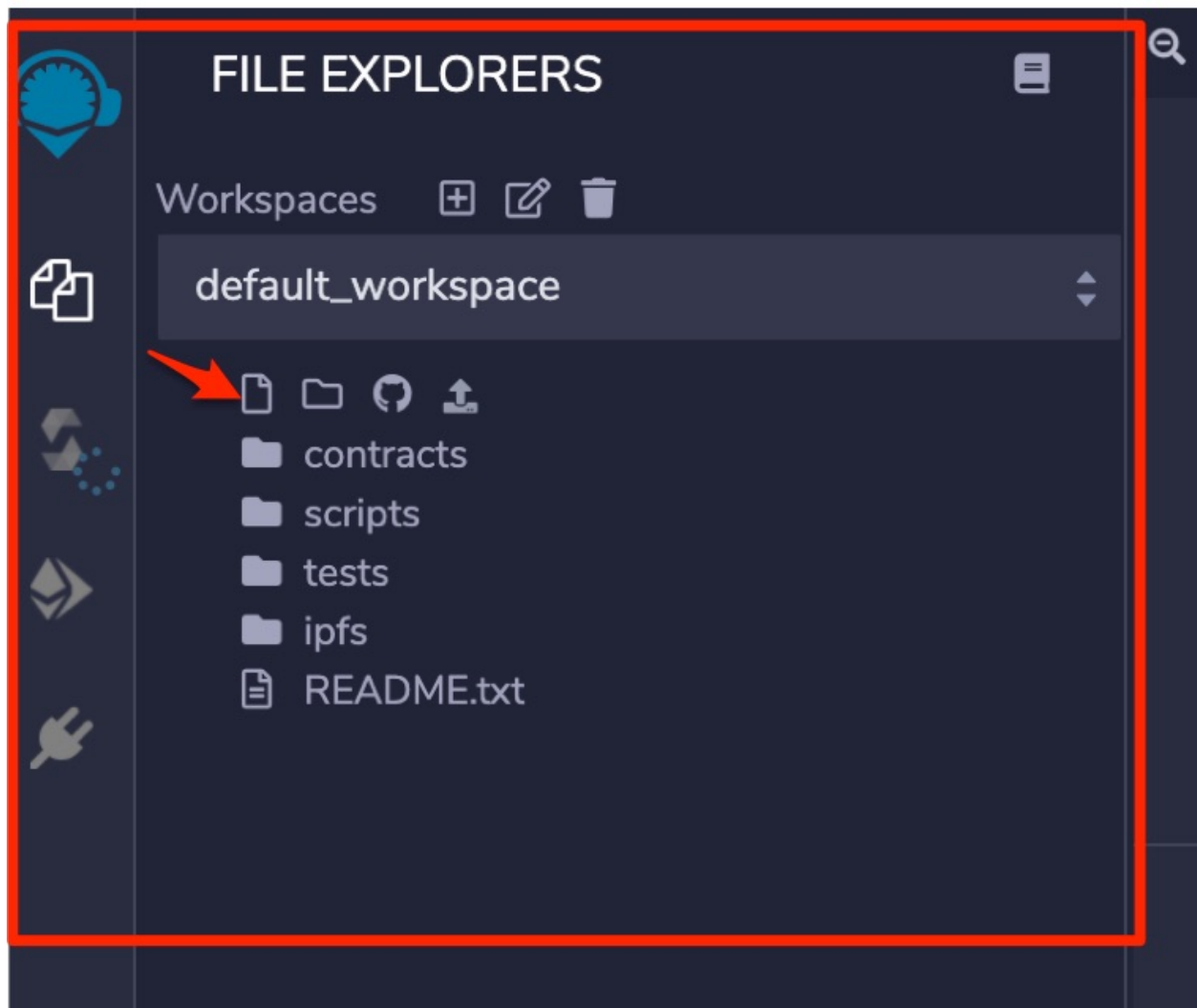
Remix Environment

You can try out code examples in this book directly in your browser with the Remix IDE. The remix is a web browser-based IDE that allows you to write, deploy and administer Solidity smart contracts, without the need to install Solidity locally.

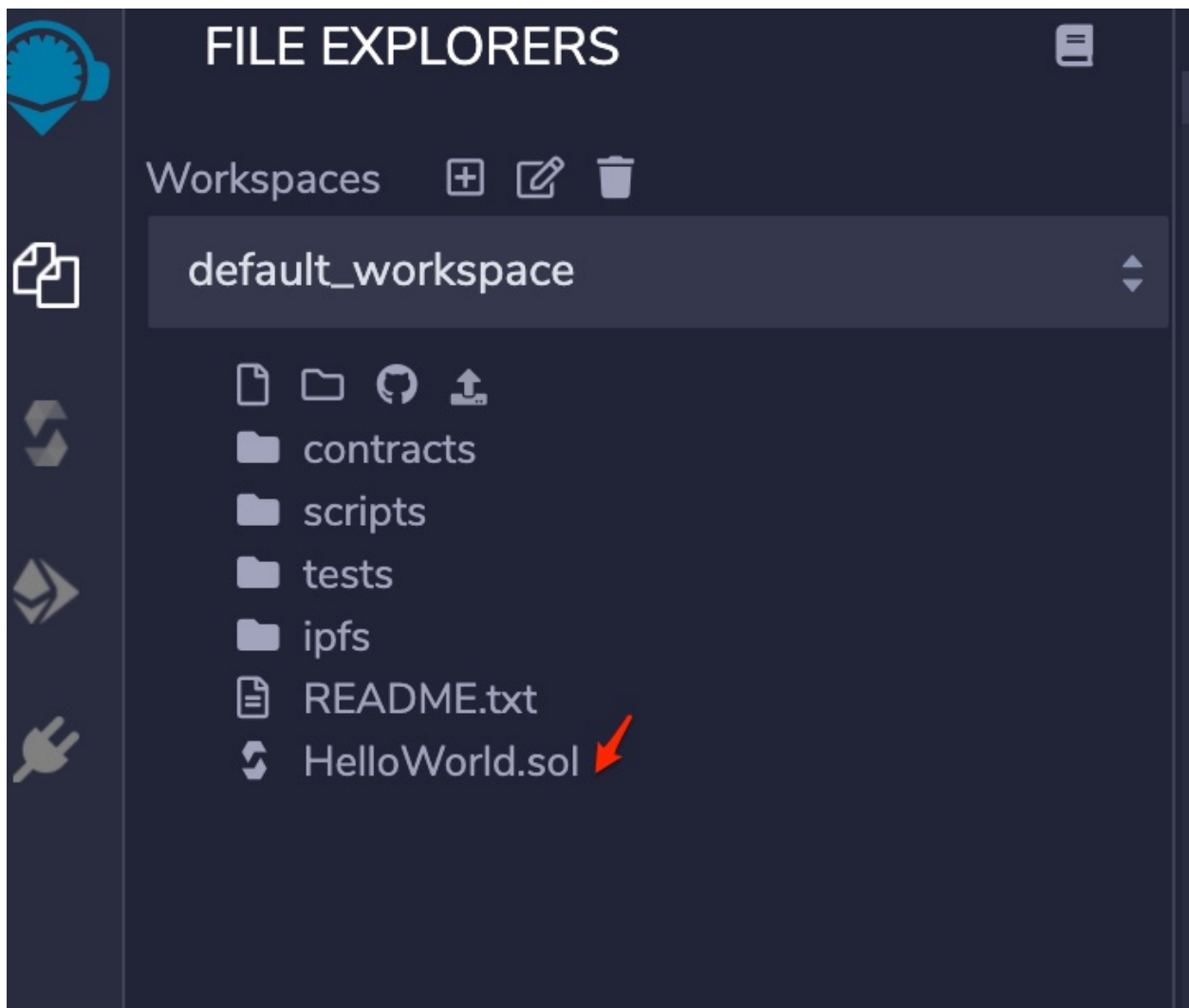
Open the <https://remix.ethereum.org/> you will be presented with the entire remix IDE in the web browser and your IDE is ready to use.

HelloWorld with Remix

Leave the default setting as it is, in your chosen workspace go to the file explorer and locate the Icon shown in the below image to create a new file and this will be our first `HelloWorld.sol` smart contract deployment in Remix



Type the file name *"HelloWorld.sol"* and enter the following code into it

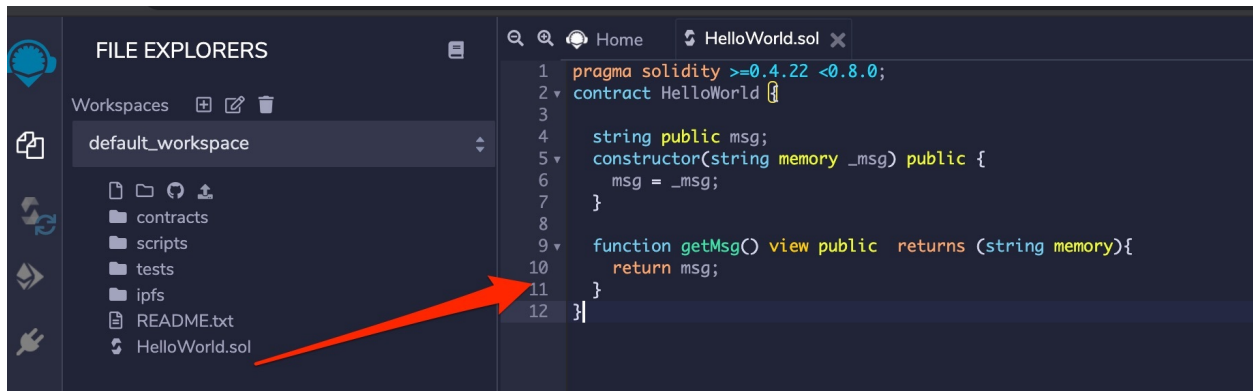


```
pragma solidity >=0.4.22 <0.8.0;
contract HelloWorld {

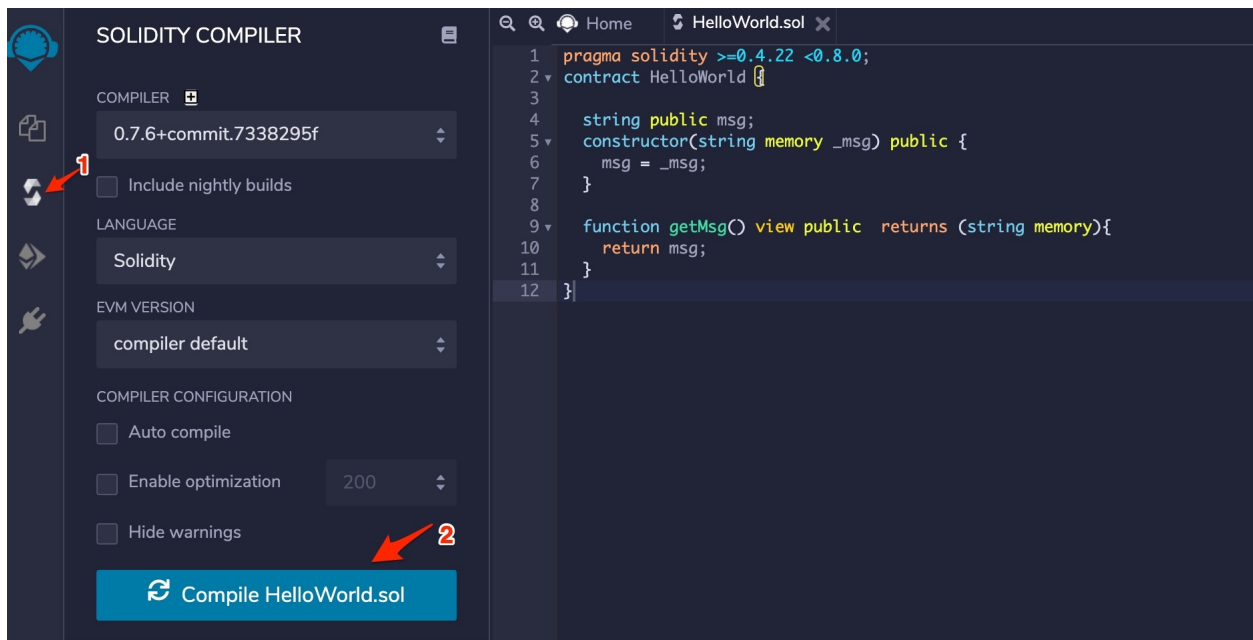
    string public msg;
    constructor(string memory _msg) public {
        msg = _msg;
    }

    function getMsg() view public returns (string memory){
        return msg;
    }
}
```

Once done click the icon just below the “file explorer” icon as shown below:



Compile your first HelloWorld.sol Contract



On successful compilation, the IDE will give you multiple Compiler options which can be explored. Now it's time to Deploy the **HelloWorld** smart contract. On file, explorer click on the Deploy option

The screenshot displays the Remix IDE interface. On the left, the 'SOLIDITY COMPILER' panel is visible, showing the compiler version '0.7.6+commit.7338295f', the language set to 'Solidity', and the EVM version set to 'compiler default'. Below these settings are checkboxes for 'Include nightly builds', 'Auto compile', 'Enable optimization' (set to 200), and 'Hide warnings'. A blue button labeled 'Compile HelloWorld.sol' is present. At the bottom of the compiler panel, a red box highlights the 'CONTRACT' section, which includes a dropdown menu showing 'HelloWorld (HelloWorld.sol)' and three buttons: 'Publish on Swarm', 'Publish on Ipfs', and 'Compilation Details'. A red arrow points to the 'Deploy Contract' button in the top-left sidebar. The main editor area on the right shows the code for 'HelloWorld.sol':

```
1 pragma solidity >=0.4.22 <0.8.0;
2 contract HelloWorld {
3
4     string public msg;
5     constructor(string memory _msg) public {
6         msg = _msg;
7     }
8
9     function getMsg() view public returns (string memory){
10         return msg;
11     }
12 }
```

A red arrow points to the 'CONTRACT' section in the compiler panel, and another red arrow points to the 'Publish on Swarm' button. A red text overlay on the right side of the editor area reads: 'Contract Compiled Successfully Next stage it to Deploy'.

Fill out the constructor parameter “Hello Solidity” required for the HelloWorld contract and click on Deploy

The screenshot shows the Remix IDE interface. On the left is the 'DEPLOY & RUN TRANSACTIONS' panel. It includes sections for 'ENVIRONMENT' (JavaScript VM), 'ACCOUNT' (0x5B3...eddC4 (100 ether)), 'GAS LIMIT' (3000000), 'VALUE' (0 wei), and 'CONTRACT' (HelloWorld - HelloWorld.sol). A red box highlights the 'Deploy' button and the 'Hello Solidity' dropdown menu. Below this is a section for 'At Address' and 'Load contract from Address'. At the bottom, it says 'Transactions recorded 0' and 'Deployed Contracts'. A message at the bottom states: 'Currently you have no contract instances to interact with.'

On the right is the 'HelloWorld.sol' contract code:

```
1 pragma solidity >=0.4.22 <0.8.0;
2 contract HelloWorld {
3
4     string public msg;
5     constructor(string memory _msg) public {
6         msg = _msg;
7     }
8
9     function getMsg() view public returns (string memory){
10         return msg;
11     }
12 }
```

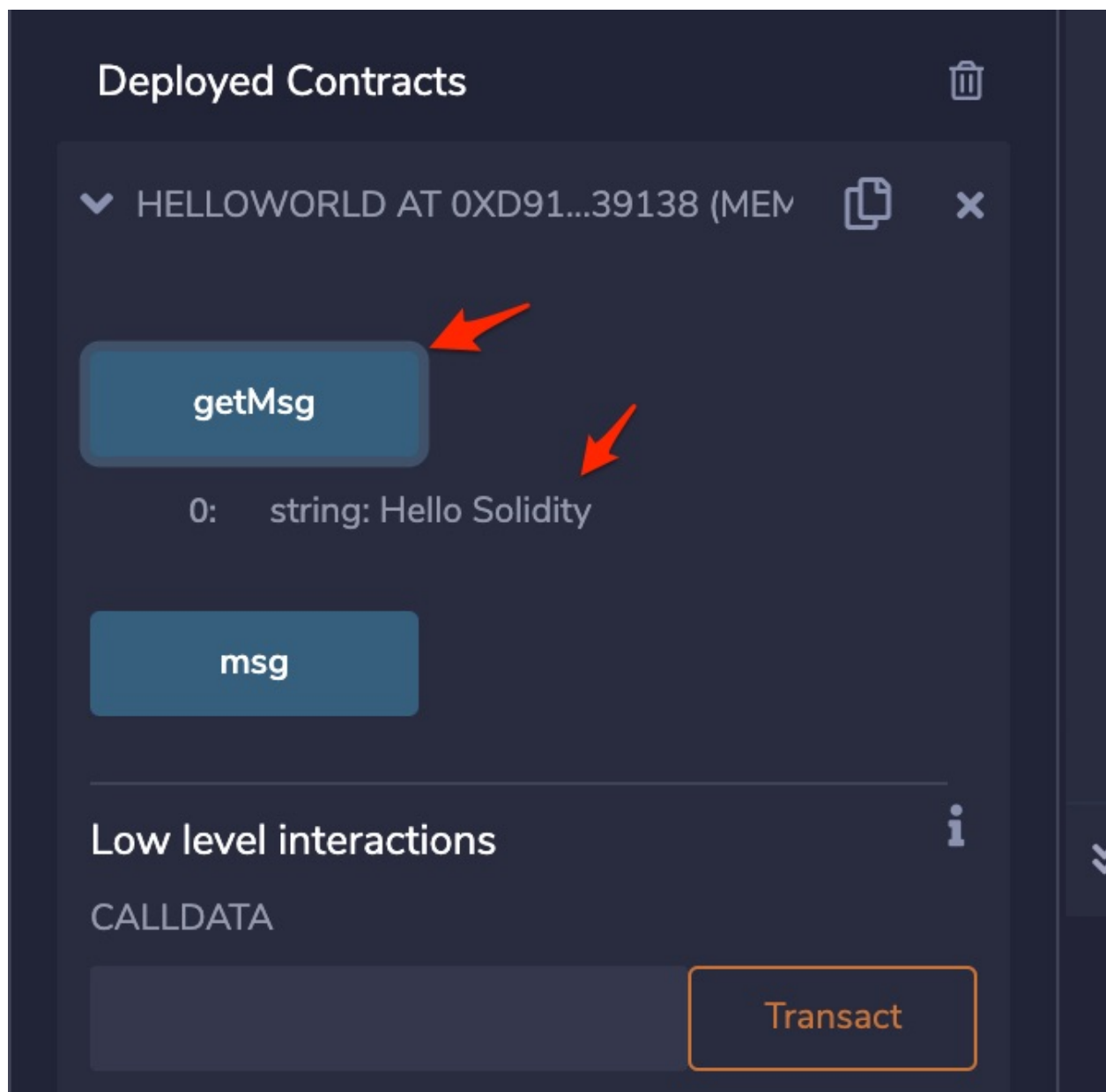
Red arrows point to the 'Deploy' button, the 'Hello Solidity' dropdown, and the 'constructor' function in the code. A red box highlights the 'Deploy' button and the 'Hello Solidity' dropdown. A red arrow points from the text 'Enter Hello Solidity and Click on Deploy' to the 'Hello Solidity' dropdown.

You will see the next screen where you can see the Deployed contracts, expand that tab and it will give you the method and state variable available for transactions. Click on `getMsg()` as defined in the contract definition

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible. It includes sections for 'ENVIRONMENT' (JavaScript VM), 'ACCOUNT' (0x5B3...eddC4), 'GAS LIMIT' (3000000), 'VALUE' (0 wei), and 'CONTRACT' (HelloWorld - HelloWorld.sol). Below these is a 'Deploy' button and a 'Publish to IPFS' checkbox. A section titled 'Transactions recorded' shows one transaction. A red box highlights the 'Deployed Contracts' section, which lists 'HELLOWORLD AT 0XD91...39138 (MEV)'. To the right of this list is a red arrow pointing to the text 'Contract Deployed'. Below the deployed contracts, there are two buttons: 'getMsg' and 'msg'. A red arrow points to the 'getMsg' button with the text 'Method Click This', and another red arrow points to the 'msg' button with the text 'State Variable'. On the right side of the interface, the Solidity code for 'HelloWorld.sol' is shown in a dark-themed editor. The code includes a pragma statement, a contract definition, a public string variable 'msg', a constructor that assigns the input to 'msg', and a 'getMsg' function that returns the value of 'msg'.

```
1 pragma solidity >=0.4.22 <0.8.0;
2 contract HelloWorld {
3
4     string public msg;
5     constructor(string memory _msg) public {
6         msg = _msg;
7     }
8
9     function getMsg() view public returns (string memory){
10         return msg;
11     }
12 }
```

The smart contract will display the output of the program



Use the Remix IDE to examine the transaction of hash, address, execution cost and others. This information will be available on the bottom pane of the remix IDE.

The Remix IDE will also show the entire calldata for e.g for the above transaction the following information is captured.

```
transaction hash      0xf772277ffa74574891ead016341ec9cd81481139c8377f6a99539453bc686207
from                  0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                    HelloWorld.getMsg() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost        24201 gas (Cost only applies when called by a contract)
hash                  0xf772277ffa74574891ead016341ec9cd81481139c8377f6a99539453bc686207
input                 0xb5f...deb23
decoded input         {}
decoded output        { "0": "string: Hello Solidity" }
logs                  []
```