深入详解以太坊
智能合约语言Solidity

— 第1课 —

熊丽兵（Tiny熊）

公众号：登链学院　　微信号：xlbxiong

登链学院
edu.upchain.pro

自我介绍

- 熊丽兵（Tiny熊）
- 深入浅出区块链learnblockchain.cn
- 登链科技CTO

登链学院
edu.upchain.pro

登链学院

- 专注区块链技术普及推广
- 让每个程序员都懂区块链

登链学院
edu.upchain.pro

课程介绍

- 课程特点
- 面向人群
- 课程内容
- 学习收获

登链学院
edu.upchain.pro

课程特点

- 每节课10钟以内
- 30节以上
- 案例主导教学
- 和常规语言对比讲解

登链学院
edu.upchain.pro

面向人群

- 对区块链有基本了解
- 写过其他语言
- 有兴趣进一步了解智能合约开发

登链学院
edu.upchain.pro

课程内容

全面掌握Solidity语言特性

- 语言类型：基本类型、结构体、映射...
- 函数、修饰器、API 、事件、错误处理...
- 继承、库、重载...

登链学院
edu.upchain.pro

课程内容

Remix IDE 、MetaMask 工具使用

- 环境搭建
- 智能合约编译、调试、部署
- Remix高级用法...

登链学院
edu.upchain.pro

课程内容

工具库

- 最佳轮子：Openzepplin/SafeMath
- 字符串库、数组库
- 时间日期库
- …

登链学院
edu.upchain.pro

课程内容

大量实战经验的总结

- 如何节约GAS?
- 合约编写有哪些陷阱?
- 有哪些地方容易引发安全问题?

登链学院
edu.upchain.pro

学习收获

- Solidity 的实战能力显著提升
- 对智能合约理解上一个档次
- 胜任任何公司的智能合约开发岗位

谢谢观看

欢迎关注微信号/公众号交流技术问题

登链学院

Tiny熊

深入详解以太坊
智能合约语言Solidity

— 第2课 —

熊丽兵（Tiny熊）

微信号：xlbxiong　　公众号：登链学院

登链学院
edu.upchain.pro

以太坊核心概念

- 智能合约
- Solidity
- EVM
- 账户
- 交易、消息调用
- 钱包
- Gas
- 以太坊网络

智能合约

- 以太坊程序
- 不受干扰的执行
- 编程语言：Solidity、Vyper

Solidity

- HelloWorld合约

```
contract HelloWorld {
    function hello() public returns(string) {
        return "Hello World";
    }
}
```

- IDE: Remix

  https://remix.ethereum.org

EVM

- Solidity -> EVM
- Java -> JVM

账户

- 地址 (Address) : 20字节
- 状态 (State)

  交易序号、余额、数据存储（StorageRoot）、代码（codeHash）

登链学院
edu.upchain.pro

账户

两类账户

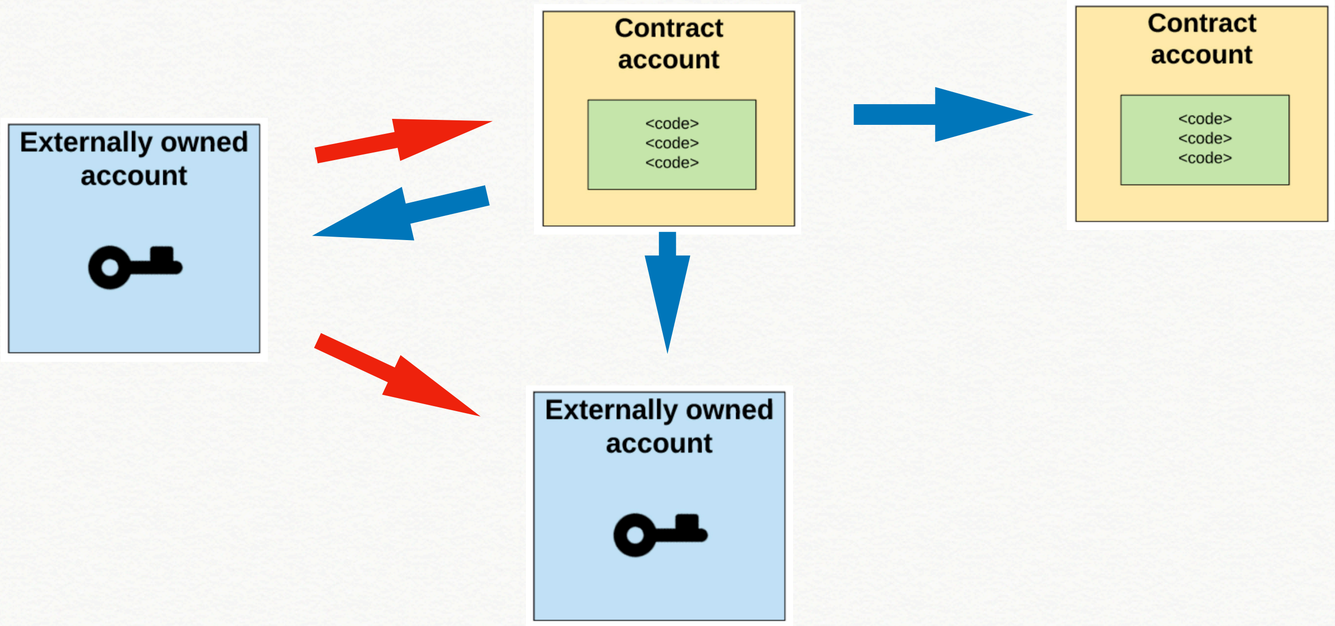
- 外部账户： EOA
- 合约账户

登链学院
edu.upchain.pro

账户

两类账户关系

账户之间可交互，但只能由外部账号发起　　合约无法定时执行



深入详解以太坊
智能合约语言Solidity

— 第3课 —

熊丽兵（Tiny熊）

微信号：xlbxiong　　公众号：登链学院

交易

- 从一个账户发送到另一个账户的消息
- 内容：以太币 + 数据payload
- 事务性（原子性）
- 触发消息调用

消息调用（内部交易）

- 来源、目标、数据、Gas、返回数据
- 消息调用由合约产生（交易由外部账户产生）
- 消息调用可以依次产生更多的消息调用
- 使用的是交易的gas
- 调用层数<=1024

谢谢观看

欢迎关注微信号/公众号交流技术问题

登链学院

Tiny熊

深入详解以太坊
智能合约语言Solidity

— 货币单位 与 Gas —

熊丽兵（Tiny熊）

微信号：xlbxiong　　公众号：登链学院

https://learnblockchain.cn/course/

登链学院
edu.upchain.pro

货币单位

- 单位：ether finney szaboo gwei wei

1 ether = 1 000 finney

1 finney = 1 000 szaboo

1 szaboo = 1 000 gwei

1 gwei = 1 000 000 000 wei

1 ether = 1 000 000 000 gwei

1 gwei = 1 000 000 000 wei

https://etherconverter.online/

登链学院
edu.upchain.pro

Gas 机制

gas（工作量单位）	汽油 （升）
gas价格（gwei）	每升汽油的价格（元）
gas Limit	提供多少升汽油（用不完可退）
矿工费：gas 用量 * gas价格	运费：汽油用量 * 汽油价格
费用由发起交易的账号支付	谁提任务谁付运费

Gas 机制

- 更高gas价格 交易更快确认

  https://etherscan.io/gastracker

- 交易复杂度决定了gas 用量

  gas Limit 需要大于gas 用量，交易才能顺利执行

  转账交易是 21000

  合约执行，gas 由合约复杂度决定
  合约的代码最终会解释为opcode，每个opcode对应的gas 消耗对应表：
  https://wiki.learnblockchain.cn/OPCODE_Gas.pdf

登链学院
edu.upchain.pro

Gas Limit

- Gas limit 太少，Out of gas异常（没油了），交易回滚

  如果异常，消耗的gas 费用被矿工被笑纳了

  gas limit 高一些，交易完成的可能性就高一些

  如果执行结束还有Gas剩余，剩余Gas将被返还

- 区块有Gas limit，限制任务量

  这也是为什么矿工优先选高gas 价格

https://ethstats.net/

深入详解以太坊
智能合约语言Solidity

— 第4课 —

熊丽兵（Tiny熊）

微信号：xlbxiong　　公众号：登链学院

钱包

- Metamask
- Geth
- Mist
- Parity
- Myetherwallet（https://www.myetherwallet.com/）

以太坊网络

- 主网
- 测试网络
- 私有链
- 模拟环境

谢谢观看

欢迎关注微信号/公众号交流技术问题

登链学院

Tiny熊

深入详解以太坊
智能合约语言Solidity

— 第5课 —

熊丽兵（Tiny熊）

微信号：xlbxiong　　公众号：登链学院

登链学院
edu.upchain.pro

Remix IDE

在线Remix

- https://remix.ethereum.org
- Remixd 神器：与本地共享目录

```
> npm install remixd -g
> remind -s <shared folder>
```

登链学院
edu.upchain.pro

Remix IDE

本地Remix

- remix-ide

```
> npm install remix-ide -g
> remix-ide
浏览器打开：http://localhost:8080
```

登链学院
edu.upchain.pro

Remix IDE

Remix App

- remix app

https://github.com/horizon-games/remix-app

其他编辑器

任意主流文本编辑器

- Atom + 插件
- VS code + 插件

谢谢观看

欢迎关注微信号/公众号交流技术问题

登链学院

Tiny熊

深入详解以太坊
智能合约语言Solidity

— 第6课 —

熊丽兵（Tiny熊）

微信号：xlbxiong　　公众号：登链学院

登链学院
edu.upchain.pro

开发环境安装

MetaMask安装

- https://metamask.io/
- 翻墙

登链学院
edu.upchain.pro

开发环境安装

MetaMask使用

- 初始化
- 切换网络
- 创建账号、导入账号
- 获取、发送以太
- 转移token

登链学院
edu.upchain.pro

开发环境安装

MetaMask连接节点

- 连接Geth

  https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum

- 连接Ganache

  https://trufflеframework.com/ganache

深入详解以太坊
智能合约语言Solidity

初探智能合约

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

初探智能合约

使用Solidity语言来编写一个简单的智能合约：
保存一个值到以太坊上

https://github.com/xilibi2003/leanSolidity

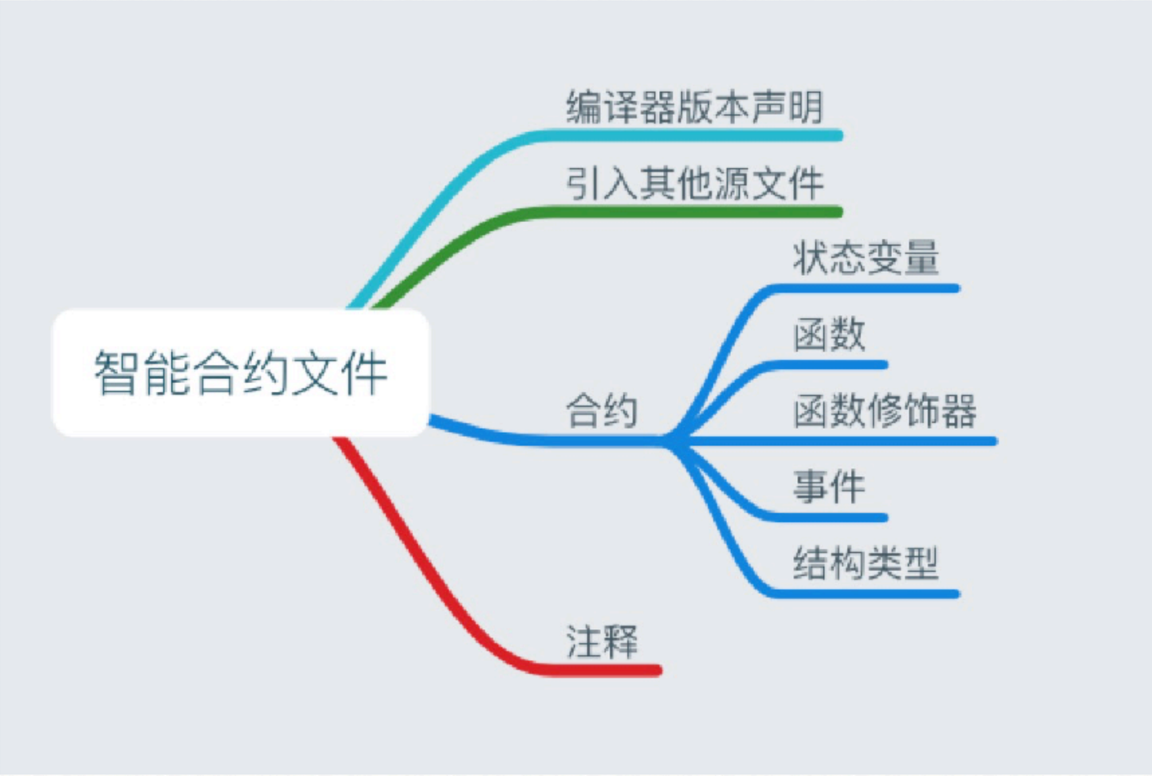
深入详解以太坊
智能合约语言Solidity

一个合约包含哪些内容

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

一个合约包含哪些内容



代码地址：https://github.com/xilibi2003/leanSolidity

深入详解以太坊
智能合约语言Solidity

Solidity 语言类型

Tiny熊

登链学院
edu.upchain.pro

Solidity语言类型

静态类型的语言

- 编译前变量类型需要确定

值类型

- 赋值或传参时，总是进行值拷贝

引用类型

- 赋值比较复杂，看情况，后面介绍

登链学院
edu.upchain.pro

值类型

- 布尔类型(Booleans)
- 整型(Integers)
- 定长浮点型(Fixed Point Numbers)
- 定长字节数组(Fixed-size byte arrays)
- 有理数和整型常量(Rational and Integer Literals)
- 字符串常量（String literals）
- 十六进制常量（Hexadecimal literals）
- 枚举(Enums)
- 函数类型(Function Types)
- 地址类型(Address)
- 地址常量(Address Literals)

注意：红色部分有亮点

https://solidity.readthedocs.io/en/v0.4.24/

https://learnblockchain.cn/2017/12/05/solidity1/

登链学院
edu.upchain.pro

bool类型

- 关键字：bool
- 取值

  true/false
- 支持运算

  !　　&&　　||　　==　　!=

https://github.com/xilibi2003/leanSolidity

深入详解以太坊智能合约语言Solidity

整型

Tiny熊

登链学院
edu.upchain.pro

整型

int

int8 int16 … int256

类型所占空间：位数

uint

uint8 … uint256

u: unsinged 无符号， 只能大于0

代码：**https://github.com/xilibi2003/leanSolidity**

登链学院
edu.upchain.pro

整型

运算

比较运算： <=, < , ==, !=, >=, >

位运算： &，|，^(异或)，~（位取反）

算术运算： +，-，-（负），*，/, %(取余数), **（幂）, << (左移位), >>(右移位)

/ （除） ： 截断 ， 除0会抛异常

x << y 和乘等价，左移1位相当于乘2

x >> y 和除等价，右移1位相当于除2

代码：https://github.com/xilibi2003/leanSolidity

整型

运算符简写

- a += e 等价 a = a + e

  -=, *=, /=, %=, |=, &=, ^=

- a++ 等价 a += 1

  - -

思考:

uint a = 1;

c = a++ 与 c = ++a

c = 1　　c = 2

深入详解以太坊智能合约语言Solidity

整型安全使用：溢出问题

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

整型溢出问题

在使用整型时，要特别注意整型的大小及所能容纳的最大值和最小值

如：uint8 最大值为0xff （255）
最小值为0

```
uint8 x = 0x80;
uint8 y = x * 2;  // y 溢出后为0

uint8 i = 0xf0;
uint8 j = 0x10;
uint8 k =  i +  j;  // k 溢出后为0

uint8 m = 0x01;
uint8 n =  m - 2;  // n 溢出后为255
```

https://learnblockchain.cn/2018/04/25/bec-overflow/

代码：https://github.com/xilibi2003/leanSolidity

整型溢出问题

如何解决

- assert
- SafeMath库（以后讲解）

深入详解以太坊智能合约语言Solidity

定长浮点型、定长字节数组

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

定长浮点型

fixed/ufixed

类似其他语言的浮点型float和double

- fixedMxN　ufixedMxN

所占空间：位数

小数点位数

M： 以8步进，可为8到256位

N：可为0到80之间

登链学院
edu.upchain.pro

定长字节数组

占空间固定的数组

bytes1, bytes2, bytes3, ..., bytes32

表示所占空间的字节数

.length成员

获取字节数组的长度

定长字节数组

- 像字符串一样使用
- 像整型一样运算比较和位运算

比较符: <=, <, ==, !=, >=, > (返回bool)

位操作符: &, |, ^ (按位异或)，~（按位取反）, << (左移位), >> (右移位)

- 像数组一样进行索引

索引（下标）访问: 如果x是bytes8，当0 <= k < 8 ，则x[k]返回第k个字节

深入详解以太坊
智能合约语言Solidity

常量（literals）

熊丽兵（Tiny熊）

常量

- 数字(有理数、整数)常量
- 字符串常量
- 十六进制常量

登链学院
edu.upchain.pro

数字（有理数、整数）常量

表达式中直接出现的数字

数字常量表达式本身支持任意精度，也就是可以不会运算溢出，除法运算也不会截断。

- 0.1
- 科学型：2e2　-2e10

-2 * 10^10

- 0.5 * 8
- 5/2 + 5/2 = ?

5

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

字符串常量

表达式中直接出现的字符串

- "登链学院"
- 支持转义：\n　　\x61　　\u718A

换行符　　a ASCII　　熊 Unicode

字符串常量不支持任何运算，如字符串拼接+

https://baike.baidu.com/item/转义字符

https://www.bejson.com/convert/unicode_chinese/

登链学院
edu.upchain.pro

十六进制常量

表达式中直接出现的字符串：内容是16进制数

- hex"001122ff"
- 转换为字节数组

深入详解以太坊
智能合约语言Solidity

枚举

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

枚举

关键字enum 用来自定义类型

- 可与整数进行转换，但不能进行隐式转换
- 枚举类型应至少有一名成员

代码：https://github.com/xilibi2003/leanSolidity

深入详解以太坊
智能合约语言Solidity

地址类型

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

地址类型

address：表示一个账户地址（20字节）

0x72ba7d8e73fe8eb666ea66babc8116a41bfb10e2（地址常量）

成员

属性：.balance　单位是为wei 1eth = 10^18wei

函数： transfer()

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

地址类型深入详解

send() 成员函数

错误时不发生异常返回false，使用时一定要检查返回值，大部分时候使用transfer()

addr.transfer(y) 等价 require(addr.send(y))

send()及transfer() 2300 gas 限制 当合约接收以太币时，转账很容易失败

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

地址类型高级用法

call() 成员函数

addr.call(函数签名, 参数)

.value() 附加以太币　　addr.call.value(y)() 功能上类似 addr.transfer(y)

.gas() 指定gas

delegatecall() 成员函数

不支持 .value()

深入详解以太坊
智能合约语言Solidity

函数类型

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

函数类型

函数是一种类型

像其他类型一样，函数类型可以作为变量类型，也可以作为返回值类型（函数式语言的特点）

两种函数类型

外部(external)函数：发起EVM消息调用，使用: 地址.函数名 进行调用

内部(internal)函数：代码跳转调用， gas小的多，但无法在合约外部调用，使用: 函数名 进行调用

默认是内部(internal)函数

登链学院
edu.upchain.pro

函数类型

声明

声明一个函数：
function foo(int);
function foo(int) external returns (int);

声明一个函数类型的变量：
function (int) foo;
function (int) external returns (int) foo;

成员.selector 属性

返回函数选择器

内部函数没有.selector 属性

代码：https://github.com/xilibi2003/leanSolidity

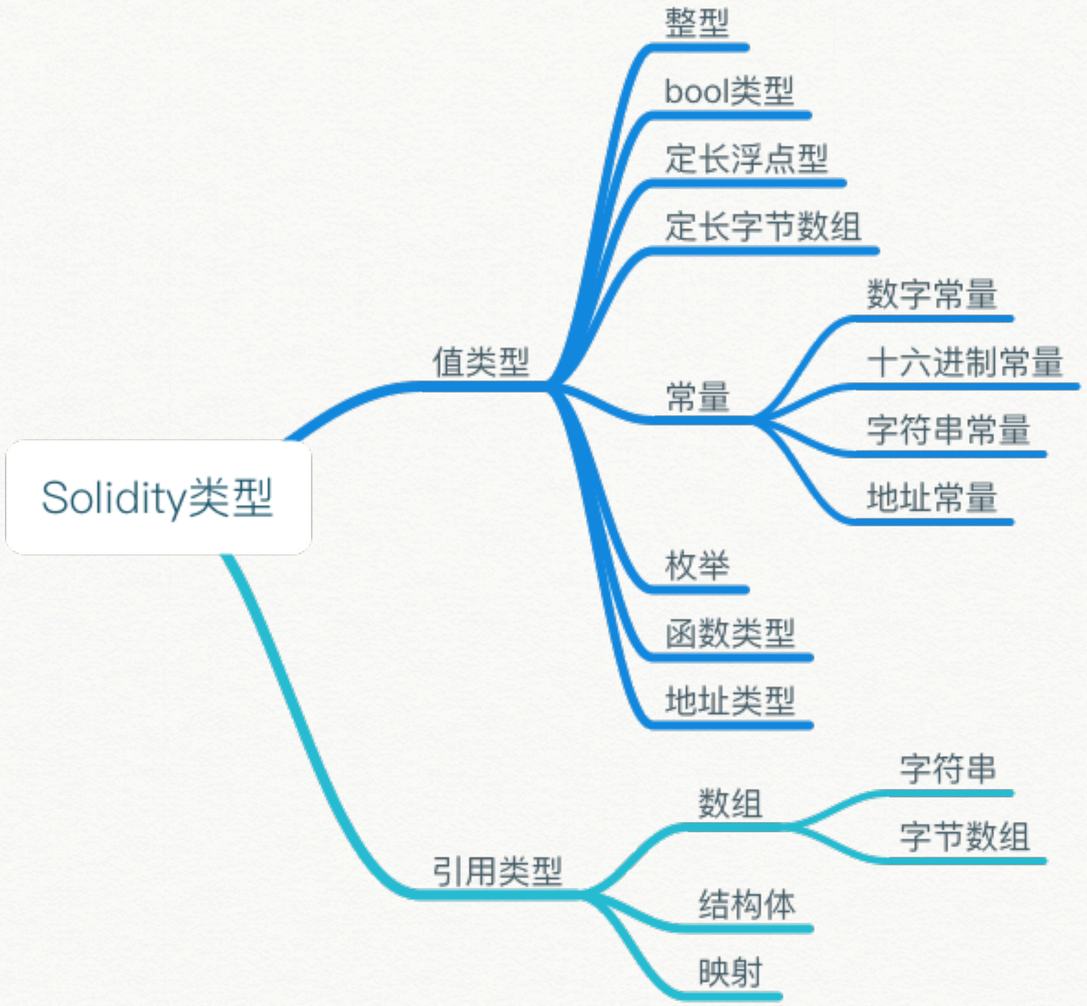
深入详解以太坊智能合约语言Solidity

引用类型之存储位置

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

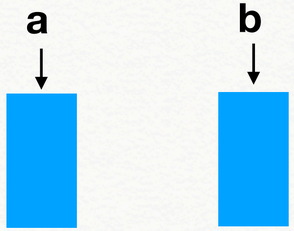
Solidity语言类型



登链学院
edu.upchain.pro

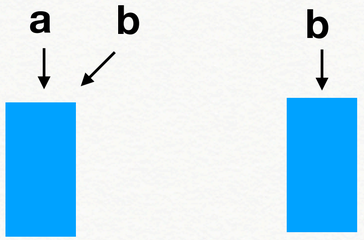
Solidity语言类型

值类型



b = a

引用类型



b = a

登链学院
edu.upchain.pro

存储位置

storage（区块链中）

1. 状态变量

2. 复杂类型的局部变量

```
contract A {
    uint a;
}
```

```
contract A {
    function fun() {
        uint[] arr;
    }
}
```

memory(EVM内存中)

局部变量及参数

```
contract A {
    function fun(uint[] a) {
        uint b;
    }
}
```

登链学院
edu.upchain.pro

存储位置

在memory和storage之间总是会创建一个完全独立的拷贝。

状态变量之间相互赋值，总是会创建一个完全独立的拷贝。

同样的数据位置之间赋值只是引用的传递

文档：https://learnblockchain.cn/2017/12/21/solidity_reftype_datalocation/

代码：https://github.com/xilibi2003/leanSolidity

深入详解以太坊
智能合约语言Solidity

数组

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

数组

声明 T[k]

T: 元素类型　k: 数组长度（可选）

uint [10] tens;　　uint [ ] us;

uint [] public u = [1, 2, 3];　　public数组状态变量会自动生成一个对应变量的函数

使用new 关键字

uint[] c = new uint[](7);

登链学院
edu.upchain.pro

数组

通过下标访问

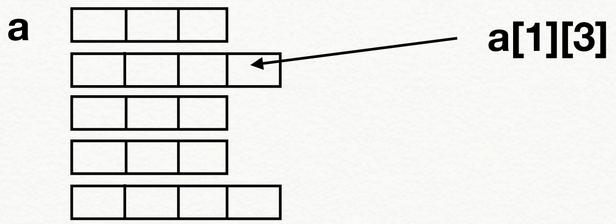
tens[0];

tens[0] = 10;

下标不能超出当前数组的长度

多维数组

uint[][5] a; //5个元素都是变长数组



代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

数组成员

.length 获取长度

uint [10] tens;　tens.length 为10

storage的变长数组，可以通过给.length赋值调整数组长度

memory数组一旦创建，大小不可调整

.push() 添加元素，返回新长度

仅针对变长数组

memory数组不支持

代码：https://github.com/xilibi2003/leanSolidity

深入详解以太坊
智能合约语言Solidity

字节数组 与 字符串

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

字节数组

字节数组bytes

类似 byte[ ]， 不过bytes作为外部函数的参数时空间更小

```
bytes bs;
bytes bs0 = "12abcd";
```

字符到utf8：https://www.qqxiuzi.cn/bianma/Unicode-UTF.php

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

字符串

字符串也是数组

string str0;

string str1 = “登链学院";

string str2 = "TinyXiong";

没有.length属性 bytes(s).length 获取字节数组的长度

bytes(s)[k]：获取字节数组下标k的UTF-8 编码

bytes: 用来存储任意长度的字节数据
string: 用来存储UTF-8编码的字符串数据

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

stringutils

https://github.com/Arachnid/solidity-stringutils

- 字符长度
- 匹配字符串： find(), startsWith(),endsWith()
- 字符串拼接
- …

代码：https://github.com/xilibi2003/leanSolidity

深入详解以太坊智能合约语言Solidity

映射

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

映射

关键字mapping

```
mapping(address => uint) public balances;

mapping(address => uint) public userLevel;
```

键类型不能是变长数组、合约类型、嵌套类型　　值类型无限制

索引访问

```
balances[userAddr];
```

如果访问一个不存在的键，返回的是默认值。

登链学院
edu.upchain.pro

映射 局限性

- 只能作为状态变量
- 无法遍历访问

没有长度

没有键集合

没有值集合

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

可遍历的映射

- iterable_mapping

https://github.com/ethereum/dapp-bin/blob/master/library/iterable_mapping.sol

代码：https://github.com/xilibi2003/leanSolidity

深入详解以太坊
智能合约语言Solidity

结构体

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

结构体

关键字struct 自定义类型

基本类型
数组
结构体
映射

```
struct  CustomType {
    bool myBool;
    uint myInt;
}
```

```
struct CustomType {
    bool myBool;
    uint myInt;
    CustomType my;
}
```

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

结构体

声明与初始化

- 仅声明变量，不初始化

```
CustomType ct1;
```

- 按成员顺序初始化

```
CustomType ct1 = CustomType(true, 2);
```

```
struct  Type3 {
        string name;
        mapping(string=>uint) score;
        int age;
    }
Type3 ct = Type3("tiny", 2);  // 跳过对mapping的初始化
```

登链学院
edu.upchain.pro

结构体

声明与初始化

- 命名方式初始化

```
CustomType ct = CustomType({ myBool: true, myInt: 2});
```

不按顺序初始化，
参数数量需一致，同样忽略mapping类型

访问与赋值

```
CustomType ct;
 ct.myBool = false;
```

登链学院
edu.upchain.pro

结构体

限制

结构体目前仅支持在合约内部使用，或继承合约内使用，
如果要在参数和返回值中使用结构体，函数必须声明internal

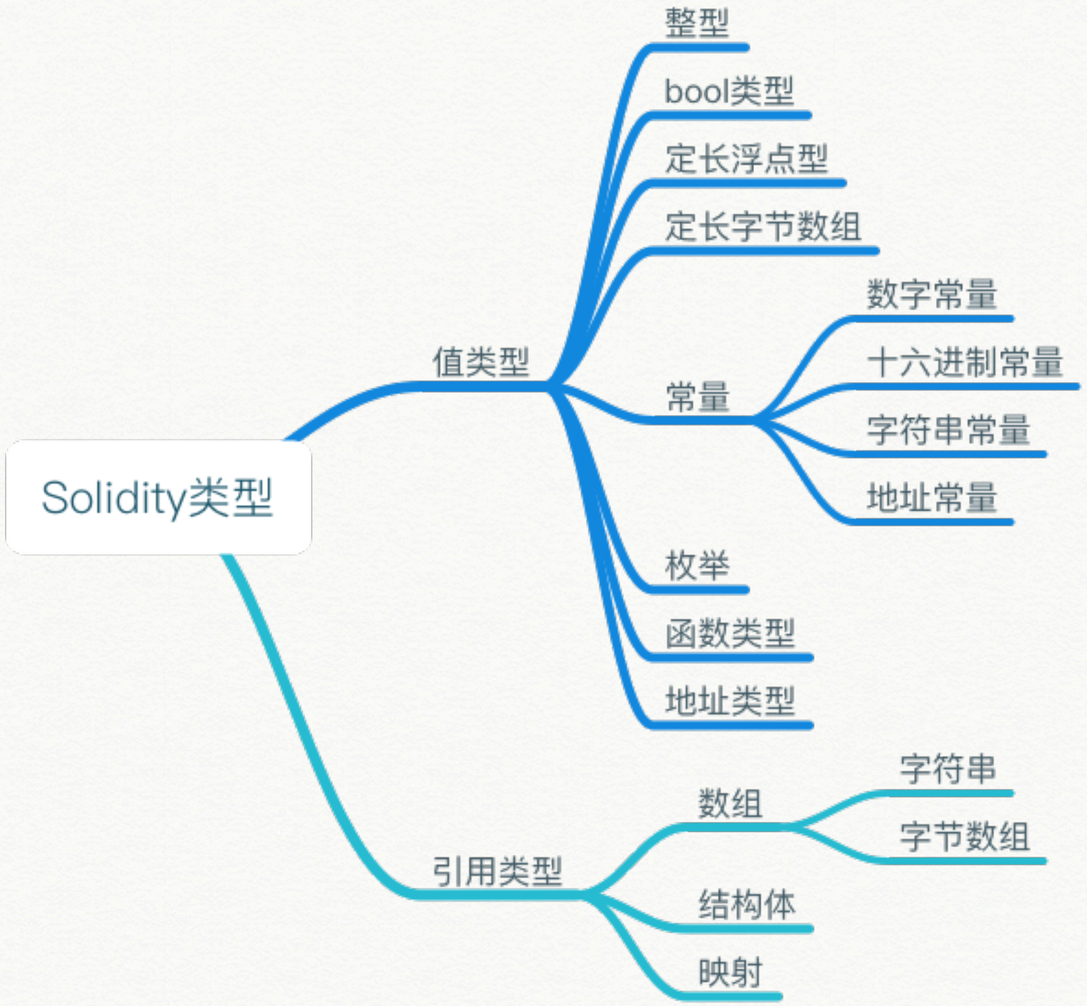
```
function interFunc(CustomType ct) internal {
    }

function exterFunc(CustomType ct) public {

    }
```

登链学院
edu.upchain.pro

Solidity语言类型



深入详解以太坊智能合约语言Solidity

类型转换、delete

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

类型转换

将一个类型转为另一个类型，转换可分为隐式转换和显式转换。

隐式转换

运算符两边有不同的类型，不会丢失数据下，编译器会尝试隐式转换类型

uint8 -> uint16,uint256

uint16,uint256 -> uint8 

类型转换

将一个类型转为另一个类型，转换可分为隐式转换和显式转换。

显式转换

如果编译器不允许隐式的自动转换，但你知道转换没有问题时，可以进行强转。

强转需谨慎

不正确的转换会带来错误

如果转换为一个更小的类型，高位将被截断

登链学院
edu.upchain.pro

delete

重置变量

bool -> false;
uint -> 0

address -> 0x0
bytes -> 0x0
string -> “"

```
uint[] memory arr = new uint[](7);
delete arr;    // a.length = 0;

CustomType memory ct = CustomType(true, 100);
delete ct;  // ct.myBool = false ; myInt = 0;
```

delete 对映射无效

delete 不影响值拷贝的变量

深入详解以太坊
智能合约语言Solidity

Solidity 时间及日期

熊丽兵（Tiny熊）

时间及日期

时间单位

seconds, minutes, hours, days, weeks, years

1 == 1 seconds

1 minutes == 60 seconds

1 hours == 60 minutes

1 days == 24 hours

1 weeks = 7 days

1 years = 365 days

时间换算并不精确

登链学院
edu.upchain.pro

时间及日期

获取当前时间

```
now;  // 当前块的时间戳 = block.timestamp
```

http://tool.chinaz.com/Tools/unixtime.aspx

如何实现一天内只能执行一次?

代码：https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

DateTime库

- 实现时间戳与日期时间的转换

https://github.com/pipermerriam/ethereum-datetime

深入详解以太坊
智能合约语言Solidity

Solidity API 之 区块和交易

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

Solidity API

- 区块和交易
- ABI 编码
- 错误处理
- 数学及加密
- 地址及合约

https://learnblockchain.cn/2018/03/14/solidity-api/

https://solidity.readthedocs.io/en/v0.4.24/units-and-global-variables.html

登链学院
edu.upchain.pro

区块和交易

blockhash（uint blockNumber） returns (bytes32)：返回给定区块号的哈希值，只支持最近的256个区块.
block.coinbase (address): 当前块矿工的地址,括号中表示返回值的类型（下同）。
block.difficulty (uint):当前块的难度。
block.gaslimit (uint):当前块的gaslimit。
block.number (uint):当前区块的块号。
block.timestamp (uint): now 当前块的Unix时间戳
gasleft() (uint256): 获取剩余gas。
msg.data (bytes): 完整的调用数据（calldata）。
msg.sender (address): 当前调用发起人的地址。
msg.sig (bytes4):调用数据(calldata)的前四个字节（例如为：函数标识符）。
msg.value (uint): 这个消息所附带的以太币，单位为wei。
tx.gasprice (uint) : 交易的gas价格。
tx.origin (address): 交易的发送者（全调用链）

https://learnblockchain.cn/2018/03/14/solidity-api/

深入详解以太坊
智能合约语言Solidity

Solidity API 之 ABI编码函数

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

ABI

ABI： Application Binary Interface 应用程序二进制接口

调用一个合约函数 = 向合约地址发送一个交易

交易的内容就是 ABI 编码数据

用ABI编码传递数据给合约，因此与合约交互离不开ABI

登链学院
edu.upchain.pro

ABI

0x60fe47b1
0000000000000000000000000000000000000000000000000000000000000001

- sha3(“set(uint256)”) == 0x60fe47b1
- 1 == 0x00 … 001

登链学院
edu.upchain.pro

ABI编码函数

abi.encode(...) returns (bytes) : 计算参数的ABI编码

abi.encodePacked(...) returns (bytes) : 计算参数的紧密打包编码

abi. encodeWithSelector(bytes4 selector, ...) returns (bytes)：
计算函数选择器和参数的ABI编码

abi.encodeWithSignature(string signature, ...) returns (bytes): 等价于
abi.encodeWithSelector(bytes4(keccak256(signature), ...)

深入详解以太坊
智能合约语言Solidity

Solidity API 错误处理

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

错误处理

什么是错误处理

错误处理是指在程序发生错误时的处理方式

传统语言

```
do something1;

try {
  do error;
} catch () {
}

do something2;
```

Solidity

```
do something1;

do error;

do something2;
```

登链学院
edu.upchain.pro

错误处理

为什么没有try catch

区块链是全球共享的分布式事务性数据库

事务性

都生效，都不生效

错误处理函数

assert(bool condition)
用于判断内部错误，条件不满足时抛出异常

require(bool condition)
用于判断输入或外部组件错误，条件不满足时抛出异常
require(bool condition, string message)
同上，多了一个错误信息。

revert()
终止执行并还原改变的状态
revert(string reason)
同上，提供一个错误信息。

登链学院
edu.upchain.pro

错误处理

assert类型异常

1. 下标（序号）越界
2. 被除数为0， 如5/0 或 23 % 0
3. 负移位。如:5<<i; i为-1时
4. 过大整数或负值转为枚举类型，则抛出异常
5. 调用未初始化内部函数类型的变量。
6. 调用assert的参数为false

....

程序代码异常

消耗掉所有剩余的gas

登链学院
edu.upchain.pro

错误处理

require类型异常

1. 调用throw
2. 调用require的参数为false
3. 外部函数调用、消息调用 出错
4. 没有payable 的public的函数 接收以太币时
5. 如果.transfer()执行失败

...

外部条件异常

剩余gas会返还给调用者

登链学院
edu.upchain.pro

require() 还是 assert()

经验总结

1. 用于检查用户输入；
2. 用于检查合约调用返回值， 如require(addr.send(1));
3. 用于检查状态，如：msg.sender == owner；
4. 通常用于函数的开头
5. 不知道使用哪一个的时候，就使用require

1. 用于检查溢出错误， 如`z = x + y ; assert(z >= x);`
2. 用于检查不应该发生的异常情况；
3. 用于在状态改变之后，检查合约状态；
4. 尽量少使用assert;
5. 通常用于函数中间或结尾;

深入详解以太坊
智能合约语言Solidity

数学及加密函数 API

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

数学函数 API

addmod(uint x, uint y, uint k) returns (uint)
计算(x + y) % k， 支持任意精度

mulmod(uint x, uint y, uint k) returns (uint)
计算(x * y) % k， 支持任意精度

https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

哈希函数

哈希函数（散列函数）：任意长度的输入，通过散列算法，变换成固定长度的输出

MD4 、 MD5、ripemd-160
SHA（Secure Hask Algorithm）密码散列函数家族

SHA家族:
SHA-1
SHA-2：SHA-224、SHA-256、SHA-384，和SHA-512
SHA-3：Keccak 算法

登链学院
edu.upchain.pro

哈希函数 API

keccak256(…) returns (bytes32)
sha3(...) returns (bytes32)

sha256(...) returns (bytes32)

ripemd160(...) returns (bytes20)

登链学院
edu.upchain.pro

加密函数 API

ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address)

通过椭圆曲线签名来恢复与公钥关联的地址

四个参数
hash: 被签名数据的哈希结果值，r，s，v分别来自签名结果串
r = signature[0:64]
s = signature[64:128]
v = signature[128:130] + 27

深入详解以太坊
智能合约语言Solidity

地址及合约 API

熊丽兵（Tiny熊）

地址相关API

balance 属性
获取余额

transfer(uint256 amount)
send(uint256 amount) returns (bool):
转账

call(...) returns (bool)
delegatecall(...) returns (bool)
底层合约函数调用

可复习地址类型章节

合约相关API

this : 表示当前合约，可以转会为一个地址

selfdestruct(address recipient)
suicide(address recipient)
销毁合约，并把它所有资金发送到给定的地址

登链学院
edu.upchain.pro

Solidity API

- 区块和交易
- ABI 编码
- 错误处理
- 数学及加密
- 地址及合约

https://learnblockchain.cn/2018/03/14/solidity-api/

https://solidity.readthedocs.io/en/v0.4.24/units-and-global-variables.html

深入详解以太坊
智能合约语言Solidity

函数修改器

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

函数修改器

modifier 关键字定义函数修改器

函数修改器(Modifiers)可以用来改变一个函数的行为。
通常用于在函数执行时检查某种前置条件。

```
modifier onlyOwner {
        require(msg.sender == owner);
        _;
}

function kill() public onlyOwner {
  // do something;
}
```

函数修改器修饰函数时，函数体被插入到 “_;”

登链学院
edu.upchain.pro

函数修改器

- 函数修改器可被继承使用
- 函数修改器可接受参数
- 多个函数修改器一起使用

空格隔开，修饰器会依次检查执行

深入详解以太坊
智能合约语言Solidity

函数修改器 进阶

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

函数修改器

理解函数修改器执行流

在修改器中或函数内return语句，仅仅跳出当前的修改器或函数，”_”后继续执行。

```
modifier noReentrancy() {
        require(!locked);
        locked = true;
        _;
        locked = false;
    }
```

```
function f() public noReentrancy returns (uint) {
        return 7;
    }
```

深入详解以太坊
智能合约语言Solidity

函数修饰符
payable view pure

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

函数修饰符-payable

- 表示一个函数能附加以太币调用
  - 普通函数
  - 构造函数
  - 回退函数

登链学院
edu.upchain.pro

函数修饰符-view

表示一个函数不能修改状态

view 和 constant 等价，constant在0.5版本之后会弃用

本地执行不消耗gas

下面几种情况认为是修改了状态：

1. 写状态变量
2. 触发事件（events）
3. 创建其他的合约
4. call调用附加了以太币
5. 调用了任何没有view或pure修饰的函数
6. 使用了低级别的调用（low-level calls）

登链学院
edu.upchain.pro

函数修饰符-pure

表示一个函数不读取状态，也不修改状态

本地执行不消耗gas

下面几种情况认为是读取了状态：

1. 读状态变量
2. 访问了.balance 属性
3. 访问了block、tx、msg 成员 (msg.sig 和 msg.data除外).
4. 调用了任何没有pure修饰的函数

深入详解以太坊
智能合约语言Solidity

继承

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

继承

- 继承的合约可以访问所有非private成员

- is 表示继承，通过复制代码的方式实现继承

```
contract  A {
}

contract B is A {
}
```

登链学院
edu.upchain.pro

继承

构造函数

派生合约需要调用父合约的构造函数，
如果有参数，派生合约需要提供参数调用父合约构造函数

```
contract Derived1 is Base(7) {
}
```

```
contract Derived2 is Base {
    constructor(uint _y) Base(_y * _y) public {
    }
}
```

登链学院
edu.upchain.pro

多重继承

多重继承

is后接多个合约，基类合约在is后的顺序很重要

继承顺序原则是从最接近基类到最接近派生类

```
contract X {}
contract A is X {}
contract C is A, X {}
```

登链学院
edu.upchain.pro

继承

抽象合约

合约存在没有函数体（实现）的的函数

合约不能通过编译，只能被继承

接口

没有任何函数是已实现的

函数不能继承其它合约，或接口

不能定义构造器、变量、结构体、枚举类

深入详解以太坊智能合约语言Solidity

库

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

库

一个特殊的合约

可以像合约一样进行部署，但是没有状态变量、不能存以太币

可重用

部署一次，在不同的合约内反复使用

节约gas，相同功能的代码不用部署一遍又一遍

库

定义库、使用库

```
library  mathlib {
    plus();
}
```

```
contract C {
   mathlib.plus();
}
```

库函数使用委托的方式调用DELEGATECALL，库代码是在发起合约中执行的

using for 扩展类型

using A for B 把库函数(从库A)关联到类型B

A库有函数add(B b), 则可使用b.add()

登链学院
edu.upchain.pro

常用库

openzeppelin

https://github.com/OpenZeppelin/openzeppelin-solidity

是库更是最佳实践

SafeMath

ERC20

ERC721

Ownership

Whitelist

Crowdsale

常用库

ethereum-libraries

https://github.com/modular-network/ethereum-libraries

ArrayUtilsLib　　StringUtilsLib

dapp-bin

https://github.com/ethereum/dapp-bin/tree/master/library

iterable_mapping可遍历Map　　StringUtils 字符串比较

linkedList 双向链表

stringutils

https://github.com/Arachnid/solidity-stringutils

深入详解以太坊智能合约语言Solidity

回退函数

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

回退函数（Fallback）

无名称 无参数 无返回值 函数

一个合约可以有一个回退函数

当给合约转以太币时，需要有payable回退函数

如果调用合约时，没有匹配上任何函数，就会调用回退函数。

登链学院
edu.upchain.pro

回退函数（Fallback）

如何回退函数在接收以太币时执行，可能仅有2300gas来执行，下面的操作消耗大于2300gas:

1. 写存储(storage)
2. 创建一个合约
3. 执行一个外部(external)函数调用，会花费非常多的gas
4. 发送ether

深入详解以太坊
智能合约语言Solidity

调试

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

调试

- 如何进入调试
- 如何设置断点
- 如何分析调试信息

https://github.com/xilibi2003/leanSolidity

深入详解以太坊智能合约语言Solidity

事件

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

事件

如何理解事件

外部获取智能合约的状态变化

事件其实是以太坊日志接口，可索引

https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

事件

如何使用

```
//声明事件，可继承
event  EventName(uint param);

//发出一个事件：
emit EventName(10);
```

https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

事件

DAPP监听使用

```
var ev = contractInstance.EvevtName();

ev.watch(function(err, result) {
      result.args.name;
})
```

区块链全栈-以太坊DAPP开发实战

https://wiki.learnblockchain.cn//course/dapp.html

深入详解以太坊
智能合约语言Solidity

总结

熊丽兵（Tiny熊）

登链学院
edu.upchain.pro

恭喜大家学完课程

这么枯燥的课程都可以学完，还有什么做不到呢?

登链学院
edu.upchain.pro

总结

- 以太坊核心概念

  账号、合约、交易、调用、GAS、钱包

- 全面介绍Solidity

  类型、修饰器、API、错误处理、继承、库、调试、事件

https://github.com/xilibi2003/leanSolidity

登链学院
edu.upchain.pro

进一步学习

- 多加练习、读源码
- 通过代币学智能合约
- 以太坊DAPP开发实战

谢 谢 ~

登链学院

Tiny熊