**CODELANDCS BLOCKCHAIN DEVELOPMENT SYLLABUS**

**WEEK 3**

**DAY 3**

**FUNCTIONS IN SOLIDITY**

Functions are blocks of code that perform specific tasks. In Solidity, functions are defined using the **function keyword**, followed by the function name, parentheses, and a block of code enclosed in curly braces.

Let's take a look at the components that makes up a function in solidity:

**functionName:** This is the name of the function. It should be descriptive of the task the function performs.

**parameters:** This is an optional list of parameters that the function takes as inputs. Parameters are separated by commas and are defined as parameterType followed by parameterName.

**[public/private]:** This specifies the visibility of the function. By default, functions are public.

**[view/pure]:** This specifies the type of function. A **view** function reads from the blockchain but does not modify it, while a **pure** function does not read or modify the blockchain.

**[payable]:** This specifies whether the function can accept Ether.

**[returns (returnType)]:** This specifies the return type of the function.

**VIEW AND PURE FUNCTIONS**

**View Functions**

In Solidity, a view function is a function that **does not modify the state of the blockchain.** It is a **read-only** function that returns a value or a set of values. When a view function is called, it does not change the state of the blockchain or **consume any gas**. It simply reads the data from the blockchain and returns it to the caller.

When a view function is called, it is executed on the local node and does **not require any network interaction.** The function reads the data from the blockchain and returns it to the caller. Because view functions do not modify the state of the blockchain, they do not consume any gas.

View functions can access the blockchain data, but they cannot modify it. This means **that they cannot create new transactions or modify existing ones.** They can only read the data from the blockchain and return it to the caller.

View functions are important in Solidity because they provide a way to read the data from the blockchain without modifying it. This is useful in situations where you need to retrieve data from the blockchain but do not need to modify it. It is also useful in situations where you want to retrieve data from the blockchain without incurring any gas costs.

View functions are also important because they help to improve the security of the smart contract. By using view functions, you can ensure that the contract cannot be modified by unauthorized users. This is because view functions do not modify the state of the blockchain and are therefore safe to execute.

**Pure Functions**

A pure function in Solidity is a function that **does not read or modify the state of the blockchain**. It is similar to a view function in that it does not modify the state, but it differs in that it also does not read any data from the blockchain. Instead, it **operates solely on the input parameters passed to it.**

In Solidity, a pure function is defined using the **"pure" keyword.**

When a pure function is called, it operates solely on the input parameters passed to it. It does not read any data from the blockchain or modify any data on the blockchain. Because it does not access the blockchain, it does not require any gas to execute.

Pure functions are executed on the local node and do not require any network interaction. They simply perform their calculations based on the input parameters and return the result.

Pure functions are important in Solidity because they provide a way to perform calculations without modifying or reading any data from the blockchain. This is useful in situations where you need to perform calculations based on input parameters without accessing the blockchain.

Pure functions are also important because they improve the security of the smart contract. Because pure functions do not access the blockchain, they cannot be used to modify or access any data on the blockchain. This reduces the attack surface of the smart contract and makes it more secure.

**FUNCTION VISIBILITY**

Function visibility in Solidity is a way to specify who can access and execute a function.

We will discuss the four types of function visibility in Solidity, namely **Public,** **External**, **Internal**, and **Private**. We will explore the use cases for each visibility level, their advantages, and their limitations.

**Public Visibility**

A public function in Solidity is visible to everyone and can be called from within the contract as well as from outside the contract. The function's return value and arguments are also visible to everyone.

One of the main advantages of public visibility is that it allows other contracts and external accounts to interact with the contract. This can be useful in situations where you want to allow external parties to execute certain functions within the contract.

However, public functions also have some limitations. For example, because anyone can execute a public function, you need to be careful about how you implement it. If the function has side effects or modifies the contract's state, you need to ensure that it is executed safely and without unintended consequences.

**External Visibility**

An external function in Solidity is similar to a public function, but with one key difference. An external function **can only be called from outside the contract,** not from within the contract. This means that an external function cannot modify the contract's state.

The main advantage of external visibility is that it provides a safe way to expose functionality to external accounts and contracts without the risk of unintended side effects. This makes external functions a good choice for utility functions or **functions that perform read-only operations**.

**Internal Visibility**

An internal function in Solidity can only be called from within the contract. This means that the function is not visible to external accounts or contracts.

The main advantage of internal visibility is that it provides a way to encapsulate the contract's logic and prevent other contracts from calling certain functions. This can be useful when you want to prevent unauthorized access to sensitive functions or when you want to keep certain functions private to the contract.

**Private Visibility**

A private function in Solidity is similar to an internal function, but with one key difference. A private function **can only be called from within the same contract** and not from within derived contracts.

The main advantage of private visibility is that it provides a way to encapsulate the contract's logic even further and prevent derived contracts from calling certain functions. This can be useful when you want to prevent derived contracts from accessing certain functions or when you want to keep certain functions private to the contract and its internal implementation.

**PAYABLE FUNCTIONS**

One of the important features of Solidity is the ability to **create payable functions** that enable the transfer of Ether from the caller to the smart contract.

A payable function is a function that can receive Ether along with the function call. The **keyword payable** must be added to the function declaration to **indicate that the function can receive Ether.** A payable function can be used to implement several use cases, such as accepting payments, buying tokens, and handling auctions.

The syntax of a payable function is similar to that of a regular function in Solidity, **with the addition of the payable keyword.**

When a payable function receives Ether, the **msg.value variable** is set to the amount of Ether sent. The received Ether can be stored in a variable, and the contract can perform various operations with it, such as transferring it to another address or storing it in a mapping.

When working with payable functions in Solidity, it's important to consider security aspects. Here are a few considerations:

* Check that msg.value is greater than 0 before performing any operations with the received Ether.
* Use a withdrawal pattern to ensure that funds can be withdrawn from the contract in case of emergency.
* Be careful when using external contracts in a payable function, as they may have their own security vulnerabilities.