以太坊Solidity编程

郑嘉文

2020/09

基本概念 *以太坊，智能合约，GAS，账户， 交易，地址，EVM*

编程相关 *Nodejs, npm,web3*

测试及部署 *Ganache，Truffle*

公共测试环境 *Ropsten，Rinkeby，Kovan*

托管节点 *Infura*

钱包 *MetaMask， Mist*

调试工具 *Remix*

区块链浏览器 *etherscan*

内容

本节课主要讲述Solidity语言的基础知识：

- 文件结构
- 变量
- 函数
- 语句
- 继承
- 其他概念

第二课目录/Contents

01

文件结构

sol文件结构

编译开关/Pragma

- pragma solidity ^0.4.0；

注释/Comment

- 代码注释：单行注释 -- // ，多行注释 -- /* */
- 文档注释：单行注释 -- ///，多行注释-- /** **/

引入/Import

- import {symbol1 as alias, symbol2} from “filename”

Contract/Library/Interface

- Contract XXX {…}
- Library XXX {…}
- Interface XXX {…}

合约文件结构

状态变量/State Variable

- 值类型：布尔，整型，定长数组，字符串…
- 引用类型：结构，不定长数组…

结构/struct

- Struct XXX{ …}

修饰符/Modifier

- 标准修饰符： external, internal， pure， view …
- 自定义修饰符

事件

- Event Read(address,int)

枚举

- enum gender {male, female}

函数

- function (<parameter types>) {internal(默认)|external} [constant] [payable] [returns (<return types>)]

02

变量，语句

布尔	• True or false • If (1) {…}是非法的，不过可以强制类型转换
整型	• Int， int8， int16 … int256 • Uint,uint8, uint16 … uint256
地址	• 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
定长字节数组	• Bytes1, bytes2… bytes32 • 定长字符串
枚举	• Enum gender {male, female}
函数	• Function XXX (uint) return (uint256)

引用类型/Reference Type

数组	• localBytes= new bytes (10)
结构	• Struct{…}
字符串	• String name = ‘this is a test for Jinse online course’ ; • 可以转换成bytes类型 • 不能被indexed,push,以及使用length
字典/映射	• mapping(uint => Address)

数组的属性/Array Property

index	• 返回指定位置的数组元素
push	• 仅仅支持动态数组
length	• 除了string类型，所有数组都支持 • 只有动态数组和Bytes支持length属性

循环语句

- For loop
- While loop
- Do while loop

条件语句

- If else

其他

- Break，continue
- Return
- ？：
- 不支持switch和goto

```
function XXX (<parameter types>)
        {internal(默认)|external}
        [constant]
        [payable]
        [returns (<return types>)]
```

参数

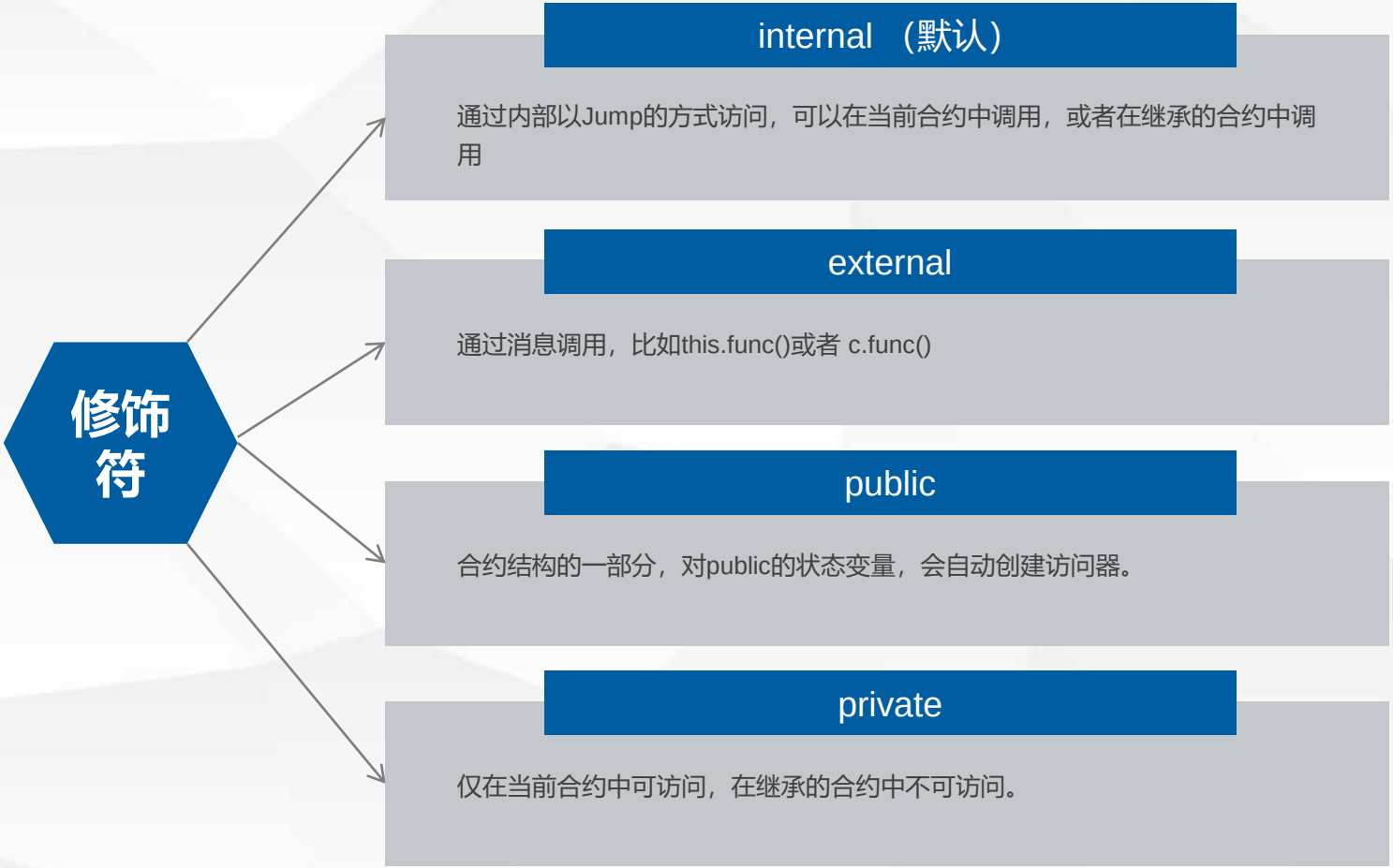
修饰符

返回值

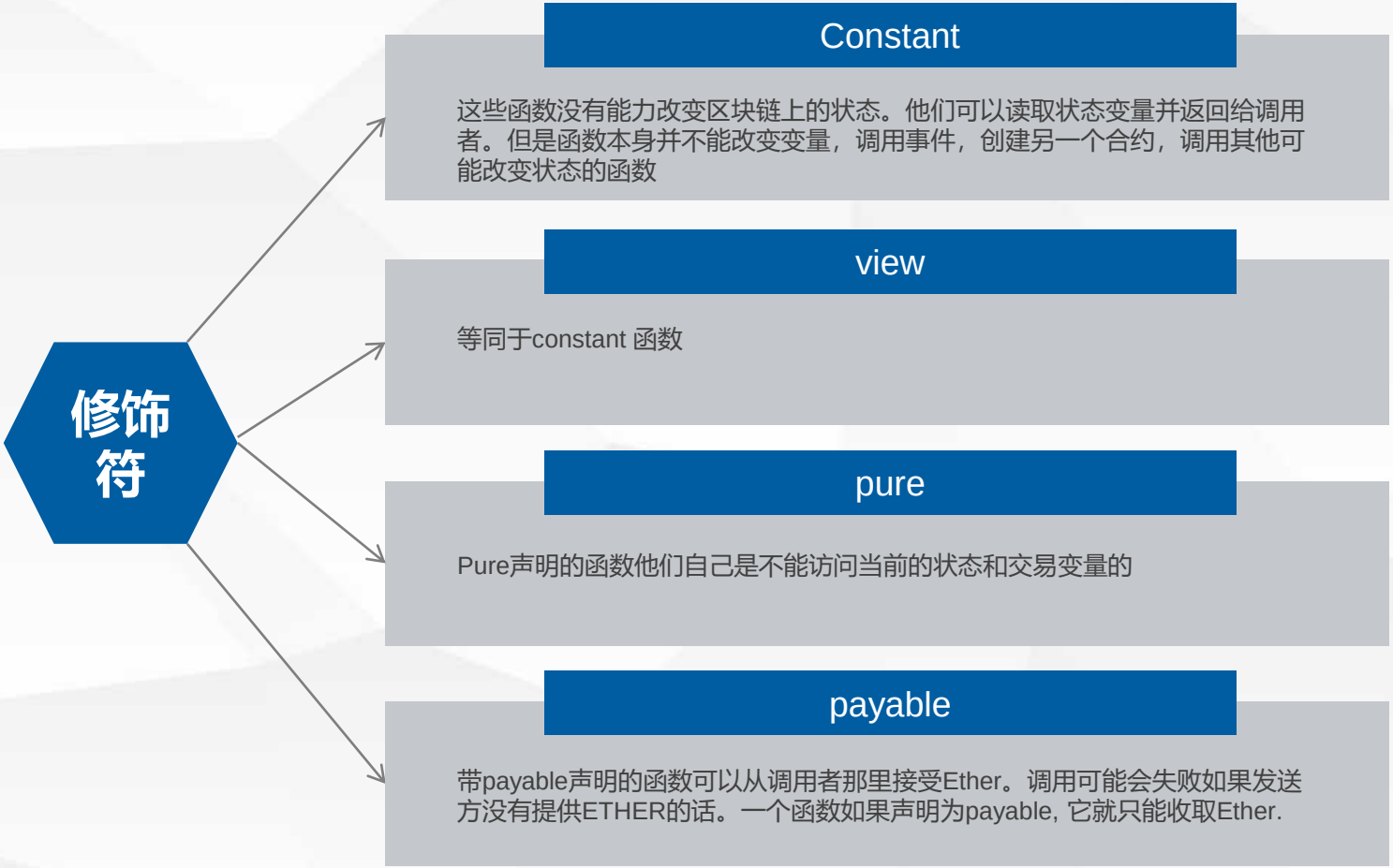
03

继承/修饰符

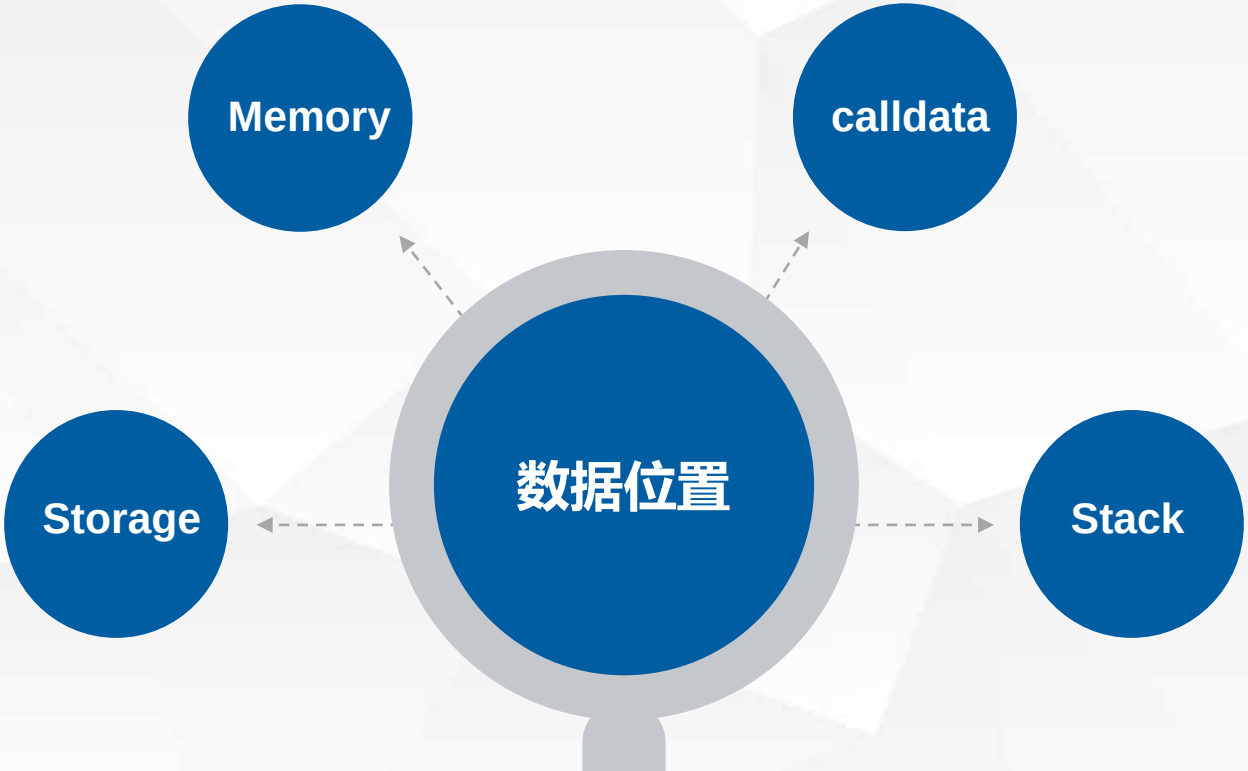
修饰符/Modifier



附加修饰符/Additional Qualifier



数据位置/Data Location



Solidity通过复制包括多态的代码来支持多重继承

```
contract owned {
    function owned() { owner = msg.sender; }
    address owner;
}

contract mortal is owned {
    function kill() {
        if (msg.sender == owner) selfdestruct(owner);
    }
}
```

Virtual

多重继承

抽象合约

04

其他/Misc

特殊变量/BlockChain Related

No	变量	说明
1	block.blockhash(uint blockNumber) returns (bytes32)	给定区块号的哈希值，只支持最近256个区块，且不包含当前区块。
2	block.coinbase (address)	当前块矿工的地址。
3	block.difficulty (uint)	当前块的难度。
4	block.gaslimit (uint)	当前块的gaslimit。
5	block.number (uint)	当前区块的块号。
6	block.timestamp (uint)	当前块的时间戳。
7	msg.data (bytes)	完整的调用数据（calldata）。
8	msg.gas (uint)	当前还剩的gas。
9	msg.sender (address)	当前调用发起人的地址。
10	msg.sig (bytes4)	调用数据的前四个字节（函数标识符）。
11	msg.value (uint)	这个消息所附带的货币量，单位为wei。
12	now (uint)	当前块的时间戳，等同于block.timestamp
13	tx.gasprice (uint)	交易的gas价格。
14	tx.origin (address)	交易的发送者（完整的调用链）

加密函数/Crypto

No	变量	说明
1	keccak256(...) returns (bytes32)	使用以太坊的（Keccak-256）计算HASH值。紧密打包。
2	sha3(...) returns (bytes32):	等同于keccak256()。紧密打包。
3	sha256(...) returns (bytes32):	使用SHA-256计算HASH值。紧密打包。
4	ripemd160(...) returns (bytes20)	使用RIPEMD-160计算HASH值。紧密打包。
5	ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address):	通过签名信息恢复非对称加密算法公匙地址。如果出错会返回0.
6	revert()：	事务回退

地址相关/Address Related

No	变量	说明
1	<address>.balance (uint256)	Address的余额，以wei为单位。
2	<address>.transfer(uint256 amount)	发送给定数量的ether，以wei为单位，到某个地址。失败时抛出异常。
3	<address>.send(uint256 amount) returns (bool)	发送给定数量的ether，以wei为单位，到某个地址。失败时返回false。
4	<address>.call(...) returns (bool)	发起底层的call调用。失败时返回false。
5	<address>.callcode(...) returns (bool)	发起底层的callcode调用，失败时返回false
6	<address>.delegatecall(...) returns (bool)	发起底层的delegatecall调用，失败时返回false。

No	变量	说明
1	this	当前合约对象，可以显式转换成Address类型
2	selfdestruct	合约自毁程序，将合约内的资金自动发送到指定的地址
3	suicide	等同于 selfdestruct

调用throw。

调用require，但参数值为false。

通过assert判断内部条件是否达成，require验证输入的有效性

当前的执行被终止且被撤销(值的改变和帐户余额的变化都会被回退)

异常还会通过Solidity的函数调用向上冒泡(bubbled up)传递

隐式类型转换

uint8->uint16 uint160->address

显示类型转换

```
int8 y = -3; uint x = uint(y);
```

类型推定

```
uint24 x = 0x123; var y = x;
```

```
for (var i = 0; i < 2000; i++) {}
```

→ 无限循环

```
Contract EventContract {
        event LogFunctionFlow(string);
        function ValidInt8(int _data) public returns (uint8){
                LogFunctionFlow("Enter function ValidInt8");
                if (_data<0 || _data>255){
                        revert();
                }
                LogFunctionFlow("Value is within expected range");
                LogFunctionFlow("Return value from function");

                return  uint8(_data);
        }
}
```

既见君子，云胡不喜

电话 **13240946967**
邮箱 zy731@hotmail.com
微信 **gavinzheng731**
博客**https://my.oschina.net/gavinzheng731/**

谢谢聆听

郑嘉文

2020/09