Solidity与智能合约编写

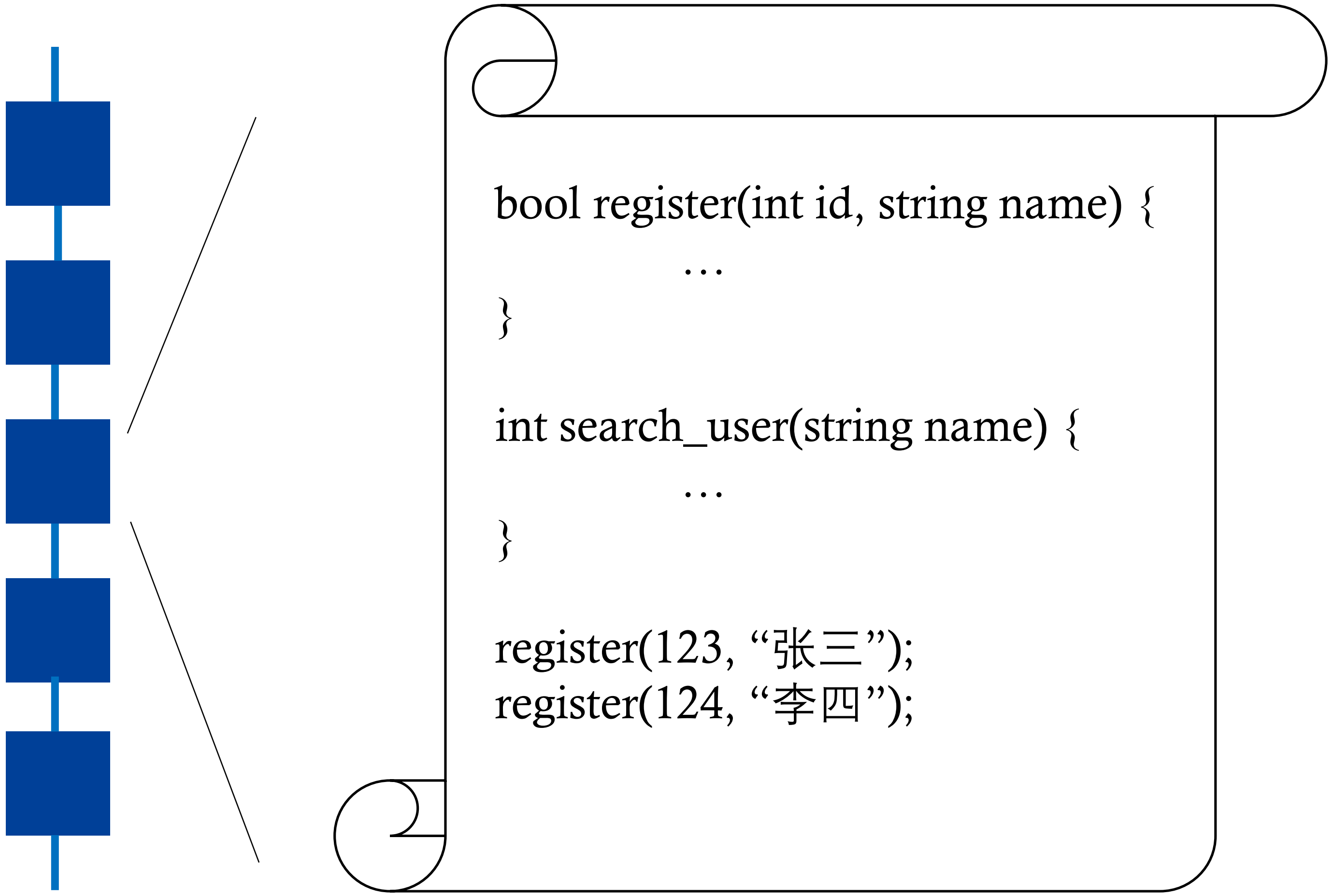
马丁
上海交通大学

一、以太坊虚拟机与Solidity

区块链共识

达成一致！

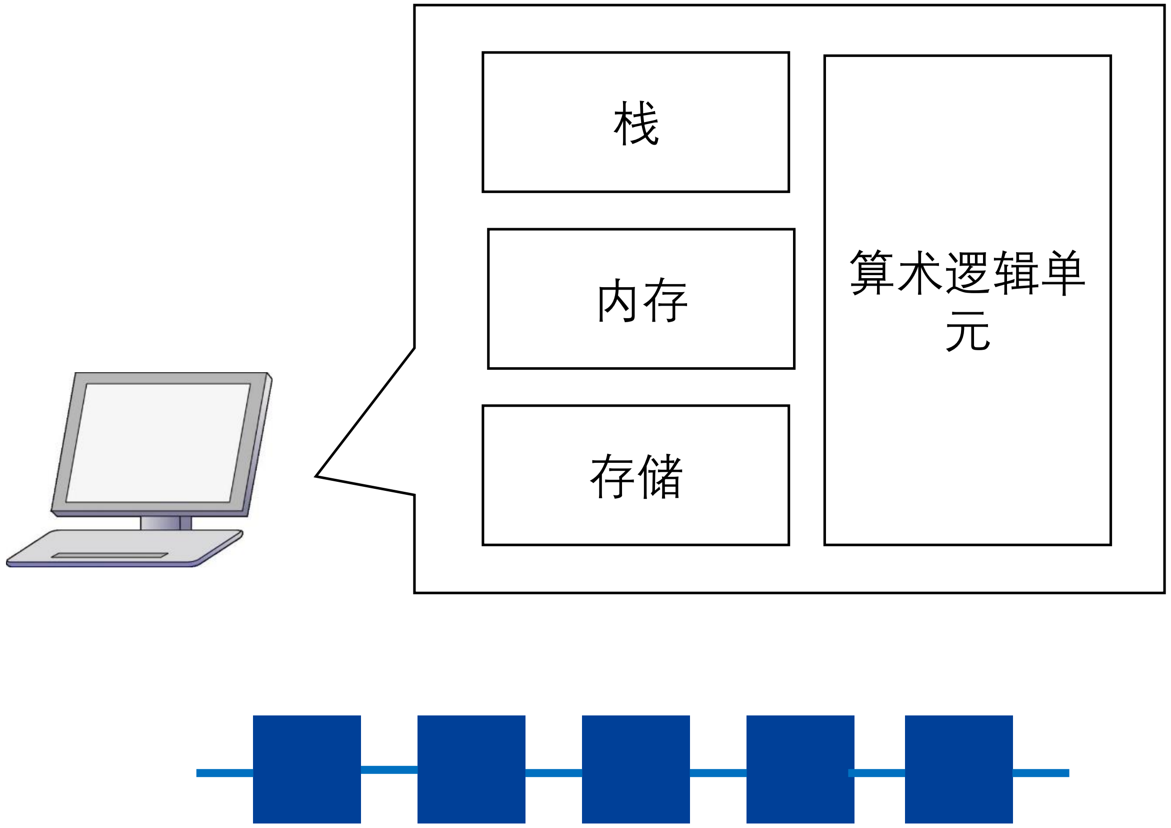
共识和代码



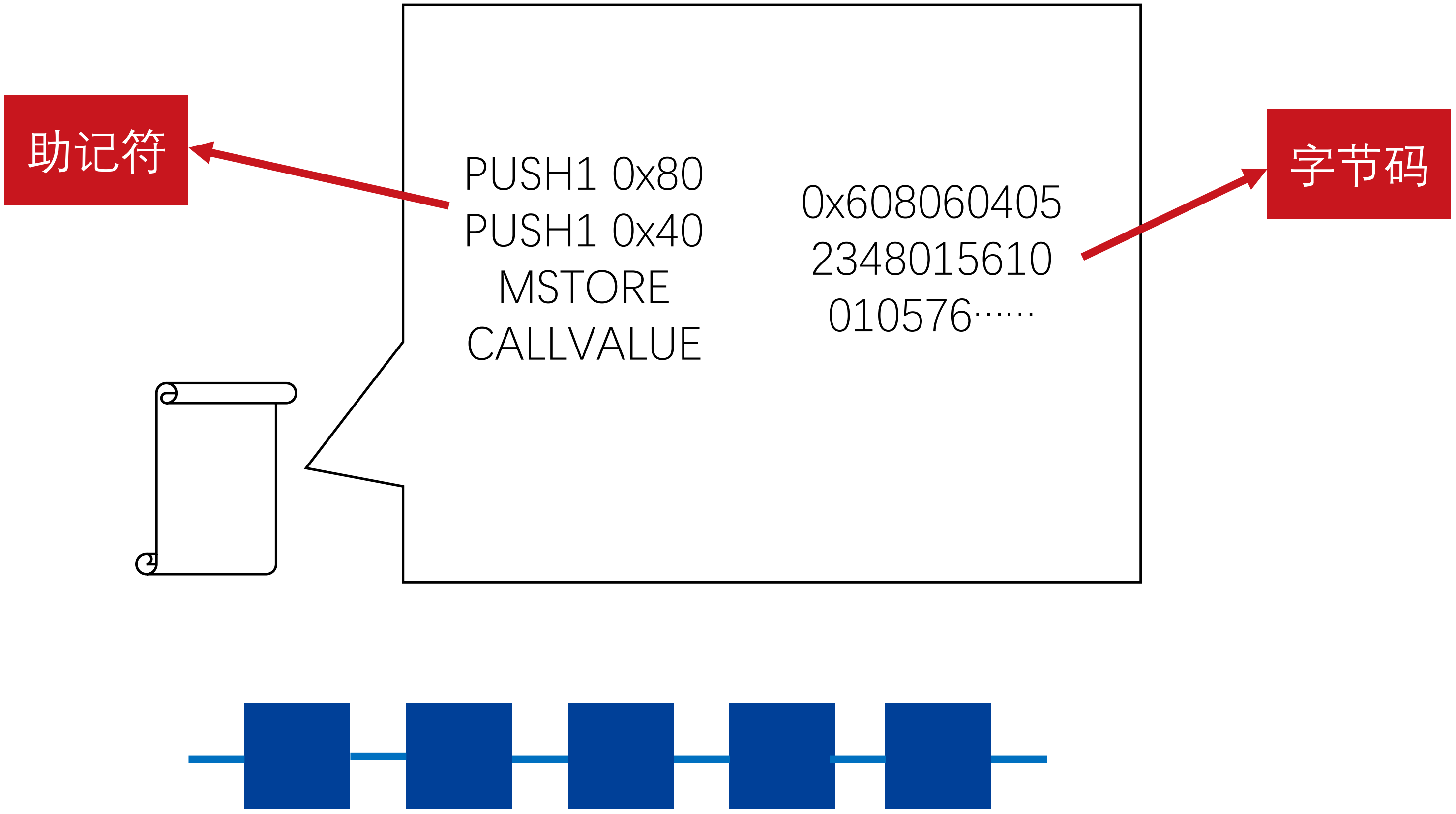
```
bool register(int id, string name) {
            …
}

int search_user(string name) {
            …
}

register(123, “张三”);
register(124, “李四”);
```

智能合约！

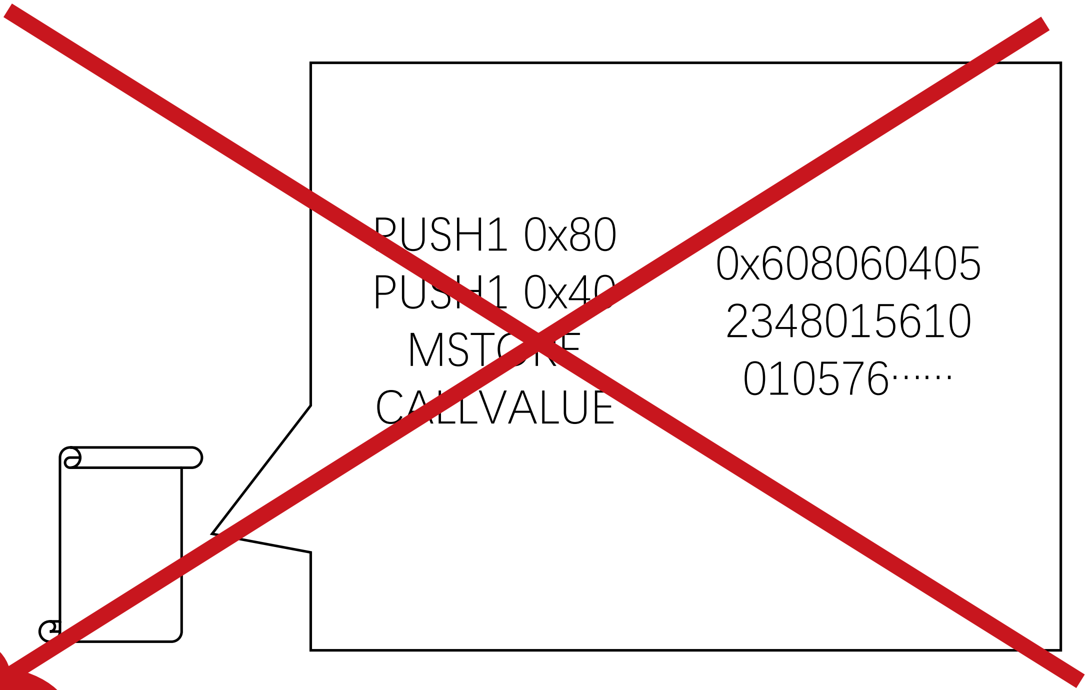
以太坊虚拟机



以太坊虚拟机


助记符
PUSH1 0x80
PUSH1 0x40
MSTORE
CALLVALUE
0x608060405
2348015610
010576……
字节码

Solidity

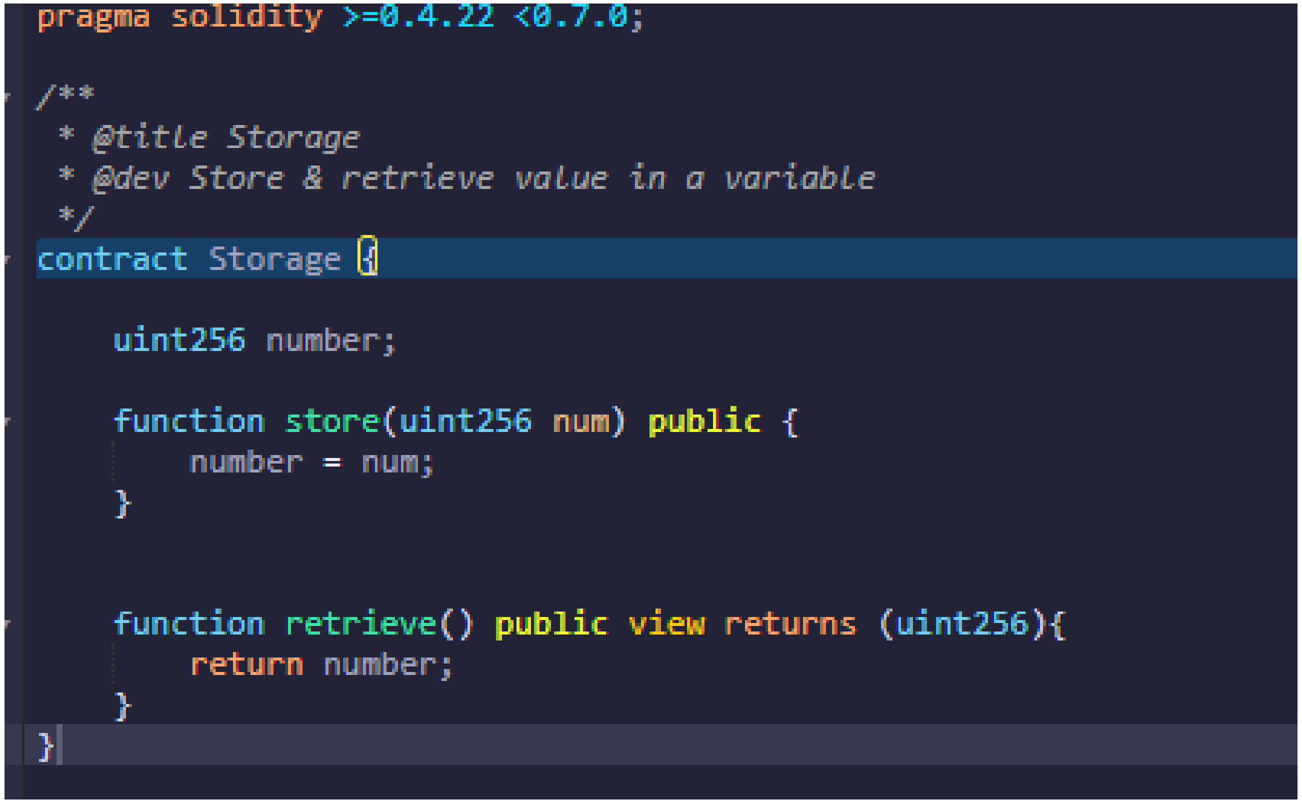


~~PUSH1 0x80~~
~~PUSH1 0x40~~
~~MSTORE~~
~~CALLVALUE~~

~~0x608060405~~
~~2348015610~~
~~010576……~~

Solidity能够帮助简
单书写合约!

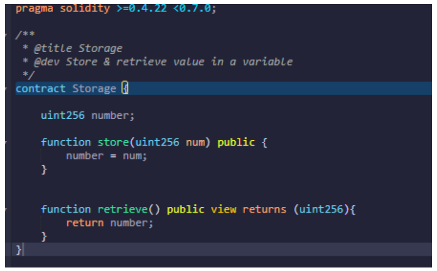
Solidity



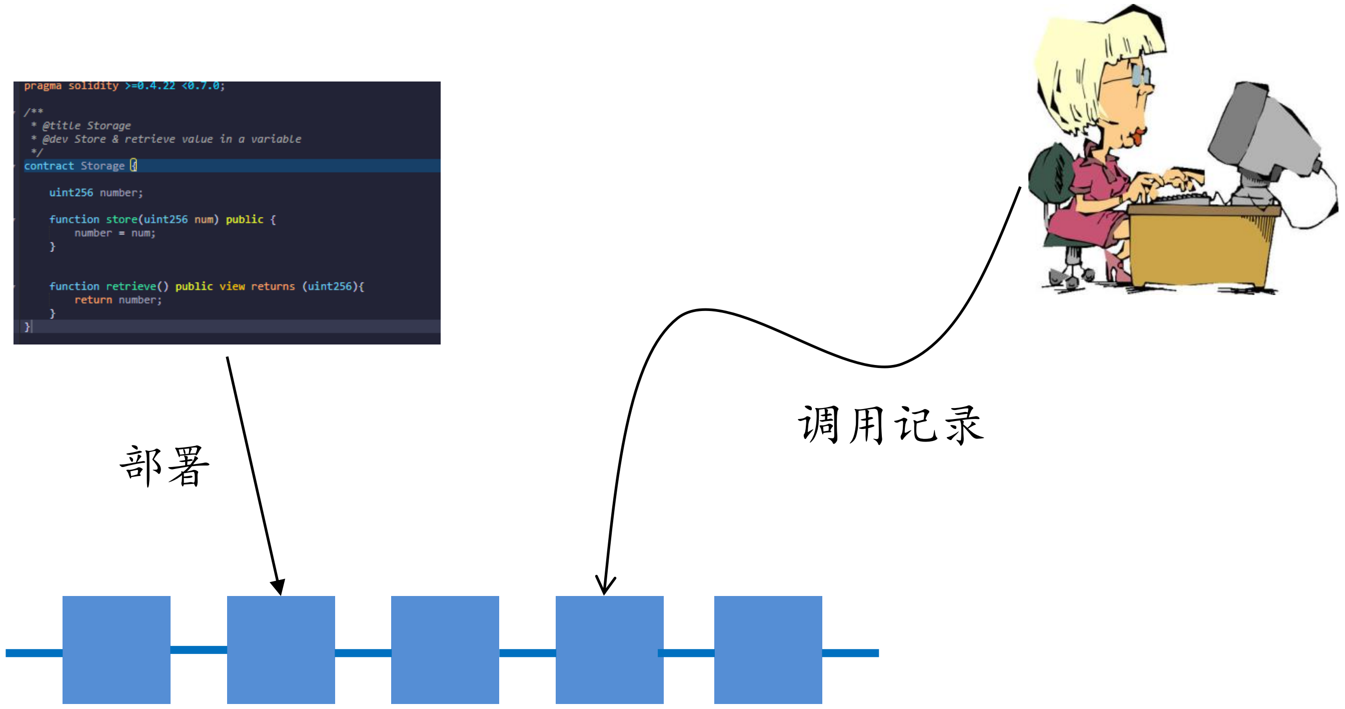
```
pragma solidity >=0.4.22 <0.7.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */
contract Storage {

    uint256 number;

    function store(uint256 num) public {
        number = num;
    }


    function retrieve() public view returns (uint256){
        return number;
    }
}
```

6080604052348015610010576000080fd5b5060c78061001f6000396000f3fe6080604052348015600f57600080fd5b506004361060325760003560e01c80632e64cec114603757806360573 61d146053575b600080fd5b603d607e…

部署和调用



```
pragma solidity >=0.4.22 <0.7.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */
contract Storage {

    uint256 number;

    function store(uint256 num) public {
        number = num;
    }


    function retrieve() public view returns (uint256){
        return number;
    }
}
```



二、开发环境搭建

开发环境

开发环境

- 单节点虚拟环境
- 真实环境

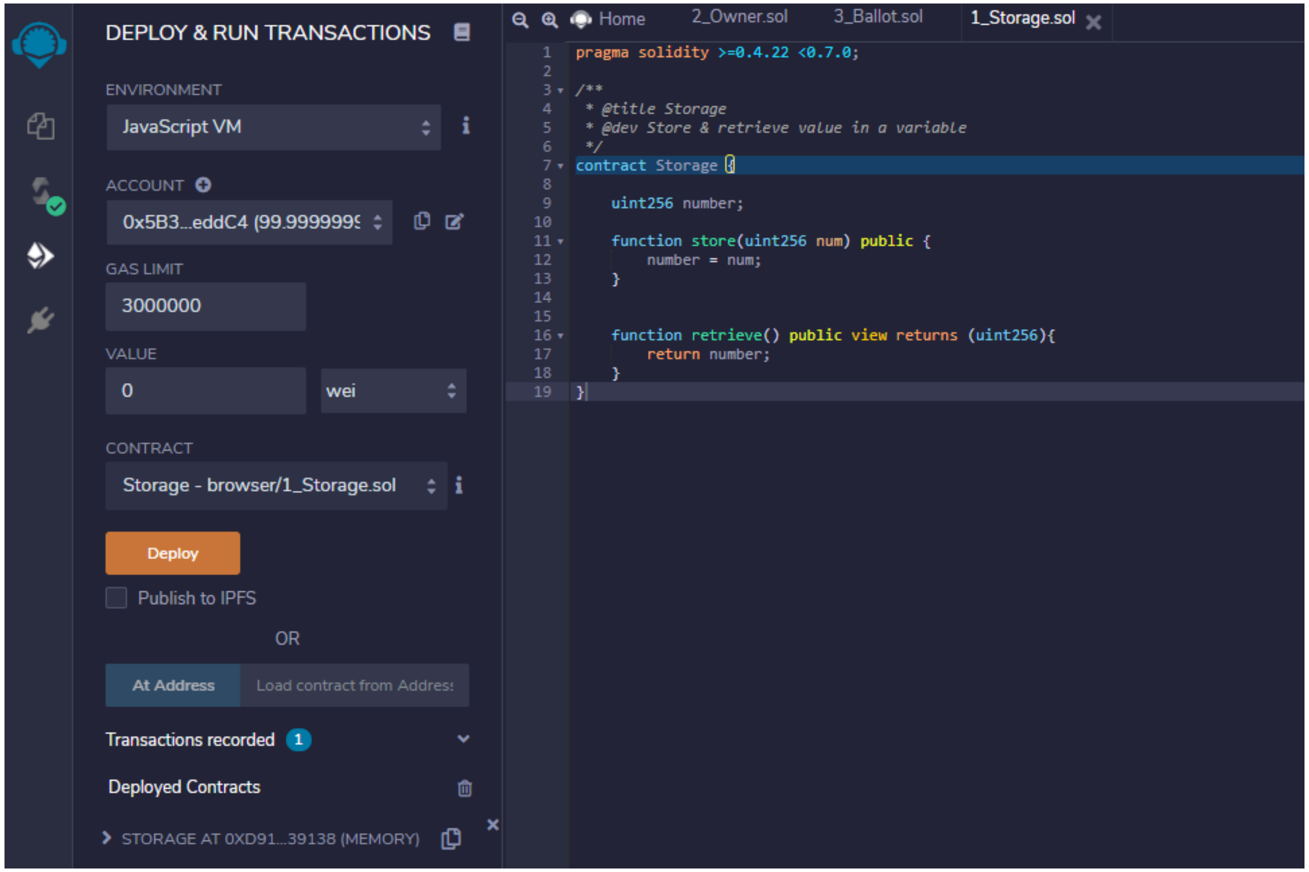
虚拟环境——JavaScript VM

- JavaScript构造的虚拟链环境
- 直接可以在网页开发使用

https://remix.ethereum.org/

虚拟环境——JavaScript VM

演示：在remix上部署并且调用一个简单的存储合约



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: JavaScript VM

ACCOUNT: 0x5B3...eddC4 (99.999999

GAS LIMIT: 3000000

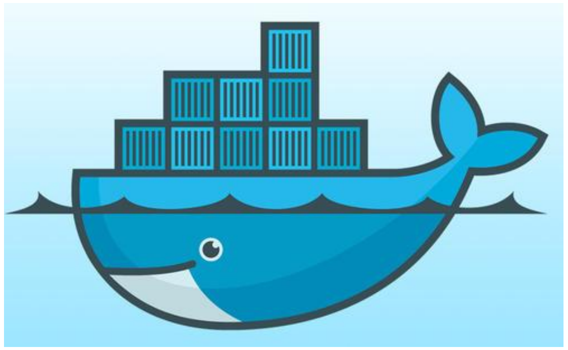
VALUE: 0 wei

CONTRACT: Storage - browser/1_Storage.sol

```
pragma solidity >=0.4.22 <0.7.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */
contract Storage {

    uint256 number;

    function store(uint256 num) public {
        number = num;
    }


    function retrieve() public view returns (uint256){
        return number;
    }
}
```

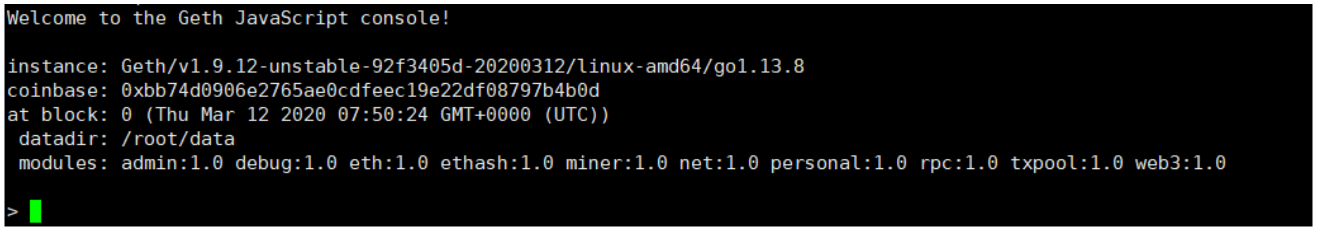
真实环境

- 利用以太坊的钱包节点geth，组成一个真正的以太坊分布式网络
- 可以利用docker，启动若干个以太坊节点，并且互联
- 存在矿工挖矿，此时可以针对某个账户，以其身份发起部署合约，待合约上链后，利用它或其它账户调用合约

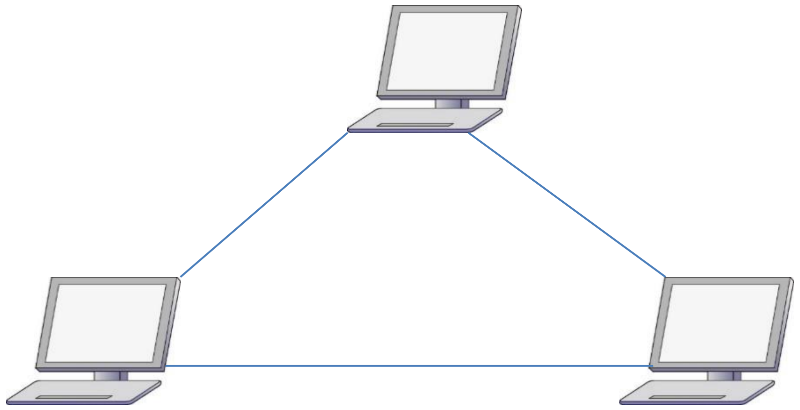


真实环境

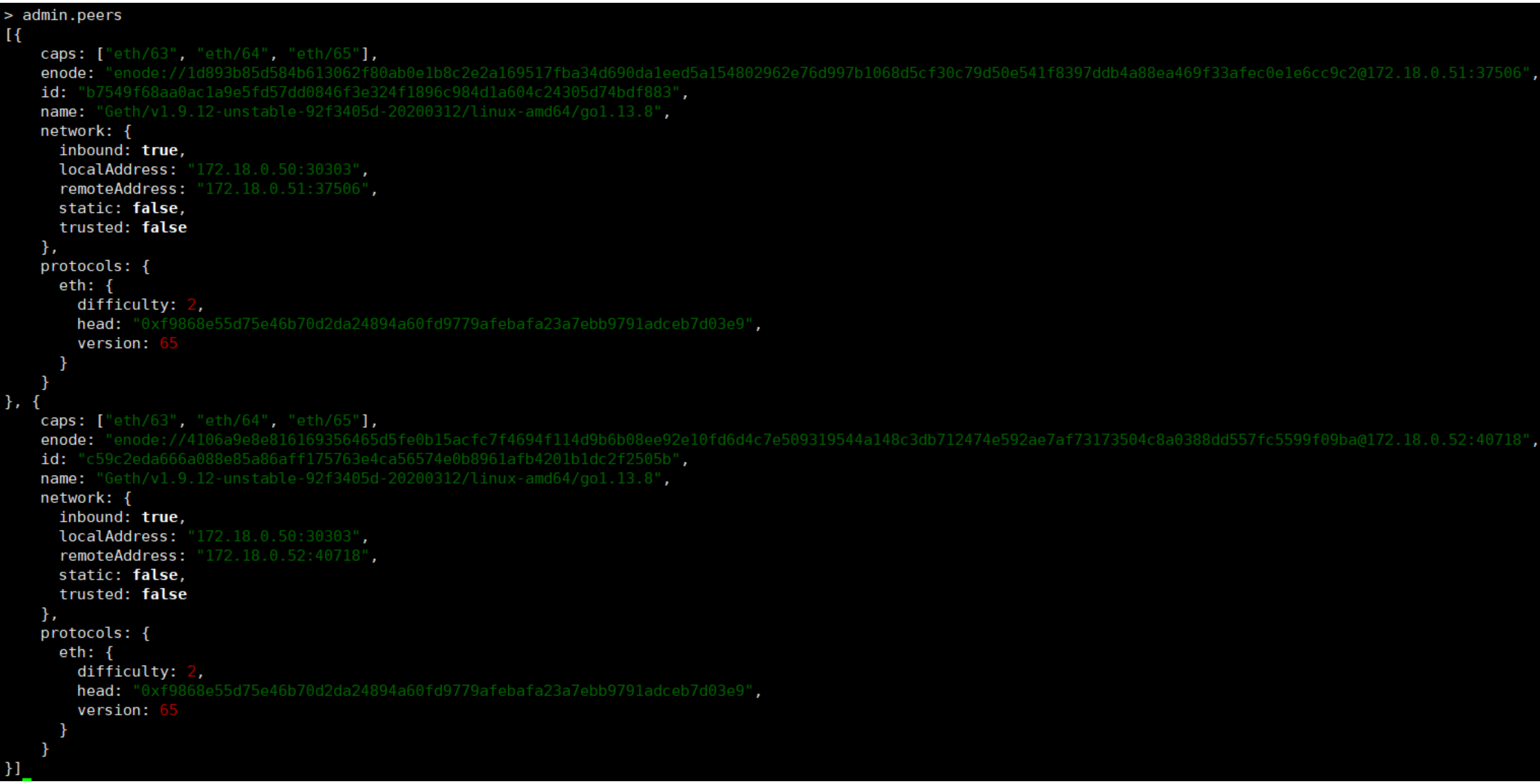
- 可以换做演示



```
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.12-unstable-92f3405d-20200312/linux-amd64/go1.13.8
coinbase: 0xbb74d0906e2765ae0cdfeec19e22df08797b4b0d
at block: 0 (Thu Mar 12 2020 07:50:24 GMT+0000 (UTC))
 datadir: /root/data
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> 
```

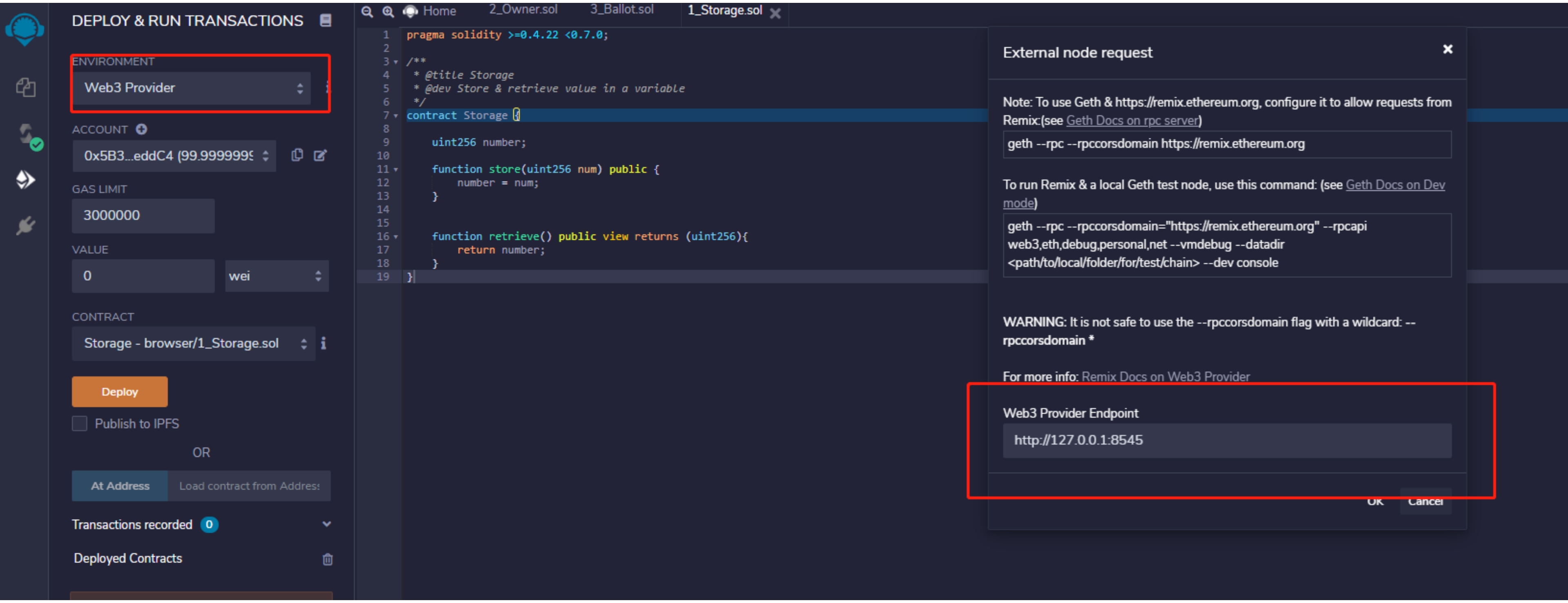


真实环境



```
> admin.peers
[{
    caps: ["eth/63", "eth/64", "eth/65"],
    enode: "enode://1d893b85d584b613062f80ab0e1b8c2e2a169517fba34d690da1eed5a154802962e76d997b1068d5cf30c79d50e541f8397ddb4a88ea469f33afec0e1e6cc9c2@172.18.0.51:37506",
    id: "b7549f68aa0ac1a9e5fd57dd0846f3e324f1896c984d1a604c24305d74bdf883",
    name: "Geth/v1.9.12-unstable-92f3405d-20200312/linux-amd64/go1.13.8",
    network: {
      inbound: true,
      localAddress: "172.18.0.50:30303",
      remoteAddress: "172.18.0.51:37506",
      static: false,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 2,
        head: "0xf9868e55d75e46b70d2da24894a60fd9779afebafa23a7ebb9791adceb7d03e9",
        version: 65
      }
    }
}, {
    caps: ["eth/63", "eth/64", "eth/65"],
    enode: "enode://4106a9e8e816169356465d5fe0b15acfc7f4694f114d9b6b08ee92e10fd6d4c7e509319544a148c3db712474e592ae7af73173504c8a0388dd557fc5599f09ba@172.18.0.52:40718",
    id: "c59c2eda666a088e85a86aff175763e4ca56574e0b8961afb4201b1dc2f2505b",
    name: "Geth/v1.9.12-unstable-92f3405d-20200312/linux-amd64/go1.13.8",
    network: {
      inbound: true,
      localAddress: "172.18.0.50:30303",
      remoteAddress: "172.18.0.52:40718",
      static: false,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 2,
        head: "0xf9868e55d75e46b70d2da24894a60fd9779afebafa23a7ebb9791adceb7d03e9",
        version: 65
      }
    }
}]
```

真实环境



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Web3 Provider

ACCOUNT: 0x5B3...eddC4 (99.999999

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: Storage - browser/1_Storage.sol

```
pragma solidity >=0.4.22 <0.7.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */
contract Storage {

    uint256 number;

    function store(uint256 num) public {
        number = num;
    }


    function retrieve() public view returns (uint256){
        return number;
    }
}
```

External node request

Note: To use Geth & https://remix.ethereum.org, configure it to allow requests from Remix:(see Geth Docs on rpc server)

```
geth --rpc --rpccorsdomain https://remix.ethereum.org
```

To run Remix & a local Geth test node, use this command: (see Geth Docs on Dev mode)

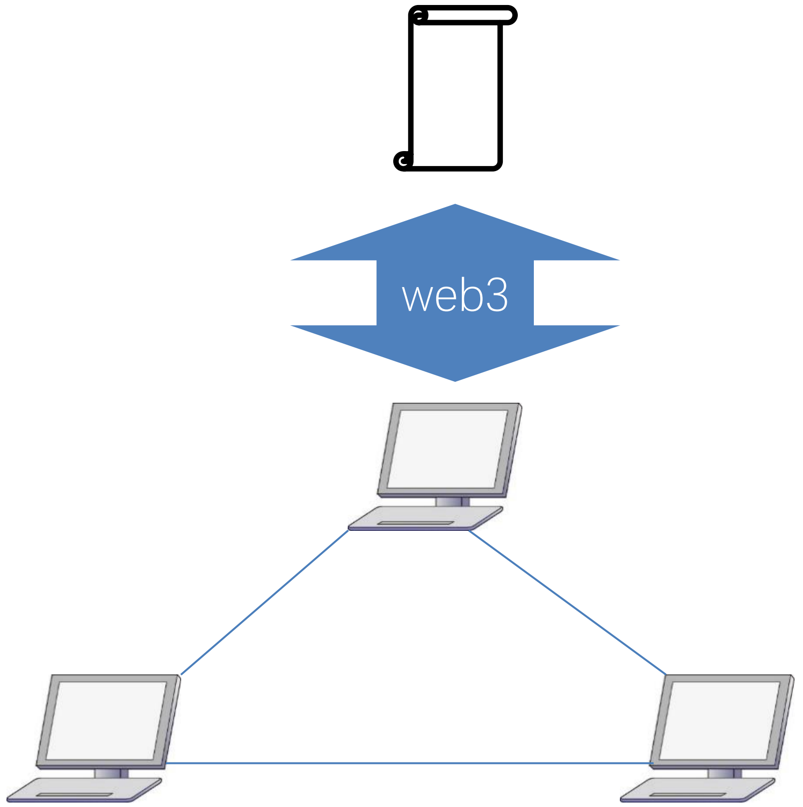
```
geth --rpc --rpccorsdomain="https://remix.ethereum.org" --rpcapi web3,eth,debug,personal,net --vmdebug --datadir <path/to/local/folder/for/test/chain> --dev console
```

WARNING: It is not safe to use the --rpccorsdomain flag with a wildcard: --rpccorsdomain *

For more info: Remix Docs on Web3 Provider

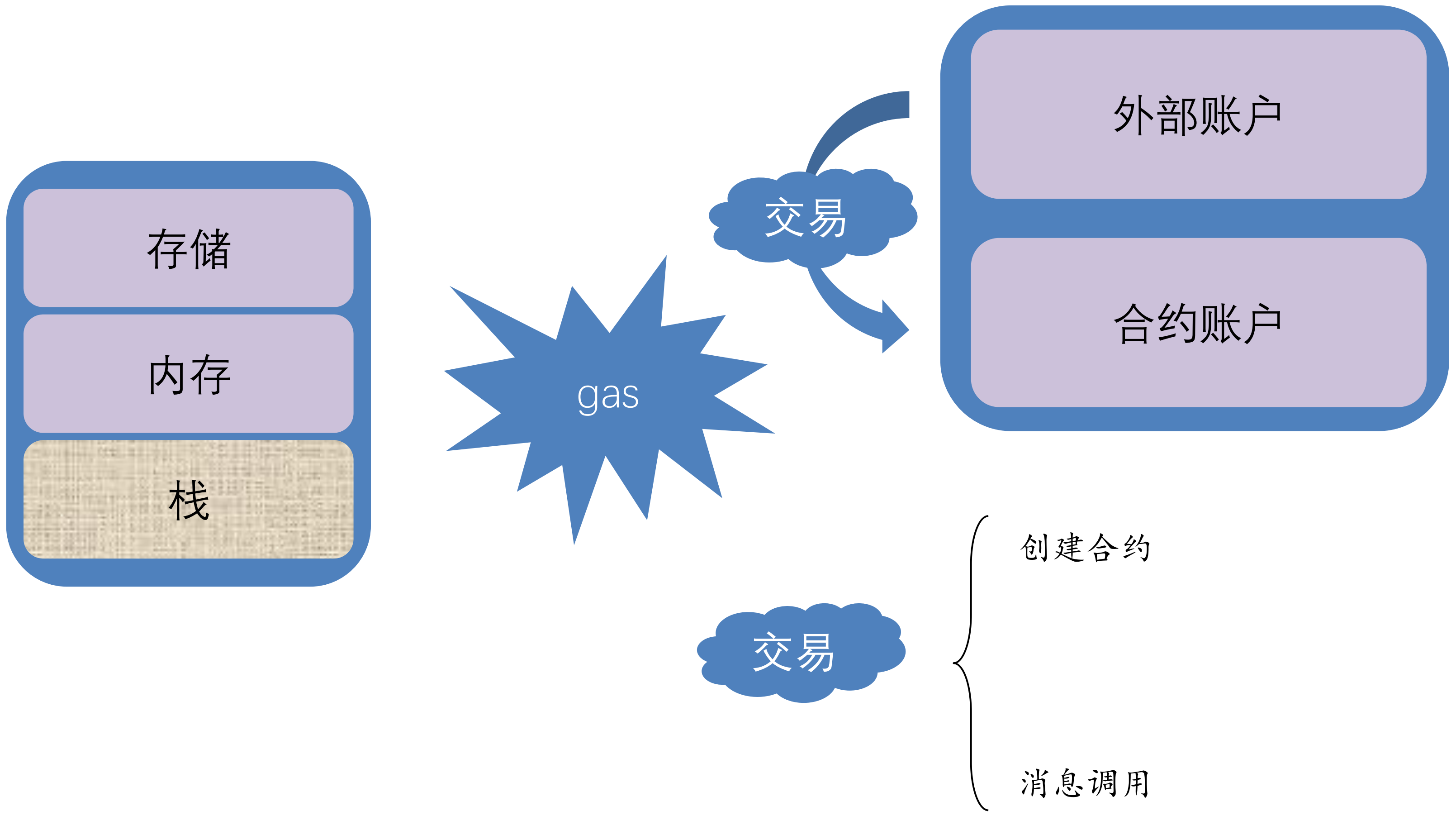
Web3 Provider Endpoint: http://127.0.0.1:8545

真实环境



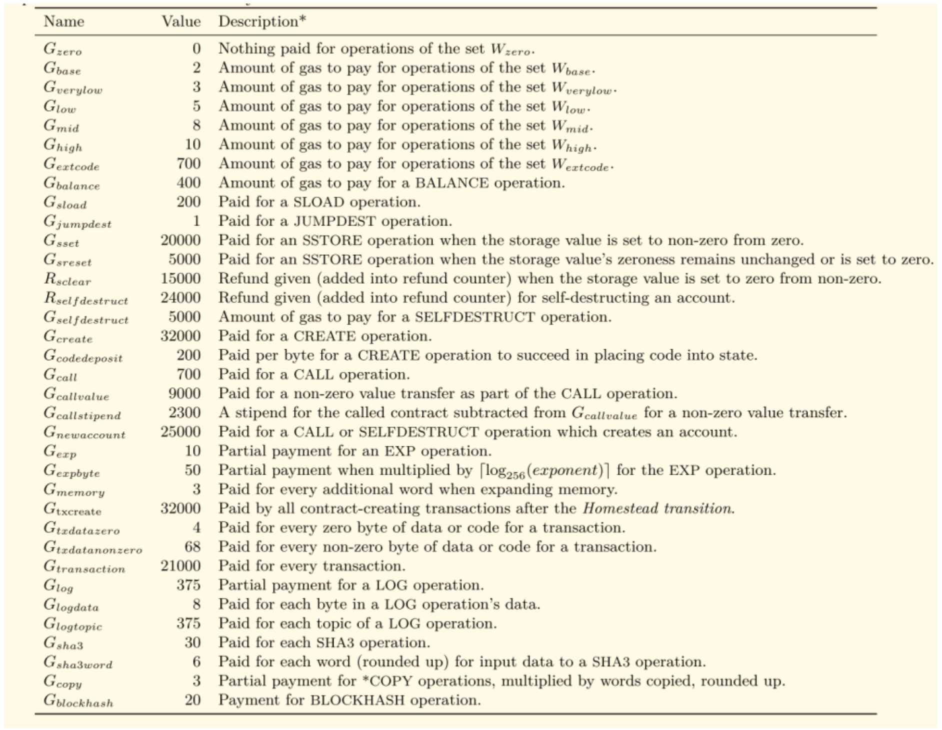
三、Solidity语法与简单合约示例

基本模型（编程者视角）



gas

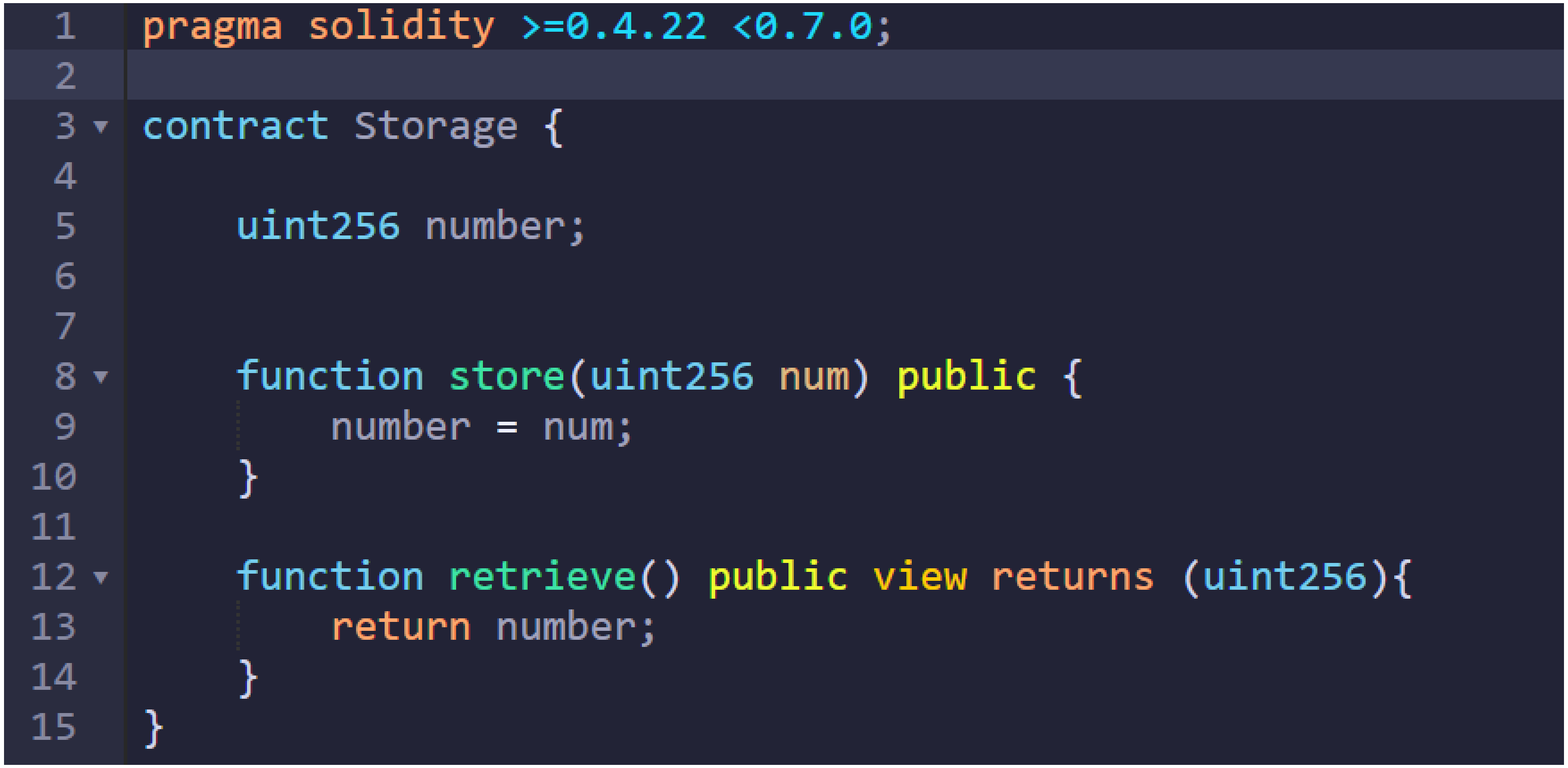
- 调用者给矿工的手续费
- 矿工收益=执行消耗gas
- 多余的gas会退回给发送者



Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead transition*.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.

基本语法

- 示例

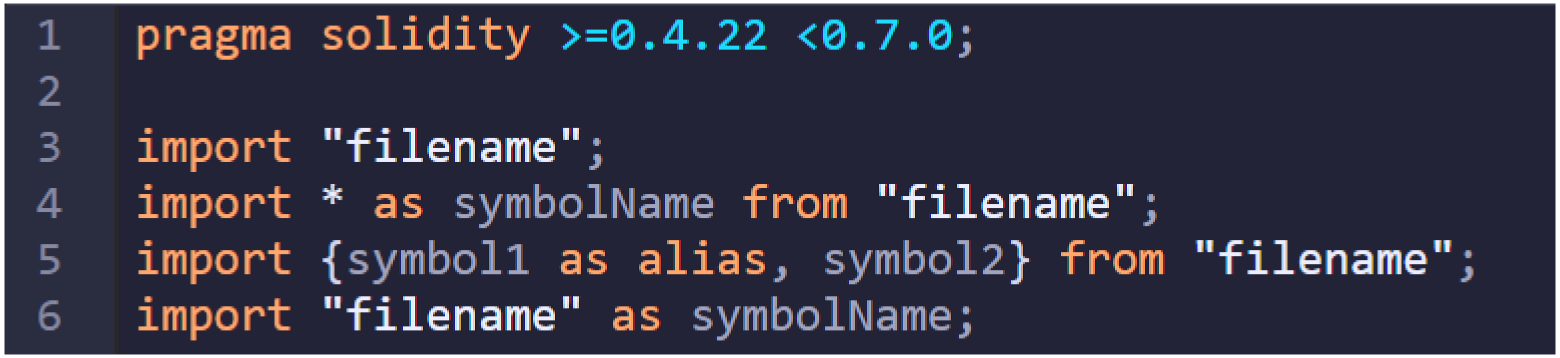


```
pragma solidity >=0.4.22 <0.7.0;

contract Storage {

    uint256 number;


    function store(uint256 num) public {
        number = num;
    }

    function retrieve() public view returns (uint256){
        return number;
    }
}
```

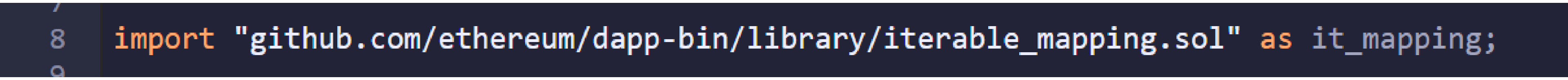
编译器和导入其他代码文件

- pragma solidity后跟语言版本

- import语句用于引入其他代码，类似于C的include



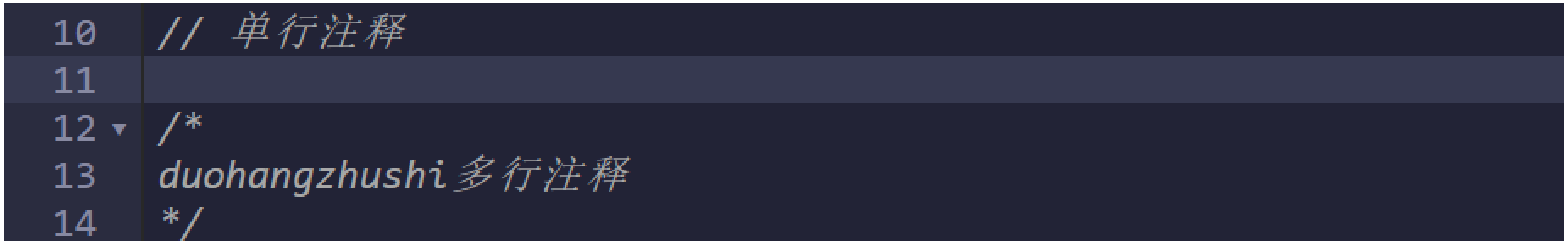
```
pragma solidity >=0.4.22 <0.7.0;

import "filename";
import * as symbolName from "filename";
import {symbol1 as alias, symbol2} from "filename";
import "filename" as symbolName;
```



```
import "github.com/ethereum/dapp-bin/library/iterable_mapping.sol" as it_mapping;
```

注释

- //　　　　　单行注释
- /*...*/　　　多行注释



```
// 单行注释

/*
duohangzhushi多行注释
*/
```

基本类型（值类型）

- 值类型在函数参数或者赋值时，会进行值拷贝

类型名	解释	说明
bool	布尔值	可取值false和true
int/uint	（无）符号整型	长度标识在后，以8为步长可从8取到256，如uint80,int248
address	地址类型	20字节，有成员 banalce：余额 transfer：对其转账
bytes1-bytes32	定长字节数组	
各种常数	字符串、整数等	

基本类型（引用类型）

- 引用类型在函数参数以及赋值中，传递的是地址
- 变长数组类型
  - 定义例如`uint[] memory a =new uint[](7);`
- 结构体类型
  - 定义例如

```
struct Player {
        address addr;
        uint id;
}
```

基本类型（map映射）

- 是一个非常常用的类型，原像可以取除了映射、变长数组、合约、枚举和结构体之外的任意类型。

- 声明例如

```
mapping (uint => address) players;
```

- 使用例如

```
players[1] = msg.sender;
```

合约结构

- 一个合约可以包含状态变量、函数、函数修饰器、事件、结构类型和枚举类型。

- 状态变量和函数

```
contract SimpleStorage [is contract1, contract2]{
        uint storedData; // 状态变量
        // ...
        function bid() public payable { // 函数
        // ... }

}
```

函数修饰器

- 改变函数行为，例如检查执行条件

```
contract owned {
		address owner;
		function owned() public {
				owner = msg.sender;
		}
		modifier onlyOwner {
				require(msg.sender == owner);
				_;
		}

		function close() public onlyOwner {
				selfdestruct(owner);
		}
}
```

事件

- 可以由Dapp监听，存储在链上

```
contract SimpleAuction {
        event HighestBidIncreased(address bidder, uint amount); // 事件
        function bid() public payable {
        // ...
        emit HighestBidIncreased(msg.sender, msg.value); // 触发事件
} }
```

```
var abi = /* abi 由编译器产生 */;
var ClientReceipt = web3.eth.contract(abi);
var clientReceipt = ClientReceipt.at("0x1234...ab67" /* 地址 */);
var event = clientReceipt.HighestBidIncreased(); // 监视变化
event.watch(function(error, result){
// 结果包括对 `HighestBidIncreased` 的调用参数在内的各种信息。
        if (!error) console.log(result);
});
```

枚举类型

- 可以构建常量值的自定义类型
- 例如购买合约

```
contract Purchase {
        enum State { Created, Locked, Inactive } // 枚举
}
```

函数

- 每个合约可能包括一个或者多个函数供调用

```
pragma solidity ^0.4.16;
contract Simple {
        function taker(uint _a, uint _b) public pure {
        // 用 _a 和 _b 实现相关功能.
        }
        function f(address _in) public pure returns (address
out) {
         // ...
        }
}
```

描述关键词

- 描述关键词用于描述函数的某些属性

view/constant	不修改状态
external	外部函数调用
public	可从合约外部调用
private	只能从合约内部调用
pure	不读写状态
internal	只能从内部和派生合约调用
payable	函数可以接收以太币

控制流、合约创建

- 支持if，else，while，do，for，break，continue，return，? :
- new 创建合约

```
contract D {
        uint x;
        function D(uint a) public payable {
                x = a;
        }
}
contract C {
        D d = new D(4);
        function createD(uint arg) public {
        D newD = new D(arg);
        }
}
```

错误控制

- require用于检查输入参数，并且抛出异常字符串（不消耗gas）
- assert用于测试内部错误

```
contract Sharer {
        function sendHalf(address addr) public payable returns (uint
balance) {
        require(msg.value % 2 == 0, "Even value required.");

        uint balanceBeforeTransfer = this.balance;
        addr.transfer(msg.value / 2);

        assert(this.balance == balanceBeforeTransfer - msg.value / 2);
        return this.balance;
        }
}
```

- revert回滚至执行前状态，剩余gas返还给调用者

接口

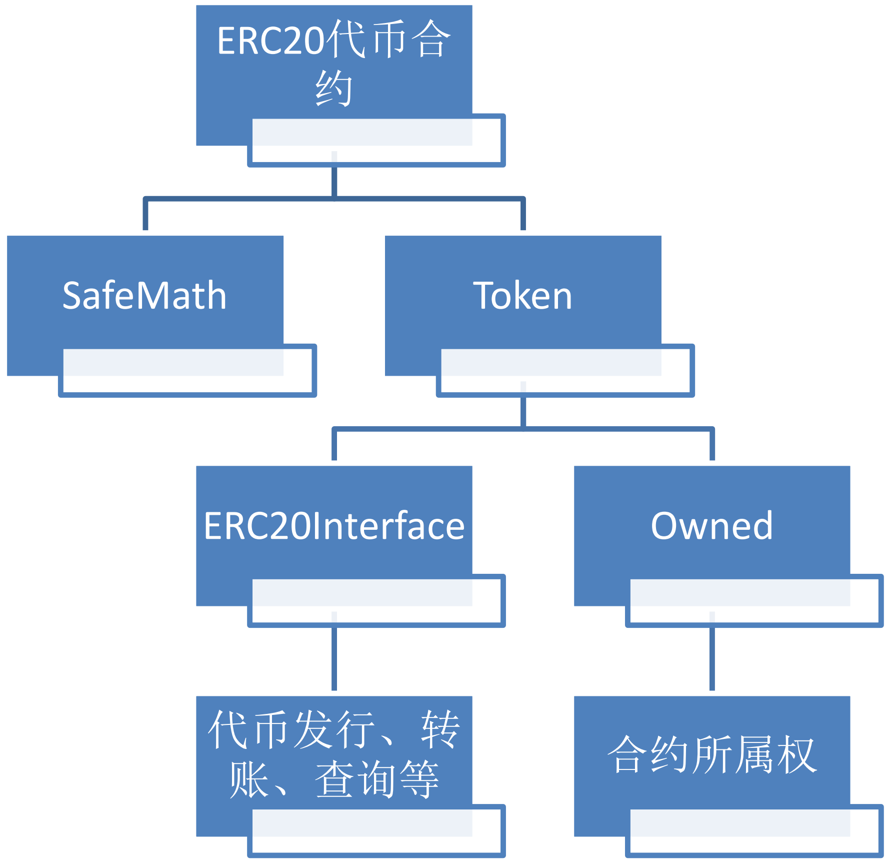
- interface指明了一个合约应该实现的函数，然后由其他合约继承

```
pragma solidity ^0.4.11;
interface Token {
	function transfer(address recipient, uint amount)
public;
}

contract ERC20 is Token {
	//…
}
```

ICO合约实例（ERC20）

- ICO（Initial Coin Offering）首次代币发行
- 发行区块链项目的代币
- mapping(address => int)
- ERC20是一种代币发行的代码标准

ERC20代码结构


ERC20代币合约
SafeMath
Token
ERC20Interface
Owned
代币发行、转账、查询等
合约所属权

SafeMath

```
pragma solidity ^0.4.24;

contract SafeMath {
      function safeAdd(uint a, uint b) public pure
returns (uint c) {
            c = a + b;
            require(c >= a);
      }
      function safeSub(uint a, uint b) public pure
returns (uint c) {
            require(b <= a);
            c = a - b;
      }
      //… 乘法、除法
}
```

ERC20Interface

```
contract ERC20Interface {
        function totalSupply() public constant returns
(uint);
        function balanceOf(address tokenOwner) public
constant returns (uint balance);
        function allowance(address tokenOwner, address
spender) public constant returns (uint remaining);
        function transfer(address to, uint tokens) public
returns (bool success);
        function approve(address spender, uint tokens)
public returns (bool success);
        function transferFrom(address from, address to,
uint tokens) public returns (bool success);
        event Transfer(address indexed from, address
indexed to, uint tokens);
        event Approval(address indexed tokenOwner, address
indexed spender, uint tokens);
}
```

SJTUToken

```
contract SJTUToken is ERC20Interface, Owned, SafeMath {
      string public name;
      uint public _totalSupply;
      mapping(address => uint) balances;
      mapping(address => mapping(address => uint))
allowed;
      constructor() public {
            name = ”SJTU Token”;
            _totalSupply = 1000000000000000000000000000;

     balances[0x5A86f0cafD4ef3ba4f0344C138afcC84bd1ED222]
= _totalSupply;
            emit Transfer(address(0),

      0x5A86f0cafD4ef3ba4f0344C138afcC84bd1ED222,
_totalSupply);
      }
```

SJTUToken(2)

```
function totalSupply() public constant returns (uint) {
      return _totalSupply - balances[address(0)];
}

function balanceOf(address tokenOwner) public constant
returns (uint balance) {
      return balances[tokenOwner];
}


function transfer(address to, uint tokens) public returns
(bool success) {
     balances[msg.sender] = safeSub(balances[msg.sender],
tokens);
      balances[to] = safeAdd(balances[to], tokens);
      emit Transfer(msg.sender, to, tokens);
      return true;
}
```

SJTUToken(3)

```
function approve(address spender, uint tokens) public returns
(bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
}

function transferFrom(address from, address to, uint tokens)
public returns (bool success) {
        balances[from] = safeSub(balances[from], tokens);
        allowed[from][msg.sender] =
safeSub(allowed[from][msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        emit Transfer(from, to, tokens);
        return true;
}

function allowance(address tokenOwner, address spender) public
constant returns (uint remaining) {
        return allowed[tokenOwner][spender];
}
```

SJTUToken(4)

```
    function transferAnyERC20Token(address tokenAddress, uint tokens)
    public onlyOwner    returns (bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
    }
}
```

OwnerShip

```
contract Owned {
        address public owner;
        address public newOwner;
        event OwnershipTransferred(address indexed _from, 
address indexed _to);
        constructor() public {
                owner = msg.sender;
        }
        modifier onlyOwner {
                require(msg.sender == owner);
                _;
        }
        function transferOwnership(address _newOwner) 
public onlyOwner {
                newOwner = _newOwner;
        }
        //...
}
```

Q & A

谢谢