『0011』 - Solidity Types - 地址(Address)

> 孔壹学院：国内区块链职业教育领先品牌
>
> 作者：黎跃春，区块链、高可用架构工程师
> 微信：liyc1215 QQ群：348924182 博客：http://liyuechun.org

以太坊钱包地址位数验证

以太坊中的地址的长度为 `20` 字节，一字节等于8位，一共 `160` 位，所以 `address` 其实亦可以用 `uint160` 来声明。

我的以太坊钱包的地址为 `0xF055775eBD516e7419ae486C1d50C682d4170645` ， `0x` 代表十六进制，我们将 `F055775eBD516e7419ae486C1d50C682d4170645` 拷贝，如下图所示，将其进行二进制转换，不难发现，它的二进制刚好 `160` 位。

备注：以太坊钱包地址是以16进制的形式呈现，我们知道一个十六进制的数字等于 `4` 个字节， `160 / 4 = 40` ，你自己验证一下，钱包地址 `F055775eBD516e7419ae486C1d50C682d4170645` 的长度为40。

我的以太坊钱包地址：

```
0xF055775eBD516e7419ae486C1d50C682d4170645
```

对应的二进制为：

```
0b1111 0000 0101 0101 0111 0111 0101 1110 1011 1101 .....0100 0101
```

PS:通过工具进行转换完的二进制有问题，所以自己去计算一下即可。

```
pragma solidity ^0.4.4;


contract Test {

    address _owner;
    uint160 _ownerUint;
    function Test() {
        _owner = 0xF055775eBD516e7419ae486C1d50C682d4170645;
        _ownerUint = 1372063746939811866332913075223978746532890871365;
    }

    function owner() constant returns (address) {
        return _owner;
    }

    function ownerUint160() constant returns (uint160) {
        return uint160(_owner);
        // 1372063746939811866332913075223978746532890871365
    }


    function ownerUintToAddress() constant returns (address) {
        return address(_ownerUint);
    }

}

// 0x 45
// 0b 0100 0101
// 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128
// 1 + 0 + 4 + 0 + 0 + 0 + 64 + 0 = 69


// 1 2 3 4 5 6 7 8 9 A B C D E F


// 0xEAEC9B481c60e8cDc3cdF2D342082C349E5D6318
// 0xCa8Aec018a4822275EEA6d5c8296d840E0823553

// 0xF055775eBD516e7419ae486C1d50C682d4170645
// 1372063746939811866332913075223978746532890871365
```

```
// 0b1111 0000 0101 0101 0111 0111 0101 1110 1011 1101 .....0100 0101
//


// 8  4 2 1


// 0 0 0 1
// 1000

// 9

// 4 2 1 0
// 1 0 0 1
// 1001


// 40 * 4 = 160

// uint160

// address
// uint160
```

不可不知的几个常识

- 合约拥有者

`msg.sender`就是当前调用方法时的发起人，一个合约部署后，通过钱包地址操作合约的人很多，但是如何正确判断谁是合约的拥有者，判断方式很简单，就是第一次部署合约时，谁出的`gas`，谁就对合约具有拥有权。

```
pragma solidity ^0.4.4;


contract Test {

    address public _owner;

    uint public _number;

    function Test() {
        _owner = msg.sender;
        _number = 100;
    }
```

```
    function msgSenderAddress() constant returns (address) {
        return msg.sender;
    }

    function setNumberAdd1() {
        _number = _number + 5;
    }

    function setNumberAdd2() {
        if (_owner == msg.sender) {
            _number = _number + 10;
        }
    }

}

// 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
```

- 合约地址

```
pragma solidity ^0.4.4;


// 0x903ad08970c70d10e5fb5b3c26f7b714830afcf6
// 0x62e40877f4747e06197aa1a2b9ac06dd9bb244a3

// 0xf055775ebd516e7419ae486c1d50c682d4170645

// 0xe7795e05d15f7406baf411cafe766fc28eccf35f
// 0xe7795e05d15f7406baf411cafe766fc28eccf35f

contract Test {

    address public _owner;

    uint public _number;

    function Test() {
        _owner = msg.sender;
        _number = 100;
    }

    function msgSenderAddress() constant returns (address) {
        return msg.sender;
    }
```

```
    function setNumberAdd1() {
        _number = _number + 5;
    }

    function setNumberAdd2() {
        if (_owner == msg.sender) {
            _number = _number + 10;
        }
    }


    function returnContractAddress() constant returns (address) {
        return this;
    }



}
```

一个合约部署后，会有一个合约地址，这个合约地址就代表合约自己。

- this是人还是鬼

`this`在合约中到底是`msg.sender`还是`合约地址`，由上图不难看出，`this`即是当前合约地址。

支持的运算符

```
pragma solidity ^0.4.4;



contract Test {

    address address1;
    address address2;

    // <=, <, ==, !=, >=和>


    function Test() {
        address1 = 0xF055775eBD516e7419ae486C1d50C682d4170645;
        address2 = 0xEAEC9B481c60e8cDc3cdF2D342082C349E5D6318;
    }
```

```
    // <=
    function test1() constant returns (bool) {
        return address1 <= address2;
    }

    // <
    function test2() constant returns (bool) {
        return address1 < address2;
    }

    //  !=
    function test3() constant returns (bool) {
        return address1 != address2;
    }

    // >=
    function test4() constant returns (bool) {
        return address1 >= address2;
    }

    // >
    function test5() constant returns (bool) {
        return address1 > address2;
    }
}
```

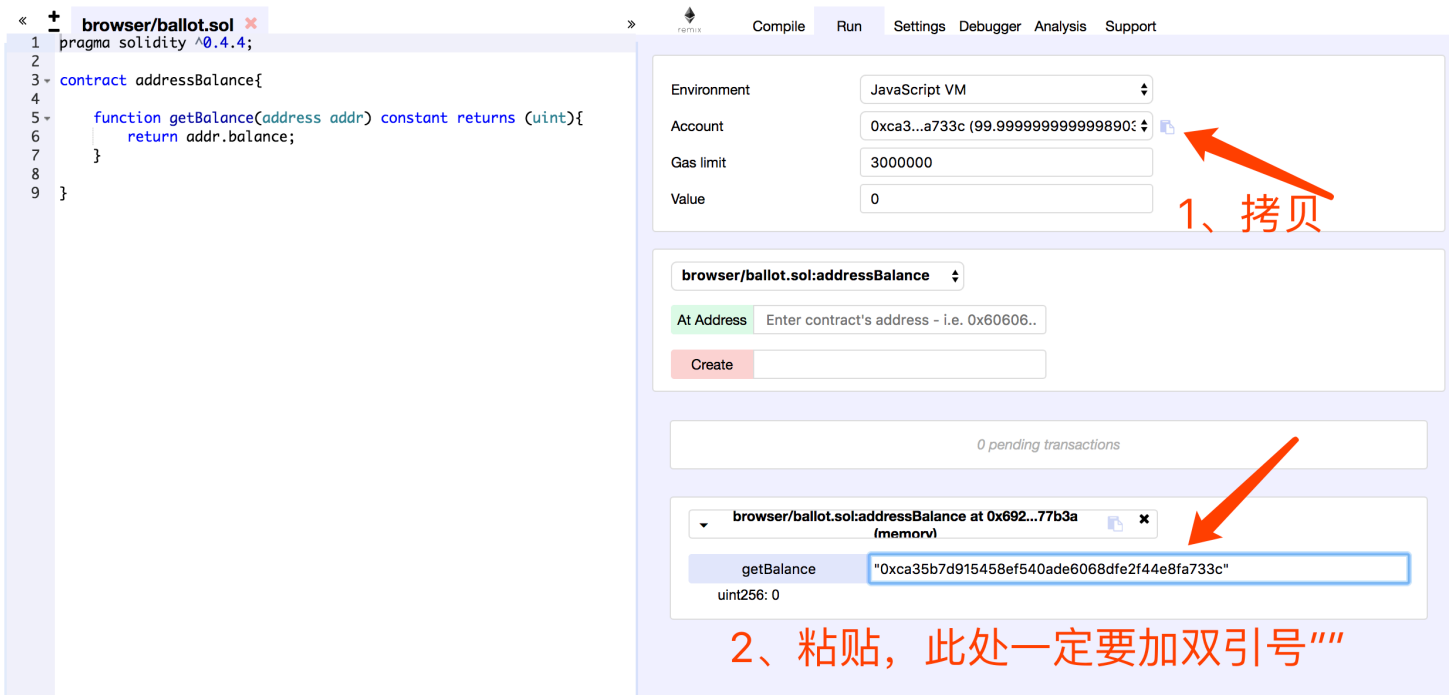
- `<=`, `<`, `==`, `!=`, `>=`和`>`

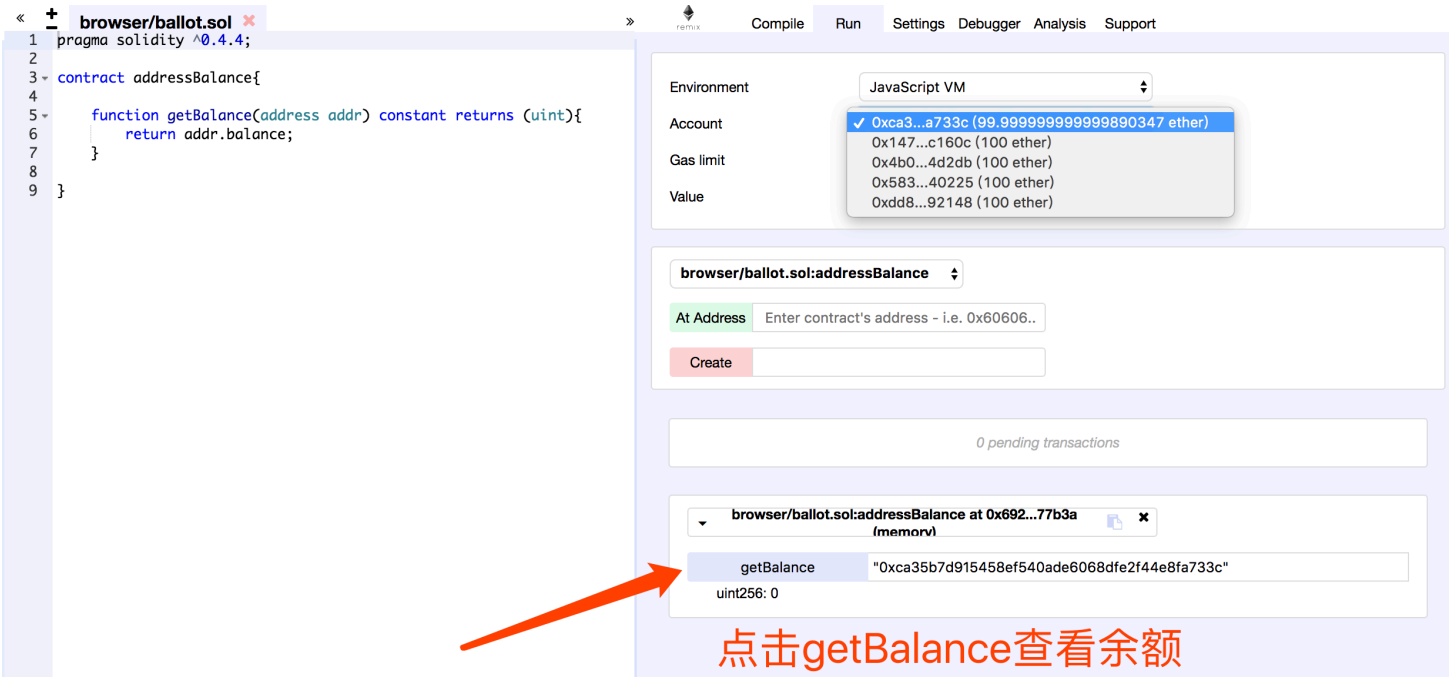
成员变量和函数

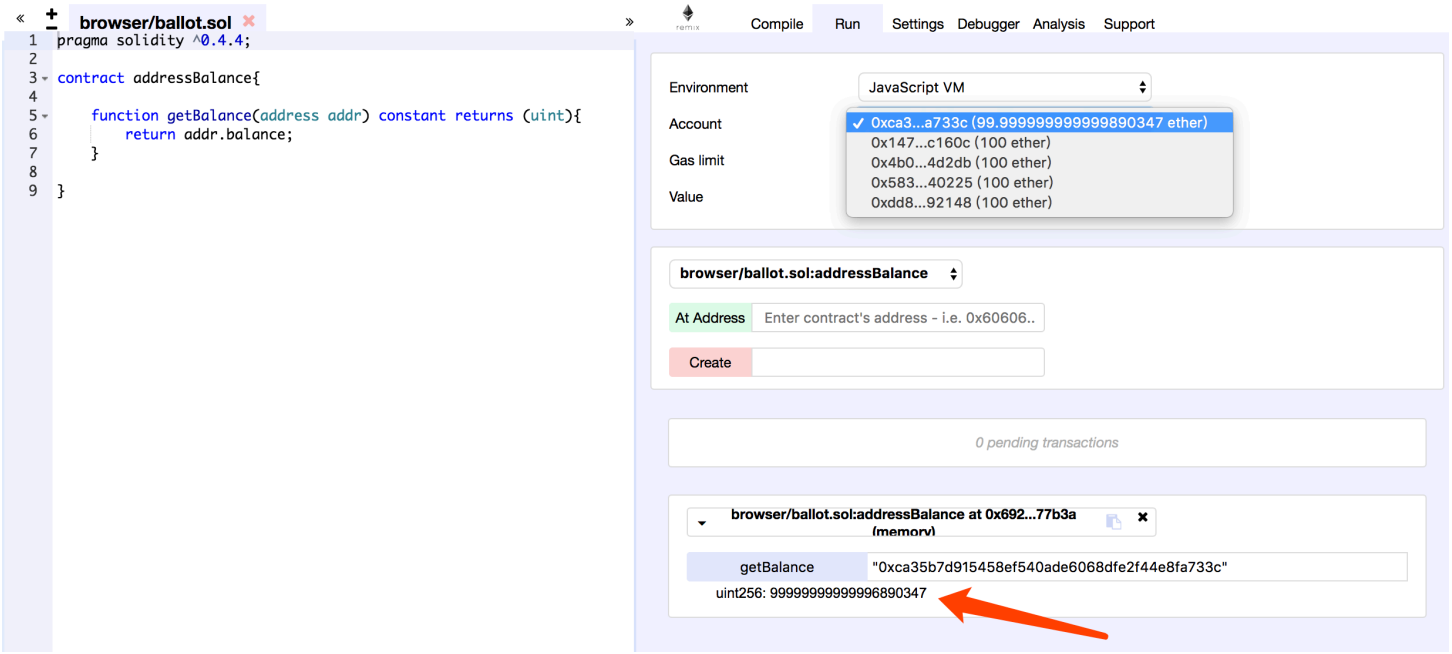
一、 balance

如果我们需要查看一个地址的余额，我们可以使用`balance`属性进行查看。

```
pragma solidity ^0.4.4;

contract addressBalance{

    function getBalance(address addr) constant returns (uint){
        return addr.balance;
    }

}
```


browser/ballot.sol
pragma solidity ^0.4.4;
contract addressBalance{
function getBalance(address addr) constant returns (uint){
return addr.balance;
}
}
Compile
Run
Settings
Debugger
Analysis
Support
Environment
JavaScript VM
Account
0xca3...a733c (99.9999999999998903
Gas limit
3000000
Value
0
1、拷贝
browser/ballot.sol:addressBalance
At Address
Enter contract's address - i.e. 0x60606..
Create
0 pending transactions
browser/ballot.sol:addressBalance at 0x692...77b3a (memory)
getBalance
"0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
uint256: 0
2、粘贴，此处一定要加双引号""


browser/ballot.sol
pragma solidity ^0.4.4;
contract addressBalance{
function getBalance(address addr) constant returns (uint){
return addr.balance;
}
}
Compile
Run
Settings
Debugger
Analysis
Support
Environment
JavaScript VM
Account
0xca3...a733c (99.999999999999890347 ether)
0x147...c160c (100 ether)
0x4b0...4d2db (100 ether)
0x583...40225 (100 ether)
0xdd8...92148 (100 ether)
browser/ballot.sol:addressBalance
At Address
Enter contract's address - i.e. 0x60606..
Create
0 pending transactions
browser/ballot.sol:addressBalance at 0x692...77b3a (memory)
getBalance
"0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
uint256: 0
点击getBalance查看余额

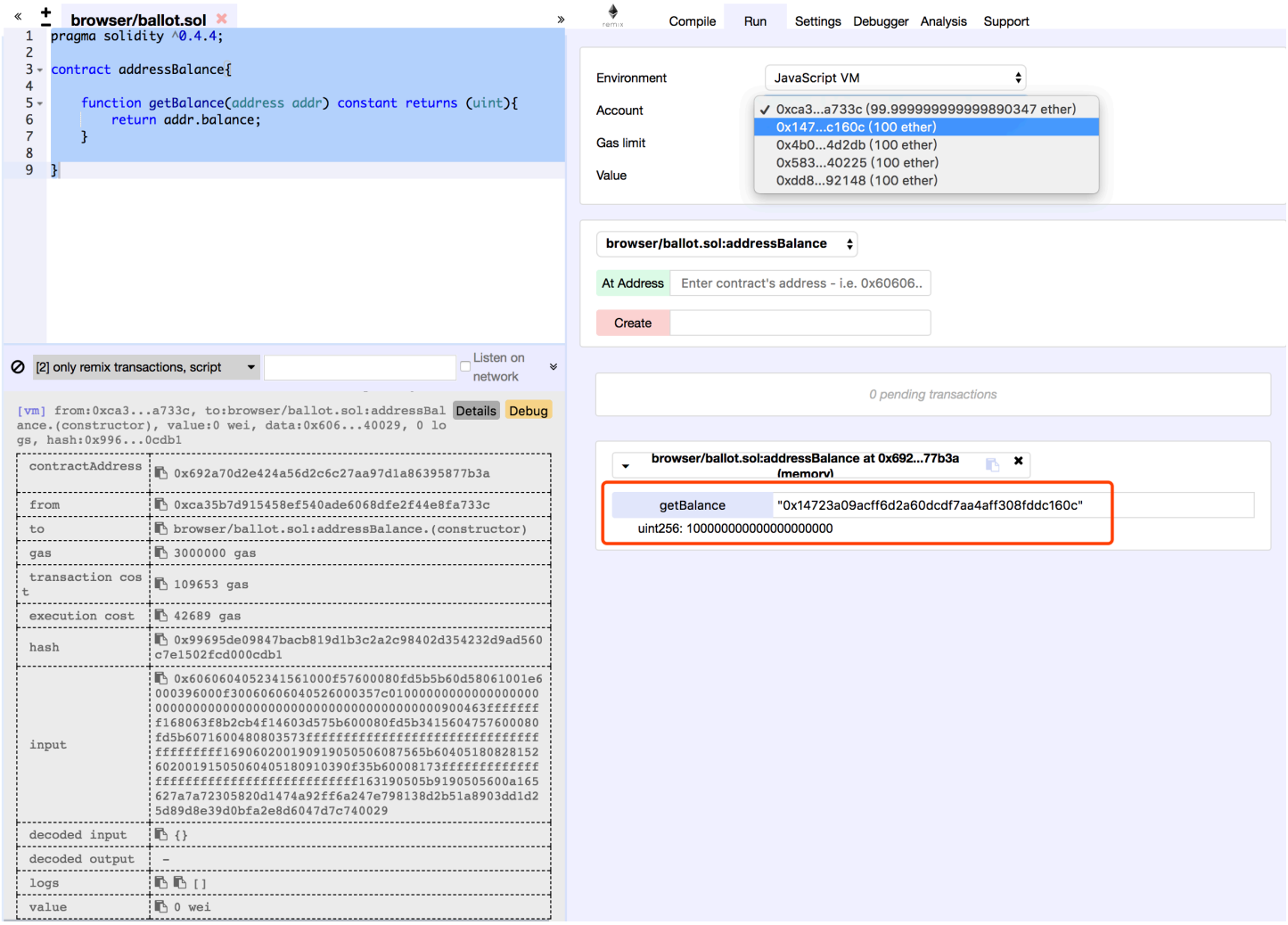

browser/ballot.sol
pragma solidity ^0.4.4;
contract addressBalance{
function getBalance(address addr) constant returns (uint){
return addr.balance;
}
}
Compile
Run
Settings
Debugger
Analysis
Support
Environment
JavaScript VM
Account
0xca3...a733c (99.999999999999890347 ether)
0x147...c160c (100 ether)
0x4b0...4d2db (100 ether)
0x583...40225 (100 ether)
0xdd8...92148 (100 ether)
browser/ballot.sol:addressBalance
At Address
Enter contract's address - i.e. 0x60606..
Create
0 pending transactions
browser/ballot.sol:addressBalance at 0x692...77b3a (memory)
getBalance
"0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
uint256: 9999999999996890347

在 `Account` 一栏中，会自动生成 `5` 个钱包地址供我们测试使用，在我们点击 `Create` 按钮时， `Account` 一栏选中的是哪个钱包地址，我们部署合约时，花费的 `gas` 就从哪个钱包地址里面扣除，**【PS，这5个钱包地址每次都是系统临时生成，所以在我们开发测试过程中，每次的地址不会相同】**。因为在本案例中，我们部署合约时，用的是第一个 `Account` ，所以 `gas` 自然从它里面扣除，大家会发现，其它四个钱包地址中的余额是 `ether` ，而第一个钱包地址中不到 `100` 个 `ether` 。

当我们点击 `getBalance` 获取某个钱包地址的余额时，获取到的余额的单位是 `Wei` ，一个 `ether` 等于 `1000000000000000000Wei` ， `Wei` 是最小单位，相当于我们的 `1元RMB` 等于 `100分` 。**【PS： `1ether` 等于 `10的18次方Wei` 】**

9999999999996890347 wei == 99.999999999996890347 ether

我们重新换一个钱包地址查看它的余额试试：

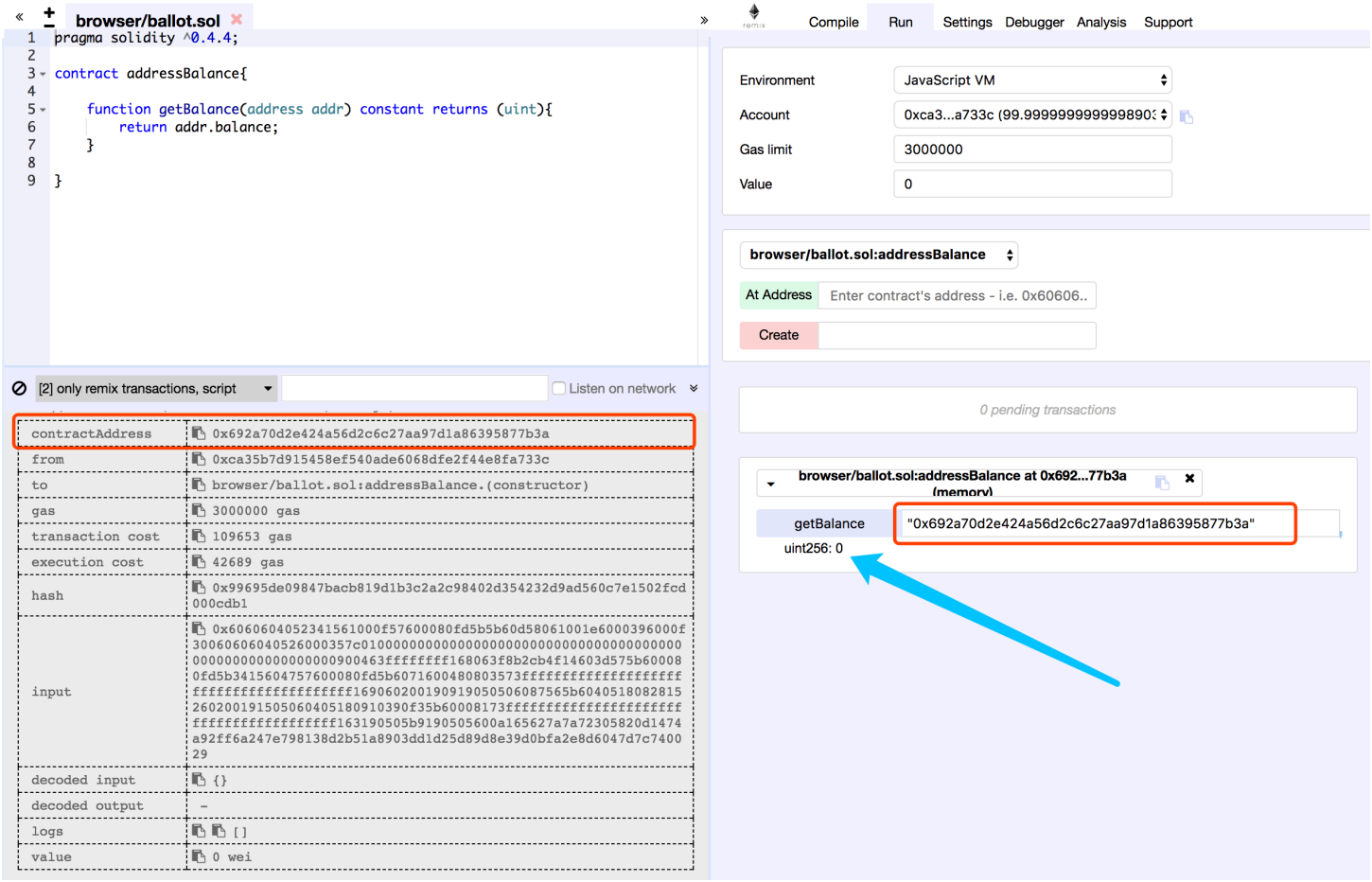


我用的是第二个钱包地址，余额刚好为 `100000000000000000000 wei` ，也就是 `100 ether` 。

二、this 查看当前合约地址余额

```
pragma solidity ^0.4.4;
```

```
contract addressBalance{

    function getBalance(address addr) constant returns (uint){
        return addr.balance;
    }

}
```

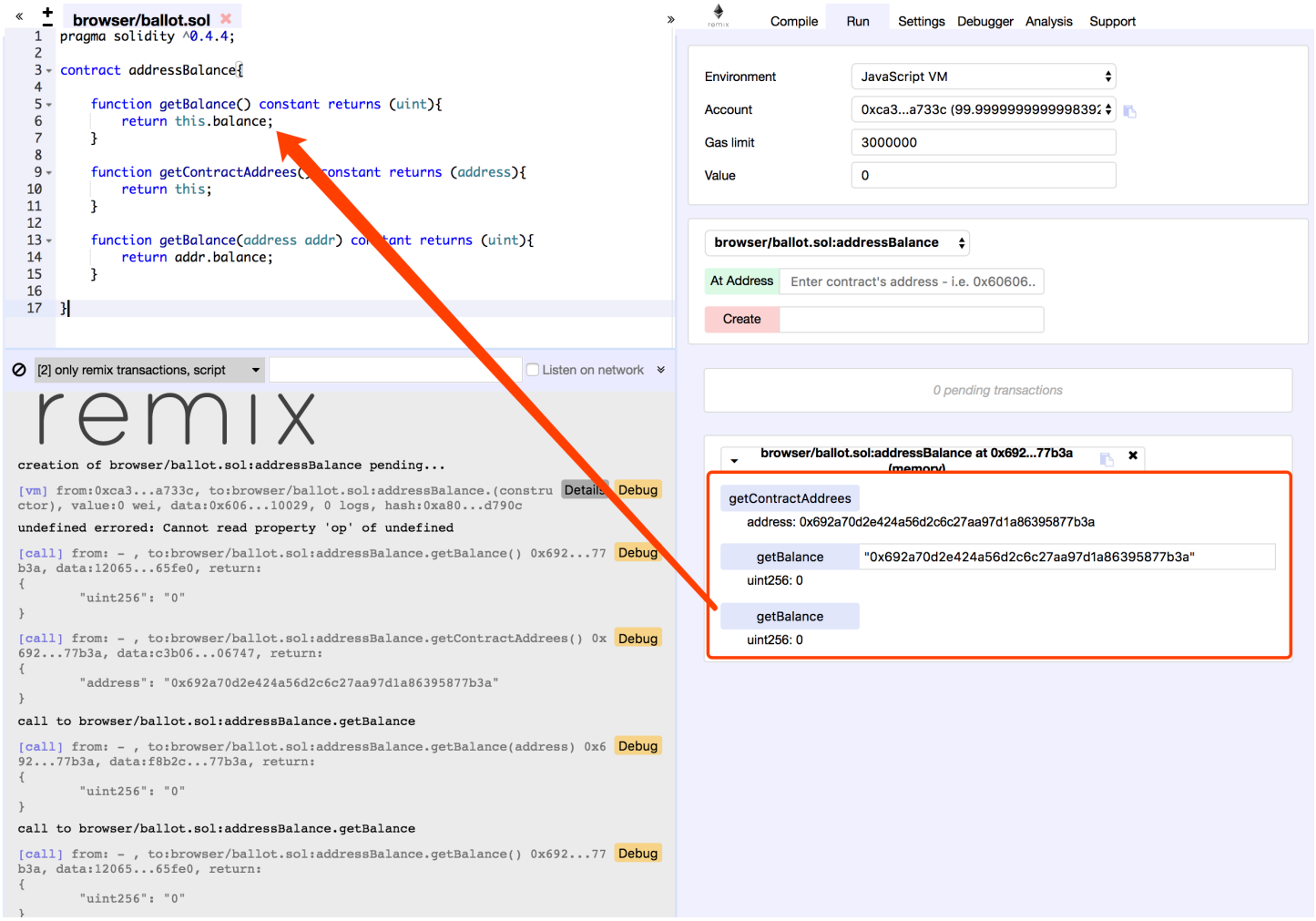


将当前合约地址作为参数，由上图所示，当前合约地址余额为 `0 wei` 。

在上面的文章中我们说过， `this` 代表当前合约地址，那么如果在代码中我们只是想简单查询当前合约余额，我们可以直接通过 `this.balance` 进行查询。

```
pragma solidity ^0.4.4;

contract addressBalance{

    function getBalance() constant returns (uint){
        return this.balance;
    }

    function getContractAddrees() constant returns (address){
        return this;
```

```
    }

    function getBalance(address addr) constant returns (uint){
        return addr.balance;
    }

}
```

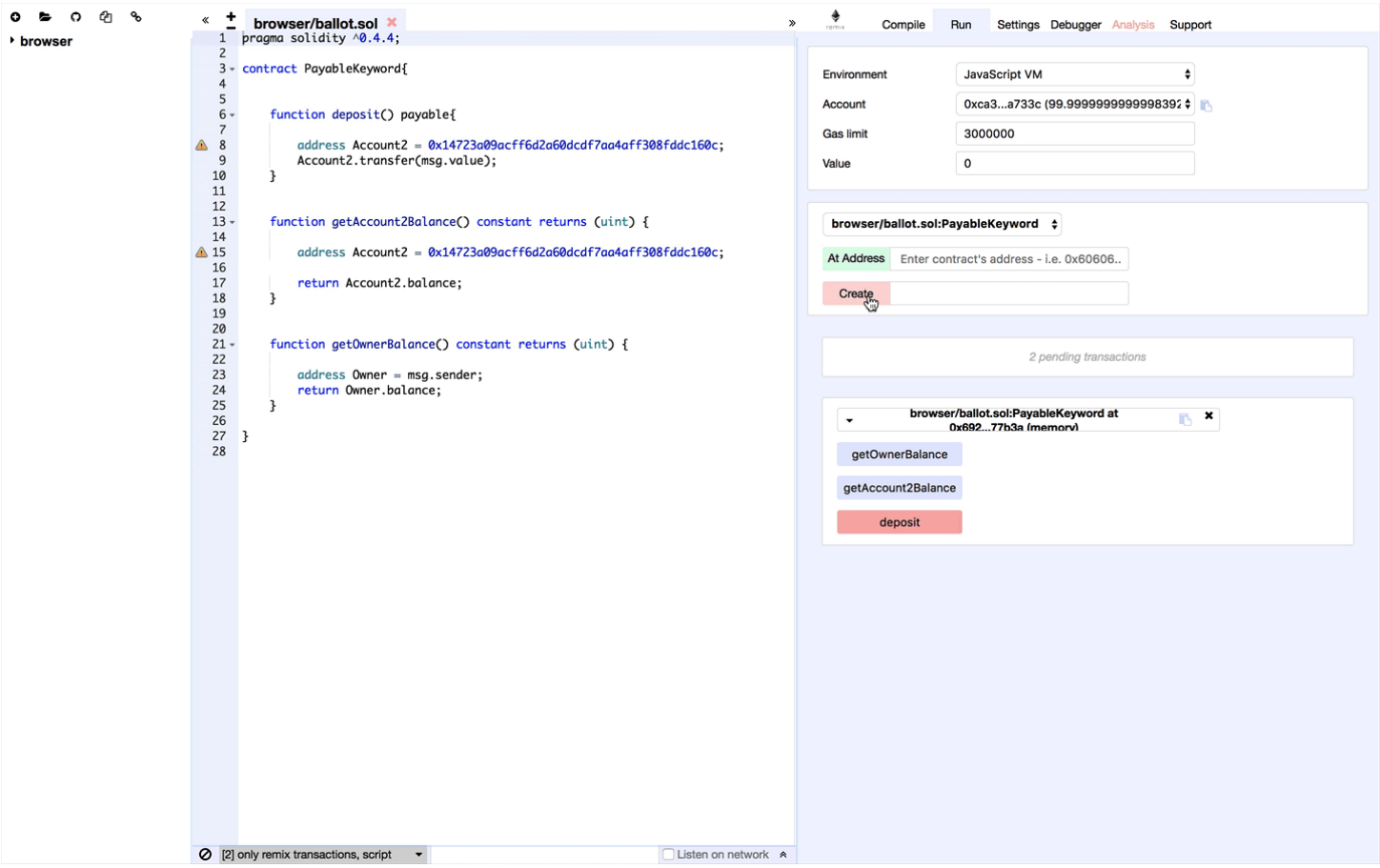


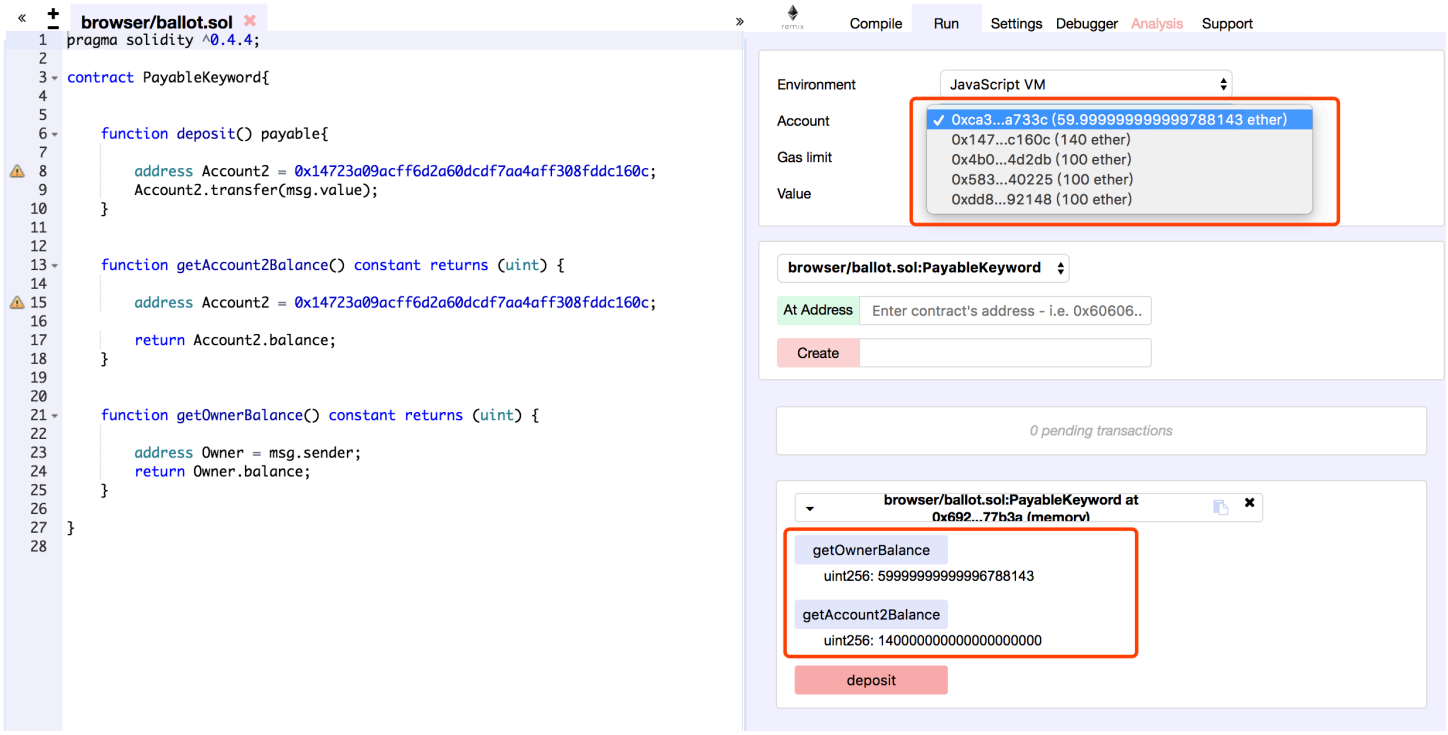
三、transfer

transfer：从合约发起方向某个地址转入以太币(单位是wei)，地址无效或者合约发起方余额不足时，代码将抛出异常并停止转账。

```
pragma solidity ^0.4.4;

contract PayableKeyword{


    // 从合约发起方向 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c 地址转入 msg.value 个以太币，单位是 wei
    function deposit() payable{
```

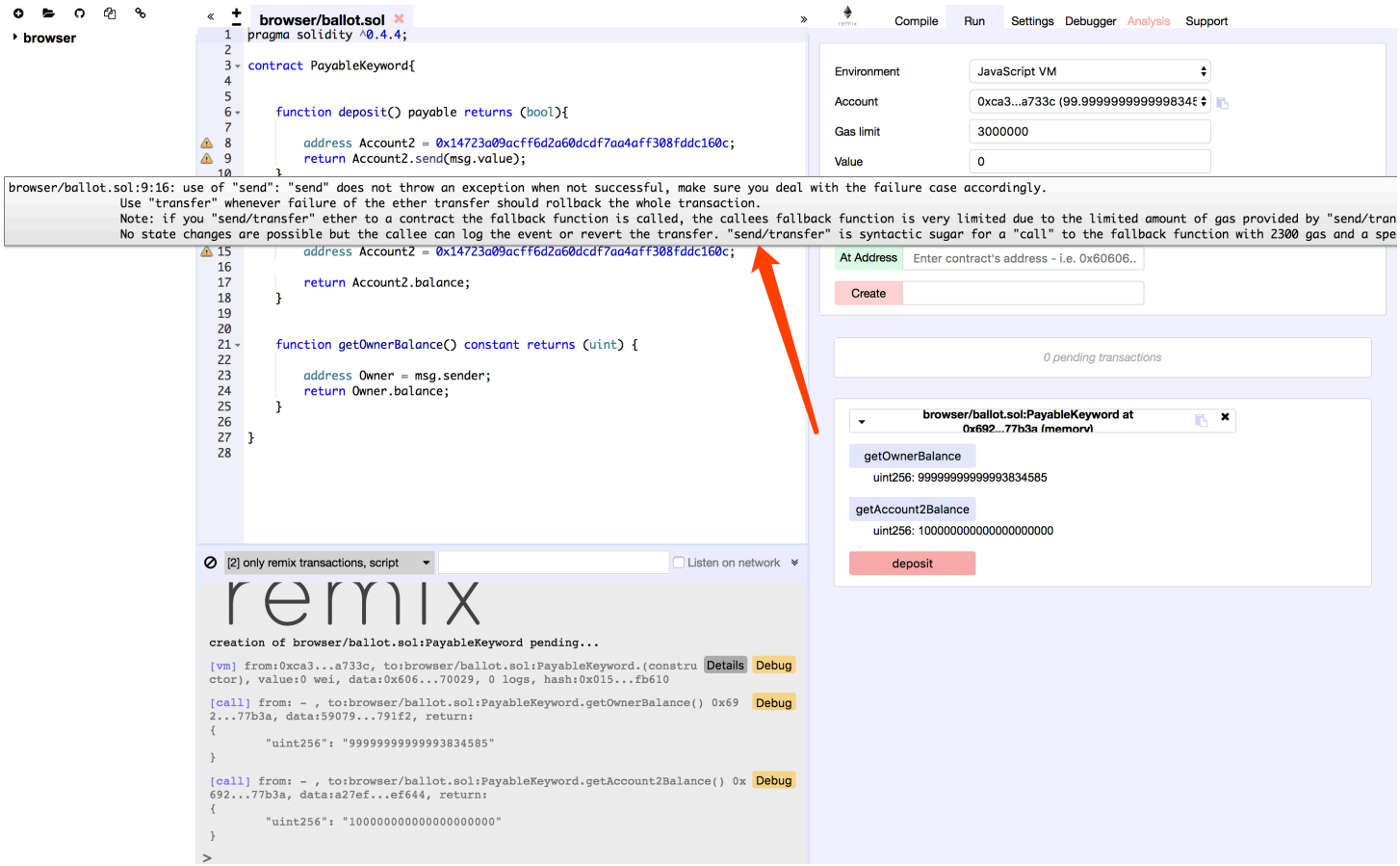
```
        address Account2 = 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c;
        Account2.transfer(msg.value);
    }


    // 读取 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c 地址的余额
    function getAccount2Balance() constant returns (uint) {

        address Account2 = 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c;

        return Account2.balance;
    }

    // 读取合约发起方的余额
    function getOwnerBalance() constant returns (uint) {

        address Owner = msg.sender;
        return Owner.balance;
    }

}
```





四、send

send： `send` 相对 `transfer` 方法较底层，不过使用方法和 `transfer` 相同，都是从合约发起方向某个地址转入以太币(单位是wei)，地址无效或者合约发起方余额不足时， `send` 不会抛出异常，而是直接返回 `false` 。



```
pragma solidity ^0.4.4;

contract PayableKeyword{


    function deposit() payable returns (bool){

        address Account2 = 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c;
        return Account2.send(msg.value);
    }


    function getAccount2Balance() constant returns (uint) {

        address Account2 = 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c;

        return Account2.balance;
    }


    function getOwnerBalance() constant returns (uint) {

        address Owner = msg.sender;
        return Owner.balance;
    }

}
```

备注：这是测试代码，测试方法和 `transfer` 一样。

⚠️Warning

send()方法执行时有一些风险

- 调用递归深度不能超1024。
- 如果gas不够，执行会失败。
- 所以使用这个方法要检查成功与否。
- `transfer` 相对 `send` 较安全