Dapp智能合约开发

深圳ABI - 30.05.2018

凯东 - Antoine Dahan

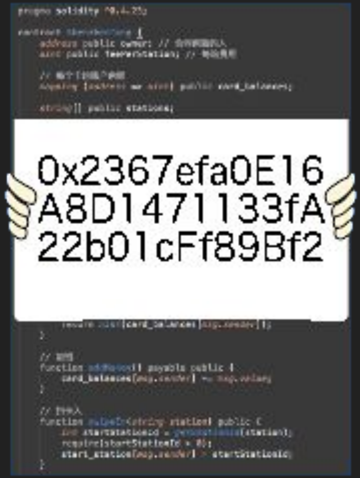
比特币和以太坊有什么有什么区别?

比特币

1Hz96kJKF2HL
PGY15JWLB5m
9qGNxvt8tHJ

以太坊

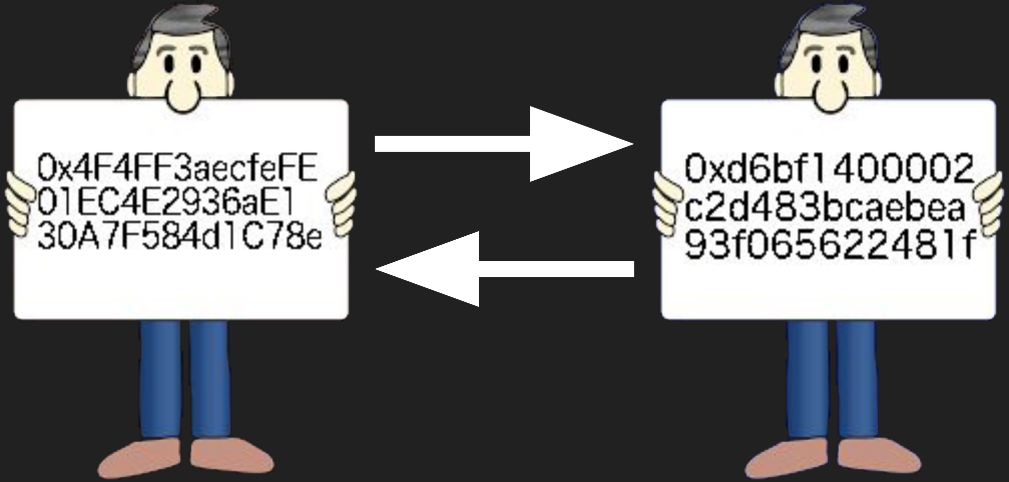
0x4F4FF3aecfeFE
01EC4E2936aE1
30A7F584d1C78e



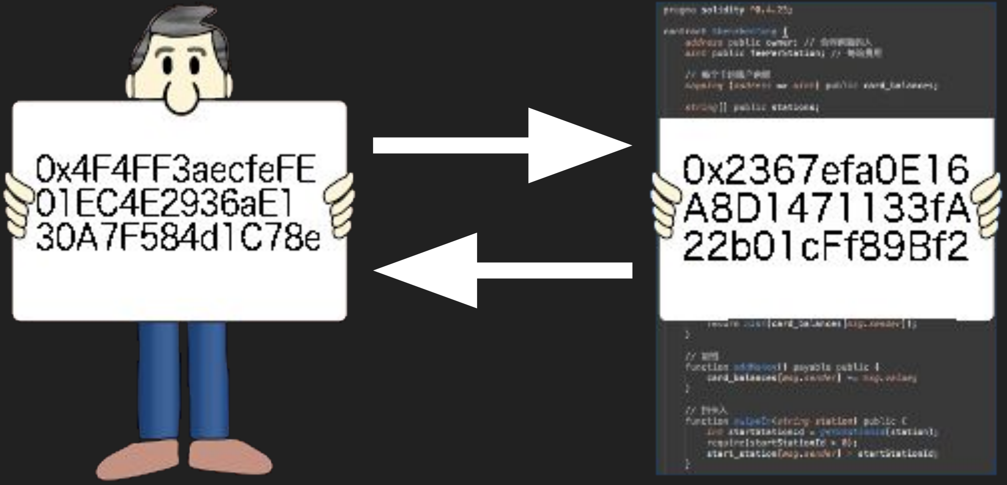
```
pragma solidity ^0.4.25;

contract [illegible] {
    address public owner; // [illegible]
    uint public feePerStation; // [illegible]

    // [illegible]
    mapping (address => uint) public card_balances;

    string[] public stations;
```

```
        return uint(card_balances[msg.sender]);
    }

    // [illegible]
    function addMoney() payable public {
        card_balances[msg.sender] += msg.value;
    }

    // [illegible]
    function swipeIn(string station) public {
        int startStationId = getStationId(station);
        require(startStationId >= 0);
        start_station[msg.sender] = startStationId;
    }
```

以太坊


0x4F4FF3aecfeFE
01EC4E2936aE1
30A7F584d1C78e
0xd6bf1400002
c2d483bcaebea
93f065622481f

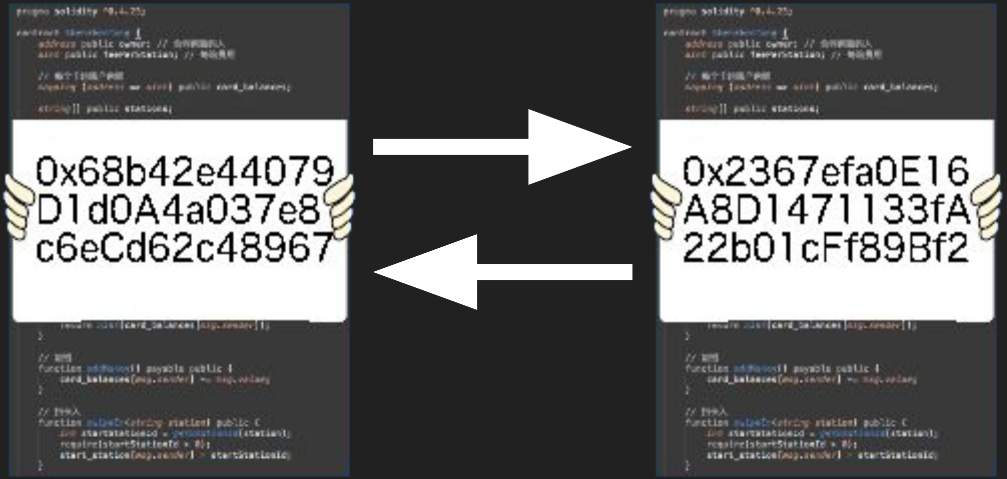
以太坊



```
pragma solidity ^0.4.25;

contract [illegible] {
    address public owner; // [illegible]
    uint public fee[illegible]; // [illegible]

    // [illegible]
    mapping (address => uint) public card_balances;

    string[] public stations;
```

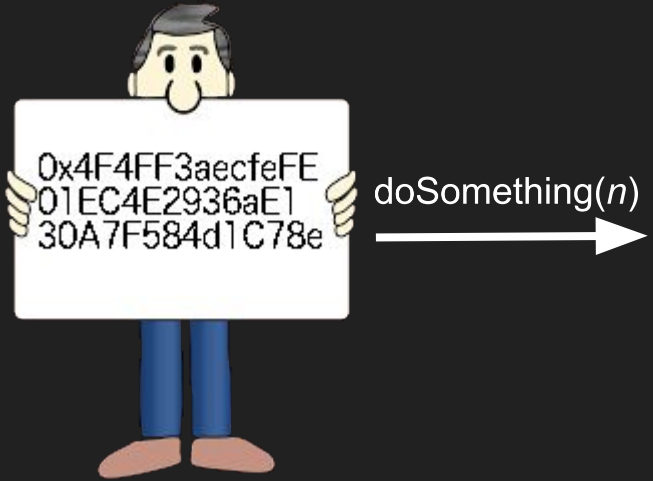
```
        return uint(card_balances[msg.sender]);
    }

    // [illegible]
    function addMoney() payable public {
        card_balances[msg.sender] += msg.value;
    }

    // [illegible]
    function [illegible](string station) public {
        int startStationId = [illegible](station);
        require(startStationId >= 0);
        start_station[msg.sender] = startStationId;
    }
```

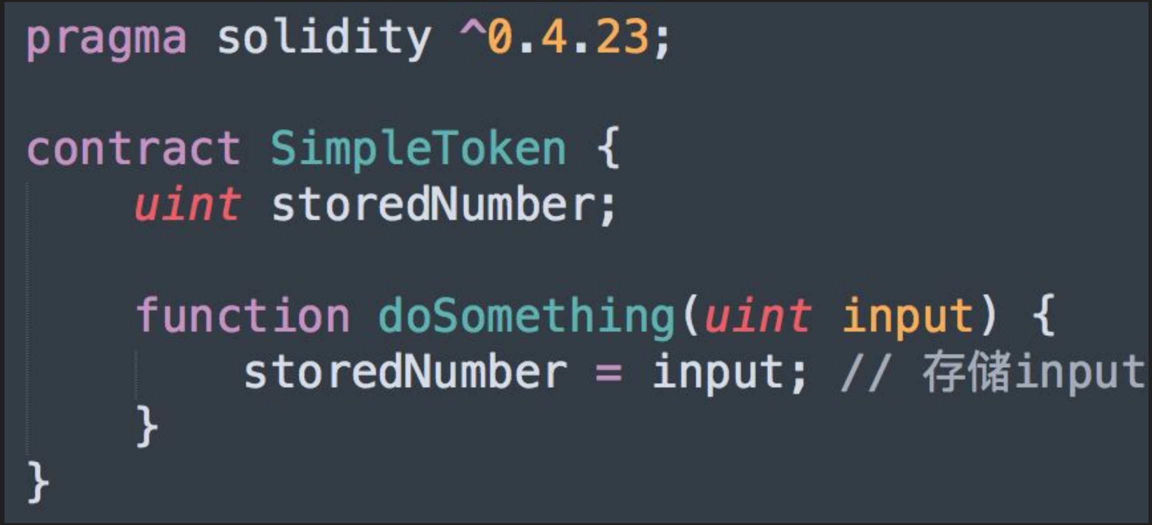
以太坊


0x68b42e44079D1d0A4a037e8c6eCd62c48967
0x2367efa0E16A8D1471133fA22b01cFf89Bf2

```
pragma solidity ^0.4.25;

contract [illegible] {
    address public owner; // [illegible]
    uint public [illegible]; // [illegible]

    // [illegible]
    mapping (address => uint) public card_balances;

    string[] public stations;
[illegible]
        return [illegible](card_balances[msg.sender]);
    }

    // [illegible]
    function [illegible]() payable public {
        card_balances[msg.sender] += msg.value;
    }

    // [illegible]
    function [illegible](string station) public {
        int startStationId = [illegible](station);
        require(startStationId > 0);
        start_station[msg.sender] = startStationId;
    }
```

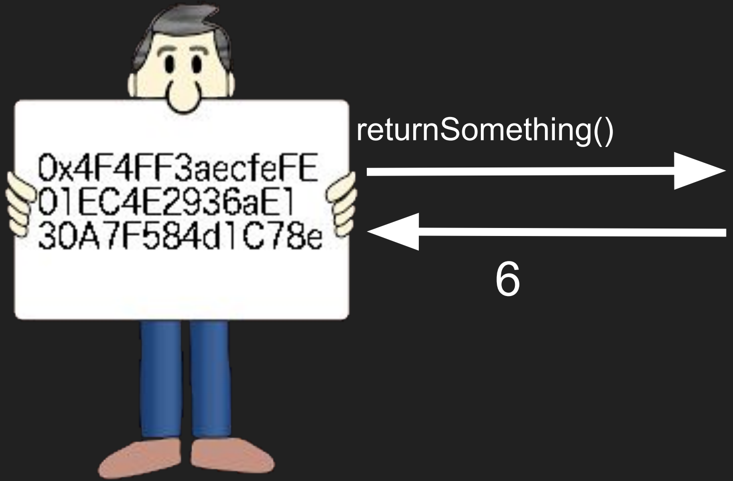
怎么用一个智能合约?

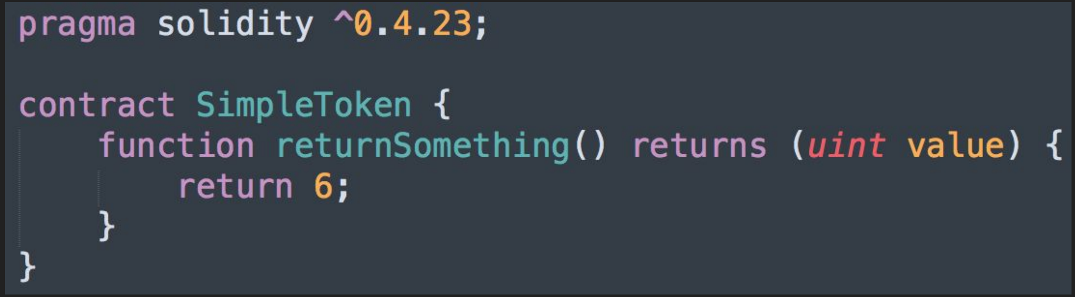




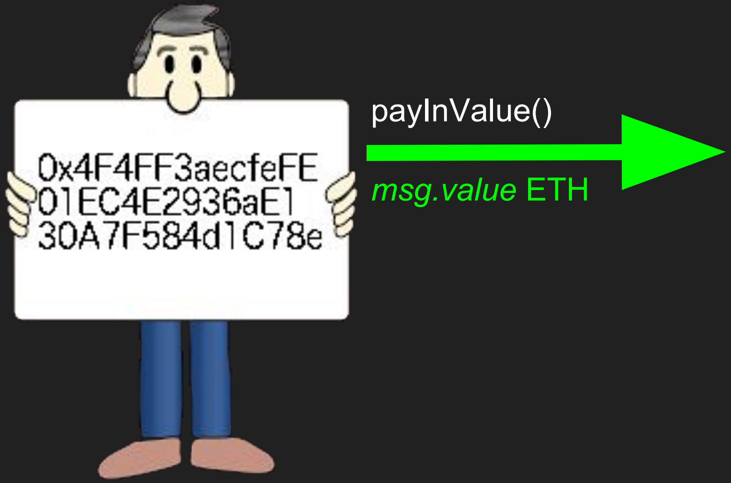
```
pragma solidity ^0.4.23;

contract SimpleToken {
    uint storedNumber;

    function doSomething(uint input) {
        storedNumber = input; // 存储input
    }
}
```

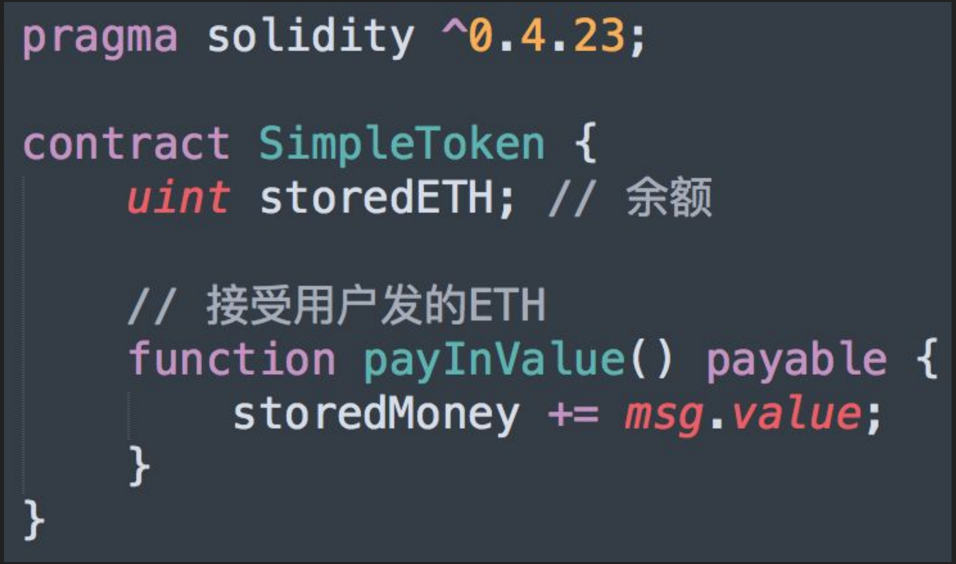
怎么用一个智能合约?




```
pragma solidity ^0.4.23;

contract SimpleToken {
    function returnSomething() returns (uint value) {
        return 6;
    }
}
```

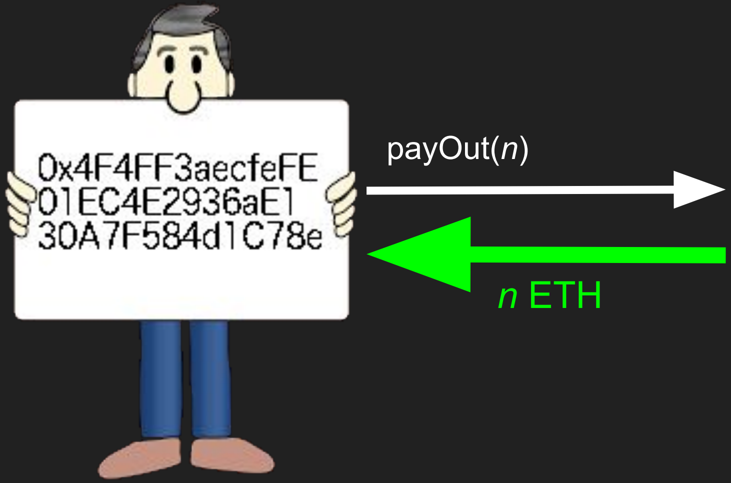
怎么用一个智能合约?

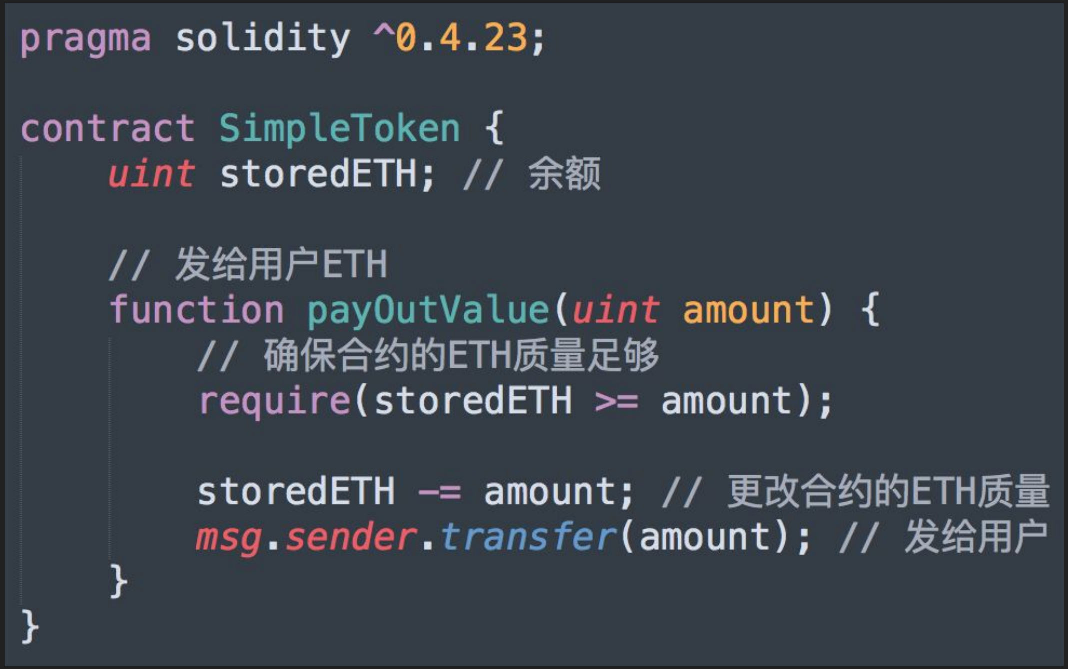




```
pragma solidity ^0.4.23;

contract SimpleToken {
    uint storedETH; // 余额

    // 接受用户发的ETH
    function payInValue() payable {
        storedMoney += msg.value;
    }
}
```

怎么用一个智能合约?



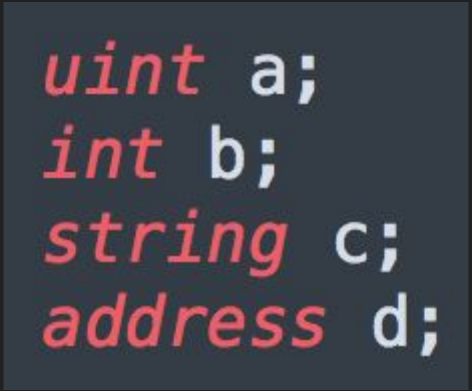


```
pragma solidity ^0.4.23;

contract SimpleToken {
    uint storedETH; // 余额

    // 发给用户ETH
    function payOutValue(uint amount) {
        // 确保合约的ETH质量足够
        require(storedETH >= amount);

        storedETH -= amount; // 更改合约的ETH质量
        msg.sender.transfer(amount); // 发给用户
    }
}
```

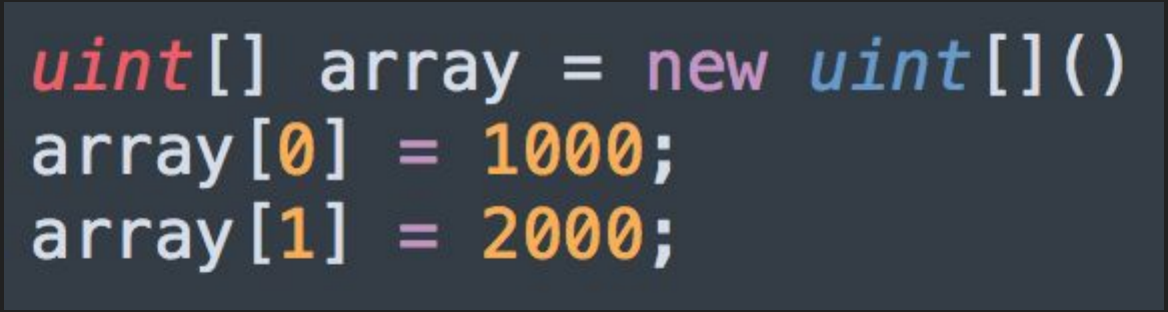
Solidity

https://solidity.readthedocs.io/en/latest/types.html

Solidity



```
uint a;
int b;
string c;
address d;
```

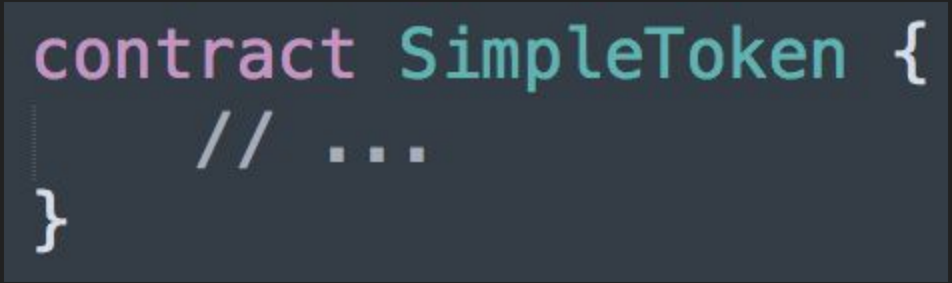


```
uint[] array = new uint[]()
array[0] = 1000;
array[1] = 2000;
```

Solidity

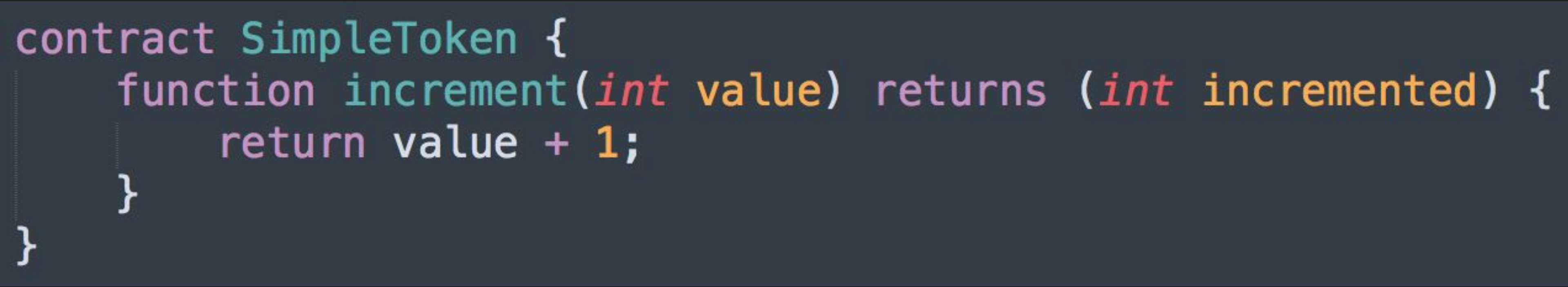
```
mapping (address => uint) public balances; // 每个ETH地址的令牌余额
address myAddress = 0x123; // 我的地址
uint myBalance = balances[myAddress]; // 检查我的令牌余额
```

Solidity



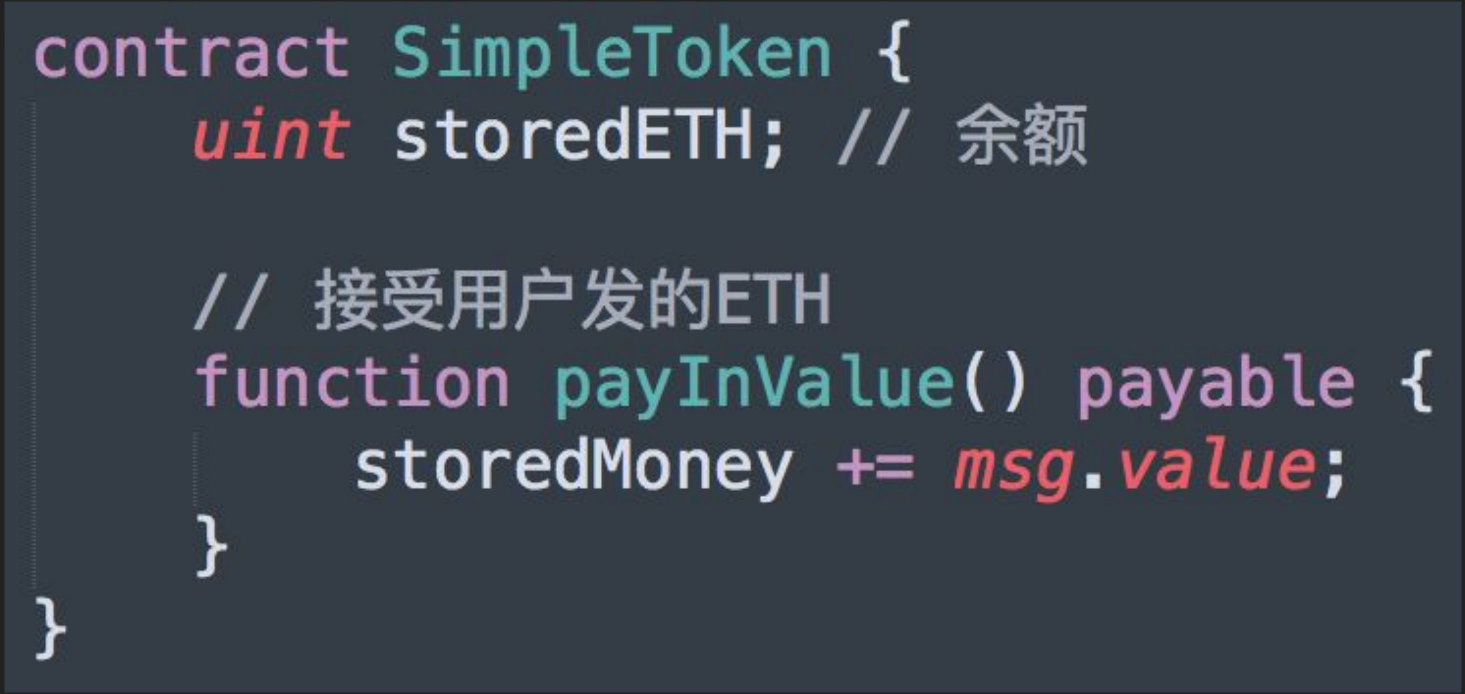
```
contract SimpleToken {
    // ...
}
```

Solidity



```
contract SimpleToken {
    function increment(int value) returns (int incremented) {
        return value + 1;
    }
}
```

Solidity



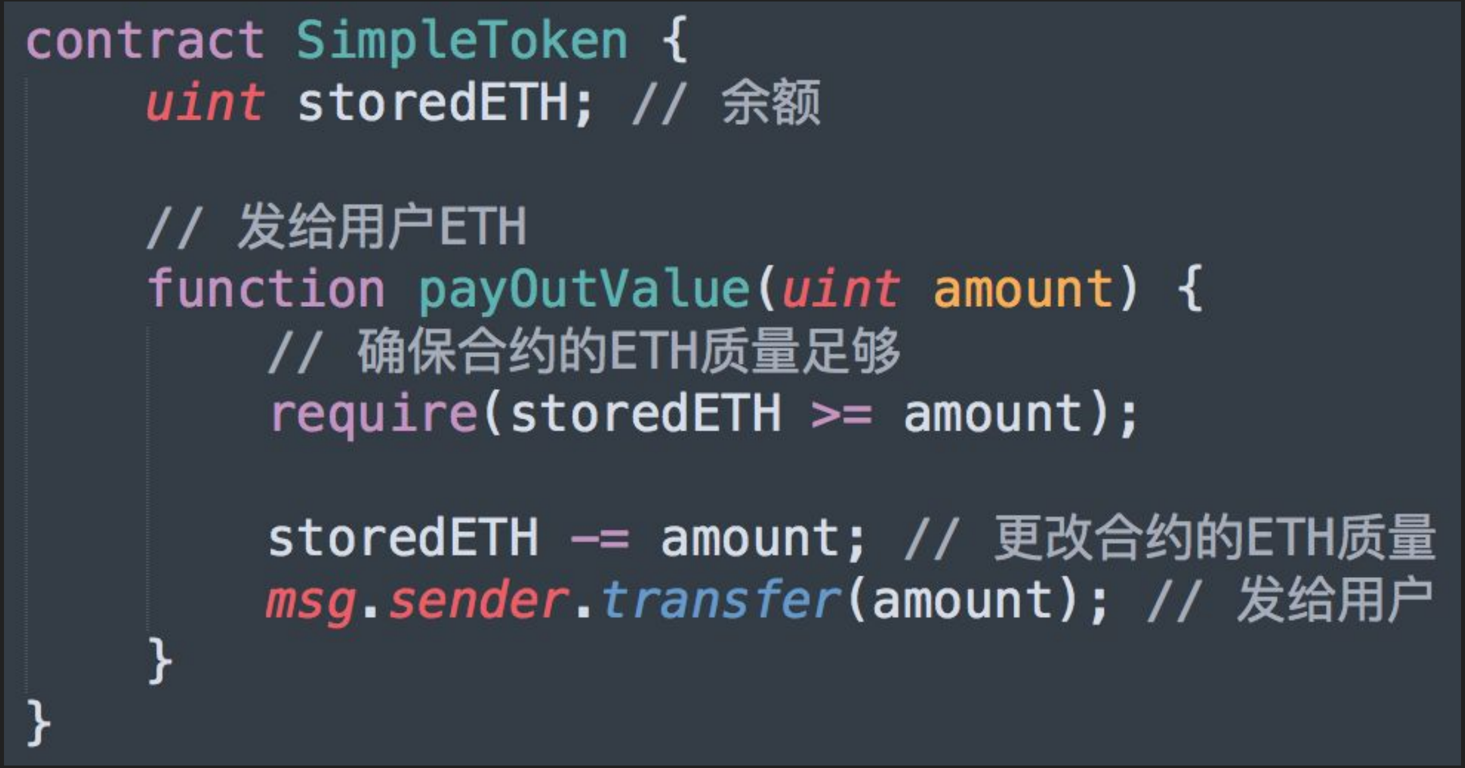
```
contract SimpleToken {
    uint storedETH; // 余额

    // 接受用户发的ETH
    function payInValue() payable {
        storedMoney += msg.value;
    }
}
```

Solidity

```
require(n > 10); // require(条件)
```

```
// 发给另外的ETH地址令牌
function send(address recipient, uint amount) {
    require(balances[msg.sender] >= amount);
    balances[msg.sender] -= amount;
    balances[recipient] += amount;
}
```

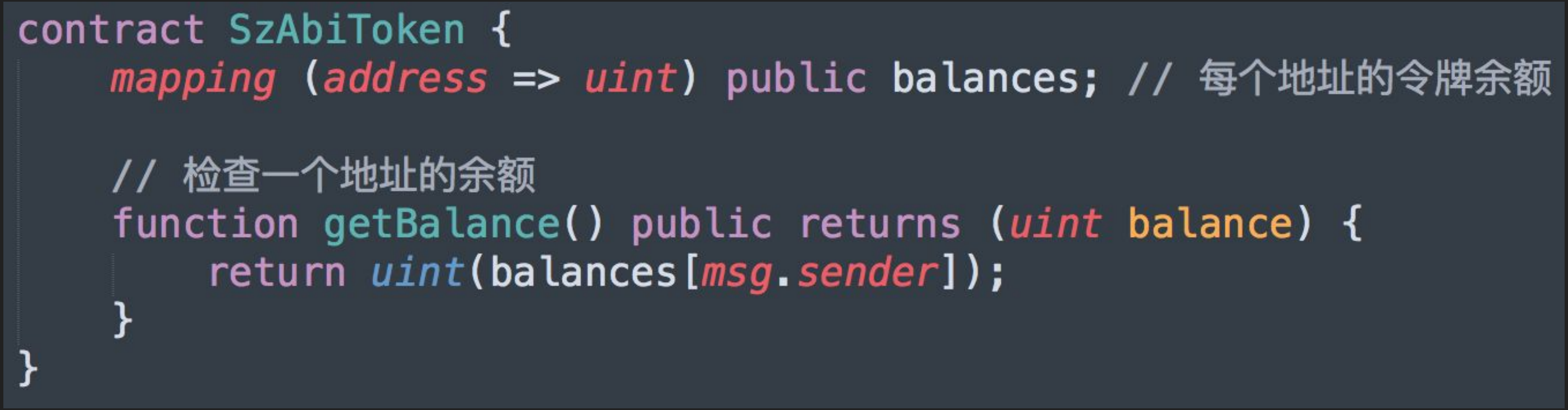
Solidity



```
contract SimpleToken {
    uint storedETH; // 余额

    // 发给用户ETH
    function payOutValue(uint amount) {
        // 确保合约的ETH质量足够
        require(storedETH >= amount);

        storedETH -= amount; // 更改合约的ETH质量
        msg.sender.transfer(amount); // 发给用户
    }
}
```

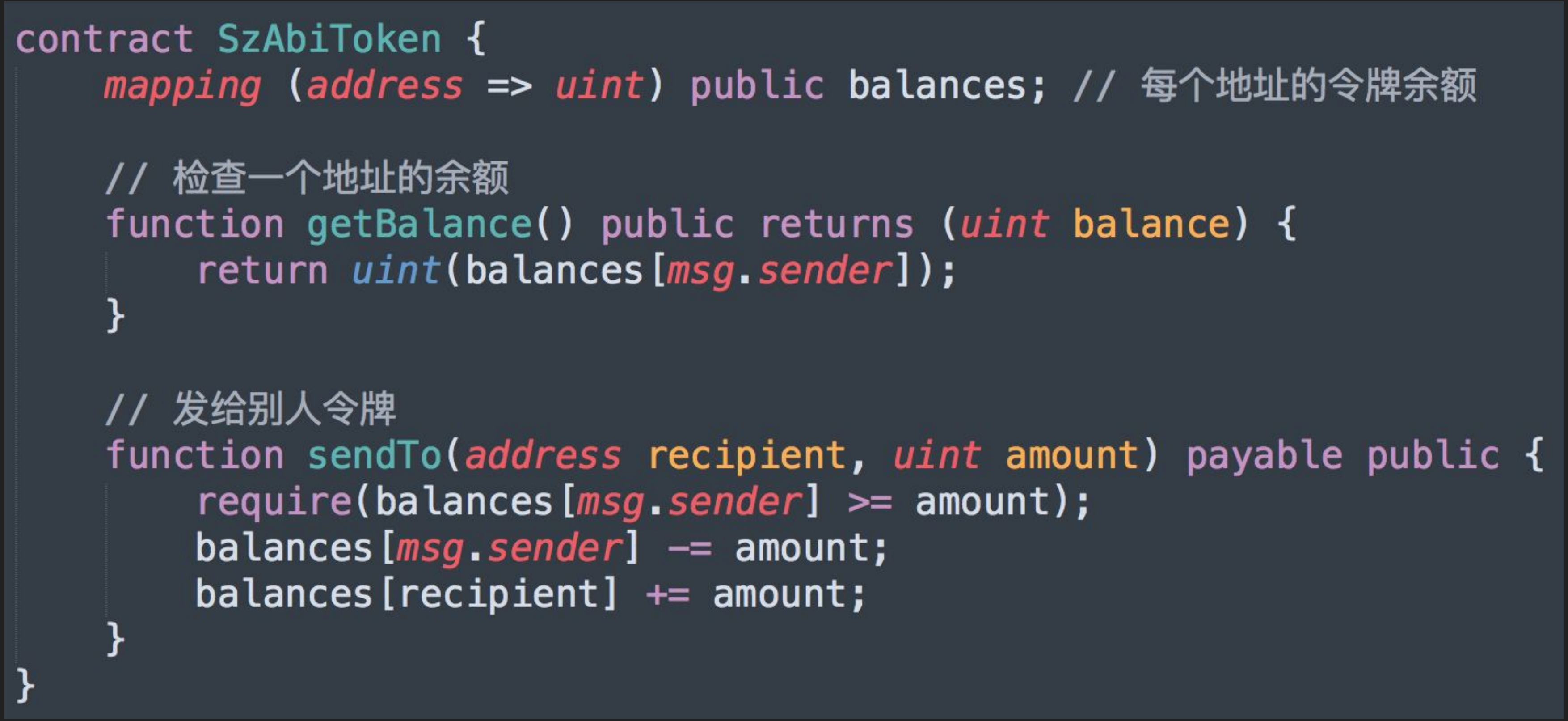
写一个ICO的代笔

让用户检查他的代笔余额

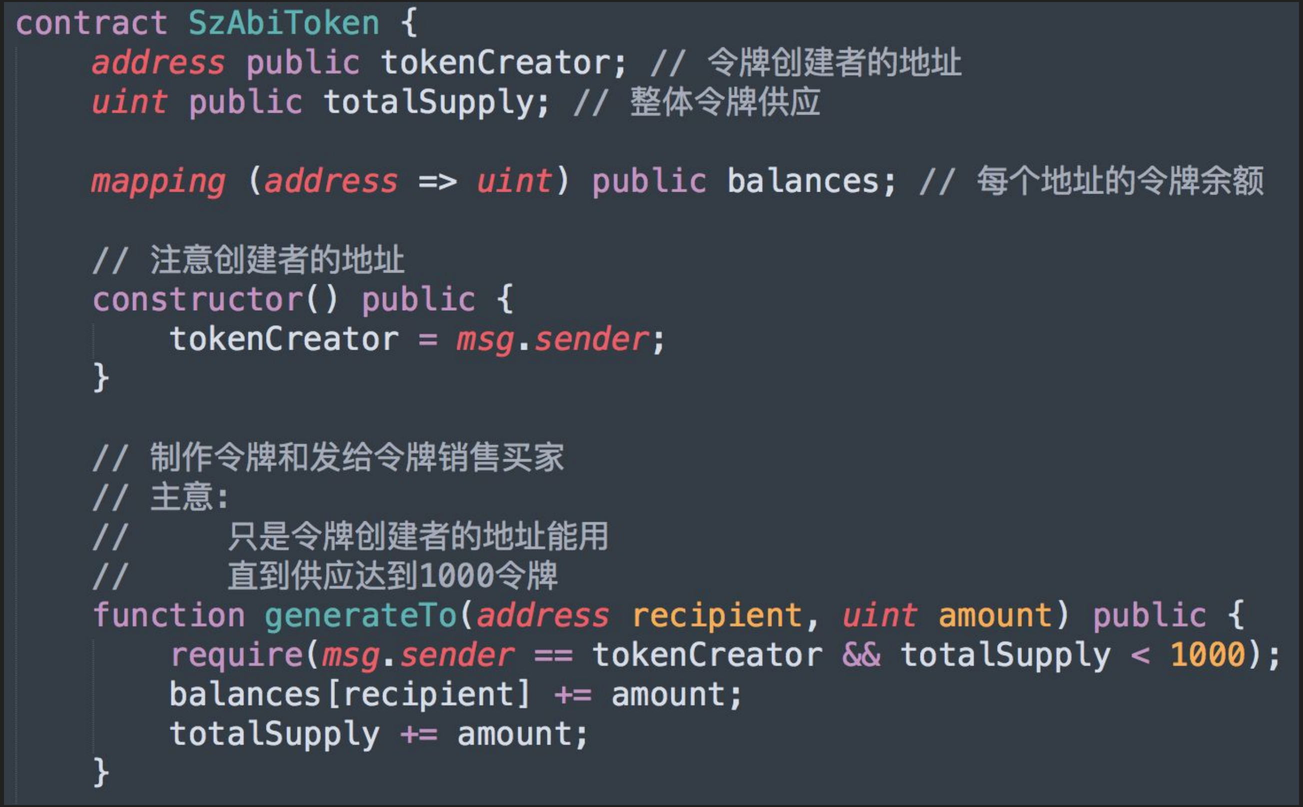


```
contract SzAbiToken {
    mapping (address => uint) public balances; // 每个地址的令牌余额

    // 检查一个地址的余额
    function getBalance() public returns (uint balance) {
        return uint(balances[msg.sender]);
    }
}
```

让用户发给别人代笔

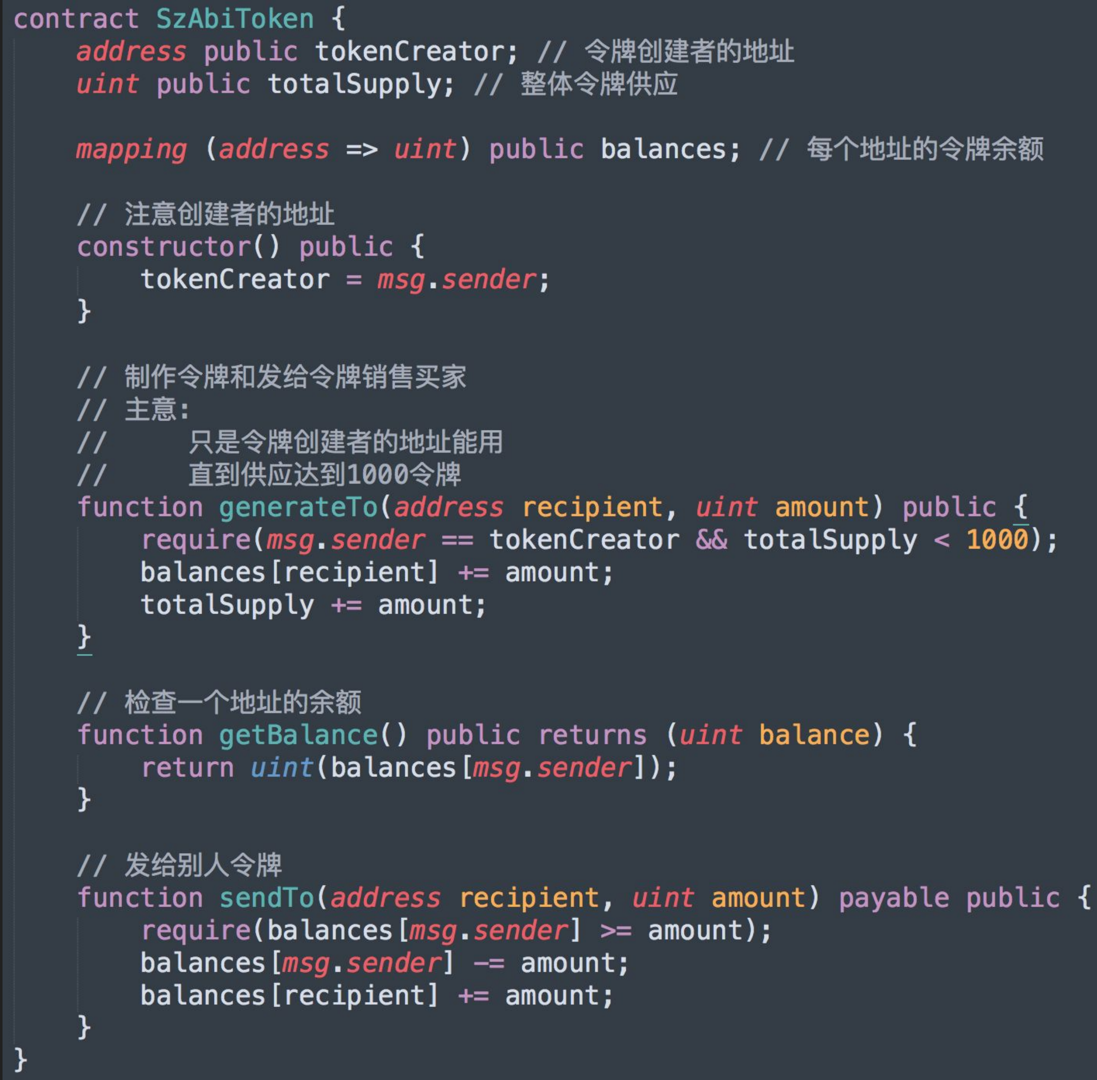

```
contract SzAbiToken {
    mapping (address => uint) public balances; // 每个地址的令牌余额

    // 检查一个地址的余额
    function getBalance() public returns (uint balance) {
        return uint(balances[msg.sender]);
    }

    // 发给别人令牌
    function sendTo(address recipient, uint amount) payable public {
        require(balances[msg.sender] >= amount);
        balances[msg.sender] -= amount;
        balances[recipient] += amount;
    }
}
```

让合约创造者分发代笔



```
contract SzAbiToken {
    address public tokenCreator; // 令牌创建者的地址
    uint public totalSupply; // 整体令牌供应

    mapping (address => uint) public balances; // 每个地址的令牌余额

    // 注意创建者的地址
    constructor() public {
        tokenCreator = msg.sender;
    }

    // 制作令牌和发给令牌销售买家
    // 主意:
    //      只是令牌创建者的地址能用
    //      直到供应达到1000令牌
    function generateTo(address recipient, uint amount) public {
        require(msg.sender == tokenCreator && totalSupply < 1000);
        balances[recipient] += amount;
        totalSupply += amount;
    }
```

所有的



```
contract SzAbiToken {
    address public tokenCreator; // 令牌创建者的地址
    uint public totalSupply; // 整体令牌供应

    mapping (address => uint) public balances; // 每个地址的令牌余额

    // 注意创建者的地址
    constructor() public {
        tokenCreator = msg.sender;
    }

    // 制作令牌和发给令牌销售买家
    // 主意:
    //      只是令牌创建者的地址能用
    //      直到供应达到1000令牌
    function generateTo(address recipient, uint amount) public {
        require(msg.sender == tokenCreator && totalSupply < 1000);
        balances[recipient] += amount;
        totalSupply += amount;
    }

    // 检查一个地址的余额
    function getBalance() public returns (uint balance) {
        return uint(balances[msg.sender]);
    }

    // 发给别人令牌
    function sendTo(address recipient, uint amount) payable public {
        require(balances[msg.sender] >= amount);
        balances[msg.sender] -= amount;
        balances[recipient] += amount;
    }
}
```

工具

Truffle

Ganache

Ganache

我们会一起写和部署一个完全分布的深圳通系统

https://github.com/toinetoine/SzAbiTalkDappDevCode