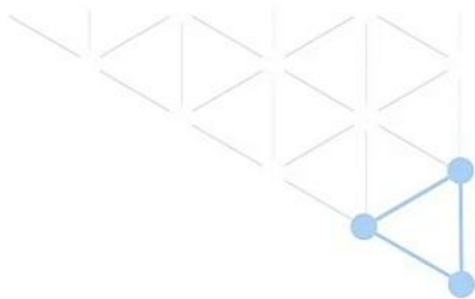


# Solidity 函数-上

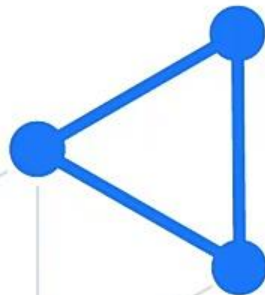
- 蚂蚁链《区块链系统开发与应用》A认证系列课程

## 课程 目标

- 了解函数的定义
- 了解函数参数与函数调用



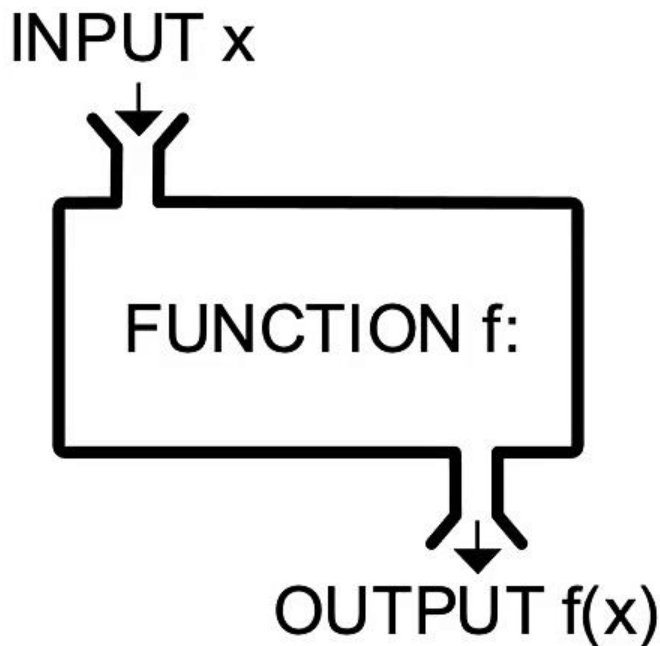
# 01 函数的定义



# 什么是函数？

## 什么是函数？

- 函数原本是数学上的概念，有着非常精确的定义：
  - 函数表示两不为空集的集合间的一种对应关系：  
输入值集合中的每项元素皆能对应唯一一项输出值集合中的元素；
  - 我们常用  $f(x)$  来表示一个函数，而  $f$  就是英文中 function 的简写；



# Solidity 函数

## Solidity 函数的概念

- Solidity 函数是一组可重用代码的包装，是合约代码中的可执行单元，接受输入，返回输出；

```
pragma solidity >=0.4.0 <0.6.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

# Solidity 函数

## 定义

- Solidity 中，我们可以使用如下方式定义一个函数：

```
function function_name(parameter1, parameter2 ...) scope returns(return_type) {  
    // Logic  
  
    // the type of the return value need to be consistent with return_type  
    return value;  
}
```

# Solidity 函数

## 与其他高级语言的比较

- Solidity 函数概念和其他高级语言中的函数概念类似，但是在表现形式上有些许不同，我们选取 Java 语言和 Javascript 语言的函数来比较一下

```
// Java
public return_type function_name(parameter1, parameter2 ...) {
    // Logic

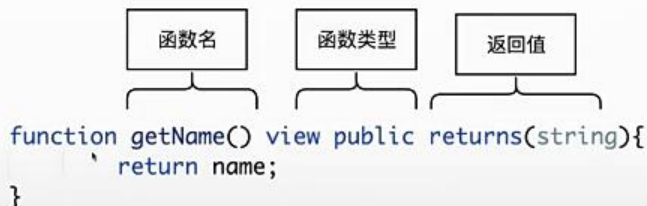
    // the type of the return value need to be consistent with return_type
    return value;
}
```

```
// Javascript
function function_name(parameter1, parameter2 ...) {
    // Logic
    return value/function;
}
```

# Solidity 函数

## 语法

- 可以看到，Solidity 语言在设计上参考了这些主流的编程语言，关于其语法我们总结如下：
  - 使用 `function` 关键字来声明一个函数；
  - 函数如果有返回值，必须在函数定义中显示申明函数的返回值类型；
  - 函数有可见性；
  - 函数参数需要用小括号括起来；



```
function getName() view public returns(string){  
    return name;  
}
```



编译

0.4.23

FunctionDemo16.sol



```
1 pragma solidity ^0.4.20;
2
3
4 contract FunctionDemo16 {
5     // 最简单的：无入参，也没有返回参数的函数
6     function |
7
8     // 有一个入参的函数
9
10    // 有多个入参的函数
11
12    // 有一个返回值的函数
13
14    // 有多个返回值的函数
15 }
```

保存



a00e36c5 - 开放联盟链...

编译

TX Hash: 0xfa1e90cd06ab24517ee6a3c86b...

function functionTest003

调用合约

function functionTest001

调用合约

function functionTest004

调用合约

tx hash

0x8676efc49e9c2f1843501b670d66...

input

output

uint256: 1608385377316

log

function functionTest002

调用合约

function functionTest005

调用合约

tx hash

0xc53284954bed1c2ef41eac2a826fe...

保存

FunctionDemo16.sol X

```
1 pragma solidity ^0.4.20;
2
3 contract FunctionDemo16 {
4     // 最简单的: 无入参, 也没有返回参数的函数
5     function functionTest001() public {
6
7     }
8
9     // 有一个入参的函数
10    function functionTest002(uint x) public {
11
12    }
13
14    // 有多个入参的函数
15    function functionTest003(uint x, uint y, string z) public {
16
17    }
18 }
```

编译

编译详情

合约分析

调试详情

交易详情

Filter

&gt; txhash: 0x913f02fd8c72b4a6d846131b408ebf59e6fde6b5b6446138584319e415f67141

&gt; txhash: 0x3d13da835e9351a696f67e7613b4410d50325edb18677447d19293f065952194

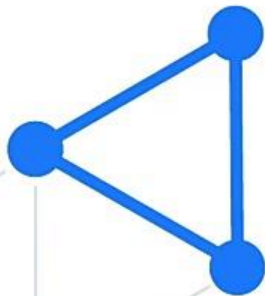
&gt; txhash: 0xdf86aeeccac0db71e189fa43aa85cd8f2d1437297ead5cd1c75816e07f9e0026

&gt; txhash: 0x7f29e2c3efab46fd1b6b674b77b2c73acdf13cb00d5da35ae1bf868b81a671a7

08:16



## 02 函数参数与函数调用



# 函数参数

## 函数参数

- 函数的入参定义与变量类似，可以省略未使用到的参数变量名，如左侧示例；
- 函数的返回参数有默认值，如整型会被初始化为0，如果没有被明确的设置值，那么它会一致保持0；

```
pragma solidity ^0.4.0;

contract Simple {
    // 函数入参的定义与变量定义类似
    function InputPara(uint a, uint) {
        // 省略未使用的参数
        a = a + 1;
    }
}
```

# 函数参数

## 函数调用

- 要进行函数调用，使用函数名加函数参数即可调用；
- 函数调用分为内部函数调用和外部函数调用；
  - 内部函数调用：只能在当前合约内被调用，调用形式为：function\_name()；
  - 外部函数调用：由地址和方法签名两部分组成，可作为外部函数调用的参数或返回值，调用形式为：  
this.function\_name() 或者 contract\_instance.function\_name();
- 内部调用和外部调用的区别在于是否是采用消息调用：
  - 内部函数调用：EVM 内部一个简单的指令跳转；
  - 外部函数调用：实际上是通过一个消息调用，而不是 EVM 内部指令跳转；
- 要调用其他合约的函数必须通过外部调用方式来完成；
- 不可以在构造函数中通过 this 调用函数；

# 随堂练习

## 练习题

- 练习创建一个无入参也无返回值的函数；
- 练习创建一个接收两个无符号整型参数的函数，并返回这两个无符号整型的乘积；
- 练习创建一个函数，调用第二个练习中的函数，并返回值；

# 谢谢



蚂蚁集团  
ANT GROUP



蚂蚁链  
ANTCHAIN