View and Pure

Table of Content

View and Pure

View and Pure Functions:

Introduction to view and pure

In Solidity, view and pure are function state mutability modifiers that provide developers with precise control over function behavior and indicate the nature of a function's impact on the blockchain state. These modifiers don't alter the state of the contract and are crucial in ensuring that functions execute predictably without modifying state variables.

Role and Functionality of view Functions

Functions marked as view promise not to modify the contract's state. These functions enable read-only access to the contract's state variables, providing a way to retrieve information without modifying anything on the blockchain. view functions are used for querying contract states and are cost-free when executed externally.

Role and Functionality of pure Functions

On the other hand, pure functions signify that a function doesn't read from or modify the contract's state. These functions solely depend on their inputs and don't interact with the contract's storage or other contracts. They are used for performing computations or transformations and are especially useful for mathematical calculations or data manipulations.

Syntax and Implementation of view and pure

In Solidity, view and pure are added as modifiers to function declarations. The syntax for declaring these functions is straightforward, using the view or pure keyword after the function visibility and before the returns.

Real-Life Examples:

An example of a view function could be a contract containing a balance-check function. This function, marked as view, provides the balance of a specific address without altering any contract state. Conversely, a pure function could include a mathematical calculation, such as finding the factorial of a number, which doesn't require interaction with the contract's state.

In Solidity, **view** and **pure** are function modifiers used to specify the behavior of functions and provide crucial information to the compiler regarding their interaction with contract state and external data.

View Function:

- **Usage:** Functions declared as **view** do not modify the state of the contract. They promise read-only access to the contract's data.
- **Purpose:** Ideal for retrieving data from the contract without making any changes to the state.

Syntax:

```
function functionName() view returns (datatype) {
// Function logic
}
```

pure Function:

- **Usage:** Functions declared as **pure** ensure that they neither read nor modify the contract's state. They don't even access external data.
- **Purpose:** Perfect for utility functions or mathematical calculations that do not require contract state or external data.

Syntax:

```
function functionName() pure returns (datatype) {
// Function logic
}
```

Code Explanation:

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

contract ViewPureDemo {
    uint256 public storedValue = 5;

    // View Function: Retrieve stored value
    function getValue() public view returns (uint256) {
        return storedValue;
    }

    // Pure Function: Multiply two numbers
    function multiply(uint256 a, uint256 b) public pure returns (uint256) {
        return a * b;
    }
}
```

getValue() Function (View):

- This function, marked as **view**, retrieves the value of **storedValue** without altering the contract's state. It provides read-only access to **storedValue**.

Multiply () Function (Pure):

- The **multiply ()** function, declared as **pure**, performs a multiplication operation on two numbers (**a** and **b**) without relying on or altering the contract state. It strictly deals with pure computation.

In a bank, **view** functions act like a customer checking their account balance without making any changes, while **pure** functions are like mathematical calculations performed without any reference to the bank's account details.

These functions in Solidity demonstrate the use of **view** and **pure** modifiers, ensuring functions are clearly marked based on their interaction with the contract's state or external data.

View:

- **Reading Data:** Functions marked as **view** do not modify the contract's state. They only read data from the blockchain.
- **Usage:** Use **view** functions when you need to fetch data from the blockchain but don't intend to modify any state variables.

Pure:

- **No State Changes:** Functions marked as **pure** don't read from or modify the contract's state. They solely perform computations.
- **Usage:** Use **pure** functions for computations or transformations that do not involve blockchain data access or state changes.

Usage in Smart Contracts:

- **View:** Use **view** functions for retrieving data from the blockchain without altering the contract's state, like getting account balances or querying contract information.
- **Pure:** Use **pure** functions for calculations or transformations that don't require blockchain interaction, such as mathematical operations or data formatting.

Real-Life Comparison:

Consider **view** as looking at a book without writing anything – you're reading the information but not altering the book's content. **Pure** is like using a calculator to perform calculations – it doesn't affect the information around you, only the computations you perform.

In Solidity, using **view** and **pure** appropriately helps clarify the function's purpose and ensures no unintended changes to the contract's state, optimizing gas usage and contract clarity.