Mastering Blockchain Programming with Solidity

Write production-ready smart contracts for Ethereum blockchain with Solidity

Jitendra Chittoda

Packt
www.packt.com

Mastering Blockchain Programming with Solidity

Write production-ready smart contracts for Ethereum blockchain with Solidity

Jitendra Chittoda

Packt

BIRMINGHAM - MUMBAI

Mastering Blockchain Programming with Solidity

Commissioning Editor: Richa Tripathi
Acquisition Editor: Shriram Shekhar
Content Development Editor: Manjusha Mantri
Senior Editor: Afshaan Khan
Technical Editor: Pradeep Sahu
Copy Editor: Safis Editing
Project Coordinator: Prajakta Naik
Proofreader: Safis Editing
Indexer: Rekha Nair
Production Designer: Shraddha Falebhai

First published: August 2019

Production reference: 1010819

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-83921-826-2

www.packtpub.com

I dedicate this book to my wife Neha, and my two daughters Arshiya and Advika with much love.

Packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Foreword

I have known Jitendra Chittoda for a good while. Together, we have audited a significant number of smart contracts for different blockchain platforms. Some of these contracts have been widely used, while others handle large amounts of funds.

Prior to working as a smart contract auditor, Jitendra was working as a developer, which gave him a broader perspective of the field, as he has observed it from a variety of standpoints. These multiple viewpoints of blockchain, and Ethereum in particular, have allowed him more general insights.

While many have heard about the power and potential of blockchain and smart contracts, I believe the blockchain journey is still in its infancy. As with many new technologies, a number of mistakes have been made. Some mistakes led to exploits, and some exploits were so large that they made news headlines. One of those attacks was the DAO reentrancy attack.

Given the novelty of the technology, people require guidance. Such guidance is required on different levels. Firstly, many people still do not sufficiently understand what blockchain is really about, what makes it special, and what differentiates it from regular databases. They don't understand the power of smart contracts—which are sometimes called programmable money—and their potential impact on future businesses.

Secondly, while other people understand the power of blockchain, they do not see a clear path to building on top of it. They are not aware of the many small mistakes that can be made along the way, which range from overly expensive smart contracts to a complete loss of control. Hence, these people require detailed and explicit technical guidance.

Given Jitendra's background, he can provide a good overview and many technical recommendations that will not only speed up learning, but can also avoid costly and potentially project-threatening errors. This book starts off with introductions to blockchain before diving into Solidity, its details, its common applications, its most useful helpers, and its security practices.

Hubert Ritzdorf

Chief Technical Officer, ChainSecurity AG

Contributors

About the author

Jitendra Chittoda is a blockchain security engineer at ChainSecurity. His day job is to perform security audit on smart contracts and expose security vulnerabilities in Solidity and Scilla contracts. He has also developed a non-custodial, decentralized, P2P lending contracts for ETHLend. The Solidity contracts that he has developed or audited handle over $100 million worth of cryptoassets. He also served as a tech and security advisor in various ICO projects.

Before finding his passion for blockchain, he coded in Java for over 11 years. He is the founder and leader of Delhi-NCR-JUG, a non-profit meetup group for Java. He holds a master's degree in computer applications and is regularly invited as a speaker at various conferences and meetups.

I would like to thank my wife, Neha, for encouraging and supporting me every single day with writing this book. A big shout-out to Manjusha Mantri, Shriram Shekhar, Pradeep Sahu, all of the technical reviewers, and the Packt team for their peer review, suggestions, and full support. I would also like to thank all my mentors and professors who always helped me in becoming a programmer. Thank you, Mom and Dad. You have given me the greatest gift: an education.

About the reviewers

Natalie Chin is a blockchain developer, college professor, and technical blogger. She is currently teaching at George Brown College in Toronto, helping to lead the first blockchain college program in the world. At STK, she built a level-two scaling solution on Ethereum, allowing instant cryptocurrency purchases. Natalie won three prizes at ETHSanFrancisco, 2018, where her team, Lending Party, built a decentralized application allowing a CDP (cryptocurrency loan) to liquidate into a bank account instantly.

She is an avid hackathon organizer, associated with DeltaHacks and Stackathon, where she inspired new developers to join the blockchain space. Natalie is a Women in Tech ambassador and is always passionate about spreading knowledge about the blockchain field.

> *A sincere thank you to Dad, for introducing me to technology at a young age, supporting me, and perpetually pushing me outside of my comfort zone to grow and be better.*
>
> *To Zichen Jiang, for always finding the silver lining of situations, and always inspiring me to embrace everything with wit, strength, and humor.*
>
> *Finally, I am eternally grateful to my mother-in-law, for embodying what it means to live a life of curiosity and positivity.*

Ben Weinberg is a self-taught computer programmer who started off in iOS development with Objective-C and Swift, and, eventually, got caught up in the world of blockchain while working in Jerusalem. He eventually started learning blockchain and web development simultaneously with the goal of entering the blockchain space as a developer. Ben worked at the Toronto-based Bitcoin Bay, writing Solidity smart contracts and leading workshops on Ethereum development and Bitcoin opcodes. He also joined George Brown College, where he assisted aspiring blockchain developers as a lab monitor for George Brown's new blockchain development course. At George Brown, he helped teach full-stack web development, smart contract development, and decentralized application development.

> *I would like to thank all the mentors who have helped me along the way in my endeavors to become a programmer. I would like to acknowledge the Toronto and Jerusalem blockchain communities for being so accessible and welcoming to me as I entered this world without knowing anyone and with little knowledge of how blockchain works. I would also like to send my most supreme thanks to my family for all they have done.*

Marc Lijour helps organizations strengthen their competitive advantage with technology such as blockchain and practices such as DevOps. Marc brings consulting experience from the public sector, Cisco, Savoir-faire Linux, and ConsenSys to executives driving innovation. He started the Metamesh Group with 30 consultants spanning across 5 continents. He helped George Brown launch the first blockchain development college program in Canada.

Marc holds degrees in mathematics and computer science, as well as an MBA in technology and innovation. He serves on the board of several not-for-profit organizations, including the **Information and Communications Technology Council** (**ICTC**), ColliderX, TechConnex, and the Toronto French Business Network.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface

Blockchain technology is at its nascent stage. However, technology is slowly moving forward and new developments using blockchain are emerging. This technology has the power to replace trusted third parties with trusted blockchain networks. Bitcoin was the birth of blockchain technology and has shown the world a new peer-to-peer payment system without needing intermediaries. You could say that Bitcoin was the first generation of blockchain technology. However, Ethereum took this innovative technology to the next level—you could call it blockchain generation two—where you can write decentralized applications using smart contracts. Solidity is the most widely used and popular language for writing smart contracts for decentralized applications.

The book starts with explaining blockchain, Ethereum, and Solidity. It mostly focuses on writing production-ready smart contracts in the Solidity language for Ethereum blockchain. It covers basic Solidity language syntax and control structures, and moves on to writing your own contract. It also deep dives into different libraries that you can use while writing contracts. Later on, it covers tools and techniques to write secure, production-ready smart contracts.

Who this book is for

This book is for professional software developers who have started learning blockchain, Ethereum, and the Solidity language and who want to make a career in writing production-ready smart contracts. This book is also aimed at developers who are interested in building decentralized applications over Ethereum blockchain. This book will help you learn the Solidity language for building smart contracts from scratch and make you proficient in building production-ready smart contracts for decentralized applications.

What this book covers

`Chapter 1`, *Introduction to Blockchain*, starts with how blockchain technology came into existence through the innovation of Bitcoin. This chapter discusses the different properties of a blockchain. It also introduces Ethereum and how it is different from the Bitcoin blockchain. Later, this chapter introduces smart contracts.

`Chapter 2`, *Getting Started with Solidity*, starts with the basic Solidity language syntaxes and the structure of a contract. You will learn about the different data types available in Solidity. This chapter also discusses globally available variables present in the Solidity language that you can use while writing your smart contract.

`Chapter 3`, *Control Structures and Contracts*, deep dives into the control structures of Solidity contracts, the different types of functions supported, contract inheritance, and event logging.

`Chapter 4`, *Learning MetaMask and Remix*, discusses setting up your MetaMask plugin and creating wallets using it. This chapter also discusses using the online Remix IDE to create, compile, deploy, and interact with your Solidity contracts.

`Chapter 5`, *Using Ganache and the Truffle Framework*, discusses installing and setting up your local blockchain instance using Ganache. This chapter also discusses installing and setting up the Truffle framework, learning how to use its commands, setting up your new Truffle project, writing migration scripts for the Truffle framework, and adding test cases to a project.

`Chapter 6`, *Taking Advantage of Code Quality Tools*, discusses improving the quality of your contracts by using open source tools such as `surya`, which helps in generating different kinds of reports. This chapter also discusses using Solidity linters to lint your code and fix common issues present in your contracts, as well as running Solidity coverage tools to generate code coverage reports.

`Chapter 7`, *ERC20 Token Standard*, covers the introduction of the ERC20 token standard. This chapter deep dives into its full implementation details, provides an in-depth study of each function of the standard, and covers different events, optional functions, and a number of advanced functions.

`Chapter 8`, *ERC721 Non-Fungible Token Standard*, starts with an introduction to the ERC721 standard and the difference between the ERC20 and ERC721 standards. This chapter deep dives into each of the state variables, functions, and events associated with ERC721 implementation. This chapter also discusses some other optional contracts that can be used with the ERC721 standard.

`Chapter 9`, *Deep Dive into the OpenZeppelin Library*, starts with an introduction to OpenZeppelin library contracts. This chapter will help you learn how to install and use OpenZeppelin library contracts in your Truffle project. It also provides in-depth studies of library contracts such as `Ownable`, `Claimable`, `Roles`, `PauserRole`, `Pausable`, `ERC20`, `SafeERC20`, `DetailedERC20`, `ERC20Mintable`, `ERC20Burnable`, `ERC20Pausable`, `Math`, `SafeMath`, `Crowdsale`, `Address`, and `ReentrancyGuard`.

`Chapter 10`, *Using Multisig Wallets*, The multisig wallets are special contracts that require multiple signatures to execute a transaction. This chapter provides an introduction to multisig contracts and their usage. This chapter also covers installing, creating, and setting up your own multisig wallet, as well as how to use and control it.

`Chapter 11`, *Upgradable Contracts Using ZeppelinOS*, provides an introduction to the ZeppelinOS development platform. Topics covered include creating a new project using `zos` commands, creating an upgradable contract, deploying, re-deploying to upgrade the contract, and some precautions to take when using ZeppelinOS for writing upgradable contracts.

`Chapter 12`, *Building Your Own Token*, helps you learn how to create your own ERC20 token contract from scratch. You will learn how to draft the specification of a contract, set up a Truffle project, create a contract, choose which OpenZeppelin libraries to use, write migration scripts, write test cases, execute migration scripts and test cases in Ganache, and finally, test your contracts on a testnet such as Rinkeby.

`Chapter 13`, *Solidity Design Patterns*, introduces different Solidity design patterns. These are divided into five categories: *Security patterns*: Withdrawal, Access restriction, and Emergency stop; *Creational patterns*: Factory pattern; *Behavioral patterns*: State machine, Iterable map, Indexed map, Address list, and Subscription; *Gas economic patters*: String comparison and Tight variable packing; and *Life cycle patterns*: Mortal and Auto deprecate pattern.

`Chapter 14`, *Tips, Tricks, and Security Best Practices*, helps you to learn different Solidity smart contract best practices, such as avoiding floating pragma, the 15-second blocktime rule, rounding errors, and gas consumption. It also helps you to learn about different security attacks, such as front-running, reentrancy, signature replay, integer overflow and underflow, and how to prevent these attack patterns.

To get the most out of this book

You should have knowledge of any of the existing programming languages. Solidity language syntaxes are very similar to Java, JavaScript, and Python syntaxes. A developer who has created projects using Java, JavaScript, or Python can pick up and learn Solidity very easily. You should also have some basic knowledge of OOPS concepts, blockchain, wallets, private key, public key, consensus algorithms, and exchanges.

One very important thing to note here is that, in order to write smart contracts for decentralized applications, you need to have a completely new mindset. Up until now, you have probably been building applications using Java, JS, or another language in which you can fix bugs later on, even in the production environment when something goes wrong. However, smart contracts are immutable. If your contract is deployed in production, you are done. If there is a bug left in the contract, you will be unable to fix it. Hence, you need to ensure that all your smart contracts are well-tested and have no bugs present in them.

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packt.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest versions of the following:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Mastering-Blockchain-Programming-with-Solidity`. If there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it at `https://static.packt-cdn.com/downloads/9781839218262_ColorImages.pdf`.

Code in action

To see the code being executed please visit the following link: `http://bit.ly/2Yv6kpm`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Solidity supports different data types, including `uint`, `int`, `address`, and many more."

A block of code is set as follows:

```
contract VariableStorage {
   uint storeUint; //uint256 storage variable
    //...
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
contract VariableStorage {
    uint storeUint; //uint256 storage variable
    //...
}
```

Any command-line input or output is written as follows:

```
$ npm install -g ganache-cli
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "As mentioned earlier, you can start a local Ganache blockchain instance by just clicking on the **QUICKSTART** button."

Warnings or important notes appear like this.