Blockchain Study Notes Day 7:

Module 2 - Solidity Basics
Chapter 3 - Immutable Variables in Solidity

Introduction to Immutable Variables

Immutable variables in Solidity are similar to constants but differ in the way they are initialized. While constants are set at compile time, immutable variables are set at deployment time and remain unchanged afterward.

---

1. What Are Immutable Variables?

- **Definition**:
  Immutable variables are declared using the `immutable` keyword. Their value is assigned only once, typically in the constructor, and cannot be changed thereafter.
- **Purpose**:
  They provide flexibility by allowing a value to be determined at deployment while still ensuring immutability during contract execution.

Key Characteristics:

- Value is set at deployment.
- Cannot be modified after being set.
- More gas-efficient than state variables but slightly less so than constants.

---

2. Syntax for Immutable Variables

Declaration:

```
uint public immutable myImmutableVariable;

constructor(uint _value) {
    myImmutableVariable = _value;
}
```

---

3. Difference Between `constant` and `immutable`

Feature	`constant`	`immutable`

Feature	`constant`	`immutable`
Value Assignment	At compile time	At deployment time
Modifiable	No	No
Gas Efficiency	Most efficient	Less efficient than `constant`
Usage Flexibility	Fixed at code level	Set dynamically during deployment

---

4. Example Program Using Immutable Variables

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

contract ImmutableExample {
    // Immutable variable
    address public immutable owner;

    // Constructor to set the immutable variable
    constructor() {
        owner = msg.sender;  // Assigns the deployer's address as the owner
    }

    // Function to return the owner's address
    function getOwner() public view returns (address) {
        return owner;
    }
}
```

Explanation:

1. **`owner`**: Declared as an immutable variable, storing the address of the contract deployer.
2. **`constructor`**: Sets the value of `owner` during contract deployment.
3. **`getOwner`**: A function to retrieve the value of the immutable variable.

---

5. Benefits of Immutable Variables

- **Gas Optimization**: Cheaper than regular state variables.
- **Deployment Flexibility**: Value can be dynamically assigned at deployment.
- **Security**: Immutable variables enhance security by preventing accidental or malicious modifications.

Home Task

1. **Modify the Example Program**:

    - Add an immutable variable for deployment timestamp (`uint public immutable deploymentTime`).
2. **Write a New Program**:
    - Create a contract that uses an immutable variable to store an admin address, with functions to check if the caller is the admin.
3. **Research**:
    - Compare the gas costs of using `immutable` vs. `constant` vs. regular state variables.

Conclusion

Immutable variables provide a balance between flexibility and efficiency in Solidity. They allow developers to set values dynamically at deployment while ensuring immutability during the contract's lifetime. By mastering their use, developers can create secure and optimized smart contracts.

Day 7 Notes

Prepared by Munawar Johar