№3 «Solidity - первое применение»

Цель работы: Вы узнаете о смарт-контрактах в сети Ethereum и о том, как они выполняются виртуальной машиной Ethereum. Мы расскажем, как создать смарт-контракт на языке Solidity и как опубликовать его в приватной сети.

Remix — это IDE для Solidity (https://remix.ethereum.org/) со встроенным отладчиком и средой тестирования.Remix подключается к любой сети Ethereum. Для подключения к нашему локальному блокчейну убедитесь, что вы запустили узел майнинга.

В сети Ethereum имеется возможность создавать так называемые смарт-контракты (SmartContracts). Смарт-контракт представляет собой программный код, работающий в среде виртуальной машины Ethereum. Он запускается на всех узлах сети, и результаты его работы также реплицируются на все узлы.

С помощью смарт-контрактов очень удобно отслеживать выполнениетранзакций, например, имеющих отношение к поставке товаров (особенно электронных, таких как файлы книг или доступы к сервисам), автоматизируя оплату при помощи криптовалютных средств. Если смарт-контракт получил тем или иным способом подтверждение выполнения условий сделки, то он может сам, автоматически, перевести средства поставщику.

Если условия сделки были выполнены не полностью или не выполнены вовсе, смарт-контракт может вернуть средства покупателю или перевести сумму штрафа на счет пострадавшей стороны.

Одной особенностью смарт-контракта является то, что его нельзя оспорить. Если логика смарт-контракта сработает таким образом, что средства будут переведены, эти средства уже невозможно будет вернуть.

Запускаем RemixIDE, после чего у нас откроется такое окно (рис. 8):

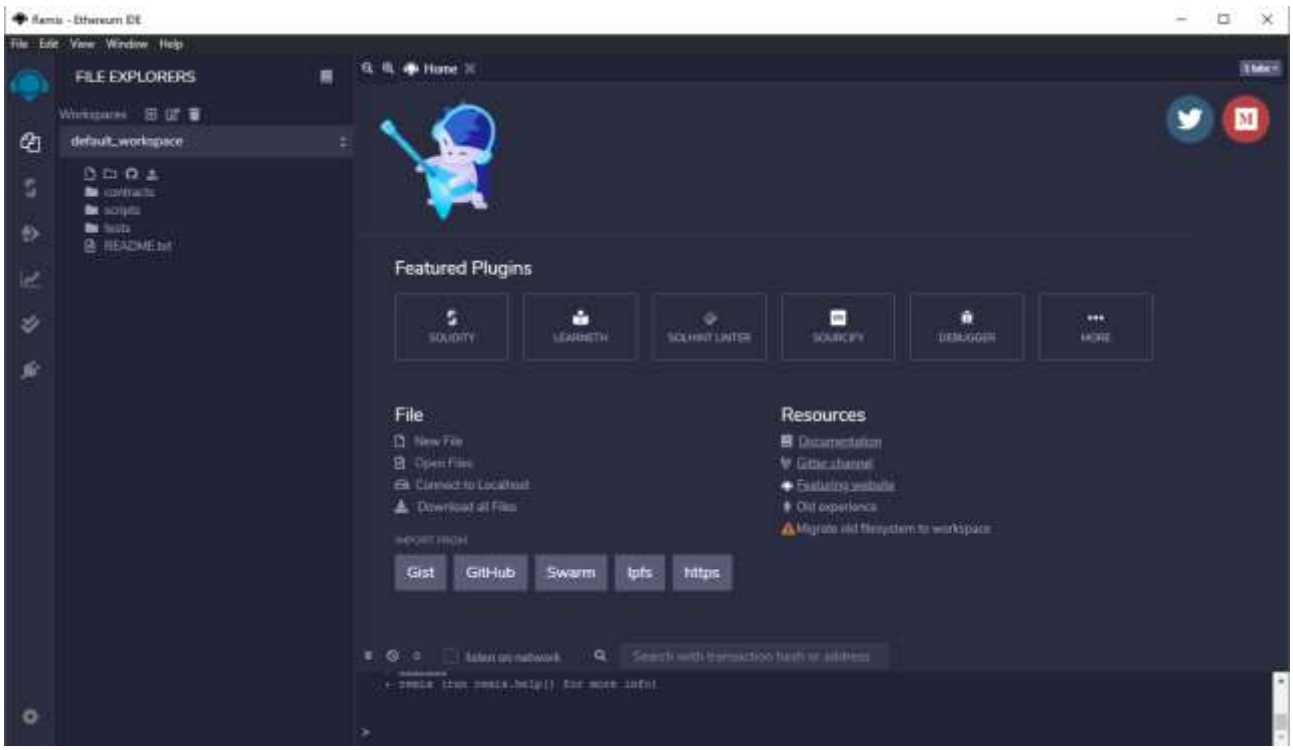

Рис. 8 - Remix IDE

Для подключения ***remix*** к нашей локальной сети нужно изменить **JavaScript VM** на ***Web3 Provider***, это делается на вкладке **Deploy &*Run transactions***вправойчастиокна (рис. 9)

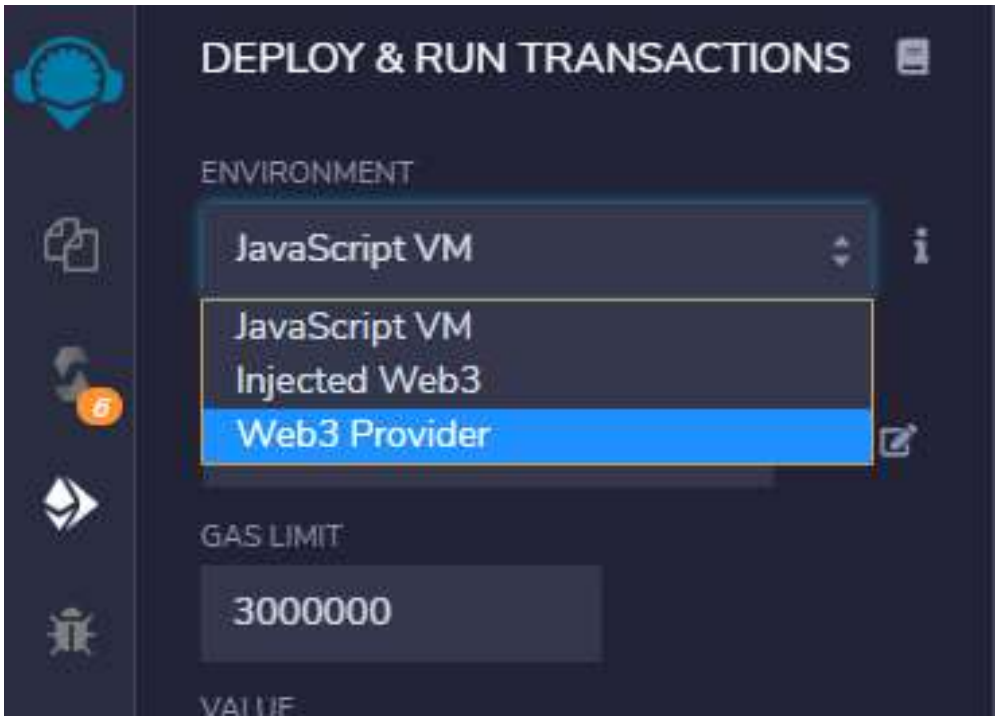

Рис. 9 - Выбор среды Web3

При этом изменении remix попросит указать **Web3 providerEndpoint**. По умолчанию узел для майнинга стартует на порту 8545, нажимаем ОК и готово.

Заметим, что при отладке смарт-контрактов процесс майнинга останавливать не нужно, иначе ваш узел не сможет обрабатывать транзакции, публиковать контракты и вызывать методы контрактов.

Заходим во вкладку **Fileexplorers**в левой части окна, переходим на пространство по умолчанию, нажимаем на папку **contracts**и создаём в этой папке новый файл. Называем его Solidity, переходим по нему и вписываем данный код (рис. 10) в раздел кода RemixIDE.

```
1  pragma solidity 0.5.0;
2  contract SimpleStorage {
3      uint storedData;
4      function set(uint x) public {
5          storedData = x;
6      }
7      function get() public view returns (uint) {
8          return storedData;
9      }
10 }
```

Рис. 10 - Простой файл Solidity

Первая строка - это директива pragma, которая сообщает, что исходный код написан для Solidity версии 0.5.0

Директива прагмы всегда является локальной для исходного файла, и если вы импортируете другой файл, прагма из этого файла не будет автоматически применяться к импортируемому файлу.

Таким образом, прагма для файла, который не будет компилироваться ранее, чем версия 0.5.0, и он также не будет работать на компиляторе, начиная с версии 0.6.0, будет записана следующим образом:**pragma solidity ^0.5.0;**

Контракт Solidity - это набор кода (его функций) и данных (его состояния), который находится по определенному адресу в блокчейне Ethereum .

Строка **uintstoredData;** объявляет переменную состояния с именем storeData типа uint, а функции set и get можно использовать для изменения или получения значения переменной.

Переходим во вкладку компиляции (**Solidirycompiler**), нажимаем на кнопку CompileSolidity.sol, ждём когда код проверится и загорится зеленная галочка рядом с иконкой компиляции (рис. 11)

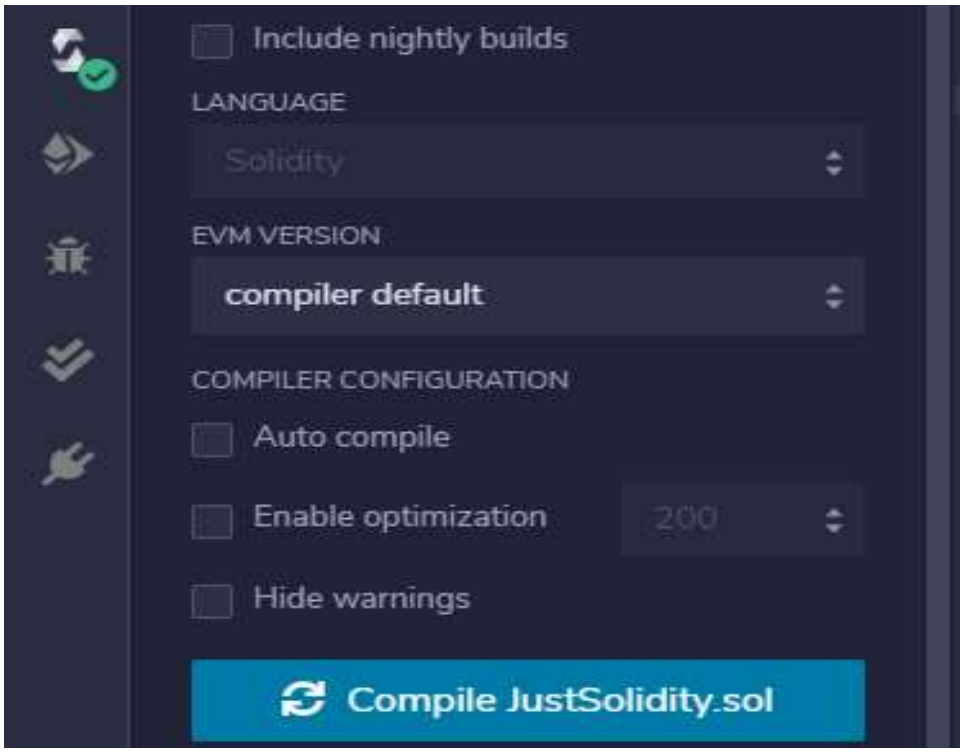

Рис. 11 – Solidirycompiler

Для автоматической компиляции можно поставить галочку напротив «Autocompile».

Снова переходим во вкладку «Развернуть и Выполнить» (**Deploy&RunTransactions)**, нажимаем кнопку **Deploy** и ожидаем записи как на рисунке ниже (рис. 12):

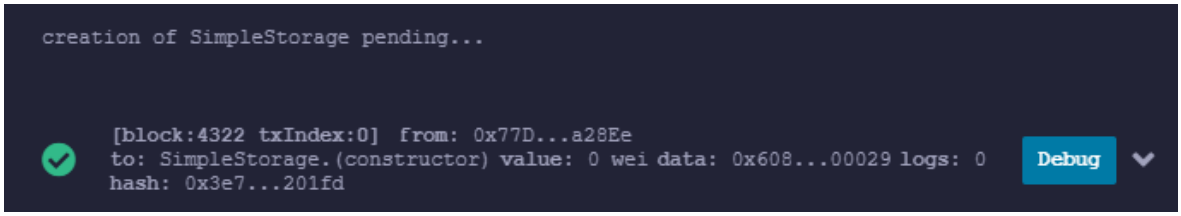

Рис. 12 – Завершениеразвёртывания

Вслучае, есливысветитсянадпись: **transact to greeter.setGreeting errored: Error: Returned error: authentication needed: password or unlock,** при выполнении смарт-контрактов вам понадобиться снова разблокировать аккаунт в консоли Git.

Далее мы раскрываем созданный нами контракт под **DeployedContracts**, вписываем любой текст в первом окне**,** нажимаем на кнопку **set**, ожидаем запись, (рис. 12) после чего нажимаем на **get** где нам выводит введённый нами текст (рис. 13)

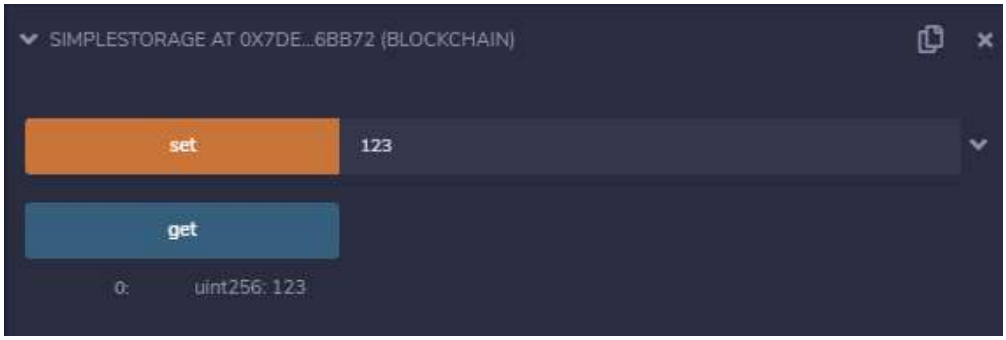

Рис. 13 - Пример выполненного контракта

Итоги работы: В этой практической мы научились подключать remix к нашей локальной сети, написали простой смарт-контракт для понимания что он из себя представляет и научились компилировать, и выполнять написанный нами смарт-контракт.

Вопросы:

1. Что представляет из себя смарт-контракт и каковы его функции?

2. Что сделает смарт-контракт если условия сделки были выполнены не полностью?