FORESIGHT X

<OpenBuild Hack@HK/>

<Solidity Basics Workshop />

Frank

Security Tech Lead @ Beosin

Foresight X
Gateway to
Web3 in Hongkong

OpenBuild
HACK.
SERIES


BLOCKCHAIN ACADEMY
BEOSIN
Blockchain Security
Solidity基础

Contents

01 初窥Solidity智能合约

02 Solidity 基本变量类型

03 函数、修饰器、事件、错误概念

04 Remix的基本使用

05 合约编译、部署、ABI调用合约

01 初窥Solidity

Solidity智能合约文件结构

```
1  pragma solidity ^0.4.0;
2  import "filename";
3
4  contract SimpleStorage {
5      uint storedData;
6
7      struct Voter { // 结构
8          uint weight;
9          bool voted;
10         address delegate;
11         uint vote;
12     }
13
14     enum State { Created, Locked, Inactive } // 枚举
15
16     function set(uint x) public { //function
17         storedData = x;
18     }
19
20
21     function get() public view returns (uint) {
22         return storedData;
23     }
24
25
26     modifier onlySeller() { // 修饰器
27         require(
28             msg.sender == seller,
29             "Only seller can call this."
30         );
31         _;
32     }
33
34     function abort() public onlySeller { // Modifier usage
35         // ...
36     }
37
38     event HighestBidIncreased(address bidder, uint amount); // 事件
39
40     function bid() public payable {
41         // ...
42         emit HighestBidIncreased(msg.sender, msg.value); // 触发事件
43     }
44
45     }
46 }
```

- **#L1 pragma**
  指代码由0.4.0版本开发的，并且在0.4.0以上 0.5.0（不包括）以下的编译器编译运行不会出问题。

- **#L2 import**
  语句把“filename”全局语句导入当前工程中，同时也全局导入“filename”里包含的所有全局导入文件，支持向后兼容性

- **#L4 contract**
  指构造合约，SimpleStorage合约关键字，符合大驼峰写法；【solidity中合约指的是一组代码（函数）和数据（他的状态），它位于以太坊区块链的一个特定地址上】

- **#L5 unit**
  指数据类型为256位无符号数的变量声明，变量名为storedData，符合小驼峰写法。这里可以理解为数据库中存储的某个位置，可以通过函数set以及get来获取或者更改变量的值。

- **#L7-12 struct**
  结构是可以将几个变量分组的自定义类型。

- **#L14 enum**
  枚举可用来创建由一定数量的“常量值”构成的自定义类型。

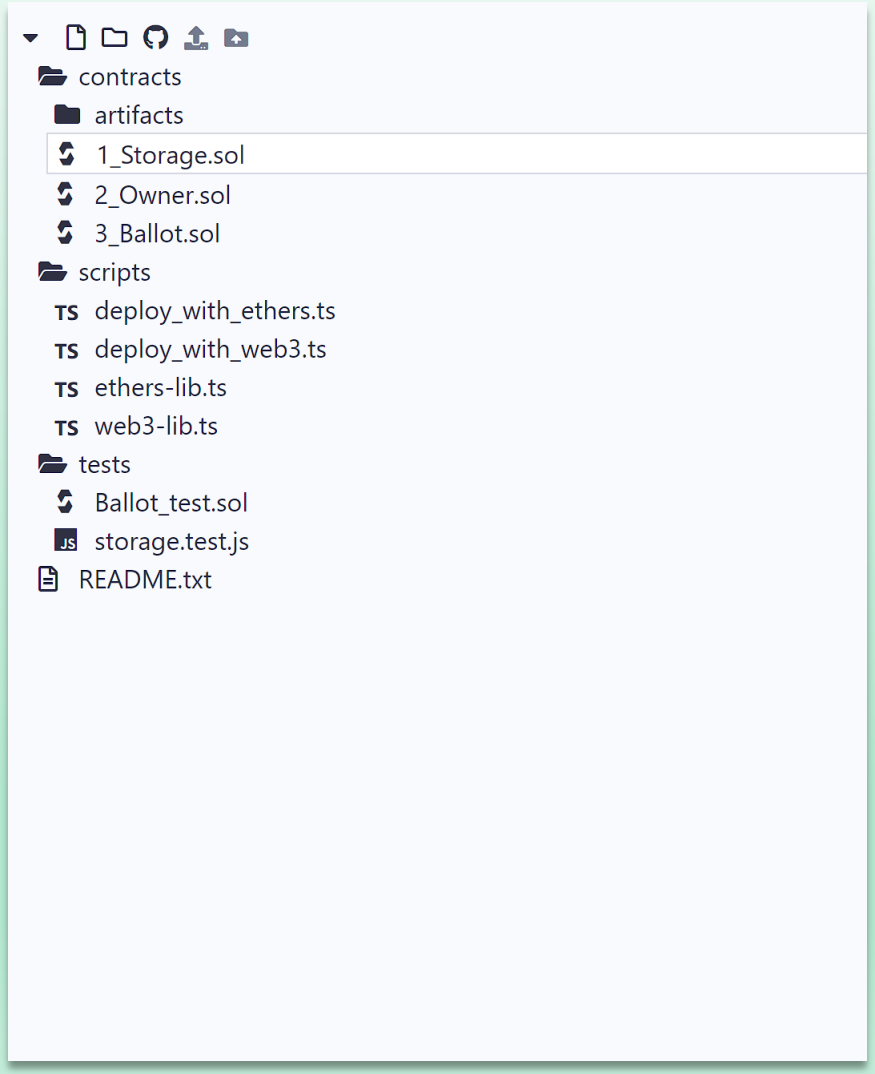
- **#L16 function**
  指的是申明方法set()并将外部传入的值X 赋予合约内部的storedData变量，public 指函数修饰符

- **#L26-32 modifier**
  函数修饰器可以用来以声明的方式改良函数语义。

- **#L38 event**
  事件是能方便地调用以太坊虚拟机日志功能的接口。

- **#L42 emit**
  触发固定事件关键字

Solidity智能合约文件类型



- **contracts**
  存放合约的目录，合约扩展名为.sol。
- **scripts**
  存放各类函数处理脚本的目录，图示中的脚本为Type Scripts编写的脚本。
- **tests**
  存放各类测试脚本的目录，图示中存有由solidity和Type Scripts分别编写的脚本
- **artifacts**
  存放solidity合约编译后后的文件

02 Solidity的基本变量类型

整形

- **有符号(int)**

  Int <m>类型位数默认为256，取值范围为0<m<=256,m需能被8整除。

  int8 占1个字节 取值范围： -2^4 到 2^4 -1

  int16 占2个字节 取值范围 ： -2^8 到2^8 -1

  .....

  int256 占32个字节 取值范围 ： -2^128 到2^128 -1

- **无符号(uint)**

  Uint <m>类型m默认为256，取值范围为0<m<=256,m需能被8整除。

  uint8 占1个字节 取值范围： 0 到 2^8 -1

  uint16 占2个字节 取值范围 ： 0 到2^16 -1

  .....

  uint256 占32个字节 取值范围 ： 0 到2^256 -1

其他类型

- **地址(address):**

  通常用作地址声明，地址类型存储一个 20 字节的值（以太坊地址的大小）。 地址类型也有成员变量，并作为所有合约的基础。

  例如：0x12a0E25E62C1dBD32E505446062B26AECB65F028

- **布尔型(bool):**

  可能的取值为字面常数值 true 和 false。

- **枚举(enum):**

  有限个可选数值，一般用enum enum_name {}方式声明

  例如：enum state { Created ,Locked, Active }

其他类型

- **元组类型(uint x , uint y , int x)**

元素数量固定的对象列表，列表中的元素可以是不同类型的对象。这些元组可以用来同时返回多个数值，也可以用它们来同时给多个新声明的变量或者既存的变量

例如：(uint x , bool y ,int z) = (12 , false , -1)

- **结构体类型(struct)**

Solidity提供了一种以Struct形式定义新类型的方法。Struct是自定义类型，可以对多个变量进行分组。结构struct中同样可以包含mapping，array，同样可以当做元素映射到mapping和数组中。

例如：struct Voter{

```
    uint weight;
    ……
    bool voted ;
}
```

数组类型

- **定长数组<type>[M]**

有 M 个元素的定长数组，M >= 0，数组元素为给定类型。特别地，定长字节数组仅支持bytes1、bytes2…bytes32

例如：uint256[2] a;

bytes32 b;

- **非定长数组<type>[]**

<type>[]，数组元素为给定类型

bytes：动态大小的字节序列

string：动态大小的Unicode字符串，通常呈现为UTF-8编码

例如：uint256[2] a;

bytes32 b;

03函数、修饰器、事件、错误概念

函数

函数是代码的可执行单元。函数通常在合约内部定义，但也可以在合约外部定义。

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.1 <0.9.0;

contract SimpleAuction {
    function bid() public payable { // Function
        // ...
    }
}

// Helper function defined outside of a contract
function helper(uint x) pure returns (uint) {
    return x * 2;
}
```

函数形式

函数分为函数标识、参数、函数函数修饰符、函数返回值4个部分，形式如下：

function (<parameter types>) {puclic|private|internal|external} [pure|constant|view|payable]

[returns (<return types>)]

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.1 <0.9.0;

contract SimpleAuction {
    function bid() public payable { // Function
        // ...
    }
}

// Helper function defined outside of a contract
function helper(uint x) pure returns (uint) {
    return x * 2;
}
```

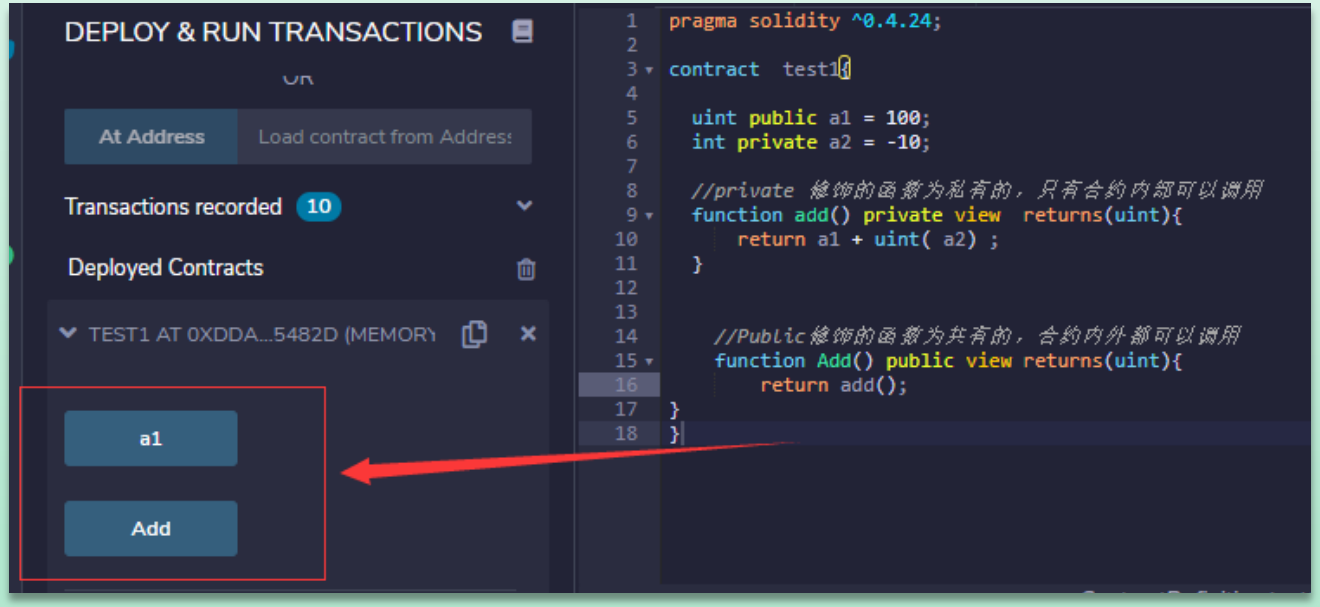
函数修饰符

在Solidity中函数修饰符规定了函数的行为、调用规则。在Solidity语言中预置的修饰符常见的有public、private、external、internal等

修饰符	说明
public	公有,任何人(拥有以太坊账户的)都可以调用
private	私有, 只有智能合约内部可以调用
external	仅合约外部可以调用，合约内部需使用this调用
internal	仅合约内部和继承的合约可以调用
view/constant	函数会读取但是不会修改任何contract的状态变量
pure(纯净的)	函数不使用任何智能合约的状态变量
payable	调用函数需要付钱，钱付给了智能合约的账户
returns	返回值函数声明中使用

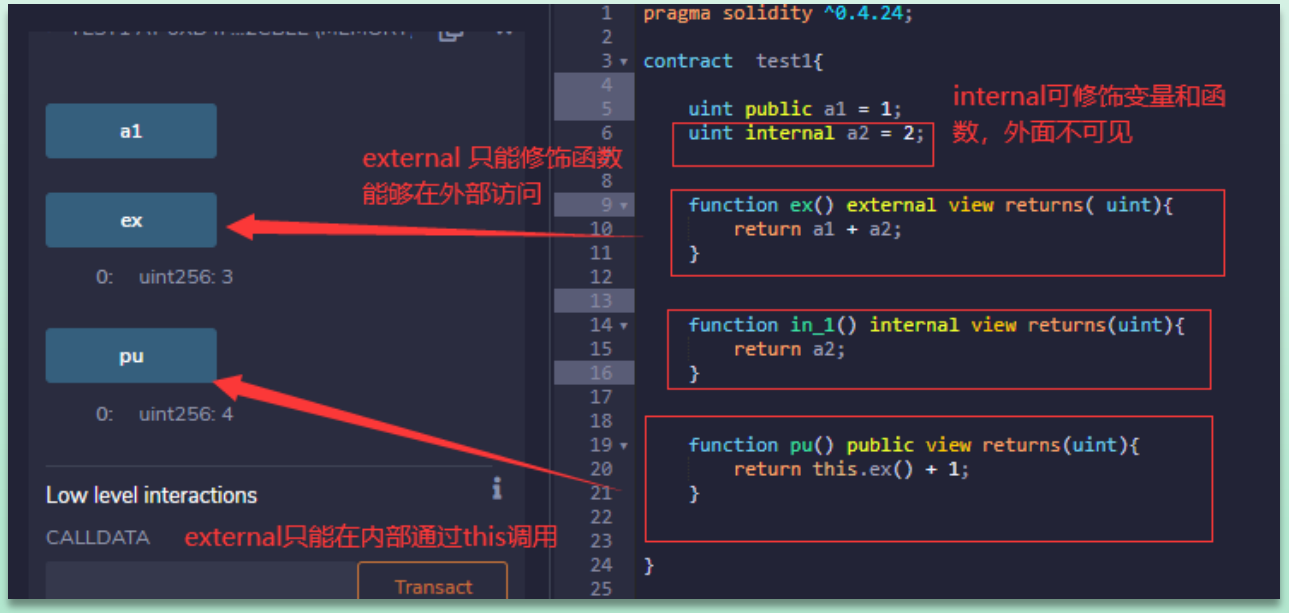
函数修饰符

- public:在外部和内部均可见（创建存储/状态变量的访问者函数）
- private:仅在当前合约中可见



函数修饰符

- external: 只有外部可见（仅对函数）在合约的内部使用this.func进行调用
- internal: 只有内部可见（变量默认是internal）



函数修饰符

- payable：允许函数在调用同时接收Ether

任何函数，只要修饰为payable，那么就可以在调用这个方法的时候，对value字段赋值，然后将价值value的钱转给合约。若这个函数没有指定payable，但是对value赋值了，那么本次调用会报错。

```
pragma solidity ^0.4.24;


contract test1 {
    uint public num;

    //如果构造函数中未指定payable关键字，那么创建合约时不允许转账
    //如果指定了payable，则可以转账
    constructor() public  {
        //构造函数
        //仅在部署合约时调用一次，完成对合约的初始化。可以在创建合约时转钱到合约

    }

    //任何函数，只要指定了payable关键字，这个合约就可以接受转账，调用时，也可以转0
    function giveMoney() public payable {

    }
}
```

重载函数

合约中可以具有不同参数的同名函数，同时也可以继承。同名函数可以减少同类功能的函数取名较多的问题。

```
pragma solidity ^0.5.0;

contract Test{


    uint256 public a = 1 ;
    uint256 public b = 2;

    int public c = 3;
    int public d = 4;

    function add(uint256 x1, uint256 x2) public view returns(uint256){ 1
        return x1 + x2;
    }

    function add(int256 x1, int256 x2) public view returns(int256){  2
        return x1 + x2;
    }

    function add(uint256 x1, uint256 x2, string memory data) public view returns(uint256){ 3
        require(add(x1,x2) > 0,data);  1
        return x1 + x2;
    }

    function test() public returns(uint256, int256){
        uint256 test1= add(a,b,"test false"); 3
        int256 test2= add(c,d); 2

        return (test1,test2);
    }
}
```

构造函数

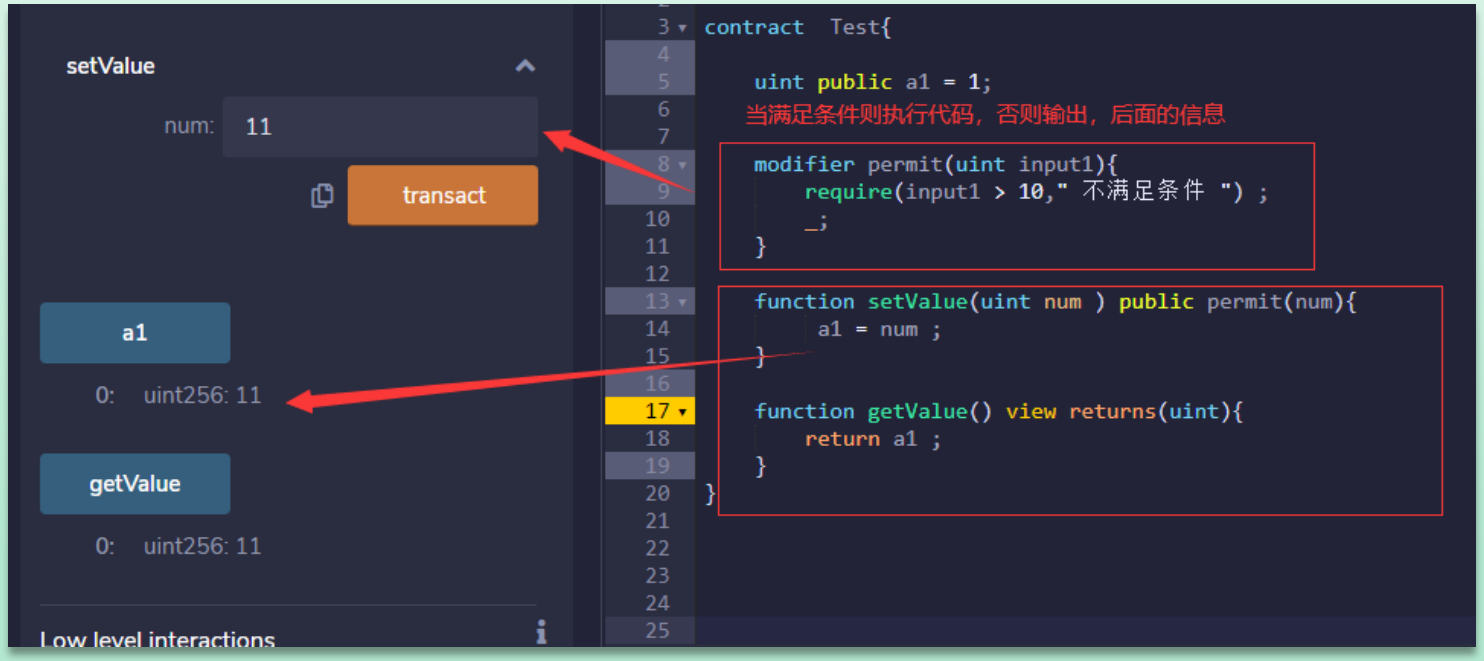
在0.4.24之前可以创建同合约名的函数快速赋值给合约的参数；
自0.4.22颁布起，废除function+合约名称来定义构造函数，而采用 “constructor(…) { … }”来声明构造函数。

```
pragma solidity 0.4.12;

contract Test{


    uint256 public a2 ;

    function Test(uint256 num)  {
        a2 = num;
    }


}
```

```
pragma solidity ^0.5.0;

contract Test{


    uint256 public a1 ;

    constructor(uint256 _num){
        a1 = _num;
    }

}
```

函数修饰器

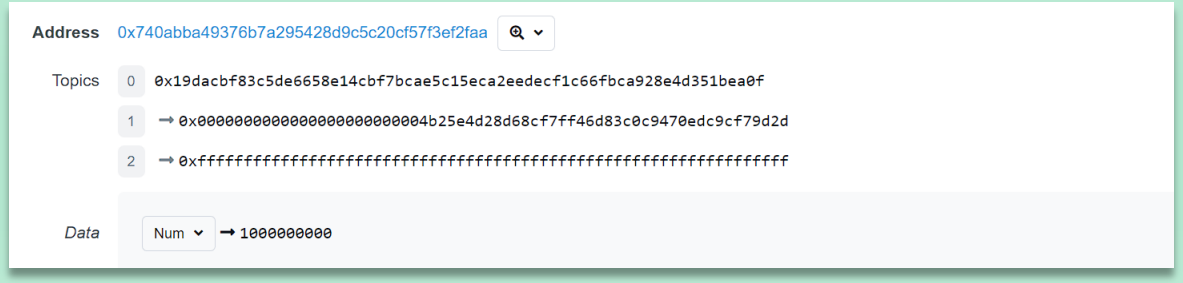
函数修饰器(Function Modifiers): 修饰器(Modifiers)可以用来轻易的改变一个函数的行为。比如用于在函数执行前检查某种前置条件。修饰器是一种合约属性，可被继承，同时还可被派生的合约重写(override)，简单来说就是对原有函数进行修改。



事件

事件是与 EVM 日志记录工具的便捷接口，应用程序可以通过以太坊客户端的 RPC 接口订阅和监听事件。并且事件参数可以使用indexed标识，使用indexed标识的参数会添加到 “topics” 的特殊参数结构。

```solidity
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.21 <0.9.0;

contract ClientReceipt {
    event Deposit(
        address indexed from,
        bytes32 indexed id,
        uint value
    );

    function deposit(bytes32 id) public payable {
        // Events are emitted using `emit`, followed by
        // the name of the event and the arguments
        // (if any) in parentheses. Any such invocation
        // (even deeply nested) can be detected from
        // the JavaScript API by filtering for `Deposit`.
        emit Deposit(msg.sender, id, msg.value);
    }
}
```



错误

错误允许开发者为故障情况定义描述性名称和数据。Solidity 使用状态恢复异常来处理错误。这种异常将撤消对当前调用（及其所有子调用）中的状态所做的所有更改，并且还可向调用者标记错误。

- **require**：用于确认条件有效性
- **assert**：只用于测试内部错误，并检查确认条件有效性，无错误信息
- **revert**：用来标记错误并恢复当前的调用，程序执行遇到该语句直接停止执行
- **throw**：可以抛出异常并回退(但无法返回错误消息)

从 0.4.13 版本开始，**throw**被弃用，并且将来会被逐渐淘汰，不推荐使用

错误

错误允许开发者为故障情况定义描述性名称和数据。Solidity 使用状态恢复异常来处理错误。这种异常将撤消对当前调用（及其所有子调用）中的状态所做的所有更改，并且还可向调用者标记错误。

```
contract Demo{

    function demo1(uint256 par) public pure {
        require(par > 365 ,"err msg!");
    }

    function demo2(uint256 par) public pure{
        assert(par >365);
    }


    function demo3(uint256 par) public pure {
        if (par >365){
            revert("err msg!");
        }
    }

    error ErrorDemo();

    function demo4(uint256 par) public pure {
        if (par > 365) revert ErrorDemo();
    }
}
```

错误

错误允许开发者为故障情况定义描述性名称和数据。Solidity 使用状态恢复异常来处理错误。这种异常将撤消对当前调用（及其所有子调用）中的状态所做的所有更改，并且还可向调用者标记错误。

```
contract Demo{

    function demo1(uint256 par) public pure {
        require(par > 365 ,"err msg!");
    }

    function demo2(uint256 par) public pure{
        assert(par >365);
    }


    function demo3(uint256 par) public pure {
        if (par >365){
            revert("err msg!");
        }
    }

    error ErrorDemo();

    function demo4(uint256 par) public pure {
        if (par > 365) revert ErrorDemo();
    }
}
```

BLOCKCHAIN ACADEMY × BEOSIN Blockchain Security

03编译、部署和用ABI调用合约

编译

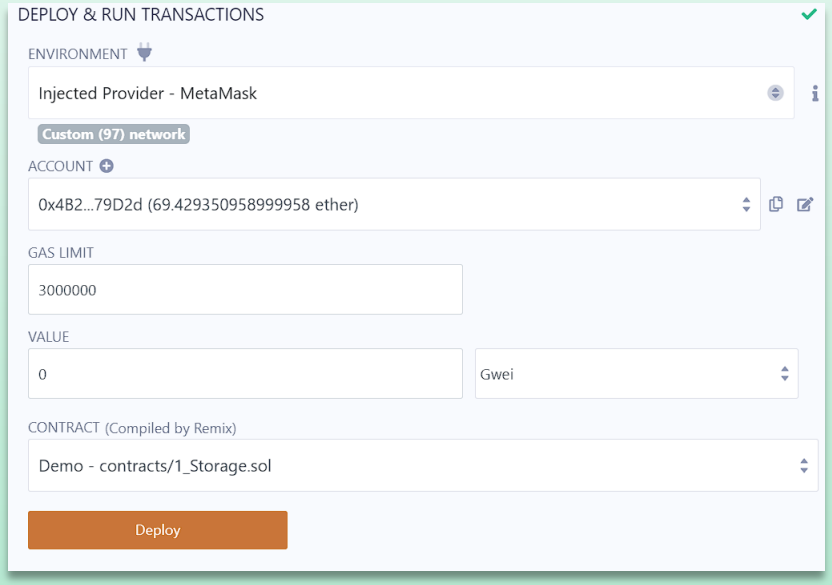
将solidity编写的合约解释为EVM能够执行的二进制编码。

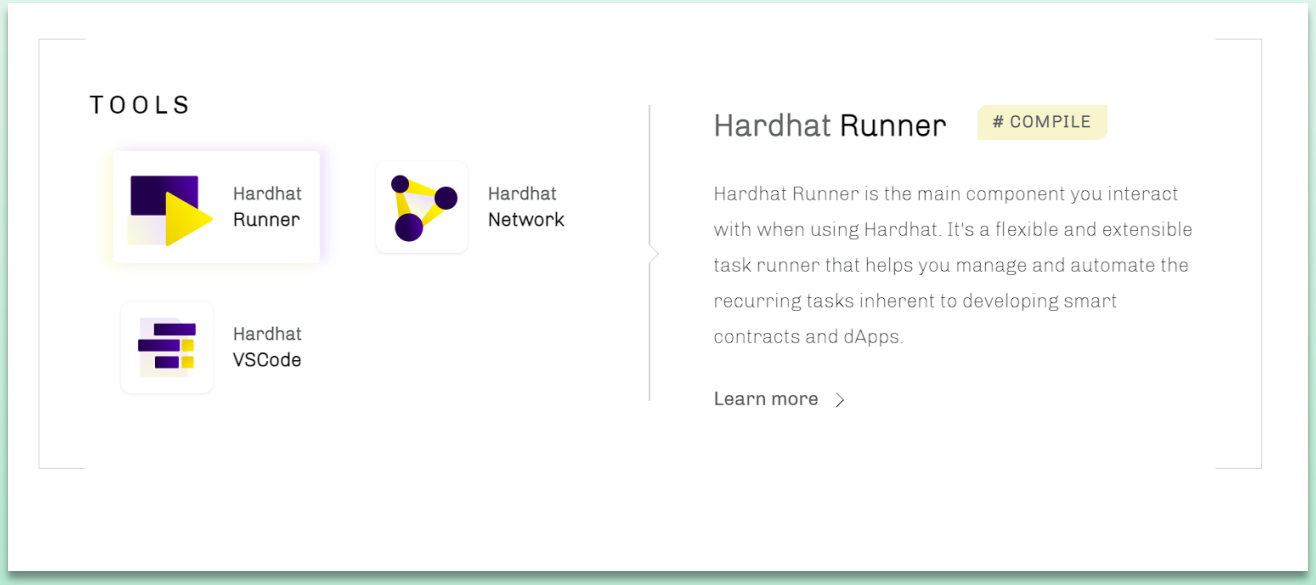
- **solc**：是 Solidity 源码库的构建目标之一，它是 Solidity 的命令行编译器
- **Solcjs**：使用 npm 可以便捷地安装Solidity编译器solcjs，该项目是利用 Emscripten 从 C++ 版的 solc 跨平台编译为 JavaScript 的

常用的开发工具已经集成编译，因此不需要开发者特别注意。

合约部署

将合约字节码发布到链上的过程。可通过多个开发者工具实现部署合约的目的。





ABI调用

合约应用程序二进制接口（ABI）是与以太坊生态系统中的合约进行交互的标准方式，无论是从区块链外部还是用于合约到合约的交互。ABI的描述提供了交互数据的编码方式。

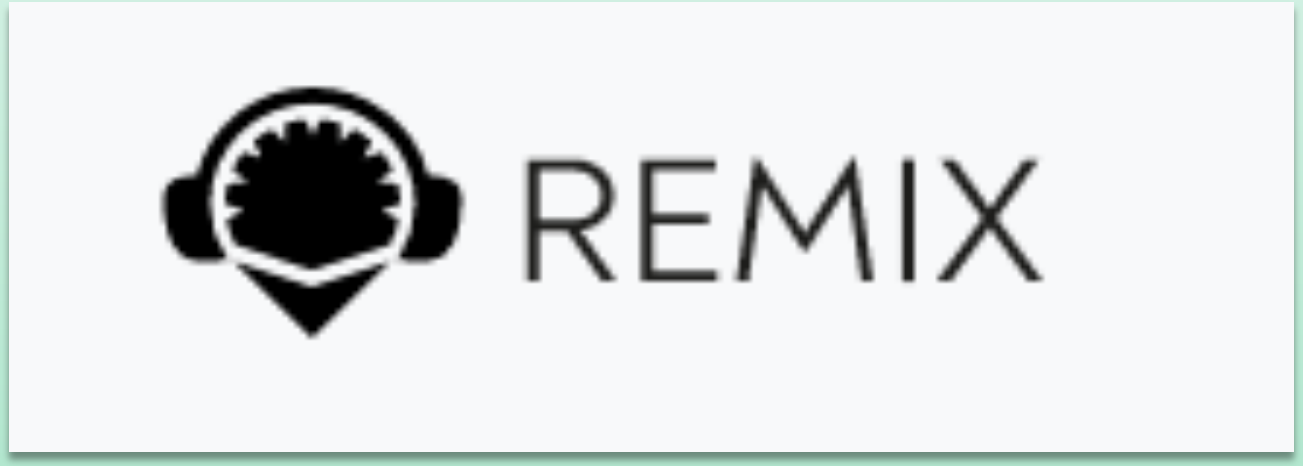

```
"abi": [
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": true,
                "internalType": "address",
                "name": "from",
                "type": "address"
            },
            {
                "indexed": true,
                "internalType": "bytes32",
                "name": "id",
                "type": "bytes32"
            },
            {
                "indexed": false,
                "internalType": "uint256",
                "name": "value",
                "type": "uint256"
            }
        ],
        "name": "Deposit",
        "type": "event"
    },
    {
        "inputs": [
            {
                "internalType": "bytes32",
                "name": "id",
                "type": "bytes32"
            }
        ],
        "name": "deposit",
        "outputs": [],
        "stateMutability": "payable",
        "type": "function"
    }
],
```


```
signed_txn = web3.eth.account.signTransaction(dict(
    nonce=web3.eth.getTransactionCount(_from),
    gasPrice=0,
    gas=8000000,
    to=_to,
    value=0,
    data=data,
    chainId=100
), privateKey)
signed = web3.eth.sendRawTransaction(signed_txn.rawTransaction)
print(signed.hex())
```

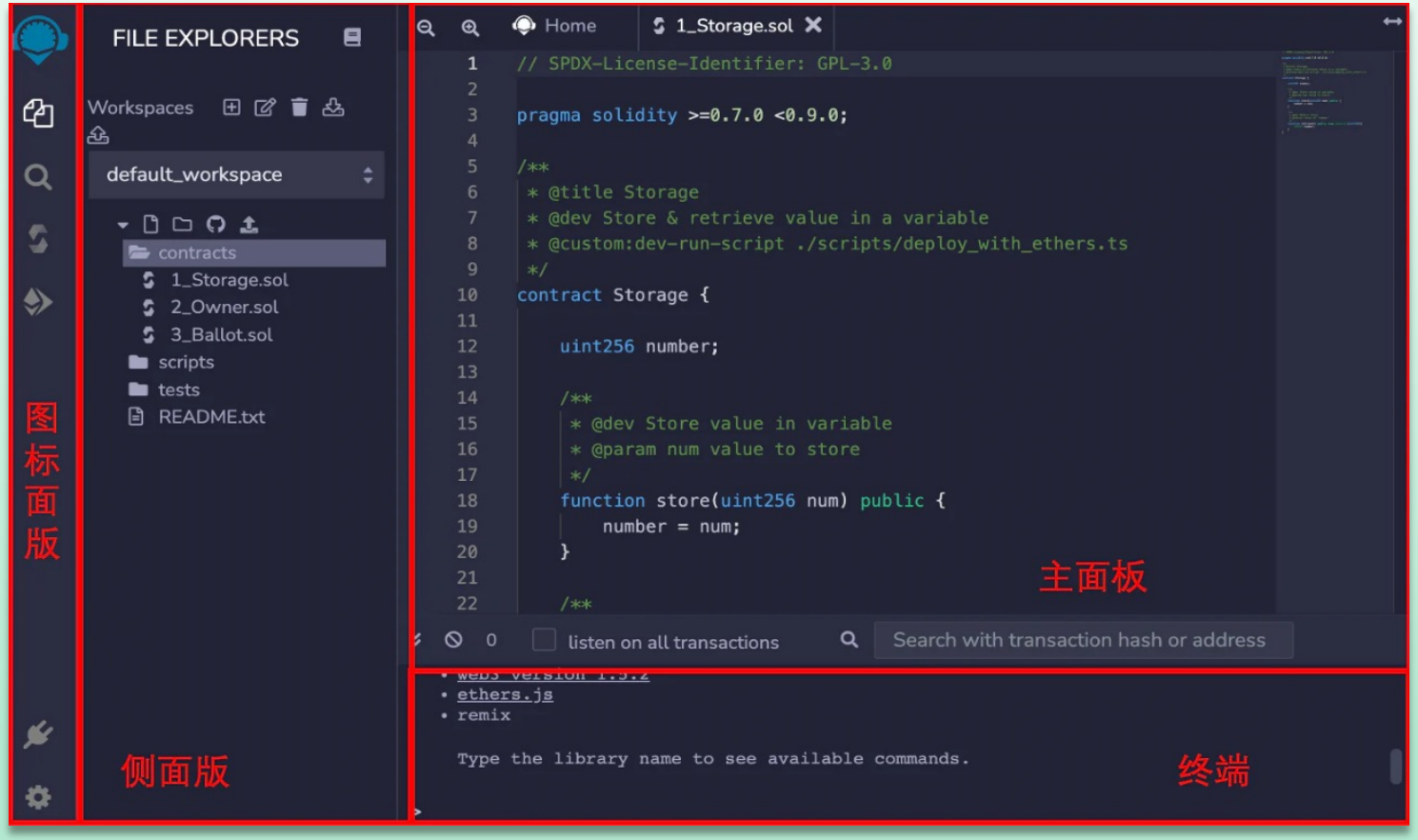
04 Remix基本使用

Remix 介绍

Remix是以太坊官方推荐的智能合约开发IDE，非常易用。Remix是可视化的，用户可以直接在浏览器中快速部署测试智能合约，而不需要安装配置任何程序。这一节我们将完整的介绍如何使用Remix部署合约，并调用其中的函数。



Remix IDE面板介绍



首先在浏览器打开Remix的网址

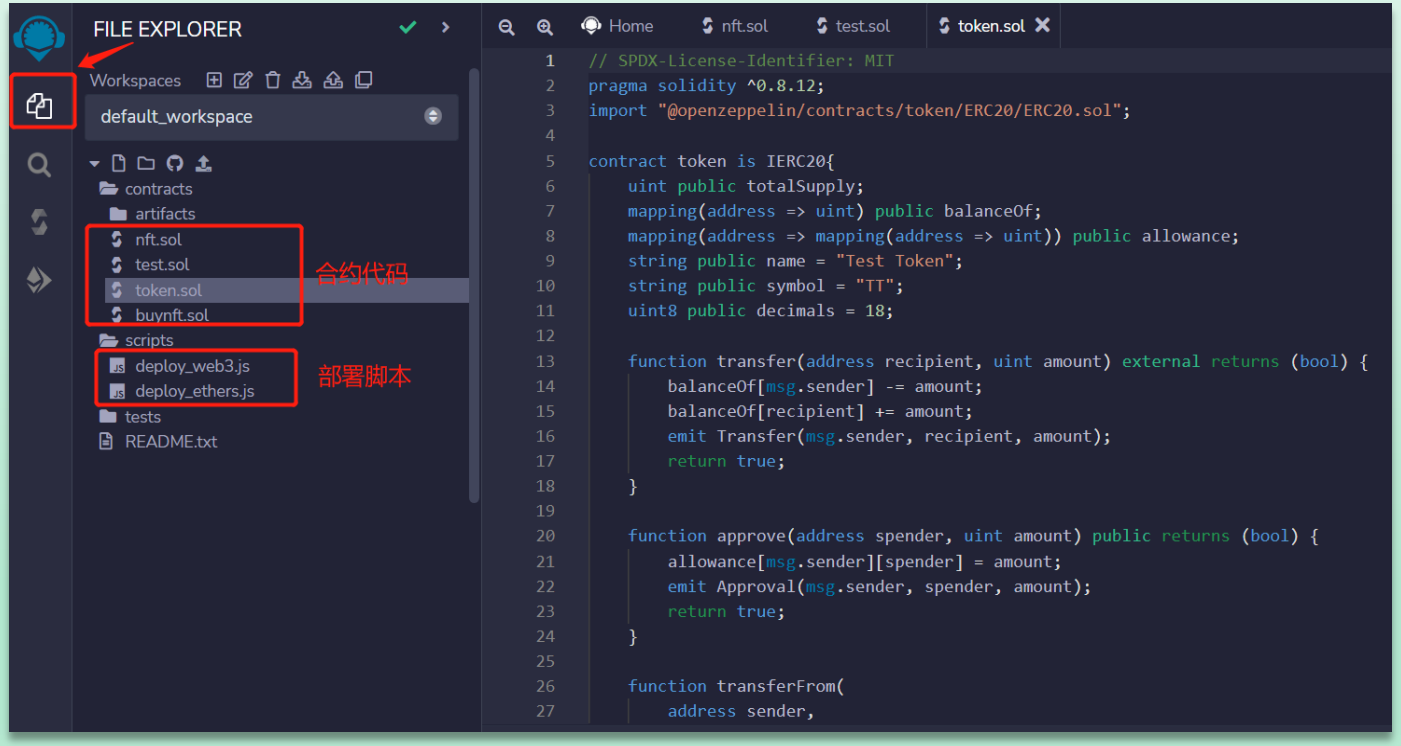
https://remix.ethereum.org/

可以看到默认Remix由四个面板组成

- 图标面板：上面有代表不同功能的图标，点击后，相应的功能个会显示在侧面版上。
- 侧面版：各种功能的 GUI。
- 主面板：代码编辑器。
- 终端：将显示与 GUI 交互的结果，也可以在此处运行脚本。

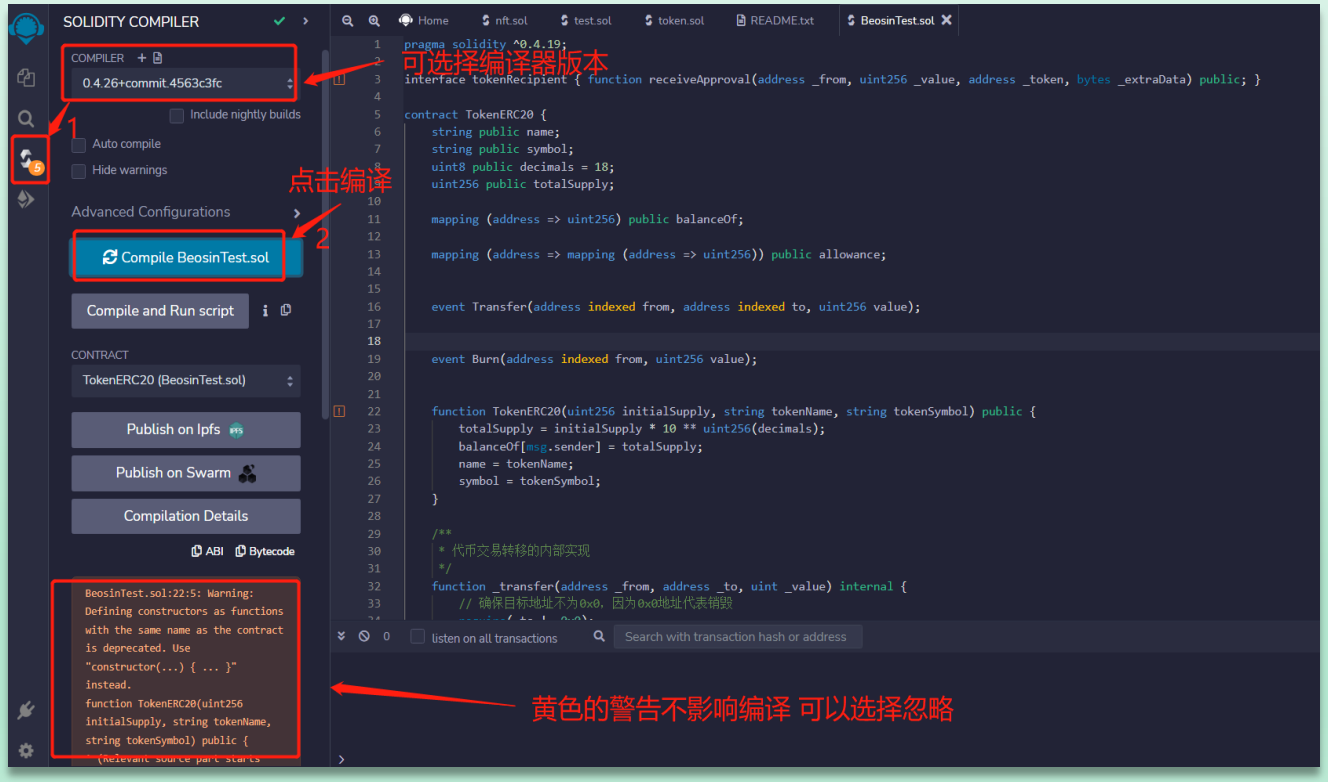
Remix IDE面板介绍

点击第一个图标，可以看到默认创建的工作空间，里面会有一些默认的合约样例代码，点击可以看到代码，这是一个ERC20代币合约。



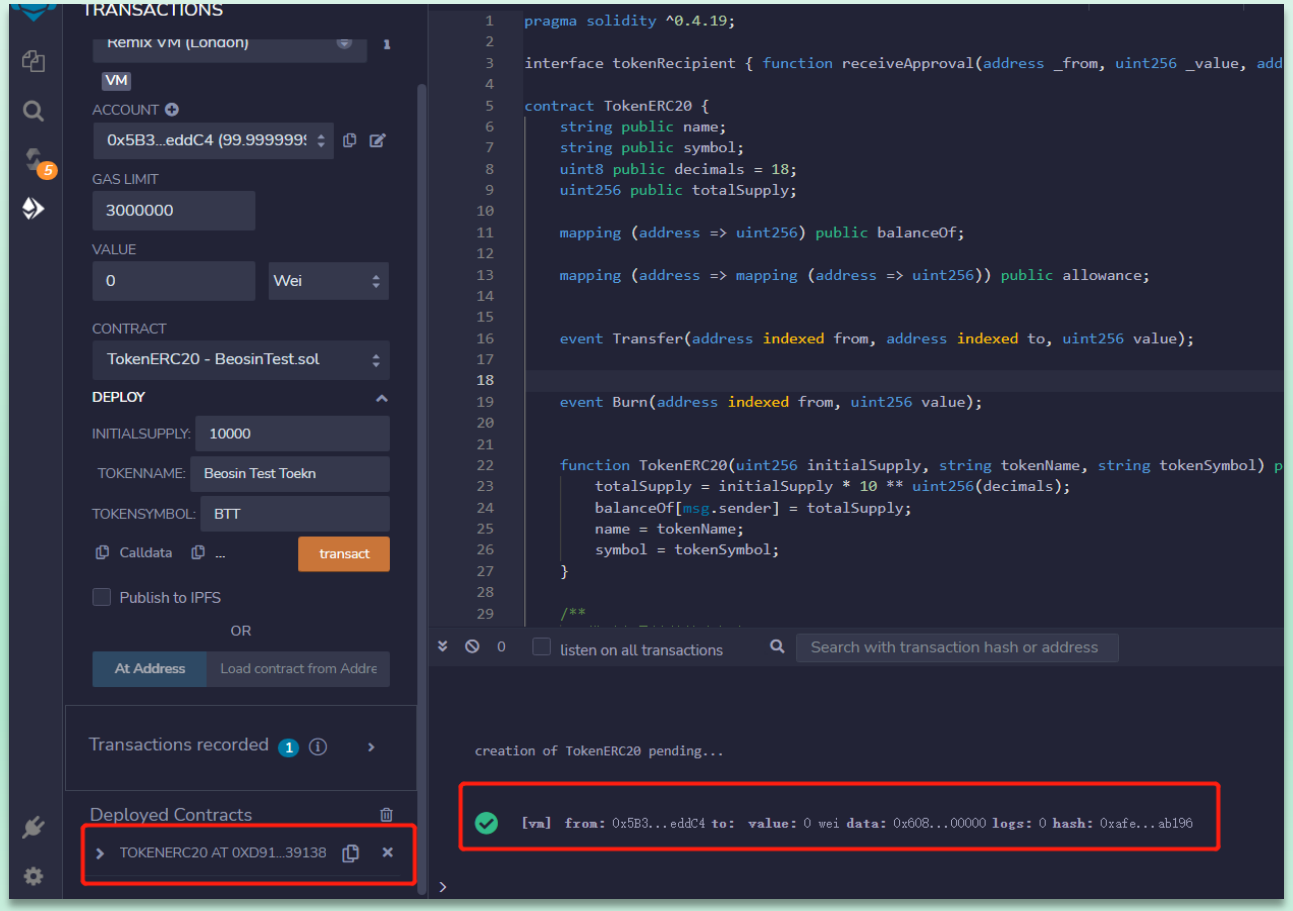
Remix IDE面板介绍

点击左边编译按钮跳转到编译功能页，在COMPILER处可选择编译器版本(一般点击编译后会自动选择)，点击Compile进行编译。



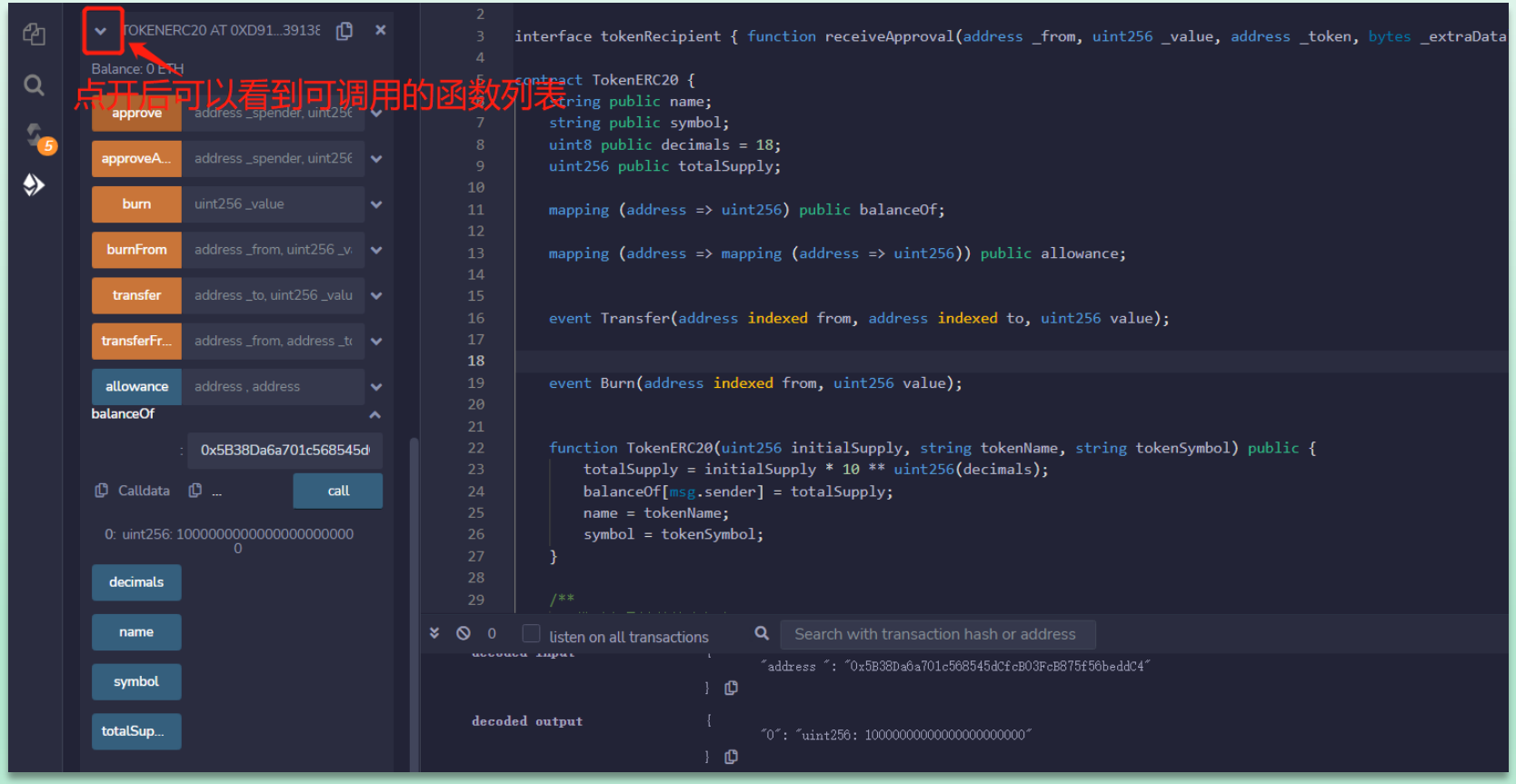
Remix IDE面板介绍

当看到Deployed Contracts下面新增了合约时，并且终端也返回成功的信息，此时我们的合约就已经在本地虚拟机上面成功部署了。



Remix IDE面板介绍

在remix中也可直接与部署的合约合约进行交互测试，这里调用balanceOf可以看到我们部署账户的余额。



FORESIGHT X

<OpenBuild Hack@HK/>

<Solidity Basics Workshop />

Frank

Security Tech Lead @ Beosin

workshop sign up code

Foresight X
Gateway to
Web3 in Hongkong

OpenBuild
HACK.
SERIES


BLOCKCHAIN ACADEMY
BEOSIN
Blockchain Security
课程练习

课程练习

- 完成一个包含所有基本元素的智能合约
- 使用Remix本地部署该合约
- 使用Remix调用合约进行交互

About Beosin

Beosin is a leading global blockchain security company co-founded by several professors from world-renowned universities and there are 40+ PhDs in the team. It has offices in Singapore, UAE, Korea, Japan and other 10+ countries. With the mission of "Securing Blockchain Ecosystem", Beosin provides "All-in-One" blockchain security solution covering Smart Contract Audit, Risk Monitoring & Alert, KYT/AML, and Crypto Tracing. Beosin has already audited more than 3000 smart contracts including famous Web3 projects PancakeSwap, Uniswap, DAI, OKSwap and all of them are monitored by Beosin EagleEye. The KYT AML are serving 100+ institutions including Binance.

- 3,000+ Smart Contracts Audited
- $500 Billion Protected Assets
- 85,000+ Identified Code Vulnerabilities
- 1 Million+ Users
- 1 Billion+ Crypto Addresses Labeled
- 20+ Years Cybersecurity Experience
- 150+ Members 40+ PhD Security Experts
- Formal Verification AI Data Analysis Over 97% Accuracy

BLOCKCHAIN ACADEMY × BEOSIN Blockchain Security

CONTACT US

- Website: www.beosin.com
- Email: contact@beosin.com
- Official Twitter: twitter.com/Beosin_com
- Alert Twitter: twitter.com/BeosinAlert
- Telegram: t.me/beosin
- Discord: discord.com/invite/B4QJxhStV4
- Linkedin: linkedin.com/company/beosin

FORESIGHT X

<OpenBuild Hack@HK/>

<Solidity Basics Workshop />

Frank

Security Tech Lead @ Beosin

workshop sign up code

Foresight X
Gateway to
Web3 in Hongkong

OpenBuild
HACK.
SERIES