

I. Hello World

. Sử dụng Framework Remix (công cụ của ethereum, như 1 playground để hỗ trợ viết smartcontract)

- . Workspace // nơi tạo ra các màn hình làm việc
- . contract // nơi chứa các hợp đồng dạng file .sol
- . test // lưu trữ các test case
- . Trong lúc gõ có lỗi Remix sẽ hỗ trợ compile realtime
- . Solodity compiler
 - . Autocompile // tự động biên dịch khi gõ
 - . Hide warning // giấu các cảnh báo
- . Deploy n run transaction
 - . Enviroment // môi trường test EVM
 - . Account // tài khoản để test
 - . Contract // hợp đồng muốn test

II. Biến và kiểu dữ liệu

- . Variable - Biến (giống các ngôn ngữ static khác)
- . User Define
 - . bool // true/false
 - . uint // chữ số
 - . address // 1 chuỗi dạng 0x.....
 - . bytes32 // chỉ chứa ký tự binary độ lớn 32 byte
 - > thường dùng để lưu string vì smartcontract xử lý string không tốt
 - . bytes // giống bytes 32, nhưng lưu bao nhiêu sẽ chiếm bấy nhiêu bộ nhớ
 - . mapping(a --> b) // lưu giá trị a vào b
 - . struct User
 - {
 - uint id;
 - string name;

```

        bool isFriend;
    }
    . enum Color
    {
        red,
        green,
        blue
    }
    . Build in
    . msg.sender //người gọi hàm
    . msg.value //giá trị hàm gọi đó

```

III.Hàm và Contructor

```

. Hàm
. Định nghĩa tên hàn bằng function ____ (type parameter)
. returns() // định nghĩa kiểu dữ liệu trả về
. return // trả về dữ liệu
. parameter //tham số truyền vào tham số
. type // kiểu dữ liệu của tham số
. Contructor
. Là hàm đặc biệt chỉ chạy 1 lần khi deploy smartcontract

```

IV.Functions, variable visibility

```

. Visibility (hàm và biến tương tự nhau)
. private // chỉ được gọi trong smartcontract
. internal // có thể gọi trong smartcontract này hoặc các smartcontract khác được enharit với nó
. external // chỉ được gọi từ bên ngoài, bên trong không được gọi
. public // ở đâu cũng gọi được

```

- . với biến mặc định là private nếu không khai báo
- . private trong biến chỉ trong smartcontract này chứ không phải với toàn blockchain
- . Chỉ cần biết biến deploy ở đâu và đọc đúng slot thì vẫn xem được đó là biến gì
- . với biến để public thì không cần viết hàm get mà Solidity sẽ tự tạo ra hàm get

V. Control Structures

- . if else // check giá trị logic
- . for // duyệt từng thành phần cho đến khi hết logic
- . while // chạy vòng lặp khi điều kiện đúng
- > hạn chế sử dụng vòng lặp do nếu chạy nhiều quá sẽ khiến chi phí rất contract
- . nếu sử dụng vượt gas limit thì smartcontract sẽ không bao giờ được thực hiện

VI. Storage Array

- . storage Array
 - . Được lưu trữ trên blockchain
 - . Sẽ có bài toán duyệt array -> tốn phí gas
 - . Khai báo bên ngoài, trong contract
- . CRUD (create, read, update, delete)
 - . uint[] myArray
 - . myArray.push() // thêm phần tử trong mảng
 - . myArray[index] // trả về giá trị tại vị trí index
 - . myArray[index] = __ // gán về giá trị tại vị trí index
 - . delete myArray[index] // trả về giá trị mặc định 0 tại vị trí index
- . memory Array
 - . Lưu trong blockchain, không tốn gas
 - . Khai báo trong function, không được khai báo ở ngoài function
 - . cần khai báo bộ dài để sử dụng
 - . Cần xác định kiểu dữ liệu chứ không có quyền khai báo luôn

. CRUD

. uint[] memory newArray = new uint[](length)

. newArray [index] = __ //gán giá trị

. array as parameter

. function myFunction (uint[] calldata myArgs) external {}

. function myFunction (uint[] memory myArgs) internal {}

. function myFunction (uint[] memory myArgs) public {}

VII.Mapping

. Mapping giống literal object trong java script

. có key và value, access key để truy cập vào value

. {

 a : 1

 b : 2

 c : 3

}

. mapping(address => uint)balance; // khai báo 1 mapping

. balances[msg.sender] = value; // gán giá trị

. balances[msg.sender]; // lấy giá trị

. delete balances[msg.sender]; // xóa giá trị

. balances[KeyNotExist] = false; // giá trị mặc định

. Nested mappings

. mapping(address1 => mapping(address2 => bool)) approve;

. Address1 là địa chỉ của người gửi, address2 là địa chỉ người xài, sử dụng khi cấp phép cho người kia xài hay không

. approved[msg.sender][sender] // trả về true or false xem bên kia có được gửi tiền hay không

VII.Struct

. Struct sử dụng để lưu trữ nhiều biến có cấu trúc

```
. struct player
```

```
{  
    uint id;  
    address addres;  
    string Name;  
}
```

```
. player memory player1 = player(1, msg.sender, _name); // khởi tạo 1 biến dạng struct
```

```
. player memory player2 = player(2, msg.sender, _name); // thêm biến nữa, hai biến này độc  
lập hoàn toàn với nhau
```

```
. player1.Name; // trả về field name
```

```
. player1.Name = "___"; // update field name
```

```
. delete player2; // xóa biến
```

```
. player.push(player({ID:1, address: msg.sender, Name : "MyName" }));
```

```
. listofPlayer[msg.sender] = player1({})
```

VII.Event

. Event là sự kiện mà smartcontract đẩy ra ngoài cho các bên thứ 3 nắm bắt và xử lý

```
. function deposit() external
```

```
{  
    //transfer token  
    emit depossit(msg.sender, amount);  
}
```

VIII.Gửi/nhận Ethereum trên Smart Contracts

. Gửi

```
. function SendEther(address payable to, uint amount) external {  
    to.tranfer(amount);  
}
```

. //payable là khi gọi hàm đó có khả năng nhận được ether khi người ta gọi nó, có thể check được số dư của smartcontract

. address(this).balance //check số dư của smartcontract

. Nhận

. receive() external payable {} // hàm này có thể nhận ether

IX.Error handling & Modifiers

. Handle lỗi

. Khi lỗi các biến sẽ revert về giá trị ban đầu nhưng vẫn tốn gas

. required sẽ kiểm tra các điều kiện trước khi chạy sang lệnh tiếp theo

. required(a == value, "Log");

. Modifiers

. Sử dụng khi có nhiều required, giúp clean code khi không cần lặp đi lặp lại require

. modifier CheckA

{

 required(a == value, "Log");

} //chạy hết các hàm modify mới chạy đến function kia

. có thể truyền tham số của hàm vào modifier

X.Inheritance: Tính thừa hưởng trong solidity

. 1 smartcontract có thể thừa hưởng từ một smartcontract khác bằng cách import file

. import "./helloworld.sol"

 contract Bonjour is Hello {}

. chỉ sử dụng được với hàm public

. Solodity hỗ trợ multiple Inheritance (1 smartcontract có thể kế thừa từ nhiều smartcontract khác)

. trong constructor 1 smartcontract có thể gọi constructor của smartcontract mà nó kế thừa