

Smart Contract 2

- 陳博宇 @ 東吳大學 -

目錄

- 1 上週回顧 + HW 1解答
- 2 合約進階介紹及注意事項
- 3 ERC20介紹與實作
- 4 ERC 721 介紹與實作

```
pragma solidity ^0.6.0;

contract MuscleMan {
    string TA;

    function set() public {
        TA = "BillHsu";
    }

    function get() public view returns (string memory) {
        return TA;
    }
}
```

- 上週回顧 + HW 1解答 -

變數宣告

● 變數型態

- ☐ bool
- ☐ int / uint
- ☐ bytes
- ☐ address
- ☐ string
- ☐ array
- ☐ mapping

+

● 能見度

- ☐ public
- ☐ private
- ☐ internal

+

● 變數名稱

```
int8 public age;  
bool private isOwner;  
string name;
```

函數宣告

● 函數名稱(參數) + ● 能見度 + ● 回傳值

○ public

○ private

○ internal

○ external ← **this.funtion()**

```
function funName() private {...}  
function funName2(uint num) external returns(uint8) {...}  
function deposit() public payable {...}
```

calldata v.s memory

- 如果函數參數是 **struct** 、 **array (string => bytes[])** 、 **mapping** 時
 - **calldata** => **external**
 - **memory** => **public**
 - **storage** 、 **memory** => **internal**
- 在函數回傳參數中
 - **memory** => **public** 、 **external**
 - **storage** 、 **memory** => **internal**

函數宣告

● View function

● Pure function

- 不改變合約狀態
- 函數執行不消耗 gas
- 不需經過礦工驗證

```
function viewFun(uint256 a, uint256 b) public view returns (uint256) {  
    return a * (b + 42) + now;  
}  
  
function pureFun(uint256 a, uint256 b) public pure returns (uint256) {  
    return a * (b + 42);  
}
```

- 合約進階介紹及注意事項 -

Struct

- 自定義變數型態
- mapping 也可以用 (**only value**)
- 可複製，但是 mapping 部分無法

```
struct 變數名稱 {  
    成員型態 成員變數名稱;  
    成員型態 成員變數名稱;  
}
```

Struct 練習

```
contract StructExample1 {
    struct Student {
        string studentName;
    }

    Student student;

    function setStudent(string calldata studentName) external {
        student.studentName = studentName;
    }

    function getStudent() external view returns (string memory) {
        return student.studentName;
    }
}
```

Struct 練習 — mapping

```
contract StructExample2 {
    struct student {
        string studentId;
        string studentName;
    }

    mapping(address => student) public studentAdd;

    function addStudent(string studentId, string studentName) {
        studentAdd[msg.sender] = student(studentId, studentName);
    }
}
```

Struct 練習 — mapping 複製

```
contract StructExample3 {
    struct student {
        address studentAdd;
        mapping(string => string) idToName;
    }

    student student1;
    student student2;

    function setStudent() public{
        student1.idToName["0612221"] = "Gaga";
        student1.studentAdd = msg.sender;
        student2 = student1;
    }
}
```

Event

- 合約內部函數觸發
- 額外的儲存空間，很便宜
- 將觸發參數存進 log 中
- 方便 DAPP 監聽事件
- Contract 無法直接取 log 的資料
- 搭配 **emit** 使用

```
event 事件名稱( 參數型態1 參數名稱1, 參數型態2 參數名稱2, ... );
```

Event 練習

```
contract EventExample {  
    event buyer(string buyerName, address buyerAdd);  
  
    function register(string calldata name, address add) external {  
        emit buyer(name, add);  
    }  
}
```

Function Modifiers

- 提供函數執行前的檢查、預先處理。
- 支援繼承屬性
- 可接收參數
- 可被覆寫
- 可以多個 modifier

```
modifier 名稱() {  
    條件檢查式; //可以有很多個  
    _;  
}
```

Function Modifiers 練習

```
contract FunctionModifiers {
    address payable owner;

    modifier isOwner() {
        require(msg.sender == owner);
        _;
    }

    function kill() public isOwner {
        selfdestruct(owner);
    }
}
```


Inheritance

- 支援多重繼承
- **virtual** 函數在繼承中可以被修改
- 要修改 **virtual** 函數，要先宣告 **override**

```
contract Shop is Owned {
    string shopName;

    function setName(string calldata _shopName) external virtual {
        if (msg.sender == owner)
            shopName = _shopName;
    }
}
```

```
contract Owned {
    address payable owner;

    constructor() public {
        owner = msg.sender;
    }
}
```

Inheritance

```
contract Shop is Owned {
    string shopName;

    function setName(string calldata _shopName) external virtual {
        if (msg.sender == owner)
            shopName = _shopName;
    }
}

contract Mall is Owned, Shop {
    string[] shopArr;

    function setName(string calldata _shopName) external override {
        if (msg.sender == owner) {
            shopName = _shopName;
            shopArr.push(_shopName);
        }
    }
}
```

Abstract Contract

- 合約內有至少一個函數未實現
- 未實現的合約要宣告為 `virtual`
- 如果繼承者也未完全實現全部函數，也應宣告為 `abstract`

```
abstract contract Abstract {  
    uint8 num;  
    function setNum(uint8) public virtual;  
    function getNum() public view returns (uint8) {  
        return num;  
    }  
}
```

Interface

- 與 abstract contract 相似，但它沒有實現任何函數
- 不能繼承其他合約或介面
- 不能定義 **constructor**、**變數**，但是 **struct**、**enum** 可以
- 所有的 function 必須是 **external**，且預設都是 **virtual**
- 就像是蓋房子前的藍圖

```
interface Member {  
    function setName(string calldata) external;  
    function setAge(uint8) external;  
    function getInfo() external returns (string memory, uint8);  
}
```

Library

- 函式庫合約不能有狀態儲存
- 不能被 destroyed
- 不能繼承或被繼承
- 不能接受 Ether

[OpenZeppelin](#)

```
library Set {  
    function insert(Data storage self, uint value) public returns (bool) {...}  
    function remove(Data storage self, uint value) public returns (bool) {...}  
    function contains(Data storage self, uint value) public returns (bool){...}  
}
```

Enum

- 自定義變數、不需要指定型態
- 裡面的值依照宣告的順序從零開始遞增
- 可搭配 Struct 使用，表示 Struct 中的某個狀態

```
enum 變數名稱 { 成員1, 成員2, ... }
```

Enum 練習

```
contract Enum {
    struct clothes{
        uint8 clothesNum;
        size clotheSize;
    }

    enum size{large, medium, small}

    clothes public tShirt;

    function set(uint8 num, size _size) public {
        tShirt.clothesNum = num;
        tShirt.clotheSize = _size;
    }
}
```

Getter Function

- 對於所有 public 變數宣告後會自動生成
- Getter function 為 external

```
contract Getter {  
    uint8 public num = 10;  
  
    function getNum() public view returns (uint8) {  
        return num;  
    }  
  
    function getNum2() public view returns (uint8) {  
        return this.num();  
    }  
}
```


Function Overloading

```
function fun1(uint8 num1) external pure returns (uint8) {  
    return num1;  
}  
  
function fun1(uint8 num1, uint8 num2) external pure returns (uint8) {  
    return num1 + num2;  
}
```

Function Multiple Returns

```
function multipleReturn() external view returns (uint256, bool, address) {  
    return (block.number, block.number % 2 == 0, msg.sender);  
}
```

- ERC 20 介紹與實作 -

- ERC 721 介紹與實作 -
