第2章 智能合约概述

目录

CONTENTS

腾讯云

CONTENTS

章节

第四章 智能合约结构

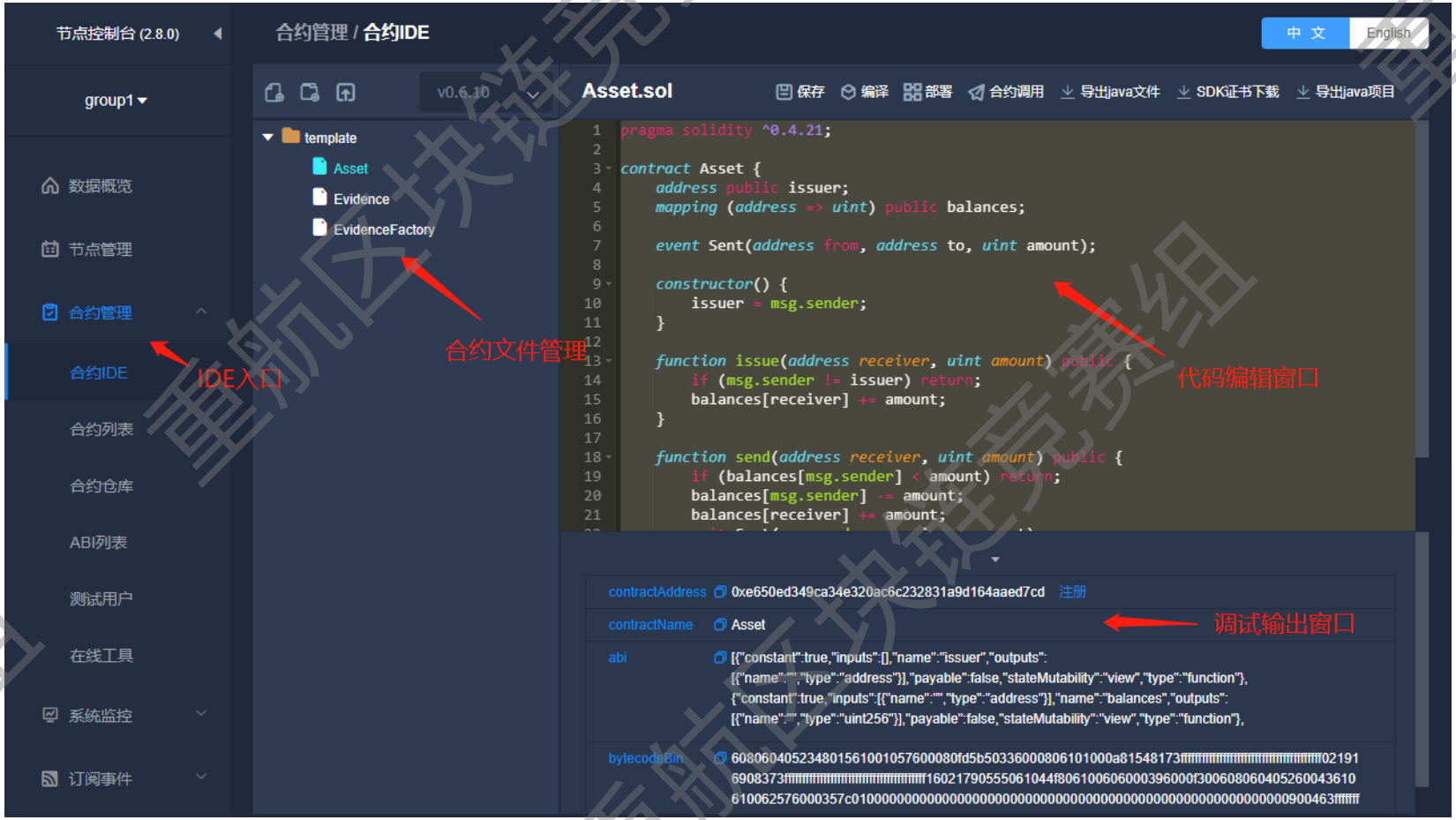
第四章 智能合约结构

CONTENTS

章节

智能合约IDE

- 选择“合约管理”-“合约IDE”菜单，进入合约IDE界面



智能合约IDE

- EvidenceFactory.sol文件示例

EvidenceFactory.sol

```
pragma solidity ^0.4.4;
import "Evidence.sol";

contract EvidenceFactory{
        address[] signers;
        event newEvidenceEvent(address addr);
        function newEvidence(string evi)public returns(address)
        {
            Evidence evidence = new Evidence(evi, this);
            newEvidenceEvent(evidence);
            return evidence;
        }

        function getEvidence(address addr) public constant returns(string,address[],address[]){
            return Evidence(addr).getEvidence();
        }


        function addSignatures(address addr) public returns(bool) {
            return Evidence(addr).addSignatures();
        }

        constructor(address[] evidenceSigners){
            for(uint i=0; i<evidenceSigners.length; ++i) {
            signers.push(evidenceSigners[i]);
            }
```

Solidity文件结构

- 一个solidity文件由以下部分组成
  - pragma：版本标识，pragmatic information的简称
  - import：导入
  - comment：注释
  - contract：普通合约
  - library：库合约
  - interface：接口合约

Solidity文件结构

- pragma（版本标识）：不同的solidity版本语法可能不兼容，所以在文件开始指定solidity版本，方便编译器进行检查，版本标识只对当前文件有效，如果使用import导入其他版本solidity文件，版本标识不会从被导入的文件加入到导入的文件中
  - 版本号的形式通常是0.x.0和x.0.0
  - 例：^0.4.4表示sol文件允许使用0.4.4及以上版本编译，但不能大于0.5.0版本
- 本课程后续所有的代码案例都是使用“0.6.10”版本编写

Solidity文件结构

- import（导入）： solidity中能够导入其他solidity文件，更方便地对代码结构进行管理，降低文件之间的耦合度
  - import "filename"：把filename中所有全局符号导入当前作用域
  - 导入时可以指定从指定路径导入，如“./”是当前路径，“../”是上一级路径，这与Linux bash的路径类似

Test.sol

```
1  pragma solidity 0.6.10;
2
3  import "../template/Asset.sol";
```

Solidity文件结构

- 注释：Solidity支持单行注释和多行注释，便于阅读程序的人理解程序
  - 单行注释：使用“//”标识
  - 多行注释：使用“/**/”标识

Test.sol

```
1  pragma solidity 0.6.10;
2
3  import "../template/Asset.sol";
4
5  // 这是一个单行注释
6  /* 这是一个多行注释的单行写法 */
7  /*
8  这是一个
9  多行注释
10 */
```

Solidity文件结构

- contract（合约）：包括普通合约（contract）、库合约（library）、接口合约（interface），大部分需要完成的合约都是普通合约

Test.sol

```
1  pragma solidity 0.6.10;
2
3  contract Test {
4      // 这是一个普通合约
5  }
6
7  library TestLib {
8      // 这是一个库合约
9  }
10
11 interface ITest {
12     // 这是一个接口合约
13 }
```

合约结构

- 在 Solidity 语言中，合约类似于其他面向对象编程语言中的“类”。
- 合约由以下多个结构组成：
  - 全局变量
  - 枚举
  - 结构体
  - 函数
  - 构造函数
  - 修饰符
  - 事件

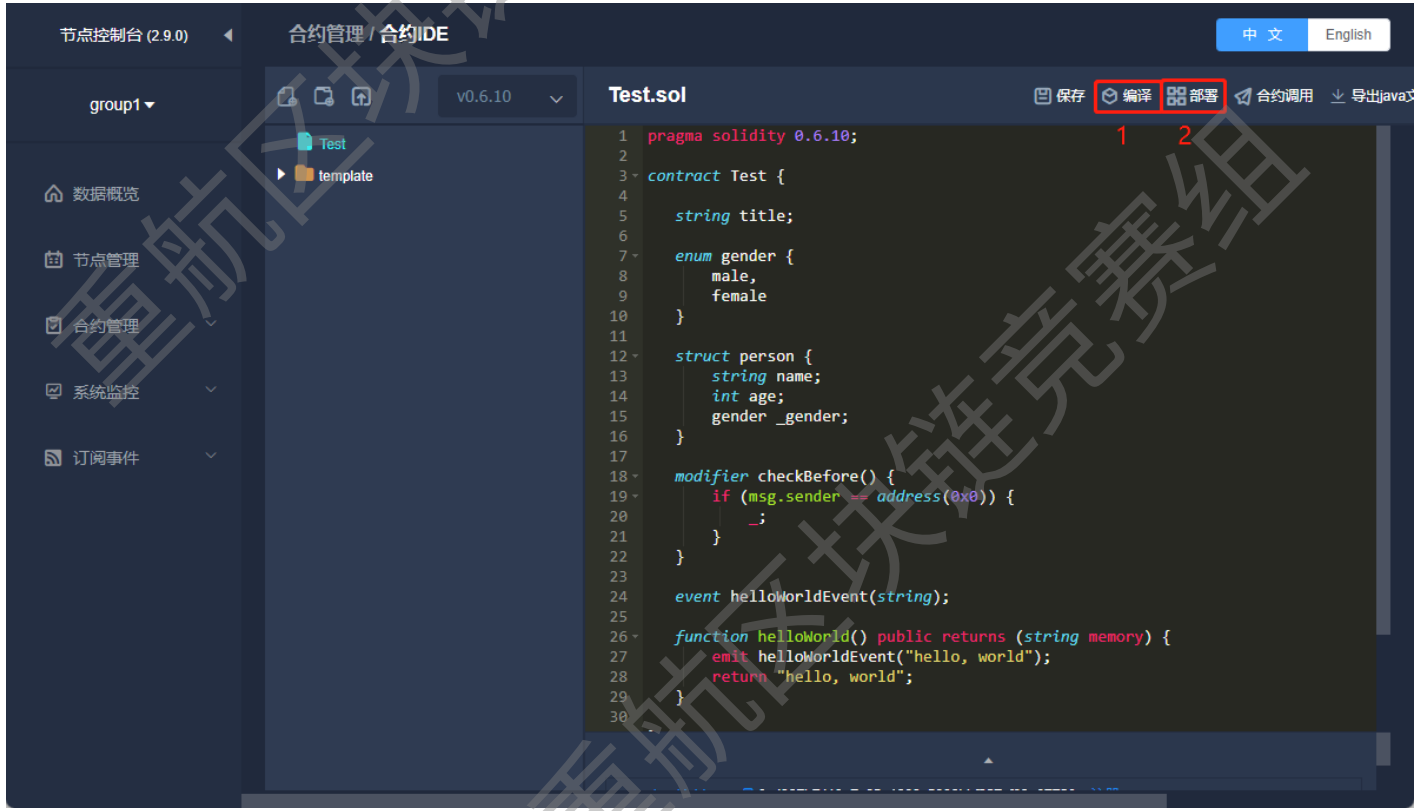
合约结构

- 合约示例

```
contract Test {
    string title;                          全局变量

    enum gender {                          枚举
        male,
        female
    }

    struct person {                        结构体
        string name;
        int age;
        gender _gender;
    }

    modifier checkBefore() {               修饰符
        if (msg.sender == address(0x0)) {
            _;
        }
    }

    event helloWorldEvent(string);         事件

    constructor() public {                 构造函数
        title = "test";
    }

    function helloWorld() public returns (string memory) {   函数
        emit helloWorldEvent("hello, world");
        return "hello, world";
    }
}
```

部署合约

- 部署智能合约，编写完智能合约后，依次点击“编译”“部署”按钮即可将智能合约部署在区块链上。



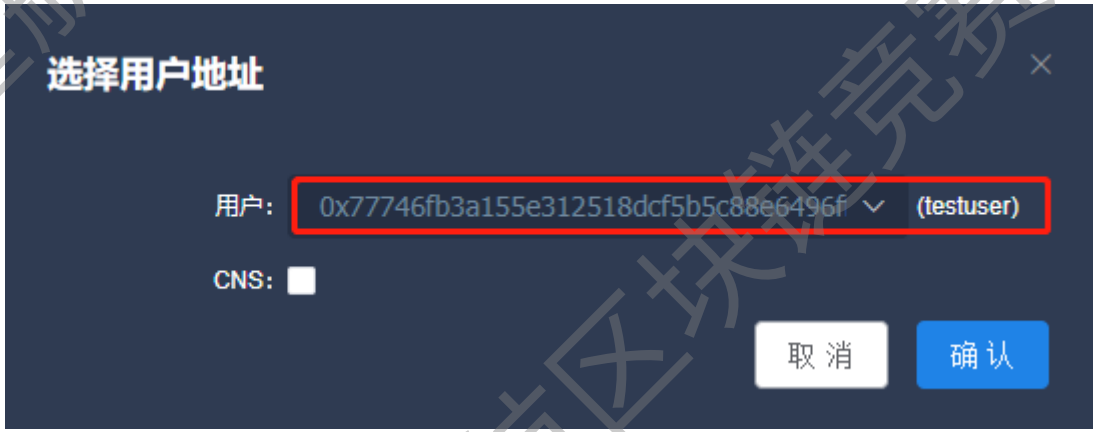
部署合约

- 部署时需要选择用户地址，在初次部署时需要新增用户，点击“新增用户”按钮，在“添加用户名”提示框中输入“testuser”，点击“确定” 按钮新增用户，新增完成后在弹出框中重新选择“testuser”用户

点击新增用户按钮

部署合约

添加用户



重新选择用户地址

部署合约

- 重新选择“testuser”用户部署合约，部署成功后在屏幕上方观察到“部署成功的提示框”，在界面下方观察到部署成功的合约地址

全局变量

- 在合约中，没有在函数内声明的变量称为全局变量，全局变量的值永久存储在当前合约中，全局变量的内存是静态分配的，在合约声明周期内，分配的内存大小不会改变。
- 全局变量可以添加相关限定符，限制变量的访问范围
  - `internal`：内部变量，默认类型，表示此变量只能在当前合约和继承当前合约的合约使用，不能被外部访问和修改
  - `private`：私有变量，在`internal`变量的基础上，即使是继承当前合约的合约，也不能使用该变量
  - `public`：公共变量，外部可以访问`public`类型变量
  - `constant`：静态变量，必须在声明时指定值，且值不可变

全局变量

- 全局变量示例

```
contract Test {
    string title; // 默认internal

    string internal title2; // internal

    string private title3; // private

    string public title4; // public

    string constant title5 = "test contract"; // constant
}
```

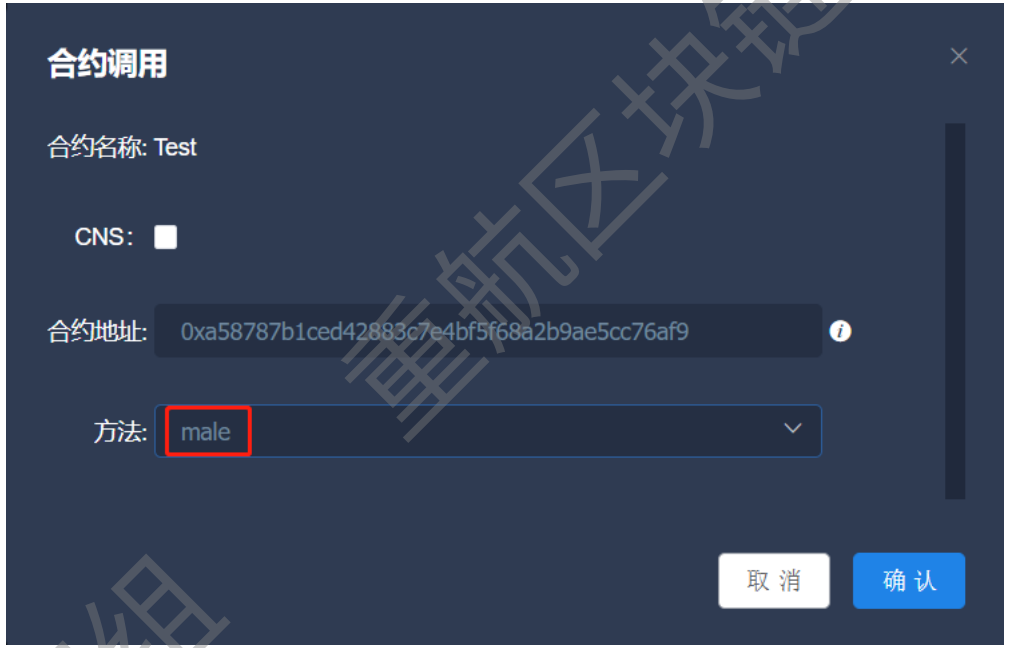
枚举

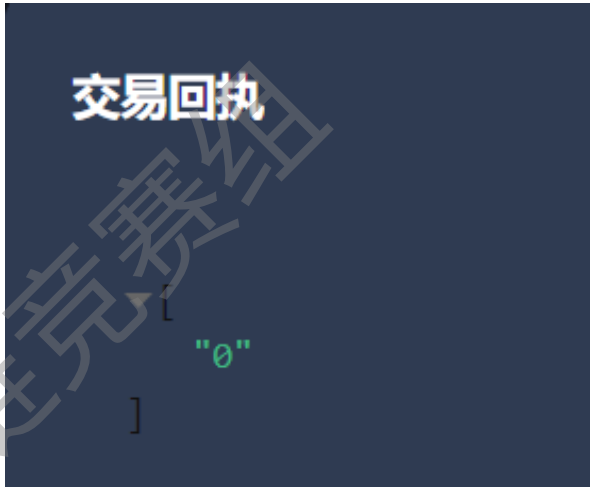
- 枚举可用来创建由一定数量的常量值构成的自定义类型，使用“enum”关键字声明，常量值从0开始

```
pragma solidity 0.6.10;

contract Test {
    // 定义枚举
    enum gender {
        male,
        female
    }

    // 定义全局变量male，值为枚举类型gender.male
    gender public male = gender.male;

    // 定义全局变量female，值为枚举类型gender.female
    gender public female = gender.female;

}
```

枚举

- 部署并调用合约，获取枚举值





枚举

- 部署并调用合约，获取枚举值

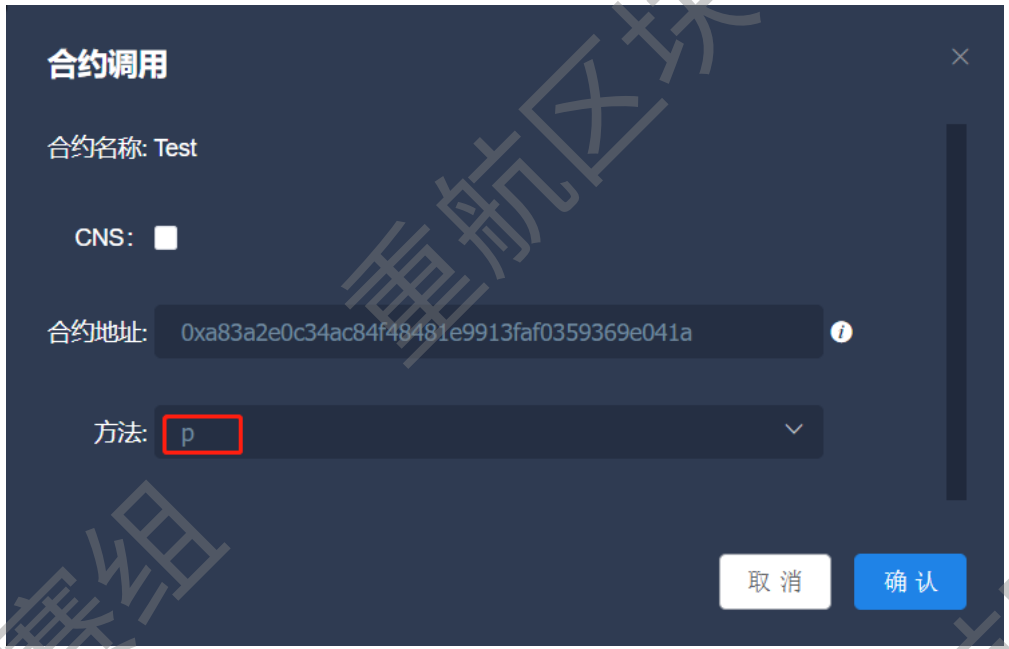
结构体

- 结构体是一种复合型数据类型，有多个不同数据类型变量组成，可以方便地实现自定义类型
- 结构体使用struct关键字声明，创建实例时使用“结构体名(参数...)”的方式

```
pragma solidity 0.6.10;

contract Test {

    // 声明person结构体，由name、age、gender三个成员组成
    struct person {
        string name;
        int age;
        string gender;
    }

    // 定义全局变量p，类型为person结构体，并初始化值
    person public p = person("jack", 18, "male");

}
```

结构体

- 部署并调用合约，获取结构体值



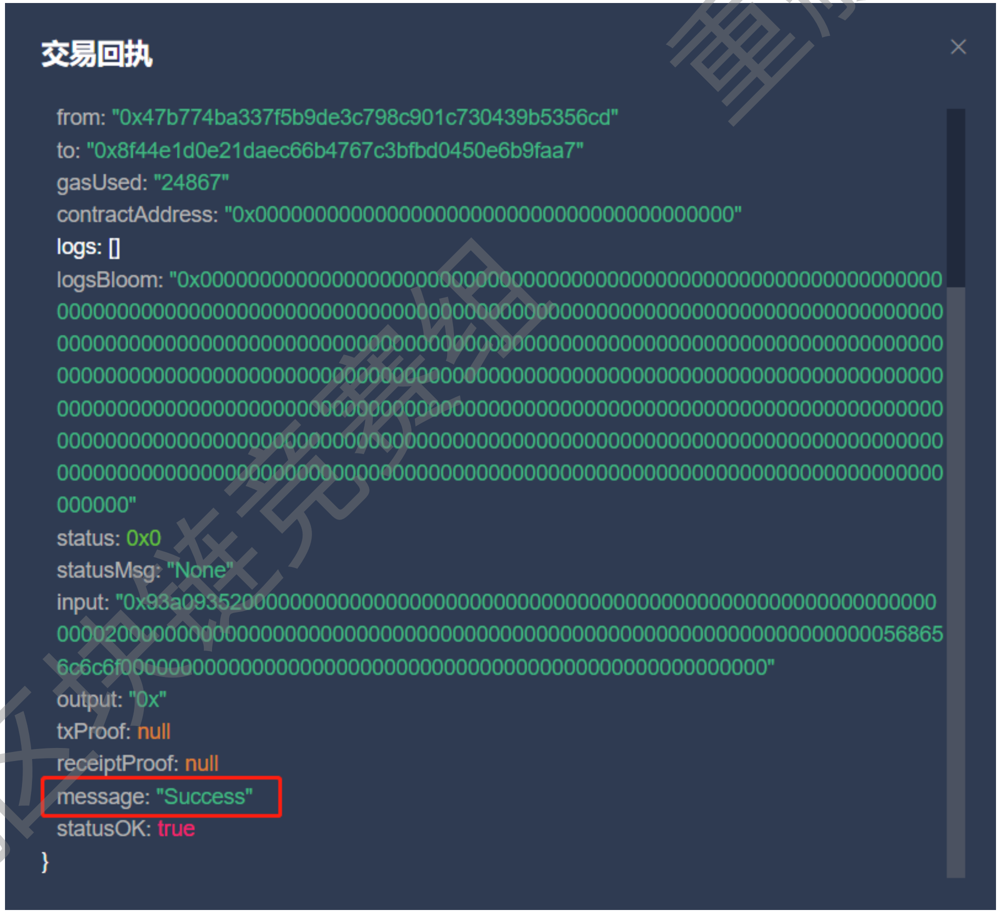
函数

- 函数是一个合约代码执行的最小单元，使用“function” 关键字创建，函数可以接收参数，执行逻辑，返回结果

```
contract Test {
    string value;

    // setValue函数的功能是设置value变量值
    function setValue(string memory v) public {
        value = v;
    }

    // getValue函数的功能是获取value变量值
    function getValue() public returns (string memory) {
        return value;
    }
}
```

函数

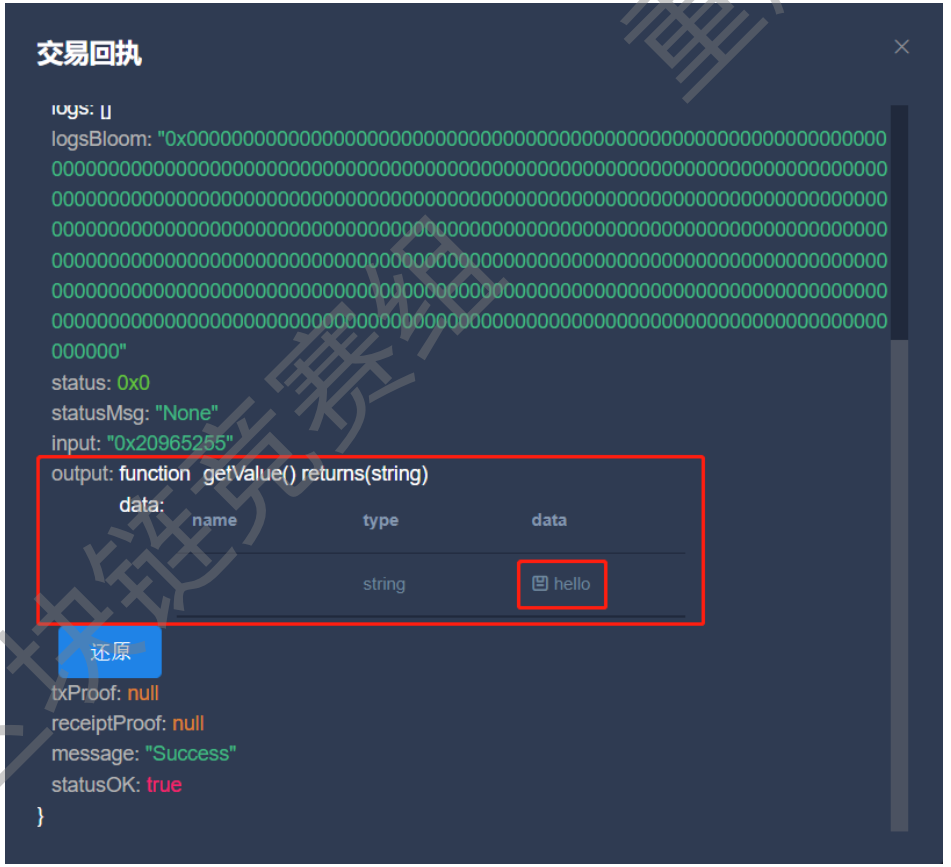
- 部署并调用合约，修改value变量值



腾讯云

函数

- 调用合约，获取修改后的value变量值



函数

- 与状态变量类似，函数也可以添加限定符
  - `internal`：函数只能在当前合约和继承当前合约的合约访问
  - `public`：函数可以直接从外部和内部访问
  - `private`：函数只能在当前合约中访问
  - `external`：函数可以在外部访问，但不能从内部访问
- 附加限定符，需结合限定符使用，限定合约全局变量的操作范围
  - `view`：函数只能读取合约全局变量，不能进行其他操作
  - `pure`：不能操作合约全局变量，包括读取

函数

- 函数限定符示例

```
contract Test {
    function internalFunc() internal {
    }

    function publicFunc() public {
    }

    function privateFunc() private {
    }

    function externalFunc() external {
    }

    function viewFunc() public view {
    }

    function pureFunc() public pure {
    }
}
```

函数

- 构造函数，在部署合约时运行的函数，使用constructor关键字声明

Test.sol

```
pragma solidity 0.6.10;

contract Test {

    string value;

    constructor(string memory v) public {
        value = v;
    }

    function getValue() public returns(string memory) {
        return value;
    }

}
```

函数

- 部署合约，设置value值，并调用getValue函数获取value值

函数

- 部署合约，设置value值，并调用getValue函数获取value值

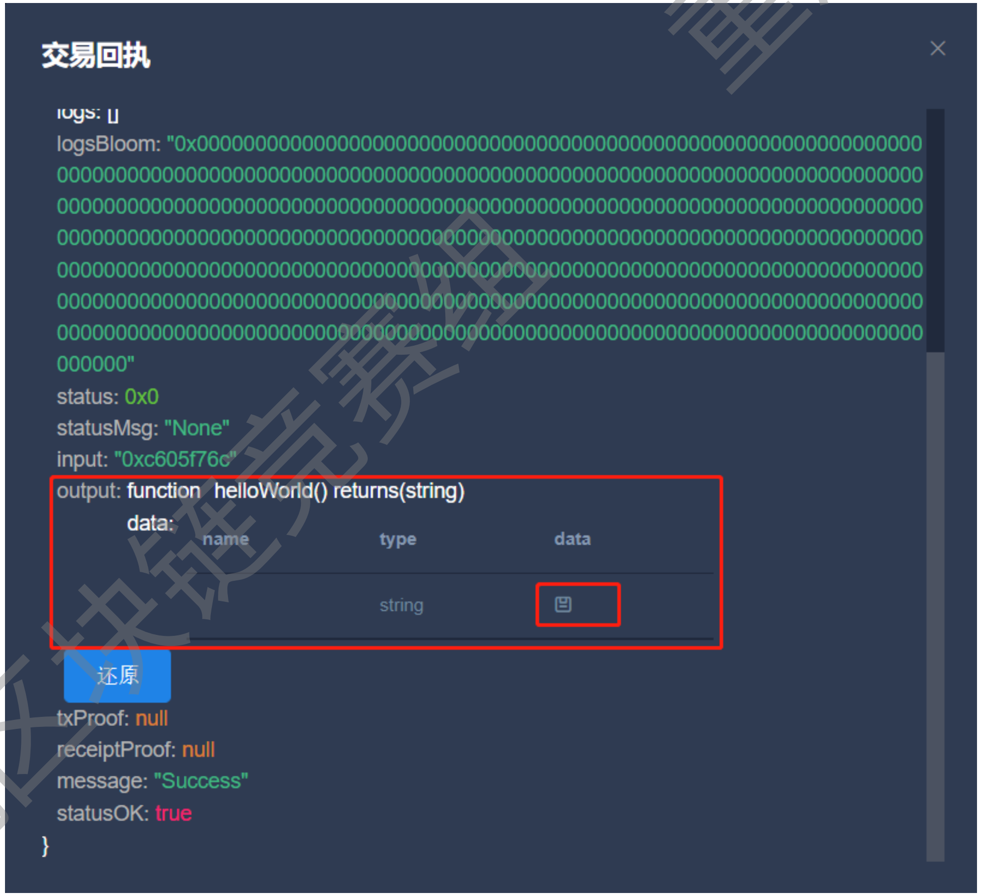
修饰符

- 修饰符与结合函数使用，可以在函数前后执行代码，常用于运行权限检查
- 修饰符使用“modifier”关键字声明，修饰符中的下划线“_”表示执行目标函数

```
pragma solidity 0.6.10;

contract Test {
    bool _switch = true;

    modifier check() {
        // 当全局变量_switch为true时，直接返回
        if (_switch) {
            return ;
        }
        // 否则执行函数
        _;
    }

    function helloWorld() check public returns (string memory) {
        return "hello,world";
    }
}
```

修饰符

- 调用合约，查看修饰符生效后的返回



腾讯云

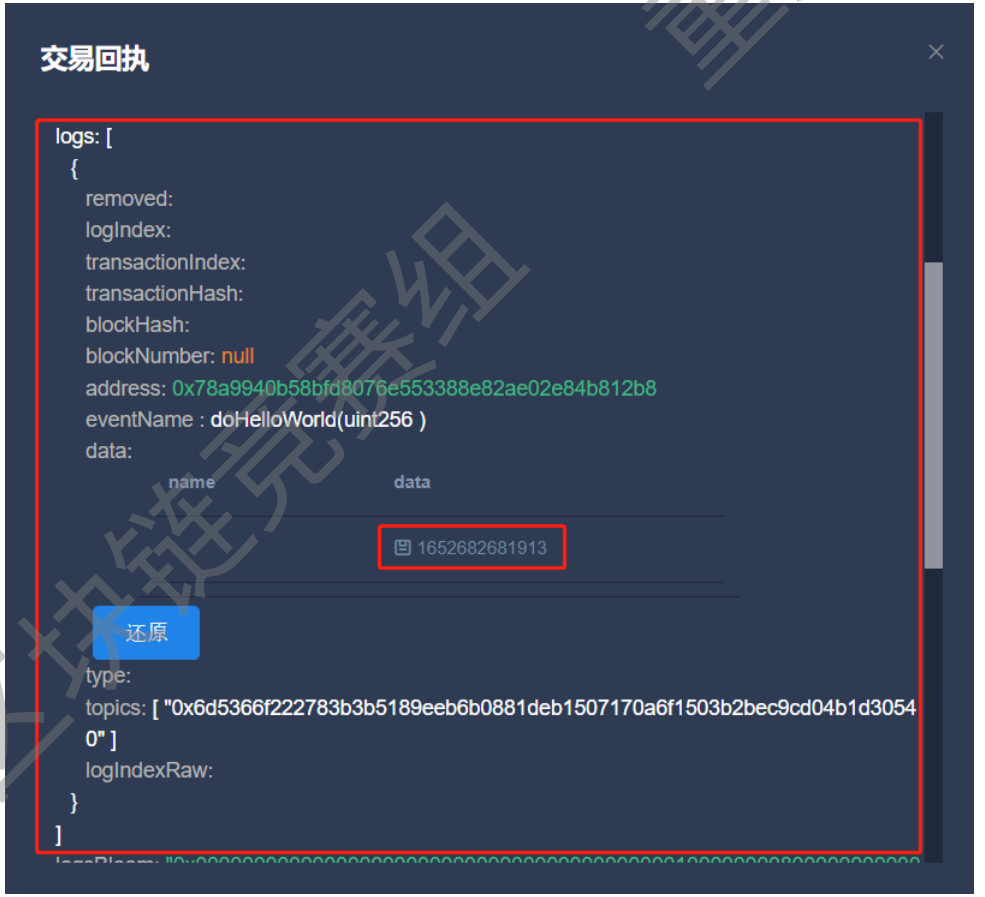
事件

- 通过事件可以在合约运行过程中记录日志，使用“event”关键字声明事件，使用“emit”关键字调用事件

```
pragma solidity 0.6.10;

contract Test {

    event doHelloWorld(uint);

    function helloWorld() public returns (string memory) {
        // 使用事件记录执行合约时的时间
        // now为内置函数，返回当前时间戳，单位为毫秒
        emit doHelloWorld(now);
        return "hello, world";
    }
}
```

事件

- 调用合约，查看事件日志输出



作业

1.一个Solidity文件包括哪些部分？

2.注释分为单行注释和多行注释，各使用什么语法标识？

3.在合约中，结构体使用什么关键字声明？

4.修饰符（modifier）的作用是什么？

5.事件（event）使用什么关键字调用？

谢谢