Best Practices & Examples

Web3 Development with Solidity

Asuma Yamada

Profile

Asuma Yamada

Sample Inc.

GitHub: **@posaune0423**
𝕏: **@0xasuma**

- 📅 2024 - Present 🌏 **PixeLAW**
  - Autonomous World on Starknet
- 📅 2023 -2024 💎 **Unikura**
  - RWA NFT Marketplace
- 📅 2021 - 2023 👀 **VWBL Protocol**
  - Decentralized Access Control Protocol

I thought what I'd do was, I'd pretend I was one of those deaf-mutes

Table of Contents

Basic Smart Contract Structure

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract SimpleStorage {
    uint256 private value;

    event ValueChanged(uint256 newValue);

    function setValue(uint256 _value) public {
        value = _value;
        emit ValueChanged(_value);
    }

    function getValue() public view returns (uint256) {
        return value;
    }
}
```

Advanced Pattern: Diamond Pattern

Key Points

- Modular contract design
- Upgradeable components
- Gas efficient
- EIP-2535 standard

```
contract Diamond {
    bytes32 constant DIAMOND_STORAGE_POSITION =
        keccak256("diamond.storage");

    struct DiamondStorage {
        mapping(bytes4 => address) facets;
    }

    function diamondStorage() internal pure
        returns (DiamondStorage storage ds) {
        bytes32 position = DIAMOND_STORAGE_POSITION;
        assembly {
            ds.slot := position
        }
    }
}
```

Common Security Patterns

Checks-Effects-Interactions

1. Check preconditions
2. Update state
3. Interact with other contracts

Re-entrancy Guard

```
modifier nonReentrant() {
    require(!locked, "Reentrant call");
    locked = true;
    _;
    locked = false;
}
```

Access Control

```
contract Ownable {
    address private _owner;

    modifier onlyOwner() {
        require(msg.sender == _owner,
            "Caller is not owner");
        _;
    }

    function transferOwnership(
        address newOwner
    ) public onlyOwner {
        require(newOwner != address(0));
        _owner = newOwner;
    }
}
```

Testing with Hardhat

```
import { expect } from 'chai'
import { ethers } from 'hardhat'

describe('SimpleStorage', function () {
  it('Should store and retrieve value', async function () {
    const SimpleStorage = await ethers.getContractFactory('SimpleStorage')
    const storage = await SimpleStorage.deploy()
    await storage.deployed()

    await storage.setValue(42)
    expect(await storage.getValue()).to.equal(42)
  })
})
```

Real-world Examples

VWBL Protocol

- NFT Access Control
- Encryption/Decryption
- On-chain Verification

PixeLAW

- Autonomous World
- Game Logic
- State Management

Thank you!

GitHub: **@posaune0423**
𝕏: **@0xasuma**