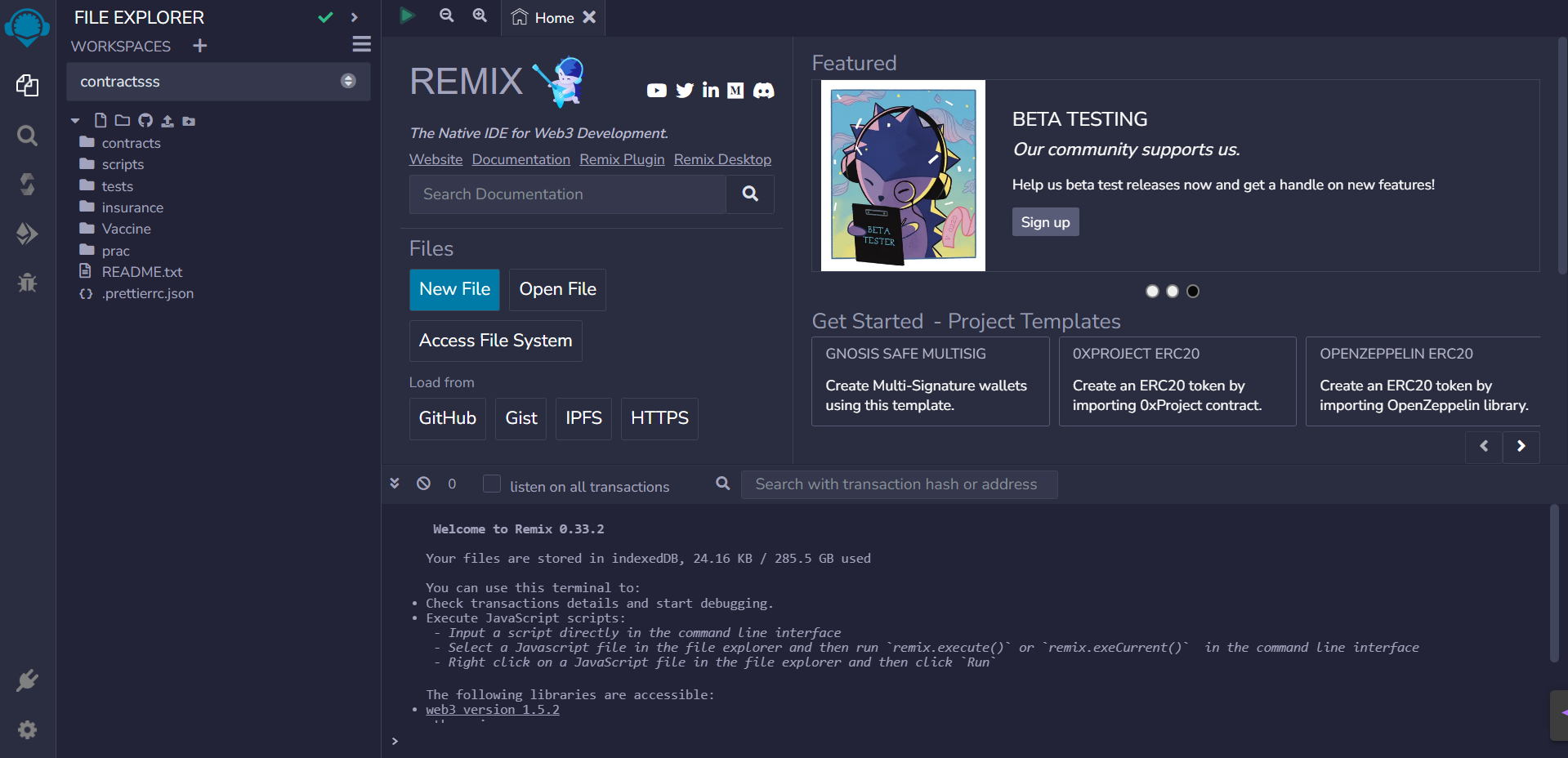
**Getting Started With Blockchain and Solidity**

For starters, we will practice our skills in the remix IDE environment.

**Step 1: Open Remix IDE**

Open your web browser and visit the Remix IDE website (<https://remix.ethereum.org>).

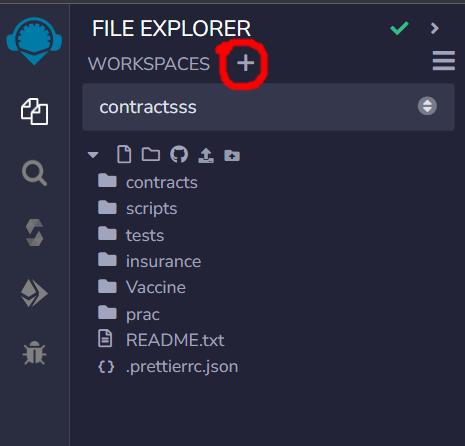
You will see the Remix IDE interface with different panels like "File Explorers," "Editor," "Compiler," "Deploy & Run Transactions," and "Solidity Compiler" on the left side.



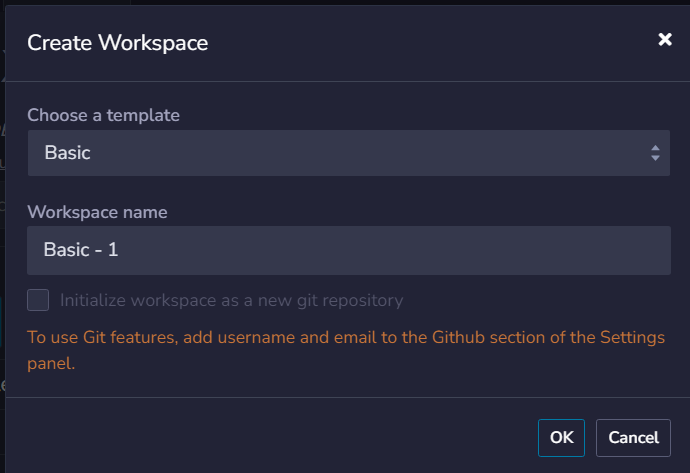
This is how the environment looks like

**Step 2: Create a Workspace**

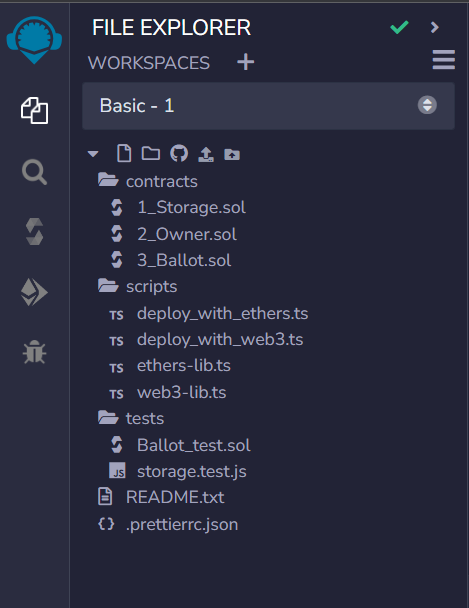
1. Click on the “+” icon beside workspace



1. Choose a basic template and give the workspace a name and click ok.



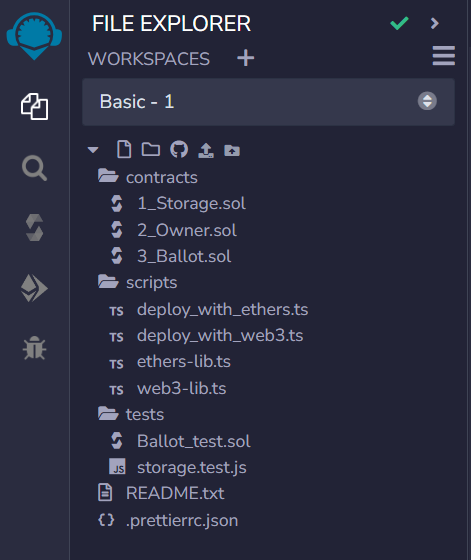
1. This is what it will look like



**Step 3: Getting to know Remix IDE**

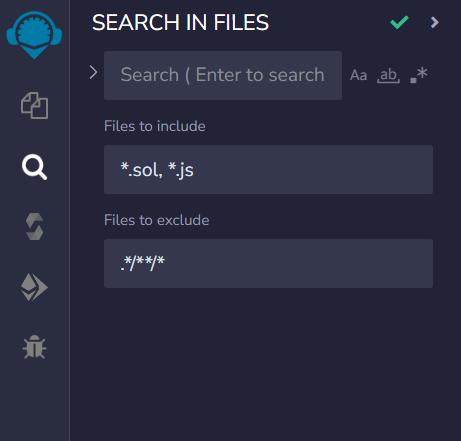
We have many plugins of remix ide in the left hand side, we will learn about this more in details,

1. File Explorer:



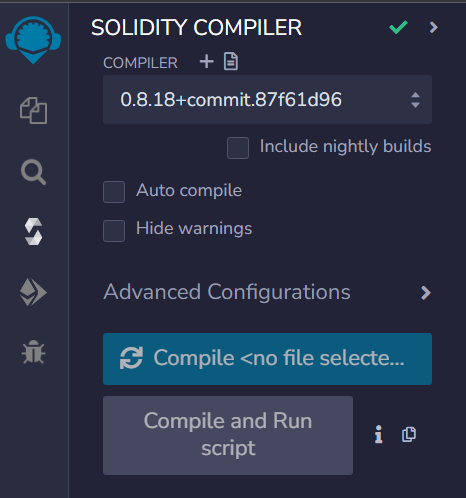
We have already see the file explorer in action. So we will move on to the next point which is,

1. Search:



The Search feature in Remix IDE enables you to quickly search for specific keywords or code snippets within your project files. It helps you locate specific functions, variables, or pieces of code without manually scrolling through the entire codebase.

1. Solidity Compiler:



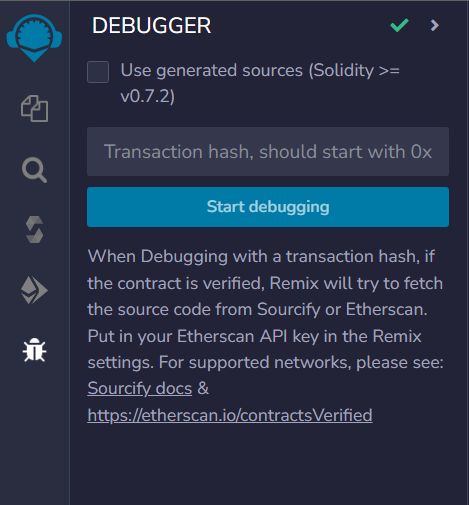
The Solidity Compiler panel is used to compile your Solidity smart contracts. It supports various versions of the Solidity compiler and allows you to select the desired compiler version for your contracts. When you compile your contracts, any errors or warnings in your code are displayed in this panel, making it easier to identify and fix issues.

1. Deploy & Run Transactions:



The Deploy & Run Transactions panel is where you can deploy your smart contracts to a blockchain network and interact with them. It provides different environment options such as JavaScript VM, Injected Web3, and Web3 Provider. With these environments, you can deploy contracts, execute transactions, and test your contract's functionality. You can also view transaction details, contract outputs, and debug messages in this panel.

1. Debugger:



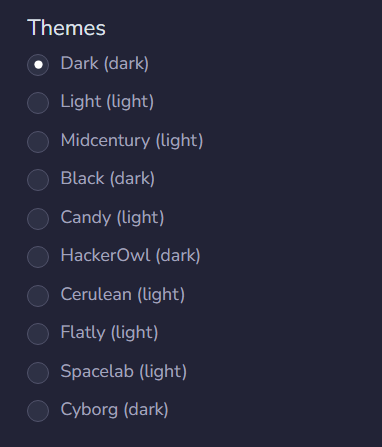
The Debugger feature in Remix IDE allows you to debug your smart contracts and analyze their execution. It provides step-by-step execution of your contract code, allowing you to observe and inspect variable values, function calls, and contract state at each step. The debugger helps in identifying and fixing issues by pinpointing the exact location of errors or unexpected behavior.

**Step 4: Changing the Theme**

1. Go to the settings from bottom left corner,

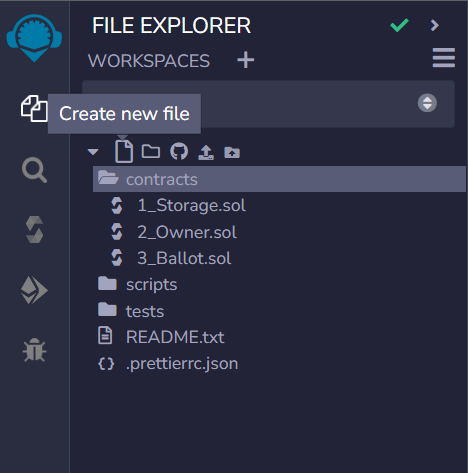


1. Scroll down to the themes, and choose what you like.

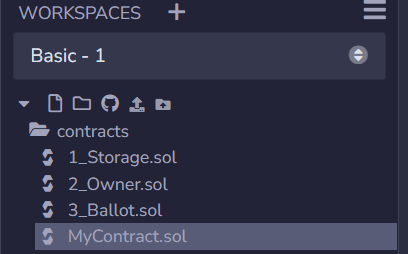


**Step 5: Create a New File**

1. Click on the contracts folder and click “Create new file” in the "File Explorers" panel to create a new file.



1. Name the file with a ".sol" extension, for example, "MyContract.sol".



1. In the "Editor" panel, you can now write your Solidity contract code.

**Step 6: Compiler**

Before writing the code we need to know about something,

**Version Pragma**

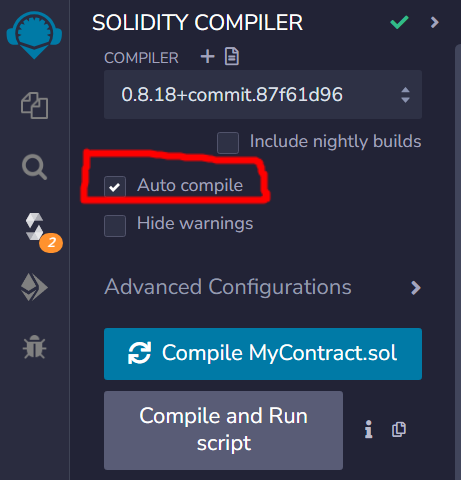
If you don't know what that pragma line is and don't want to wait several videos to understand what a pragma line is: In layman terms, it's here to configure your compiler. For example there's a version pragma, that tells the compiler "Hey, this source is made for compiler version XYZ". That's what we're going to use. Need more information? Either wait, or [read the official docs](https://docs.soliditylang.org/en/latest/layout-of-source-files.html#pragmas)

**Switch Compiler Version**

If it is necessary to switch compiler versions manually, you can always do this. You can either follow along in the videos, then use the compiler version the videos are using. Or you follow along this guide and use this solidity version.

New Compiler versions are published *very frequently*. It is very normal to find "outdated" solidity files around. Some very popular projects are using older solidity versions.

We need to make sure the ‘Auto Compile’ is enabled,



After enabling this, there will be some kind of warning, but it’s just warning. It won’t affect the coding experience.

**Step 6: Writing Our First Code**

The Contract

**Code:**

//SPDX-License-Identifier: MIT

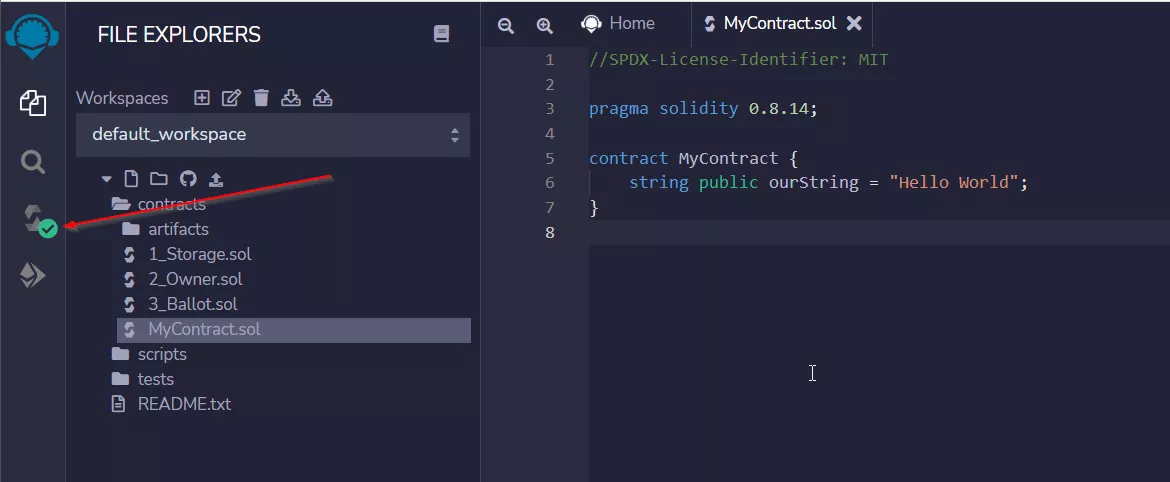
pragma solidity 0.8.14;

contract MyContract {

string public ourString = "Hello World";

}

And make sure the file is compiled without any errors or warnings!



**Wondering what that all means?**

This is a very basic version of a Smart Contract. Let's go through it line by line:

// SPDX-License-Identifier: GPL-3.0: The [The Software Package Data Exchange® (SPDX®)](https://spdx.dev/) identifier is there to clearly communicate the license under which the Solidity file will be made available. Well, if you make it available. But you should. Smart Contracts transparency and trust greatly benefit from the source being published and sometimes it's not 100% clear under which license the source is out in the wild. The [SPDX identifier is optional](https://docs.soliditylang.org/en/v0.8.1/layout-of-source-files.html#spdx-license-identifier), but recommended.

pragma solidity 0.8.14: The pragma keyword is for the compiler to enable certain features or check certain things. The *version pragma* is a safety measure, to let the compiler know for which compiler version the Solidity file was written for. It follows the [SemVer versioning standard](https://semver.org/). 0.8.14 only version 0.8.14, but if we'd write it as pragma solidity ^0.8.0 it would mean >=0.8.0 and <0.9.0.

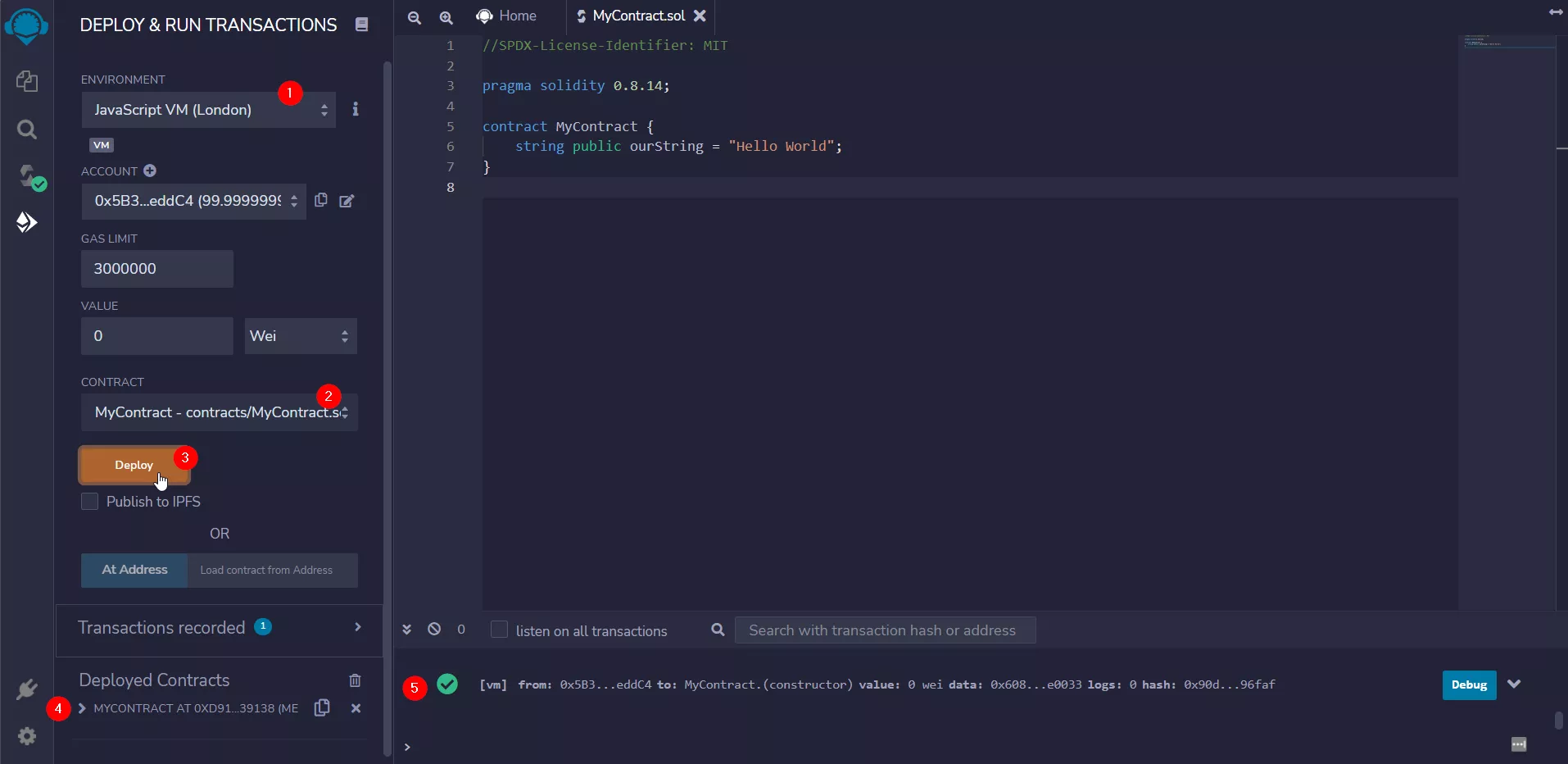
contract MyContract: That's the actual beginning of the Smart Contract. Like a Class in almost any other programming language.

string public myString = 'hello world': That is a *storage* variable. It's public and Solidity will automatically generate a getter function for it - you'll see that in a minute!

**Step-7: Deploy Smart Contract**

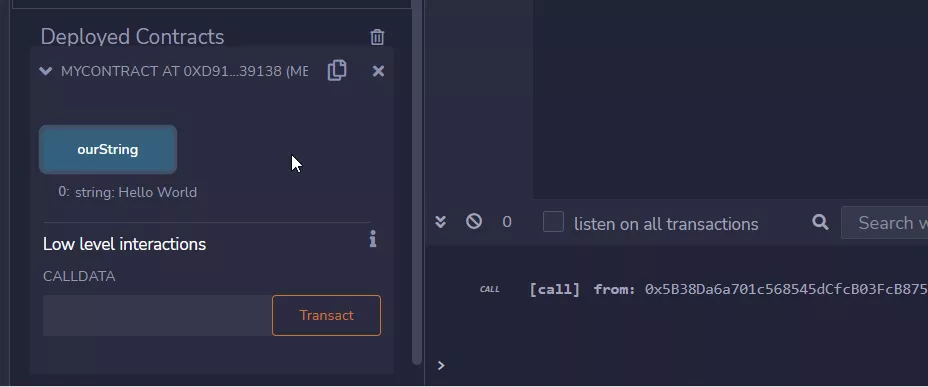
Now it's time to deploy the Smart Contract. Head over to the Deploy & Run Transactions Tab!

1. Select the JavaScript VM (currently the latest one is London) and
2. Make sure the right contract is selected from the Dropdown here. If nothing is selected, make sure the "Auto Compile" checkbox in the Compiler-Plugin is enabled.
3. deploy the Contract by clicking on "Deploy".
4. You will see a contract instance popping up on the bottom
5. And you will also witness a new transaction being logged in the logging area



**Step-8: Interact with the Smart Contract**

When you uncollapse the contract instance, then you can interact with it. Hit the little "ourString" button and it will show the "Hello World" from the string in our contract. It will also log anohter transaction in the logging area, which is now marked as "call".



**Step-9: How to stop Smart Contracts**

On the blockchain, you can't really remove contracts in the sense that also their historical data is wiped. You can "selfedestruct" them (which is a functionality that might get deprecated and removed), then you can't interact with them anymore, but their historical data is always baked into historical blocks.

In Remix, however, you just have to reload the page and the contract instances are gone. This is, because the JavaScript VM is only an in-memory blockchain simulation. If you just want to delete

1. all the contract instances, you can also click the trash-bin icon, or if you

2. just want to delete a single instance, you can click the little x-icon.

Mind: The layout might shift around from Remix version to Remix version.

