第3章 数据类型

目录

CONTENTS

腾讯云

CONTENTS

章节

第四章 类型转换

类型转换

- 编程语言中经常会出现类型转换。如将uint8类型转换为uint16类型，或者将其反向转换。这种转换常常分为隐式转换和显式转换。
  - 隐式转换：无须明确编写转换代码，且对于编译器来说，不同类型间的转换对运算结果没有影响，数据精度也不会改变，此时编译器会理解操作的含义。
  - 显式转换：某些情况下编译器不支持隐式转换，但是你很清楚你要做的结果，这种情况可以考虑显式转换。注意：显式转换可能会导致一些无法预料的后果，因此一定要进行测试。

隐式转换

- 当一个运算符能支持不同类型，编译器会隐式的尝试将一个操作数的类型，转为另一个操作数的类型，赋值同理。
- 隐式转换条件：
  - 值类型间的互相转换只要不丢失信息，语义可通则可转换。
  - 无符号整数可以被转为同样，或更大的类型，但不能反过来转换。

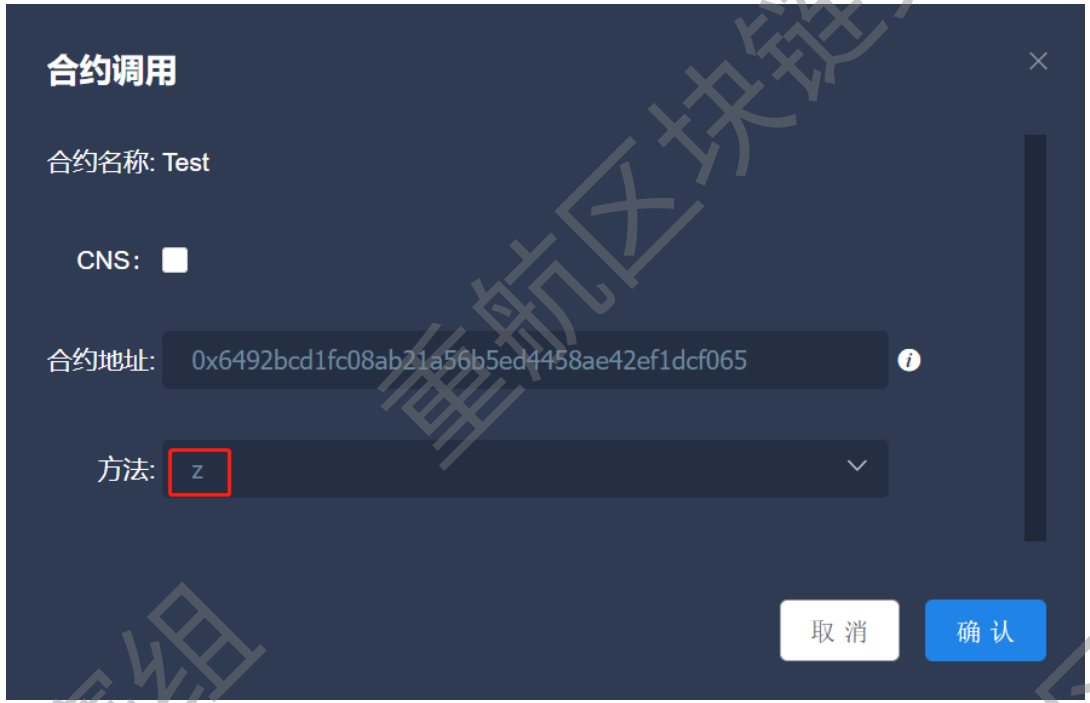
隐式转换

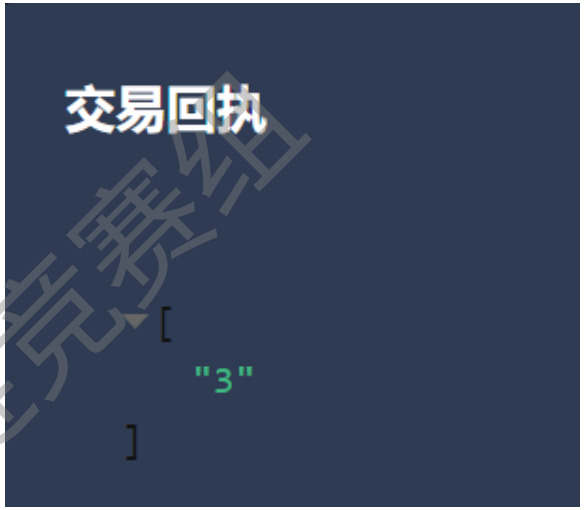
- 隐式转换示例，x和y相加时，x被隐式转换为uint16类型，之后进行赋值操作，将uint16类型隐式转换为uint32类型

```
pragma solidity 0.6.10;

contract Test {

    uint8 x = 1;
    uint16 y = 2;

    // 隐式转换为uint32类型
    uint32 public z = x + y;

    // uint32 x2 = 1;
    // uint16 y2 = 2;
    //
    // 不能将较大类型转换为较小类型
    // uint8 public z2 = x2 + y2;
}
```

隐式转换

- 调用合约，查看z变量的值





显式转换

- 某些情况下编译器不支持隐式转换，但是你很清楚你要做的结果，这种情况可以考虑显式转换，强制将变量转换为目标类型，但可能会造成无法预料的后果。
  - 将负值有符号整型转换为无符号整型时，会将负值转换为对应的补码
  - 将较大类型转换为较小类型时，高位会被舍弃

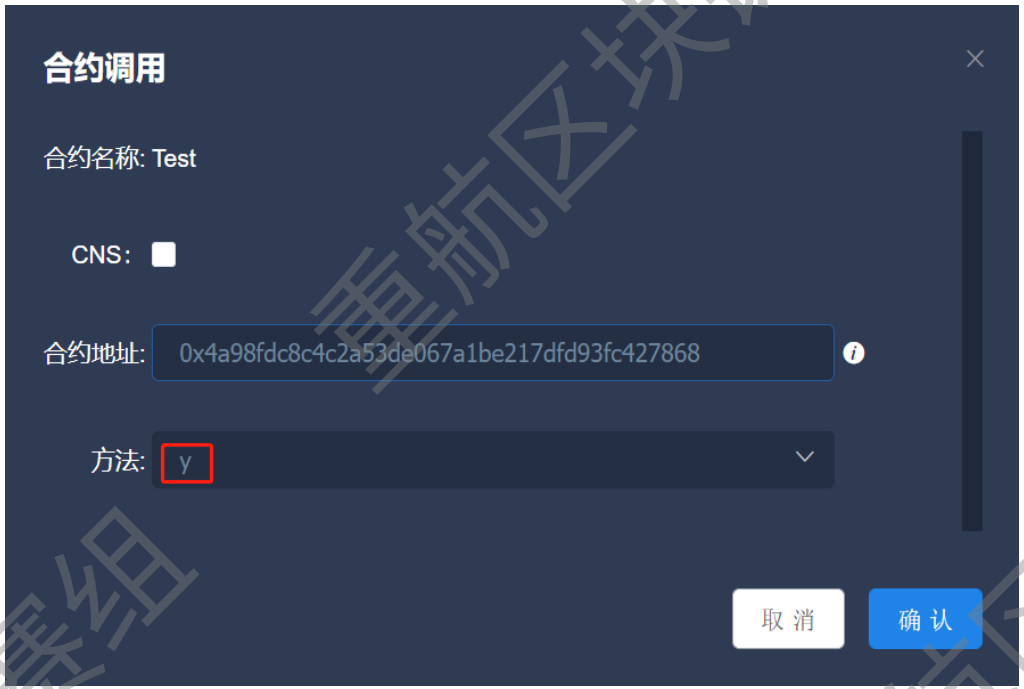
显式转换

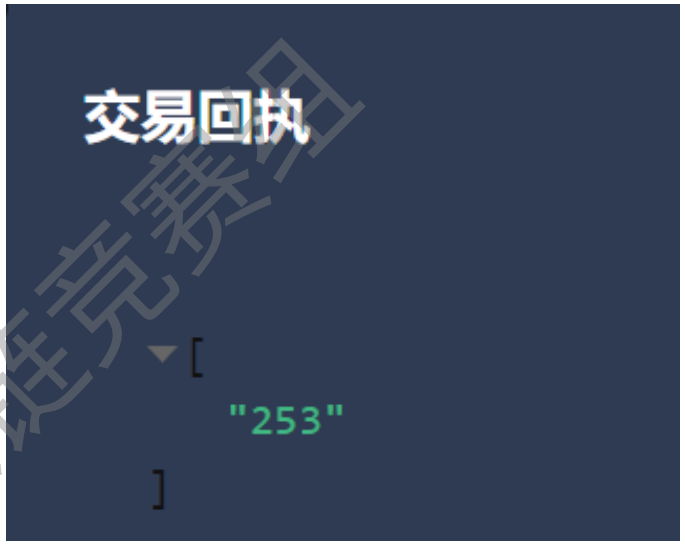
- 显式转换示例，整型之间显式转换

```
pragma solidity 0.6.10;

contract Test {

    int8 x = -3;

    // 将int8类型显式转换为uint8
    uint8 public y = uint8(x);


    uint16 x2 = 65534;

    // 将uint16类型显示转换为uint8类型
    uint8 public y2 = uint8(x2);

}
```

显式转换

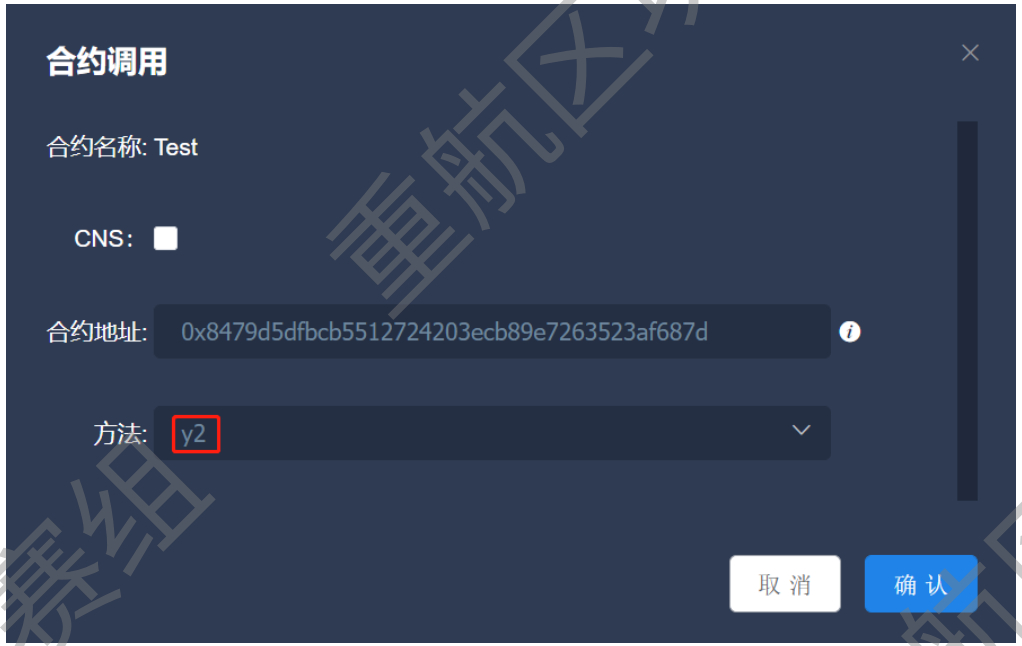
- 调用合约，查看y变量的值，253是-3的8位补码

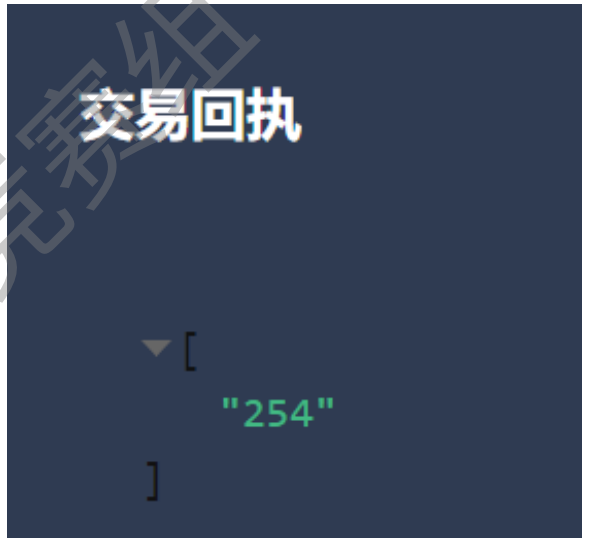




显式转换

- 调用合约，查看y2变量的值，65534 = 1111111111111110，去除高8位后值为11111110，转换为10进制为254





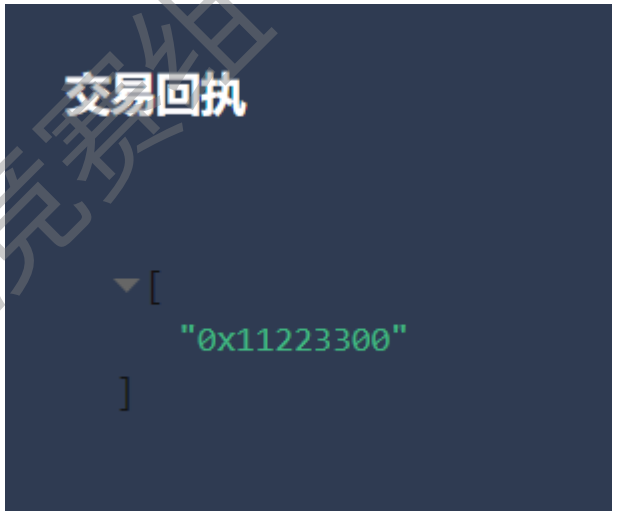
显式转换

- 显式转换示例，字节类型之间显式转换

```
pragma solidity 0.6.10;

contract Test {

    bytes3 x = 0x112233;

    // bytes3可以隐式转换为bytes4
    bytes4 public y = x;

    // 将bytes3显示转换为bytes2
    bytes2 public z = bytes2(x);
}
```

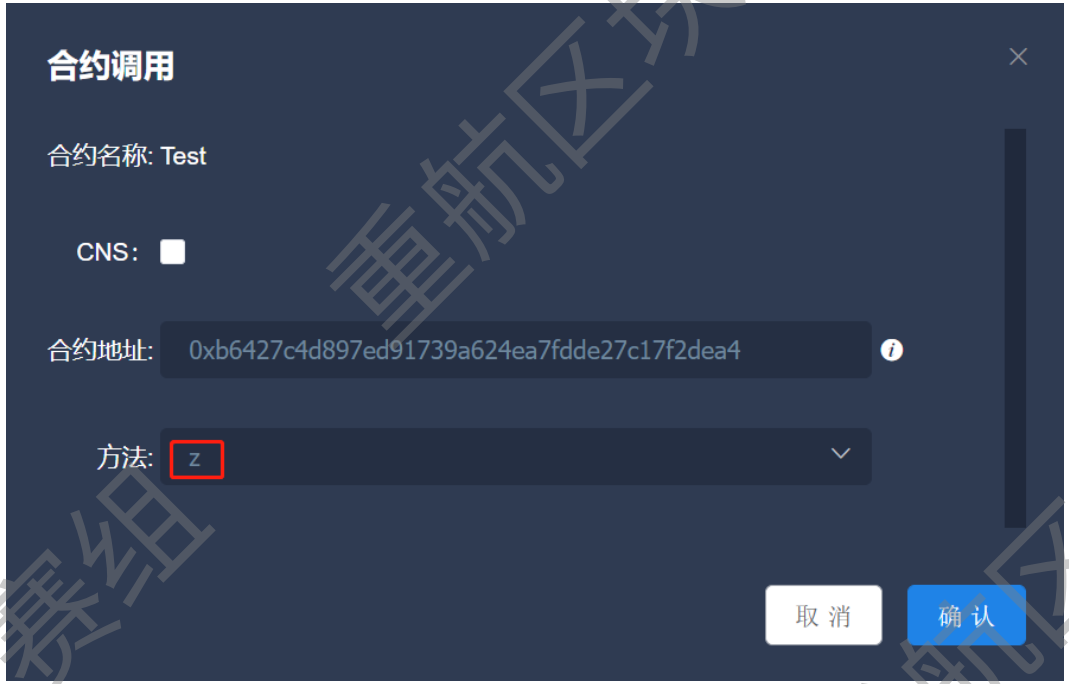
显式转换

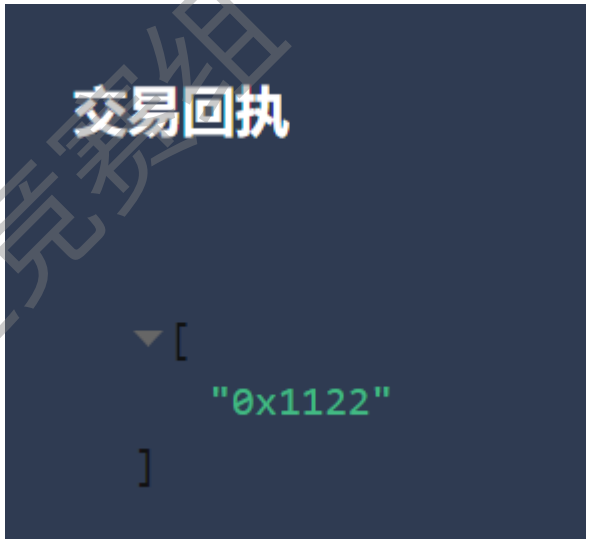
- 调用合约，查看y变量的值，将较短字节转换为较长字节时，会在尾部添加空字节



显式转换

- 调用合约，查看z变量的值，将较长字节转换为较短字节时，尾部的字节会被舍掉





显式转换

- 整型和定长字节数组在截断（或填充）时行为是不同的，如果整数和定长字节数组有相同的大小，则允许他们之间进行显式转换，如果要在不同的大小的整数和定长字节数组之间进行转换，必须使用一个中间类型来明确进行所需截断和填充的规则。

显式转换

- 整型和定长字节数组显示转换示例

```
pragma solidity 0.6.10;

contract Test {

    bytes2 a = 0x1122;

    // bytes2和uint16具有相同的长度直接转换
    uint16 public b = uint16(a);
    // bytes2和uint16大小不相同，需要中间类型uint16转换
    uint32 public c = uint32(uint16(a));
    // 使用uint截断，截断头部
    uint8 public d = uint8(uint16(a));
    // 使用bytes截断，截断尾部
    uint8 public e = uint8(bytes1(a));

}
```

显式转换

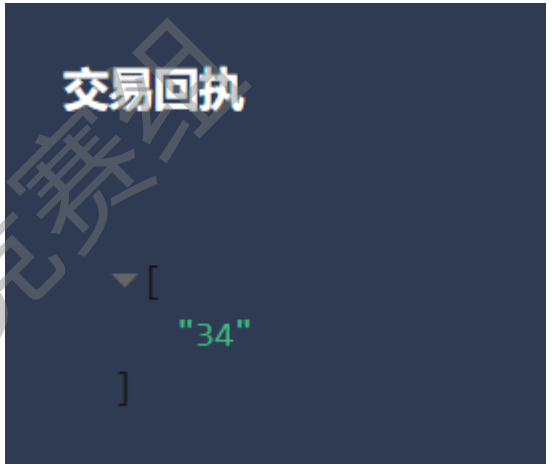
- 调用合约，查看b变量的值，4836=0x1122

显式转换

- 调用合约，查看c变量的值，4836=0x1122

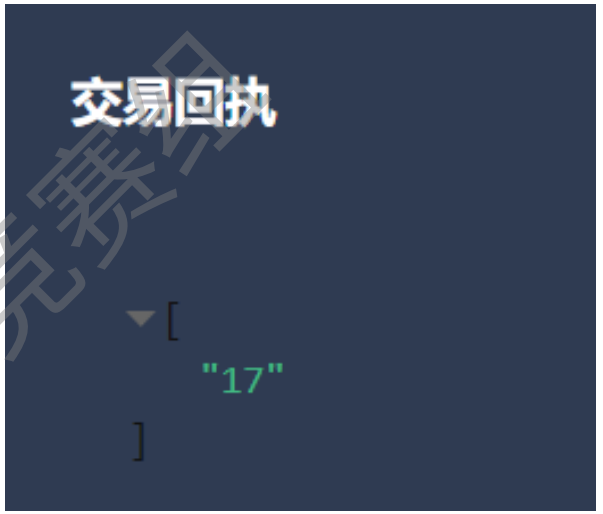
显式转换

- 调用合约，查看d变量的值，34=0x22



显式转换

- 调用合约，查看e变量的值，17=0x11



第五章 全局变量

章节

CONTENTS

全局变量

- 除了定义在合约的全局变量外，Solidity中还有一些比较特殊的内置全局变量，用于保存区块、交易信息。
  - 时间变量：seconds、minutes、hours、days、weeks
  - 区块变量：block
  - 交易变量：msg、tx
  - 地址变量：address

时间变量

- 内置时间变量包括 seconds、minutes、hours、days、weeks、years，它们的类型都是uint，换算格式如下
  - 1 seconds = 1
  - 1 minutes = 60 seconds
  - 1 hours = 60 minutes
  - 1 days = 24 hours
  - 1 weeks = 7 days

时间变量

- 时间变量合约示例

```
pragma solidity 0.6.10;

contract Test {

    uint sec = 1 seconds;

    uint min = 1 minutes;

    uint hour = 1 hours;

    uint day = 1 days;

    uint week = 1 weeks;

    function getTime() public returns (uint, uint, uint, uint, uint) {
        return (sec, min, hour, day, week);
    }

}
```

时间变量

- 调用合约的getTime函数

区块变量

- 区块变量block包含
  - blockhash(uint height) => bytes32 获取区块的哈希值（仅最近的256个区块）
  - block.gaslimit => uint 当前区块的gaslimit值，一般不需要关心该变量
  - block.number => uint 当前区块的高度
  - block.timestamp => uint 当前区块的生成时间戳，单位毫秒，有一个别名now
- 注：不要依赖blockhash，block.timestamp作为随机数的来源，这些数值不是真随机，且有被攻击的风险

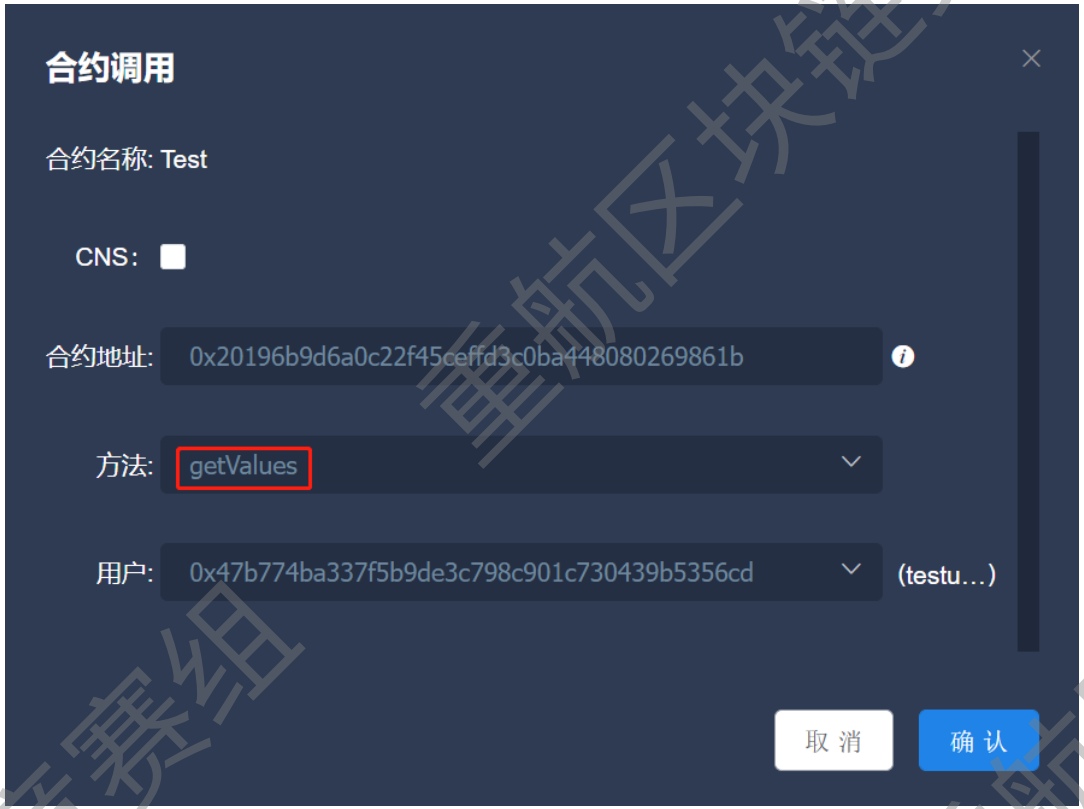
区块变量

- 合约区块变量示例

```
pragma solidity 0.6.10;


contract Test {

    function getValues() public returns (bytes32, uint, uint, uint, uint) {
        // 获取第一个区块的hash
        bytes32 blockHash = blockhash(1);
        // 获取当前区块gaslimit
        uint gaslimit = block.gaslimit;
        // 获取当前区块高度
        uint number = block.number;
        // 获取当前区块时间
        uint timestamp = block.timestamp;
        // 获取当前区块时间
        uint timestamp2 = now;
        return (blockHash,  gaslimit, number, timestamp, timestamp2);
    }

}
```

区块变量

- 调用合约，调用getValues函数，获取区块变量值



交易变量

- 交易变量msg、tx包含
  - msg.data => bytes 调用当前合约的原始数据
  - msg.sig => bytes4 调用当前合约的原始数据前4字节
  - msg.sender => address 当前合约调用者地址
  - tx.origin => address 交易发起者的地址，如A合约可以调用B合约，此时msg.sender是A合约地址，tx.origin是调用A合约的账户地址

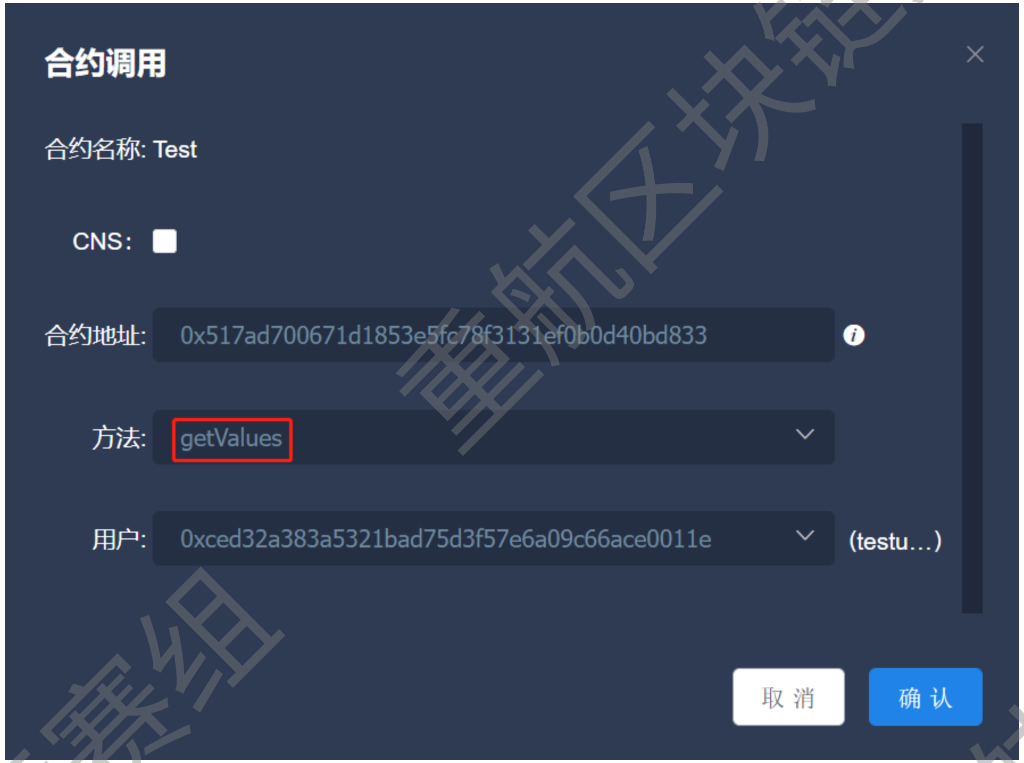
交易变量

- 合约交易变量示例

```
pragma solidity 0.6.10;


contract Test {

    function getValues() public returns (bytes memory, address, bytes4, address) {
        // 获取调用合约原始数据
        bytes memory data = msg.data;
        // 获取原始数据的钱4字节
        bytes4 sig = msg.sig;
        // 获取调用本合约的地址
        address sender = msg.sender;
        // 获取发起交易的地址
        address origin = tx.origin;

        return (data, sender, sig, origin);
    }
}
```

交易变量

- 调用合约，调用getValues函数，获取交易变量值



腾讯云

地址变量

- 在以太坊及其他公有链中，地址(address)变量存在以下几个内置函数，用于查询余额，转账等操作，但在联盟链中不涉及余额的转移，这些内置函数大部分都没有意义
  - `address.balance => uint` 获取地址余额
  - `address.transfer(uint)` 向指定地址发送指定金额
  - `address.send(uint) => bool` 向指定地址发送指定金额，失败时返回false
- 在联盟链中有意义的函数，`address(this)=>`获取合约地址

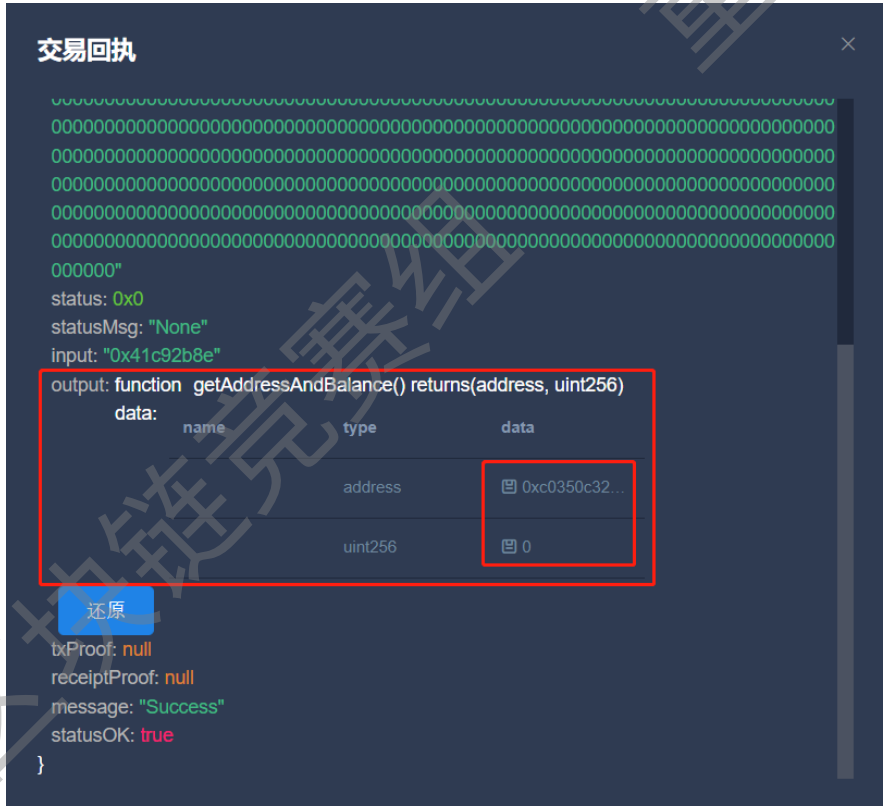
地址变量

- 地址变量实例

```
pragma solidity 0.6.10;

contract Test {


    function getAddressAndBalance() public returns (address) {
        // 获取当前合约地址
        address addr = address(this);
        // 获取当前合约地址余额
        uint balance = addr.balance;
        return address(this);
    }

}
```

地址变量

- 调用合约，调用getAddressAndBalance函数，获取合约地址和余额



作业

1.类型转换分为显式转换和隐式转换，它们有什么区别？

2.将较短字节转换成较长字节后，会发生什么变化？

3.全局变量seconds、minutes、hours是如何进行换算的？

4.区块变量block.number返回什么类型数据，含义是什么？

5.msg.sender和tx.origin有什么区别？

谢谢