Global, Local, and State Variables in Solidity

Table of Content

Global, local and state variables

Variables:

Variables in programming are placeholders to store data values. They can vary and change, hence the name "variable." In real life, think of variables as containers—a box that can hold different items at different times.

Global Variables:

Global variables in Solidity are accessible throughout the contract and often hold essential information about the blockchain's context. An example in real life could be your home address, which remains constant and can be accessed from any part of a discussion.

Local Variables:

Local variables in Solidity are confined within a specific function's scope and cease to exist once the function execution completes. Think of local variables as notes you take during a meeting that are relevant only for that discussion and aren't accessible afterward.

State Variables:

State variables are stored on the blockchain, persisting across different function calls and transactions. They represent the contract's state and retain their values between interactions. An analogy could be a bank balance that maintains its value regardless of how many transactions occur.

Syntax:

Global Variable:

```
address addr = msg.sender;
```

Local Variable:

```
function localVariableExample(uint256 newValue) public pure returns (uint256) { uint256 localVar = newValue * 2; return localVar; }
```

State Variable:

```
uint256 public stateVariable = 100;
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract VariableExample {
    // State variable - persistent across function calls and transactions
    uint256 public stateVariable = 100;

    // Global variable - available throughout the contract
    address addr = msg.sender;

    function localVariableExample(uint256 newValue) public pure returns (uint256) {
        // Local variable - exists only within this function
        uint256 localVar = newValue * 2;
        return localVar;
    }

    function updateStateVariable(uint256 newValue) public {
        // Updating the state variable
        stateVariable = newValue;

    }
}
```

Explanation of Provided Code:

- **stateVariable:**
    - Type: State variable (uint256)
    - Scope: Accessible publicly (public)
    - Initial Value: Initialized to 100
    - Purpose: Represents a variable persisting across transactions and function calls within the contract VariableExample.
- **addr:**
    - **Type: Global variable (address)**
    - Scope: Available throughout the contract

- Purpose: Stores the address of the contract deployer (msg.sender), accessing the global variable msg.sender.

- **localVariableExample Function:**
  - Type: Function with a local variable
  - Purpose: Accepts an argument newValue, performs a computation (newValue * 2) using a local variable localVar, and returns the result.
- **updateStateVariable Function:**
  - Type: Function to update a state variable
  - Purpose: Modifies the stateVariable with the provided newValue.

State Vs Local Variable

Parameter	State Variables	Local Variables
Scope	State Variables are defined at the contract level, i.e. outside the function but inside the contract. And their scope depends upon the type of access modifier we're assigning to it	The scope of local variables is limited to the function or a code block that they are defined in.
Lifetime	State variables have a persistent lifespan as they are stored on the contract storage and they can exist as long as the contract is deployed on the blockchain.	Whereas the Local variable's lifespan is limited and they are destroyed from the memory once the code block or the function they are defined in returns or completes its execution
Visibility	State variables are visible to all the functions within the contract and their visibility can be controlled by the use of access modifiers	Local Variables are visible only to the code block or the function they are defined in
Storage	State variables are stored on the blockchain which means they are allocated some space and that can contribute to the overall cost of deploying the contract	Local Variables are stored in the memory and don't cost anything which means using local variables could be cost-efficient.
Security	State variables are vulnerable as they are stored on the blockchain and can be accessed by any function. Hence it's very important to ensure that state variables are allowed access only to the appropriate components.	Since the local variable's scope is limited to the function or the code block they are defined in, they are pretty much secure compared to the state variables

Must Read:

https://www.geeksforgeeks.org/solidity-state-variables/

Global, Local, and State Variables: Understanding Their Roles

Global, Local, and State Variables in programming are like containers that hold information. Each has a specific scope and usage within a program.

Global Variables:

Think of Global Variables as messages written on a community bulletin board. They're accessible from anywhere within the community (or program). These variables store information that any part of the program can read or modify. For instance, a global variable could hold the current time or a frequently used value accessible throughout the program.

Local Variables:

Local Variables are more like notes on a personal to-do list. They're specific to a certain task or function and exist only within that function's area. These variables are temporary and get created when the function starts and disappear when it ends. They store information relevant only to that particular function, like calculations or temporary values.

State Variables:

State Variables are similar to a personal diary. They hold information about the ongoing state or condition of something, like the score in a game or the balance in a bank account. Unlike local variables, state variables persist beyond a single function; they stay and retain their value as long as the program or contract is active.

Real-Life Comparison:

Imagine a neighborhood: the community notice board (global variables) contains announcements for everyone, your to-do list (local variables) has tasks specific to your day, and your personal diary (state variables) holds ongoing records, like your fitness progress or monthly budget.

In programming, each type of variable serves a distinct purpose, aiding in organizing and managing information within the code.