Blockchain Study Notes Day 8:

Module 2 - Solidity Basics
Chapter 4 - Functions in Solidity

Introduction to Functions

Functions are the building blocks of smart contracts in Solidity. They define the behavior of the contract and allow interactions with its data. Solidity supports various types of functions for different use cases.

---

1. Types of Functions in Solidity

1.1. View and Pure Functions

- **View Functions**:
  - Read data from the blockchain but do not modify it.
  - **Example**:

```
function getName() public view returns (string memory) {
    return name;
}
```

- **Pure Functions**:
  - Neither read nor modify the blockchain state.
  - Used for computations.
  - **Example**:

```
function addNumbers(uint a, uint b) public pure returns (uint) {
    return a + b;
}
```

1.2. State-Changing Functions

- Modify the blockchain's state (e.g., update variables, transfer funds).
- **Example**:

```
function setName(string memory _name) public {
    name = _name;
}
```

1.3. Payable Functions

- Special functions that can receive Ether.
- **Example**:

```
function receiveFunds() public payable {
    balance += msg.value;
}
```

1.4. Constructor Functions

- Called once during contract deployment to initialize state variables.
- **Example**:

```
constructor(string memory _name) {
    name = _name;
}
```

---

2. Function Modifiers

- **Public**: Accessible from within and outside the contract.
- **Private**: Accessible only within the contract.
- **Internal**: Accessible within the contract and derived contracts.
- **External**: Accessible only from outside the contract.

---

3. Example Program with Functions (Using Munawar)

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

contract MunawarFunctions {
    string public name;
    uint public balance;

    // Constructor to initialize the name
    constructor(string memory _name) {
        name = _name;
    }

    // View function to get the name
    function getName() public view returns (string memory) {
        return name;
    }

    // State-changing function to set a new name
    function setName(string memory _newName) public {
        name = _newName;
    }

    // Payable function to receive Ether
```

```
    function receiveFunds() public payable {
        balance += msg.value;
    }

    // Pure function for calculation
    function multiply(uint a, uint b) public pure returns (uint) {
        return a * b;
    }
}
```

Explanation:

1. **Constructor**: Sets the initial name value during deployment.
2. **`getName`**: A `view` function to return the current name.
3. **`setName`**: A `public` function to update the name.
4. **`receiveFunds`**: A `payable` function to accept Ether and update the balance.
5. **`multiply`**: A `pure` function to multiply two numbers.

Home Task

1. **Add a Function**:
   - Write a function `isOwner` that checks if the caller is the contract owner.
2. **Implement a Withdraw Function**:
   - Create a function `withdrawFunds` to allow the owner to withdraw the balance.
3. **Experiment with Function Visibility**:
   - Use `private`, `internal`, and `external` modifiers in new functions to observe their accessibility.

Conclusion

Functions are central to Solidity, enabling developers to define contract behavior and interact with blockchain data. By mastering different types of functions and their modifiers, you can create robust and versatile smart contracts.

Prepared by Munawar Johar