Solidity Advanced（進階語法）

智能合約進階技巧 · 小學生也能懂的版本

Mappings（對照表）

一句話給小學生

Mapping 就是：**用一個東西，快速找到另一個東西**。

概念🄵明

Mapping 就像：

- 通訊🄵（名字 → 電話）
- 成績表（學生 → 分數）

在 Solidity 🄵常用來「地址 → 資料」。

基本 Mapping 範例（含中文🅵解）

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract Mapping {
    // 建立一個 mapping: 地址 => 數字
    mapping(address => uint) public myMap;

    function get(address _addr) public view returns (uint) {
        // Mapping 一定會回傳一個值
        // 如果從來🅵設定過，會回傳預設值
        // uint 的預設值是 0
        return myMap[_addr];
    }

    function set(address _addr, uint _i) public {
        // 設定或更新這個地址對應的數值
        myMap[_addr] = _i;
    }

    function remove(address _addr) public {
        // 🅵除後會回到預設值 (0)
        delete myMap[_addr];
    }
}
```

小學生重點

Mapping 🅵有「存在 / 不存在」，只有「現在是多少」。

Nested Mapping（巢狀 Mapping）

一句話

Mapping 🅵面，還可以再放 Mapping。

巢狀 Mapping 範例（含中文🅵解）

```
contract NestedMappings {
    // 地址 => (數字 => 布林值)
    mapping(address => mapping(uint => bool)) public nestedMap;

    function get(address _addr, uint _i) public view returns (bool) {
        // 就算從來🅵設定過，也可以直接讀
        // bool 的預設值是 false
        return nestedMap[_addr][_i];
    }

    function set(address _addr, uint _i, bool _boo) public {
        nestedMap[_addr][_i] = _boo;
    }

    function remove(address _addr, uint _i) public {
        delete nestedMap[_addr][_i];
    }
}
```

Enums（列舉型🅵）

小學生版

Enum = **只能從固定幾個選項🅵選一個**

🅵什🅵要用 Enum？

- 避免亂填數字
- 程式更好讀
- 限制狀態範圍

Enum 範例（含中文🅴解）

```
contract EnumExample {
    // 定義一組狀態
    enum Status {
        Pending,
        Shipped,
        Accepted,
        Rejected,
        Canceled
    }

    // 宣告一個狀態變數
    Status public status;

    function get() public view returns (Status) {
        // Enum 其實是 uint
        return status;
    }

    function set(Status _status) public {
        status = _status;
    }

    function cancel() public {
        // 直接指定某個 Enum 成員
        status = Status.Canceled;
    }
}
```

小學生記法

Enum 就像選單，不能自己亂寫答案。

Structs（自訂資料包）

一句話

Struct = **把一堆相關資料綁在一起**

Struct 範例（含中文🅵解）

```
contract TodoList {
    // 定義一個待辦事項結構
    struct TodoItem {
        string text;
        bool completed;
    }

    // 存放多個 TodoItem
    TodoItem[] public todos;

    function createTodo(string memory _text) public {
        // 方法一：直接當函式呼叫
        todos.push(TodoItem(_text, false));

        // 方法二：指定欄位名稱
        todos.push(TodoItem({ text: _text, completed: false }));

        // 方法三：先建立，再一個一個設定
        TodoItem memory todo;
        todo.text = _text;
        todo.completed = false;
        todos.push(todo);
    }

    function toggleCompleted(uint _index) public {
        // 切🅵完成狀態
        todos[_index].completed = !todos[_index].completed;
    }
}
```

View 與 Pure 函式

差🅵一句話

- view：**只看，不改**
- pure：**不看，也不改**

View / Pure 範例（含中文🅵解）

```
contract ViewAndPure {
    uint public x = 1;

    // 可以讀取狀態，但不能修改
    function addToX(uint y) public view returns (uint) {
        return x + y;
    }

    // 完全不碰狀態
    function add(uint i, uint j) public pure returns (uint) {
        return i + j;
    }
}
```

Modifiers（修飾器）

小學生版

Modifier = **先檢查，才放行**

Modifier 範例（含中文🅵解）

```
contract Modifiers {
    address public owner;

    constructor() {
        // 部署合約的人就是 owner
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "你不是擁有者");
        // _ 代表「執行原本的函式」
        _;
    }

    function changeOwner(address _newOwner) public onlyOwner {
        owner = _newOwner;
    }
}
```

Events（事件）

一句話

Event = **上鏈的公告紀🅵**

Event 範例（含中文🅵解）

```
contract Events {
    // 宣告事件
    event TestCalled(address sender, string message);

    function test() public {
        // 發出事件
        emit TestCalled(msg.sender, "有人呼叫 test()");
    }
}
```