Solidity Patterns

A compilation of patterns and best practices for the smart contract programming language Solidity

View on GitHub

Solidity Patterns

This document contains a collection of design and programming patterns for the smart contract programming language Solidity in version 0.4.20. Note that newer versions might have changed some of the functionalities. Each pattern consists of a code sample and a detailed explanation, including background, implications and additional information about the patterns.

Contents

- **Behavioral Patterns**
  - **Guard Check**: Ensure that the behavior of a smart contract and its input parameters are as expected.
  - **State Machine**: Enable a contract to go through different stages with different corresponding functionality exposed.
  - **Oracle**: Gain access to data stored outside of the blockchain.
  - **Randomness**: Generate a random number of a predefined interval in the deterministic environment of a blockchain.
- **Security Patterns**
  - **Access Restriction**: Restrict the access to contract functionality according to suitable criteria.
  - **Checks Effects Interactions**: Reduce the attack surface for malicious contracts trying to hijack control flow after an external call.
  - **Secure Ether Transfer**: Secure transfer of ether from a contract to another address.
  - **Pull over Push**: Shift the risk associated with transferring ether to the user.
  - **Emergency Stop**: Add an option to disable critical contract functionality in case of an emergency.
- **Upgradeability Patterns**
  - **Proxy Delegate**: Introduce the possibility to upgrade smart contracts without breaking any dependencies.
  - **Eternal Storage**: Keep contract storage after a smart contract upgrade.
- **Economic Patterns**
  - **String Equality Comparison**: Check for the equality of two provided strings in a way that minimizes average gas consumption for a large number of different inputs.
  - **Tight Variable Packing**: Optimize gas consumption when storing or loading statically-sized variables.
  - **Memory Array Building**: Aggregate and retrieve data from contract storage in a gas efficient way.

Bibliography

The sources used in this document can be found in this bibliography.

Disclaimer

This repository is not under active development anymore and some (if not most) sections might be outdated. There is no liability for any damages caused by the use of one of these patterns.

---

solidity-patterns is maintained by fravoll.
This page was generated by GitHub Pages.