solidity自学：第二天

目录

固定长度字节数组转动态长度字节数组

```
pragma solidity ^0.4.0;

contract fixToDynamicArray{
    bytes12 name = 0x7a68656e676a69616e78756e;

    function fixBytesTodynamicBytes() view returns(bytes){
        //return bytes(name);  !不可以这样子转换
        bytes memory newName = new bytes(name.length);
        for(uint i = 0;i < name.length;i++){//不可以是int,而是int
            newName[i] = name[i];
        }
        return newName;
    }
}
```

bytes与string的相互转换

```
pragma solidity ^0.4.0;
contract bytesToString{
    bytes name = new bytes(2);

    function Init(){
        name[0] = 0x7a;
        name[1] = 0x68;
    }

    function bytesToString_() returns(string){
        return string(name);
    }//行得通，返回“zh”
}
```

for循环

形参的传入！

高版本的solidity不能自动填充bytes32，
输入不是32个字节会报错。把输入的形参（bytes32)改成bytes就可以正常跑了

```
pragma solidity ^0.4.4;
contract bytes32ToString{
    bytes2 name = 0x7a68;

    function changeIt() returns(string){
        //return string(name);   ! 这样是不可以的
        //固定长度字节数组可以转换为bytes可变字节数组
        //bytes可变字节数组可以转换为s
    }
    function bytes32ToString_(bytes
```

```
        bytes memory newName = new bytes(_newName.length);
            newName[i] = _newName[i];
        }
        return string(newName);
        //！！解决方案1：高版本的solidity不能自动填充bytes32，
        //    输入不是32个字节会报错。把输入的形参（bytes32)改成bytes就可以正常跑了
        //!!解决方案2：将输入的0x7a68用0补全至64位，就可以正常运行了
    }
}
```

```
        for(uint i = 0;i < _newName.length;i++){
```

固定数组详解

没什么好讲的，跟Java差不多

```
pragma solidity ^0.4.4;
contract fixArray{
    uint[5] arr = [1,2,3,4,5];

    function Init(){
        arr[0] = 100;
        arr[1] = 200;
    }
    function getArrayContent() view returns(uint[5]){
        return arr;
    }
    function getGrade() view returns(uint){
        uint grade = 0;
        for(uint i = 0;i < 5;i++){
            grade = grade + arr[i];
        }
        return grade;
    }
    function changeLength() returns(uint){
        //return arr.length = 10;  ！不可以
    }
    function push(){
        //arr.push(6);  ！不可以
    }
}
```

可变长度数组深入

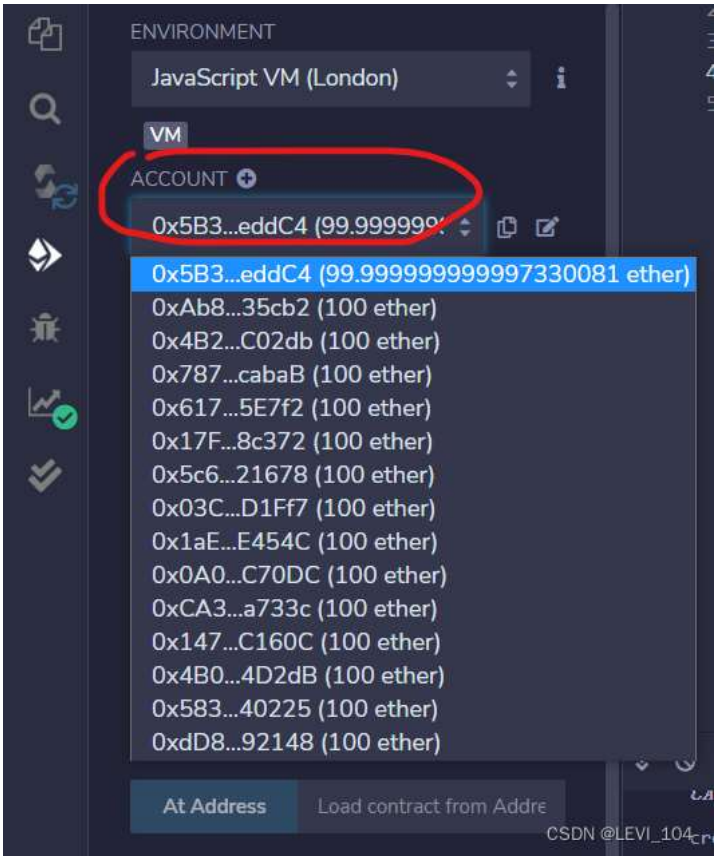
```
pragma solidity ^0.4.4;
contract dynamicArray{
    uint[] grade;//定义一个可变长度的数组

    function getContent() view returns(uint[]){
        return grade;
    }
    function getLength() view returns(uint){
        return grade.length;
    }
    function add() view returns(uint){
        uint sum = 0;
        for(uint i = 0;i < grade.length;i++){
            sum += grade[i];
        }
        return sum;
    }
    function changeLength_(){
        grade.length = 10;
    }
}
```

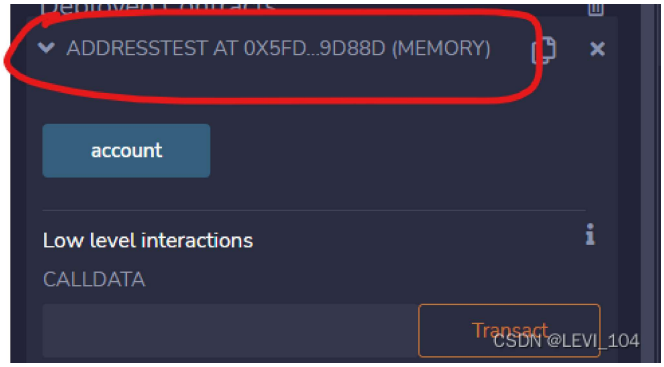
可变长度二维数组

这里就不做阐述（我也没看那几个视频），举一反三一维数组，都差不多的

以太坊地址的本质



圈住的地方就是我的账户，我的地址0x5B38Da6a701c568545dCfcB03FcB875f56beddC4（160位uint160）



圈住的是本合约在区块链中的地址

```
pragma solidity ^0.4.4;

contract addressTest{
    address public account = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;//默认值：0x0000000000000000000000000000000000000000
    //本合约用户地址：0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
    //本合约地址：0x5FD6eB55D12E759a21C09eF703fe0CBa1DC9d88D
    //因此，address使用uint160来存储的

    address account1 = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;//用户地址
    address account2 = 0xDA0bab807633f07f013f94DD0E6A4F96F8742B53;//合约地址

    function changeIt() view returns(uint160){
        return uint160(account);
        //返回：520786028573371803640530888255888666801131675076
    }

    function changeIt2() view returns(address){
        return address(520786028573
        //返回：0x5B38Da6a701c56854
```

```
    }21
    function check1() view returns(bool){
        return account > account2;
    }//F
    function check2() view returns(bool){
        return account >= account2;
    }//F
    function check3() view returns(bool){
        return account < account2;
    }//T
    function check4() view returns(bool){
        return account <= account2;
    }//T
    //这说明了，合约地址 > 用户地址
}
```

address是uint160存储；合约地址 > 用户地址