Smart Contract

A smart contract is a self-executing program deployed on a blockchain, such as Ethereum, that automatically enforces and executes the terms of an agreement when predefined conditions are met. These contracts are written in Solidity, a high-level programming language designed for the Ethereum Virtual Machine (EVM).

Key Properties of Smart Contracts

- Immutability: Once deployed, the code cannot be altered, ensuring consistent behavior.
- Deterministic: Given the same inputs, the contract will always produce the same outputs, maintaining reliability across the network.
- Permissionless: Anyone can deploy and interact with smart contracts, fostering an open and inclusive ecosystem.
- Composable**:** Smart contracts can interact with and build upon each other, enabling complex decentralized applications.

In Solidity, **value types** are types that store the actual data directly in memory. When you assign one value type to another, a copy of the value is made, and they do not share memory.

Key Value Types in Solidity:

1. Integer Types:
    - uint: Unsigned integer (non-negative numbers), e.g., uint256.
    - int: Signed integer (can be negative or positive), e.g., int256.
2. Boolean:
    - bool: Represents true or false.
3. Address:
    - address: Represents an Ethereum address, typically used for storing addresses of contracts or users.
4. Fixed-Size Byte Arrays:
    - bytes1 to bytes32: Fixed-size byte arrays where bytes1 represents 1 byte and bytes32 represents 32 bytes.
5. Enums:
    - enum: A custom type that allows you to define a collection of constants. For example, you can create a type to represent directions, statuses, or roles.
6. Structs (Can be value types when passed by value):
    - struct: A custom data type that groups multiple variables of different types. When structs are passed by value, they act like value types, meaning a copy of the struct is created.

Characteristics of Value Types:

- **Pass-by-Value**: When value types are passed to functions or assigned to other variables, a copy is made, meaning changes to the new variable do not affect the original.
- **Stack Storage**: Most value types are stored in the stack, which is cheaper and faster to access compared to the heap.

Solidity Functions

In Solidity, functions are essential building blocks for contracts. They define the logic and behavior of smart contracts.

Key Concepts of Solidity Functions

1. Visibility:
   - public: Accessible both internally and externally.
   - external: Accessible only externally.
   - internal: Accessible within the contract and derived contracts.
   - private: Accessible only within the contract.
2. State Mutability:
   - pure: Does not read or modify the blockchain state.
   - view: Reads the state but does not modify it.
   - payable: Can accept Ether (ETH).
   - Non payable: Cannot accept Ether.
3. Function Modifiers: Control function behavior (e.g., access control, validation).
4. Function Arguments & Returns: Functions can take parameters and return values.
5. Overloading: Functions can share the same name but require different parameters.
6. Function Calls: Can be called internally or externally (via transactions).

Basic Solidity Data Types

1. bool (Boolean)
2. int (Signed Integer)
3. String

Supplementary Resources

- Presentations: https://github.com/alchemyplatform/learn-solidity-presentations
- Marp Tool: https://marp.app/
- Foundry: https://book.getfoundry.sh/
- An awesome interactive resource for understanding EVM opcodes: https://www.evm.codes/