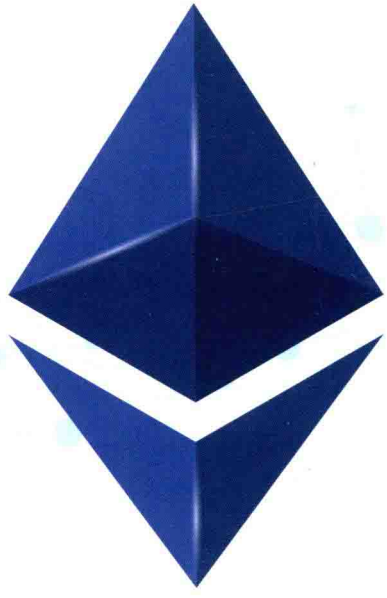
Broadview®
www.broadview.com.cn

“以太坊智能合约 + DApp”从理论到实战

精通以太坊智能合约开发

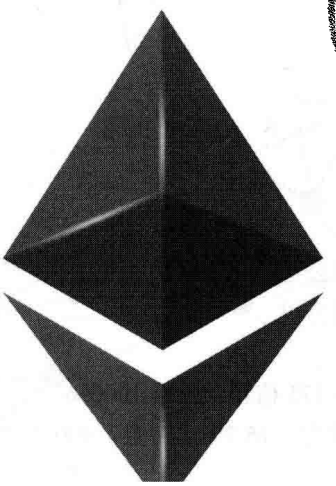
熊丽兵◎编著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

精通以太坊智能合约开发

熊丽兵◎编著



電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书系统介绍了以太坊智能合约的开发，对智能合约相关知识进行全面梳理，尤其是对智能合约开发语言 Solidity 进行了详细解读。智能合约的开发者可以从书中获得一些启发和指导。

本书可以作为一本案头手册，方便开发者在开发智能合约时随时查阅。

图书在版编目（CIP）数据

精通以太坊智能合约开发 / 熊丽兵编著. —北京：电子工业出版社，2018.9
ISBN 978-7-121-34951-5

Ⅰ. ①精… Ⅱ. ①熊… Ⅲ. ①分布式数据库－数据库系统 Ⅳ. ①TP311.133.1

中国版本图书馆 CIP 数据核字(2018)第 199195 号

策划编辑：官　杨
责任编辑：牛　勇
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：15.75　字数：300 千字
版　　次：2018 年 9 月第 1 版
印　　次：2018 年 9 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前言

本书特色

从 2017 年开始，我在博客《深入浅出区块链》中发表了 30 多篇关于区块链的入门文章，广受大家的好评，也因此收到了电子工业出版社编辑的邀请，希望我写一本关于区块链开发的书籍。我对市面上的书籍做了调查，发现介绍比特币和以太坊入门知识的书比较多，但如果想系统全面地学习智能合约开发，却并没有更好的资源。于是，在跟出版社编辑商议后，我决定写一本全面系统介绍智能合约开发的书，本书由此诞生。

本书并没有对比特币或区块链的基础概念进行过多的介绍，因为市面上已经有很多这方面的文章了，大家也可以参考我的博文《区块链技术学习指引》（https://learnblockchain.cn/2018/01/11/guide/）。本书系统介绍以太坊智能合约的开发，并尽量覆盖智能合约的方方面面，尤其是对智能合约开发语言 Solidity 进行了详细解读。因此本书可以作为一本案头手册，方便开发者在开发智能合约时随时查阅。

本书涉及的 Solidity 内容是以官方文档 0.4.24 版本（https://solidity.readthedocs.io/en/v0.4.24/）为标准的，同时加入了很多我自己的理解以及大量的实例。

另外，本书有时将智能合约简称为“合约”。

读者对象

本书适合那些对区块链有过基本了解，并想进一步学习智能合约或者去中心化应用的开发人员阅读。

本书的读者最好应了解一门语言。例如了解 C、JavaScript、Python 语言会对学习 Solidity 有帮助，因为 Solidity 中的很多思想都参考了这些语言。

本书主要适合以下人员阅读：

- 区块链应用开发者；
- 区块链技术的从业者；
- 对区块链技术感兴趣的人员。

本书内容

第 1 章初探以太坊智能合约，初步认识以太坊、智能合约、Solidity，以及如何使用 Solidity 编写一个简单的智能合约。

第 2 章介绍以太坊核心概念，例如交易、区块、费用 gas、以太坊虚拟机、以太坊账户等概念。

第 3 章介绍一个使用 Solidity 编写的智能合约的组成部分。

第 4 章介绍 Solidity 的类型系统，详细介绍 Solidity 的各种类型，例如整型、布尔类型、地址类型、函数类型、数组类型及结构体类型等。

第 5 章介绍 Solidity 中的单位，包括货币单位和时间单位，通过代码讲解单位如何换算。

第 6 章介绍 Solidity 的全局变量及函数，它们其实就是 Solidity 语言提供的 API，例如获取区块和交易的属性、有关数学及加密功能的函数等。

第 7 章介绍 Solidity 中的表达式、控制结构、变量作用范围、错误处理等。

第 8 章介绍合约，包括如何创建合约、合约函数的可见性、合约函数修饰符等。

第 9 章介绍合约编译、部署、交互、调试，包括对编译器的选择、不同工具的合约部署方法。

第 10 章介绍合约 ABI 的作用，以及如何生成 ABI。

第 11 章介绍编写合约的最佳实践，一是从编码规范来考量；二是从安全性来考量。

第 12 章介绍一些合约案例，如最常见的 ERC20 标准代币合约，介绍如何实现代币增发、资产冻结，以及如何实现一个众筹（ICO）合约，并详细介绍 ERC721 合约的实现方法。

第 13 章介绍去中心化应用的开发，重点介绍如何使用 web3 以合约进行交互，以及 Truffle 框架的使用。

勘误和支持

由于区块链是一种新兴的技术，以太坊平台也处在不断更新发展的过程中，加上作者水平有限，书中难免出现疏漏或错误。如果大家发现问题，请及时反馈给我（可添加我的微信：xlbxiong），我将在图书再版时进行修正，以提供最准确的内容。

以太坊智能合约及 Solidity 最初的内容都是以英文发表的，有个别术语还没有准确的中文翻译，因此我会在括号里注明英文原文。

为了更好地理解，本书部分内容我录制了视频课程，大家可以关注登链学院微信公众号观看学习。

本书的所有代码都被上传到了我的 GitHub（https://github.com/xilibi2003）上，也欢迎大家 Pull Request。

致谢

本书得以面世，离不开很多人的帮助，尤其是各位前辈的指导。

感谢比特币的开创者中本聪，是他带我们进入了数字货币与区块链的世界。

感谢以太坊创始人 V 神（Vitalik Buterin），是他打造了这个开放的智能合约平台，没有这个平台，就不可能有此书。

感谢电子工业出版社的编辑，他们对书稿做了专业、细致、认真的编校工作。

感谢登链科技及登链学院的同事，在我写书的时候他们帮我分担了很多工作。

感谢那些在我学习区块链技术时阅读的博客文章及书籍的作者，虽然我与他们未曾谋面，但我从他们输出的技术内容里获益颇多。

感谢小专栏平台及创始人寂小桦，在我写作博客的时候，在小专栏平台上得到了很多付费用户的认可，这也是我不断写作的动力。

最后要特别感谢我的家人，尤其是我的妻子，在写作本书的这段时间里，我的大女儿不到四岁，小女儿不到一岁，感谢她一直以来对我的支持以及对家庭的付出。

要感谢的人还有很多，难以一一列举，只希望这本书能够为区块链技术在中国的推广和发展做出尽可能多的贡献。

关于作者

熊丽兵，网络 ID：Tiny 熊。

北京航空航天大学硕士，先后加入创新工场及猎豹移动，全面负责数款千万级用户产品的开发及管理工作，2014 年作为技术合伙人参与创建酷吧时代科技。

2016 年投身于区块链技术领域，创立登链科技。CSDN 博客专家，拥有全网访问量最大的区块链技术博客《深入浅出区块链》（learnblockchain.cn），对底层公链技术、区块链技术落地都有较为深入的研究。

我的个人微信号：xlbxiong。欢迎大家向我反馈问题，或者和我一起讨论问题。二维码如下：

我的公众号为：blockchaincore。可以回复 eth_book，获取本书的代码等相关资源。二维码如下：

读者交流 QQ 群：245251041。

目录

第 1 章　初探以太坊智能合约 1

以太坊诞生 1
智能合约 2
Solidity 语言 2
一个货币合约的例子 6
本章小结 9

第 2 章　以太坊核心概念 10

区块链基础概念 10
共识协议：工作量证明（PoW） 12
以太坊虚拟机（EVM） 13
账户 13
以太坊钱包 16
交易 17
消息调用 17
费用（gas） 17
以太坊网络 18
存储、内存和栈 19
指令集 20
委托调用和库 20
日志 20
自毁 21
以太坊路线图 21
本章小结 22

第 3 章 Solidity 合约内容 23
Solidity 文件结构 23
合约结构 27
本章小结 29
第 4 章 Solidity 数据类型 30
类型概述及分类 31
布尔类型（Boolean） 32
整型（Integer） 32
定长浮点型（Fixed Point Number） 34
定长字节数组（Fixed-size Byte Array） 35
有理数和整型常量（Rational and Integer Literal） 36
字符串常量（String Literal） 37
十六进制常量（Hexadecimal Literal） 37
枚举（Enum） 38
函数类型（Function Type） 38
地址类型（Address） 44
地址常量（Address Literal） 49
数据位置（Data Location） 50
数组（Array） 52
数组成员 55
字符串 string 及字节数组 bytes 58
结构体（Struct） 60
映射（Mapping） 64
类型转换 65
var 类型推导 67
运算符 67
本章小结 71
第 5 章 Solidity 中的单位 72
货币单位（Ether Unit） 72
时间单位（Time Unit） 73
本章小结 74

第 6 章　Solidity 全局变量及函数 75
区块和交易的属性 75
地址相关属性和函数 79
合约相关属性和函数 81
本章小结 82
第 7 章　Solidity 表达式及控制结构 83
函数参数 83
控制结构 84
函数调用表达式 86
赋值表达式 89
变量声明与作用范围 90
错误处理 92
本章小结 96
第 8 章　合约 97
合约概述 98
创建合约 98
可见性 101
访问函数（Getter Function） 103
函数修改器（Function Modifier） 105
状态常量 108
视图函数（View Function） 109
纯函数（Pure Function） 110
回退函数（Fallback Function） 110
函数重载（Function Overloading） 112
事件 113
继承 116
构造函数（Constructor） 120
抽象合约（Abstract Contract） 123
接口（Interface） 124
库 124
Using for 指令 128

本章小结 131

第 9 章　合约编译、部署、交互、调试 132

Solidity 编译器 132
合约编译 134
合约部署及调用 136
使用 geth 138
使用 Remix + MetaMask 140
合约调试 144
本章小结 147

第 10 章　应用程序二进制接口（ABI） 148

简单理解 ABI 148
ABI 手册 149
本章小结 161

第 11 章　智能合约最佳实践 162

编码规范 162
代码格式 163
函数编写规范 170
安全性考虑 173
一些安全陷阱 174
编写合约的安全建议 176
本章小结 185

第 12 章　合约案例 187

代币 187
高级功能代币 196
众筹（ICO）合约 201
众筹智能合约代码 201
非同质化代币 ERC721 206
本章小结 211

第 13 章　去中心化应用开发 213
JSON RPC 214
Web3.js 215
在 geth 中使用 Web3.js 216
在应用中使用 Web3.js 216
去中心化应用案例 218
搭建测试环境 219
创建智能合约 220
合约加入事件 227
使用 Web3 监听事件、刷新 UI 227
Truffle 框架 228
安装 Truffle 228
Truffle 使用案例 229
在浏览器中运行 237
本章小结 239

第 1 章

初探以太坊智能合约

在本章中，主要带领大家初步认识以太坊、智能合约、Solidity 语言，并介绍如何使用 Solidity 语言编写一个简单的智能合约。

以太坊诞生

自 2008 年比特币出现以来，数字货币逐渐被大家所接受，人们发现其背后的区块链技术在数字货币之外同样有着广阔的应用空间，然而苦于比特币在设计之初仅考虑了数字货币的场景（虽然比特币也支持编程，但它是非图灵完备的，功能有限），因此对于很多商业应用比特币平台无法支持。

2013 年年末，Vitalik Buterin（一位俄罗斯天才少年，人称“V 神”）针对比特币系统非图灵完备性、效率低等缺点，首次提出了以太坊概念，并发布了《以太坊：下一代智能合约和去中心化应用平台》白皮书，启动了项目。

2014 年 7 月，通过 ICO 众筹到了 31529 个比特币，用来支持以太坊项目的发展。

2016 年年初，以太坊的技术得到市场认可，价格开始暴涨，也吸引了大量开发者来到以太坊的世界。

2018 年 5 月，以太币成为市值第二高的加密货币，仅次于比特币，以太坊成为目前热门的区块链应用平台。

智能合约

以太坊上的程序被称为“**智能合约**”，其包含代码和相应的状态数据。以太坊是区块链与智能合约的完美结合，通过编写智能合约可以实现强大的功能，实现去中心化的应用开发。

> 智能合约的英文是 Smart Contract，和人工智能（AI，Artificial Intelligence）的“智能”没有关系。尼克·萨博在 1995 年就提出了智能合约的概念——就是将法律条文写成可执行代码。Vitalik Buterin 把它引入到以太坊中，表示以太坊程序能自动执行及无法被干预的特点。
> 现在智能合约已经扩展到所有的区块链平台，很多时候人们把超级账本、EOS 等区块链平台的程序也称为“智能合约”。

Solidity 语言

以太坊官方推荐的智能合约开发语言为 Solidity，它是一门静态的、支持继承、类库以及复杂的自定义类型等特性的高级语言。Solidity 在设计上借鉴了 Python、JavaScript 等语言，其语法也和 JavaScript 相似。

一个简单的智能合约

我们先看看怎么使用 Solidity 语言来编写一个简单的智能合约。

```
pragma solidity ^0.4.0;

contract SimpleStorage {

    uint storedData;

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

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

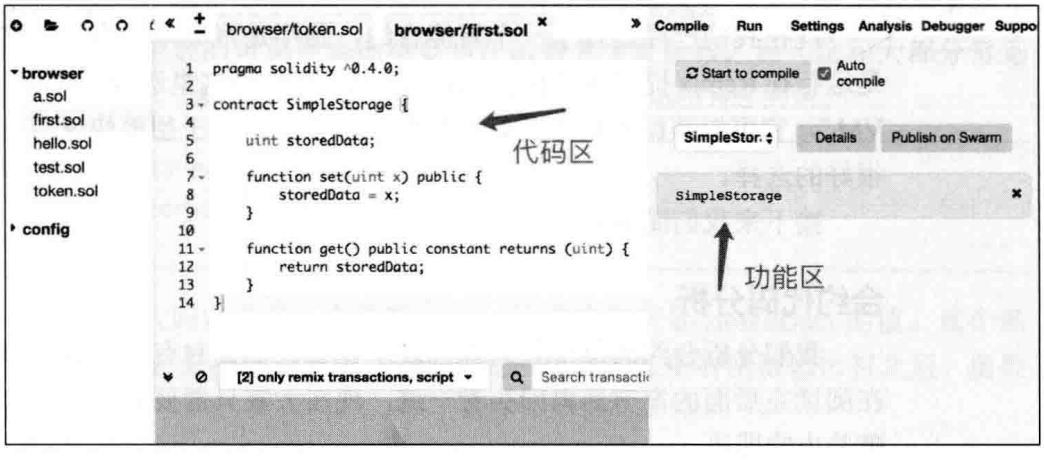
这个智能合约的作用是在区块链上存储一个变量，任何人都可以通过调用set()函数设置改变值（覆盖之前的数字），调用get()函数获取值，这个数字将会被永久留存在区块链的历史上。当然，这是一个没有多大实际意义的智能合约，这里仅仅是作为一个例子让大家看看智能合约是什么样子的。

> 这个合约就像我们在学习其他语言时，第一个应用是 Hello World 程序一样，之所以不用 Solidity 写一个 Hello World 应用，是因为智能合约运行在分布式网络中，无法像本地应用一样运行输出日志。这也是智能合约不同于传统应用的一个显著区别。
>
> 另外，用其他语言编写程序时，通常会有一个程序入口方法（如 main 方法），而智能合约没有入口方法，每一个函数都可以被单独调用，并且每一个函数也都只能在合约内部实现，没有实现全局函数。
>
> 在区块链上存储数据，是智能合约最常见、最基本的功能，这也是我们使用这个例子的原因。在本书后面的章节中，也会多次使用这个例子。

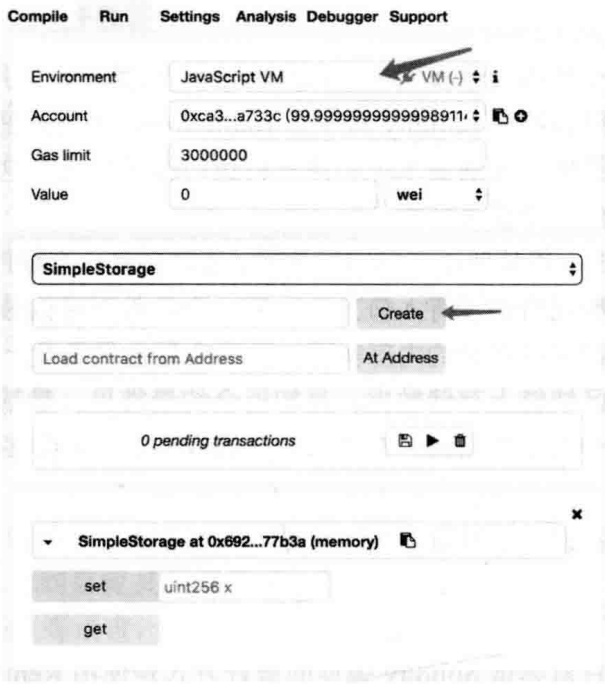
我们再来看看怎么运行这个智能合约。

运行

目前尝试 Solidity 编程的最好方式是使用 Remix。Remix 是一个基于浏览器的 Solidity IDE，它集成了 Solidity 编译器、运行环境，以及调试和发布工具。使用浏览器打开网址 https://remix.ethereum.org/，在所打开的 IDE 界面中的代码区输入上面的代码，如下图所示。



然后在功能区切换到 Run 标签页，在 Environment（环境）一栏选择“JavaScript VM”，点击 Create 按钮（注：新版 Remix 改为 Deploy 按钮了），如下图所示。



至此，第一个智能合约已经创建完成。合约创建完成之后，在功能区的下方会出现智能合约可以调用的所有函数，在这个例子中显示的是 set()和 get()函数，大家可以动手调用一下试试看。

现在我们已经运行了一个智能合约，是不是很简单？不过，这个智能合约是运行在 Remix 提供的模拟区块链环境下的。因为这里选择的环境是“JavaScript VM”，它可以帮助我们准备好账号和以太币，这对于想要快速启动一个合约是很好的选择。

接下来我们简单分析这个智能合约的代码。

合约代码分析

我们分析合约的每行代码都代表了什么，如果对有些内容不理解，则可以在阅读完后面的章节后再回头看一遍，现在大家只需要知道一个合约大概包含哪些内容即可。

```
pragma solidity ^0.4.0;
```

关键字 pragma 用来告诉编译器如何编译这段代码，^表示这里需要 Solidity 0.4.0 及以上版本（在合约版本声明章节中会进一步说明，版本号的第 3 部分可以变，留出来用于修复 bug，如 Solidity 0.4.1 的编译器有 bug，可在 Solidity 0.4.2 中修复，现有合约不用改代码），但是不能高于 Solidity 0.5.0（以避免兼容性问题）。

```
contract SimpleStorage
```

这行代码定义了一个合约，合约的名字为 SimpleStorage（这和在其他语言如 JavaScript 及 Python 中定义一个类很相似，只不过 class 关键字变成了 contract）。一个合约通常由一组代码（合约的函数）和数据（合约的状态）组成。一个合约被存储在以太坊区块链上，对应一个特殊地址。

```
uint storedData
```

这行代码声明了一个变量，在智能合约中称为“状态变量”。这个状态变量名为 storedData，类型为 uint（一个 256 位的无符号整数），可以把它理解为数据库里面的一个存储单元。

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

上面的代码定义了一个函数 set()。在 Solidity 中通过 function 关键字来定义函数，并且函数的可见性修饰符要写在函数名字及参数之后（这和大部分常见语言不一样），这里 set()函数的作用是修改变量 storedData 的值。

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

上面的代码定义了一个函数 get()，用来读取变量 storedData 的值。这个函数通过 returns 来指定返回值。returns 返回值要放在所有函数修饰符之后、函数体之前。