Introduction to Solidity

Difficulty Level : Basic • Last Updated : 11 May, 2022

Solidity is a brand-new programming language created by the **Ethereum** which is the second-largest market of cryptocurrency by capitalization, released in the year 2015 led by Christian Reitwiessner. Some key features of solidity are listed below:

- Solidity is a high-level programming language designed for implementing smart contracts.
- It is statically-typed object-oriented(contract-oriented) language.
- Solidity is highly influenced by Python, c++, and JavaScript which runs on the Ethereum Virtual Machine(EVM).
- Solidity supports complex user-defined programming, libraries and inheritance.
- Solidity is primary language for blockchains running platforms.
- Solidity can be used to creating contracts like voting, blind auctions, crowdfunding, multi-signature wallets, etc.

Ethereum

Ethereum is a decentralized open-source platform based on blockchain domain, used to run smart contracts i.e. applications that execute the program exactly as it was programmed without the possibility of any fraud, interference from a third party, censorship, or downtime. It serves a platform for nearly 2,60,000 different cryptocurrencies. Ether is a cryptocurrency generated by ethereum miners, used to reward for the computations performed to secure the blockchain.

Ethereum Virtual Machine(EVM)

untrusted code using an international network of public nodes. EVM is specialized to prevent Denial-of-service attack and confirms that the program does not have any access to each other's state, also ensures that the communication is established without any potential interference.

Smart Contract

Smart contracts are high-level program codes that are compiled to EVM byte code and deployed to the ethereum blockchain for further execution. It allows us to perform credible transactions without any interference of the third party, these transactions are trackable and irreversible. Languages used to write smart contracts are Solidity (a language library with similarities to C and JavaScript), Serpent (similar to Python, but deprecated), LLL (a low-level Lisp-like language), and Mutan (Go-based, but deprecated).

Example: In the below example, we have discussed a sample solidity program to demonstrate how to write a smart contract in Solidity.

Solidity

```
// Solidity program to
// demonstrate how to
// write a smart contract
pragma solidity >= 0.4.16 < 0.7.0;

// Defining a contract
contract Test
{

    // Declaring state variables
    uint public var1;
    uint public var2;
```

```
    // that sets the value of
    // the state variable
    function set(uint x, uint y) public
    {
        var1 = x;
        var2=y;
        sum=var1+var2;
    }

    // Defining function to
    // print the sum of
    // state variables
    function get(
    ) public view returns (uint) {
        return sum;
    }
}
```

Output:

Explanation:

1. Version Pragma:

```
pragma solidity >=0.4.16 <0.7.0;
```

Pragmas are instructions to the compiler on how to treat the code. All solidity source code should start with a "version pragma" which is a declaration of the version of the Solidity compiler this code should use. This helps the code from being incompatible

and equal to 0.4.16 but less than version 0.7.0.

2. The contract keyword:

```
contract Test{
//Functions and Data
}
```

The contract keyword declares a contract under which is the code encapsulated.

3. State variables:

```
uint public var1;
uint public var2;
uint public sum;
```

State variables are permanently stored in contract storage that they are written in Ethereum Blockchain. The line *uint public var1* declares a state variable called var1 of type uint (unsigned integer of 256 bits). Think of it as adding a slot in a database. Similarly, goes with the declaration *uint public var2* and *uint public sum*.

4. A function declaration:

```
function set(uint x, uint y) public
function get() public view returns (uint)
```

- This is a function named *set* of access modifier type *public* which takes a variable *x* and variable y of datatype *uint* as a parameter.
- This was an example of a simple smart contract which updates the value of var1 and var2. Anyone can call the function set and overwrite the value of var1 and var2 which is stored in Ethereum blockchain. This is an example of a decentralized application that is censorship proof and unaffected to the shutdown of any centralized server. As long as someone is running a single node of Ethereum blockchain, this smart contract will be accessible.
- The variable sum is calculated by adding the values of the variables var1 and var2.
- Function get will retrieve and print the value of the state variable sum.

ow to Execute The Code:

prerequisites and 4 major steps to be followed to get the smart contract running:

- **Prerequisites:**
  - Download and install node.js.
  - Install Truffle globally.
  - Install ganache-cli.
- **Objectives:**
  - Create a truffle project and configure a development network
  - Create and deploy smart contracts
  - Interact with the smart contract from Truffle console
  - Write tests for testing main features offered by Solidity

2. Online Mode: Remix IDE is generally used to compile and run Solidity smart contracts in the Online Mode. You can find the complete articles with all the steps here.

Previous Next

Related Articles

1. Solidity - Functions
2. Solidity - Inheritance

4. Solidity - View and Pure Functions
5. Solidity - Encapsulation
6. How to Install Solidity in Windows?
7. Solidity - While, Do-While, and For Loop
8. Solidity - Break and Continue Statements
9. Solidity - Variables
10. Solidity - Error Handling

Article Contributed By :

jeeteshgavande30
@jeeteshgavande30

Vote for difficulty

Current difficulty : Basic

Easy | Normal | Medium | Hard | Expert

Article Tags : Solidity-Basics, Blockchain, Solidity

Improve Article | Report Issue

feedback@geeksforgeeks.org