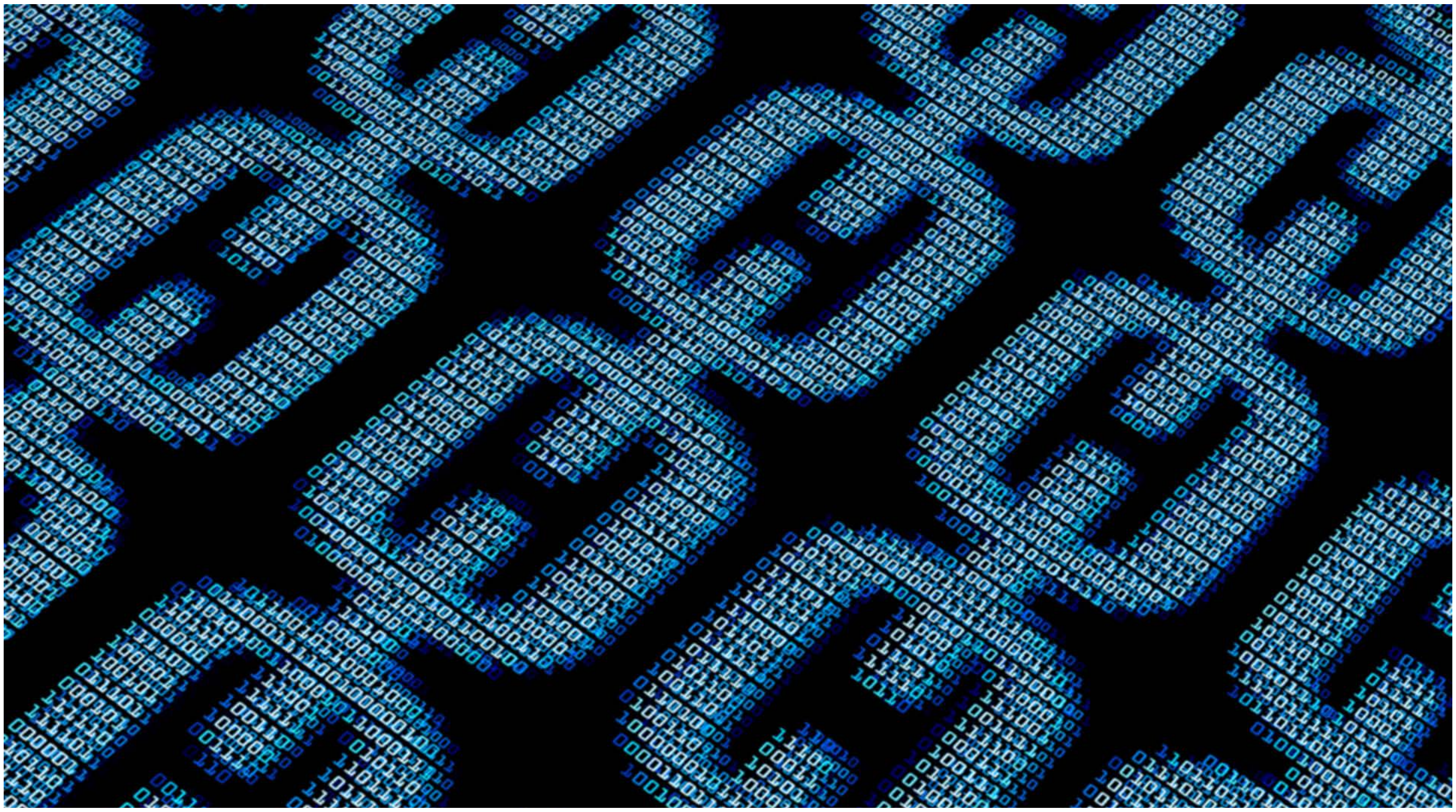
Haseeb Qureshi
Engineering @earndotcom. @Airbnb alum. Instructor @Outco. Writer. Effective Altruist. Blockchainist. Former poker pro.
Jan 27 · 21 min read

The authoritative guide to blockchain development



Cryptocurrencies, ICOs, magic internet money — it's all so damn exciting, and you, the eager developer, want to get in on the madness. Where do you start?

I'm glad you're excited about this space. I am too. But you'll probably find it's unclear where to begin. Blockchain is moving at breakneck speed, but there's no clear onramp to learning this stuff.

Since I left Airbnb to work full-time on blockchain, many people have reached out to me asking how to get into the blockchain space full-time. Consider this my authoritative (and inevitably incomplete) guide on how to get into blockchain engineering.

This guide will proceed in ten parts:

Why should you learn blockchain development?

Before I answer that question, let me first note: blockchain is a massively overvalued space right now. These prices are unsustainable, and a crash is definitely coming. This has all happened before, and will probably happen again. But if you work long-term in this space, you'll learn to shrug off prices. In the words of Emin Gun Sirer — prices are the least interesting part of cryptocurrencies. These are massively important technologies, and they are going to irrevocably change the world.

If you're unsure, I can't tell you whether or not you should jump in. But I can tell you five reasons that convinced me to take the leap:

1. **It's still early.**

Bitcoin was invented 10 years ago, but the rate of innovation has only reached a fever pitch in the last couple of years, especially with the launch of Ethereum in 2015. Most of the new companies and ideas in this space have been built on top of Ethereum, which is still very immature.

Even if you start now, you can realistically become a world-class expert within a few years. Most people just haven't been doing this that long, and it won't be that hard to catch up. Starting now would be analogous to deep learning experts who began studying the topic in the late 2000s.

2. This space doesn't have a strong talent funnel yet.

Most of the best and brightest students at universities are focusing on machine learning, web programming, or game development. Sure, blockchains are getting more sexy in the public discourse, but they're still a weird and subversive topic on which to stake your career.

Early on, blockchain was exclusively the realm of cypherpunks, paranoids, and weirdos. That's only recently begun to change. Just by being a curious and open-minded developer, you'll bring a lot of value to the space.

3. Much of the innovation is happening outside of academia.

Satoshi Nakamoto was not an academic as far as we know. There's no university or institution that offers a coherent blockchain concentration yet. Most of the innovation here has been led by aficionados, entrepreneurs, and independent researchers. Almost everything you need to know is in white papers, blog posts, public Slack channels, and open-source software. All it takes is rolling up your sleeves and jumping into the fray.

4. The demand for talent far, far exceeds supply.

There just aren't enough developers in this space, and they can't get trained fast enough. Everyone is competing to hire blockchain talent, and projects are feeling the talent crunch. Many of the best companies can't pay their people enough to stay because they have too many opportunities. If you get some skills under your belt, it'll be easy to land a job.

5. Cryptocurrencies are just really damn cool.

Where else can you build sci-fi stuff like cryptographically secured, decentralized money? It's the wild west right now—and this brings good and bad. The space could use more transparency, and regulation will eventually come. But without a doubt, cryptocurrencies are one of the most innovative areas you can be working in right now.

Naval Ravikant said in a recent interview: the key to success is to give society things that it wants, but doesn't know how to get on its own. You can't go to school for such things; if you could, the world would already have a steady supply of it.

So build something no one else knows how to build. Right now, blockchains are brand new and there's so much left to figure out. If you succeed in building the future of decentralized technology, the world will reward you handsomely.

So say you want to throw in your hat. What do you need to know before you get into the ring?

Prerequisites

I'd recommend strengthening up your understanding of fundamentals before you dive further. Blockchains are built atop decades of research in computer science, cryptography, and economics. Satoshi Nakamoto was a renegade, but he also knew well the history that preceded him. In order to understand why blockchains work, you need to understand their building blocks — what came before blockchains, and why those things didn't work.

Here are some good prerequisites to be familiar with, in order of importance.

Note, these links are just a starting point, you'll probably want to dive deeper for many of these topics.

Computer science

Data structures

You'll want to be familiar with the characteristics and complexity guarantees of the major data structures: linked lists, binary search trees, hash maps, and graphs (specifically, directed acyclic graphs which feature prominently in blockchains). It helps to have built them from scratch to better understand how they work and their properties.

Top highlight

Cryptography

Cryptography is the namesake and bedrock of cryptocurrencies. All cryptocurrencies use public/private key cryptography as the basis for identity and authentication. I'd recommend studying RSA (it's easy to learn, and doesn't require a very strong math background), then look at ECDSA. Elliptic curve cryptography requires significantly more abstract math — it's not important to understand all the details, but know that this is the cryptography that's used in most cryptocurrencies, including Bitcoin.

The other important cryptographic primitive is the cryptographic hash function. These can be used to construct commitment schemes, and are the building block for Merkle trees. Merkle trees enable Merkle proofs, one of the key optimizations that blockchains use for scalability.

Distributed systems

There are a few good textbooks on distributed systems, but it's a sprawling and difficult area of study. Distributed systems are absolutely essential to reasoning about blockchains, so you must build a foundation here before tackling blockchain programming.

Once you're no longer living on a single machine, you have to start reasoning about consistency and consensus. You'll want to know the difference between linearizable and eventual consistency models. You'll also want to learn the guarantees of fault-tolerant consensus algorithms, such as Paxos and RAFT. Know the difficulties of reasoning about time in a distributed system. Appreciate the tradeoffs between safety and liveness.

With that background, you'll be able to understand the difficulties around Byzantine fault-tolerant consensus, the primary security requirement of public blockchains. You'll want to learn about PBFT, one of the first scalable algorithms to deliver Byzantine fault-tolerant consensus. PBFT is the basis for many non-proof-of-work blockchain consensus algorithms. Once again, you don't need to understand the details of how and why PBFT is correct, but get the general idea and its security guarantees.

It's also very useful to understand the traditional methods of distributing databases (at its core, blockchains are databases after all). Learn about sharding (such as via consistent hashing), leader-follower replication, and quorum-based commits. Look into distributed hash tables (DHTs), such as Chord or Kademlia.

Networking

The decentralization of blockchains derives in large part from their peer-to-peer network topology. As such, blockchains are direct descendants of the past P2P networks.

To understand the blockchain communication model, you need to understand the basics of computer networking: this means understanding TCP vs UDP, the packet model, what IP packets look like, and roughly how Internet routing works.

Public blockchains tend to spread messages via gossip protocols using flooding. It's instructive to learn the history of P2P network design, from Napster to Gnutella, BitTorrent and Tor. Blockchains have their own place, but they draw upon the lessons of these networks and how they were designed.

Economics

Cryptocurrencies are inherently multidisciplinary — this is part of what makes them so fascinating and radical. Besides computer science, cryptography, and networking, they are also deeply interwoven with economics. Cryptocurrencies can derive many security properties through their economic structures, which is often termed *cryptoeconomics*.

As such, economics is essential to understanding cryptocurrencies.

Game theory

The most important branch of economics that plays into cryptocurrencies is game theory, the study of payoffs and incentives among multiple agents. You don't need to go *extremely* deep here, but you do need to understand the basic tools of game theoretic analysis and how you can use them to analyze incentives in one-shot and iterated games.

Two key concepts in your repertoire should be Nash equilibria and Schelling points, as they feature prominently in cryptoeconomic analysis.

Macroeconomics

Cryptocurrencies are not just protocols, they are also forms of money. As such, they respond to the laws of macroeconomics (if they can be called laws). Cryptocurrencies are subject to different monetary policies, and respond predictably to inflation and deflation. You should understand these processes and the effects they have on spending, saving, etc.

Another valuable economic concept is the velocity of money, especially as it corresponds to valuing a currency.

Microeconomics

Cryptocurrencies are also deeply interwoven with markets, which requires an understanding of microeconomics. You'll need a strong intuition for supply and demand curves. You should be able to reason about competition and opportunity costs (they'll apply frequently to cryptocurrency mining). For many coin distributions and cryptoeconomic systems, auction theory features prominently.

I expect you'll be familiar with some of these topics already. If you are, feel free to skim or skip over them entirely.

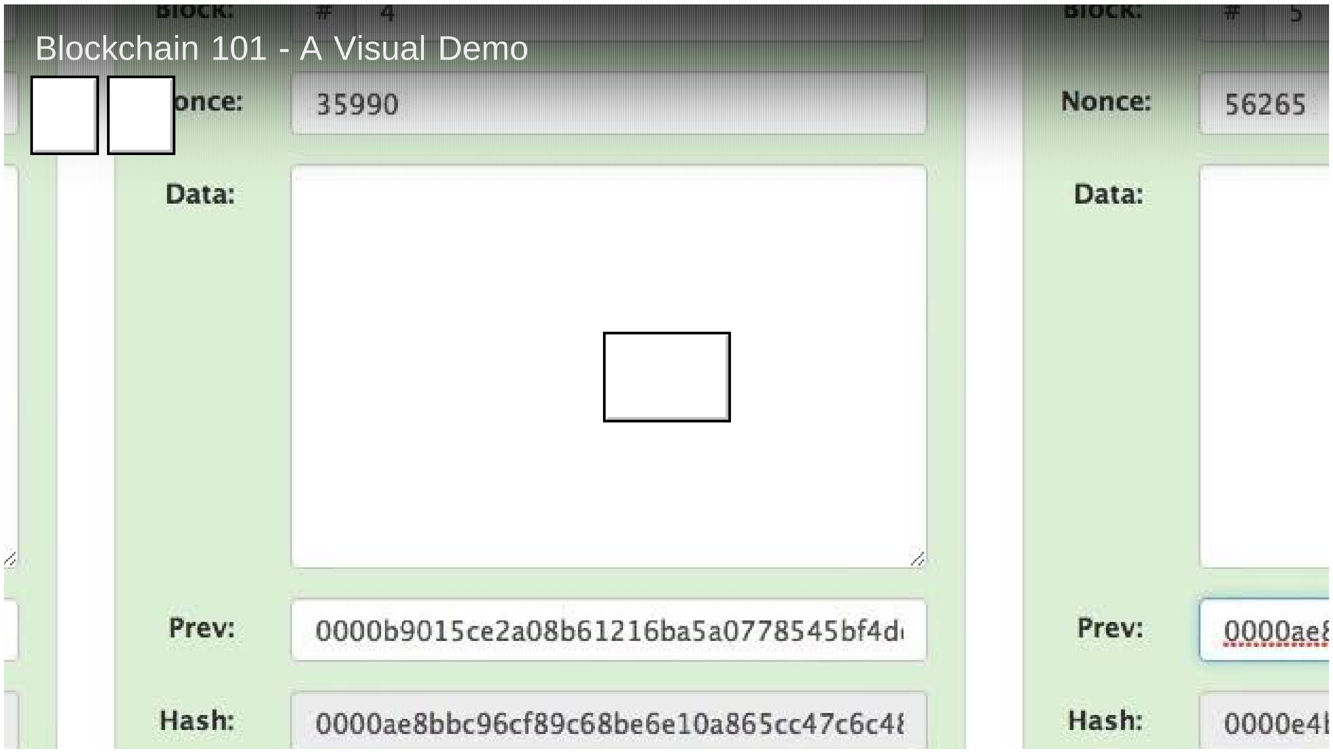
Okay, by now you've gone through and shored up your fundamentals (or maybe you skipped a bunch, who's counting?), so now that you've got your theory in check, let's get started on blockchain development.

The Theoretical Foundations of Bitcoin

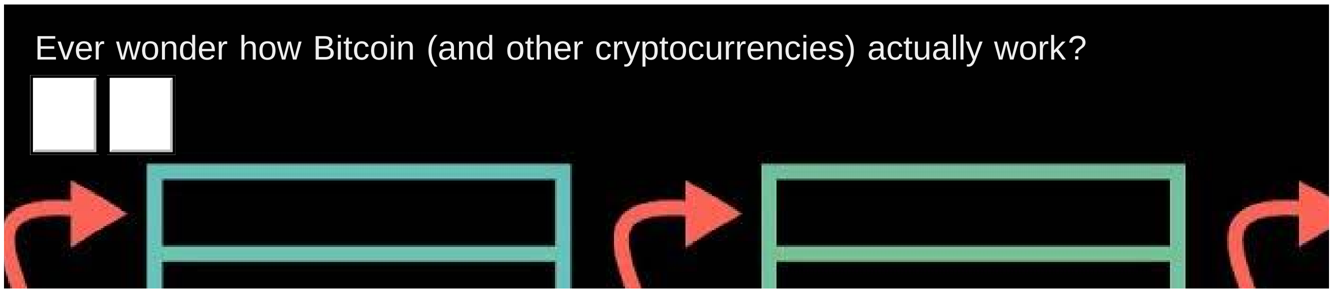
In October of 2008, Satoshi Nakamoto published a white paper in which he described a protocol for a decentralized digital currency. He called this protocol Bitcoin.

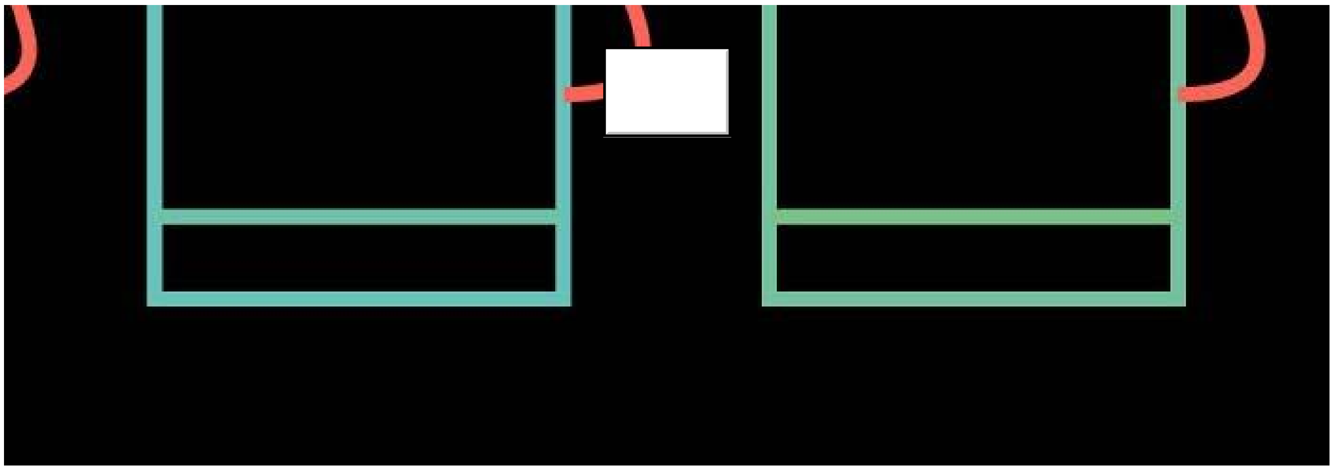
Before you can understand the big ideas behind blockchains, you have to start with Bitcoin and grasp Satoshi's original insight.

First, I recommend building your intuitions about proof-of-work and the fork choice rule (also known as Nakamoto consensus). Start here:


Blockchain 101 - A Visual Demo
Block: # 4
Nonce: 35990
Data:
Prev: 0000b9015ce2a08b61216ba5a0778545bf4d
Hash: 0000ae8bbc96cf89c68be6e10a865cc47c6c4
Block: # 5
Nonce: 56265
Data:
Prev: 0000ae
Hash: 0000e4

I recommend watching more than one video explanation to get the idea seared into your head:


Ever wonder how Bitcoin (and other cryptocurrencies) actually work?



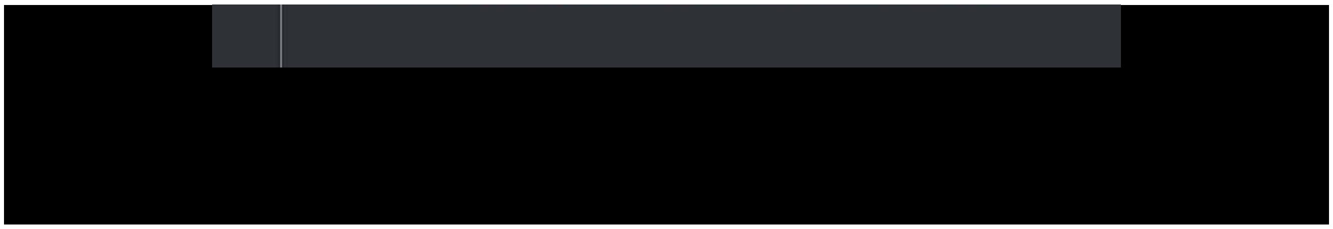
Great. Now that you've built up your intuition, this article will provide a deeper end-to-end exposition of the critical components of how Bitcoin works.

Building a blockchain yourself

Now that you have the high-level intuition, it's time to build your own proof-of-work based blockchain. Don't worry, it's easier than it sounds. Here are some good resources.

First, I have a video lecture where I walk through exactly how to do this in Ruby (I recommend watching even if you're not a Ruby programmer):

Let's build a blockchain! — A mini-cryptocurrency in Ruby (Haseeb Qureshi)
Let's build a blockchain!
A mini-cryptocurrency in Ruby



Source and slides here.

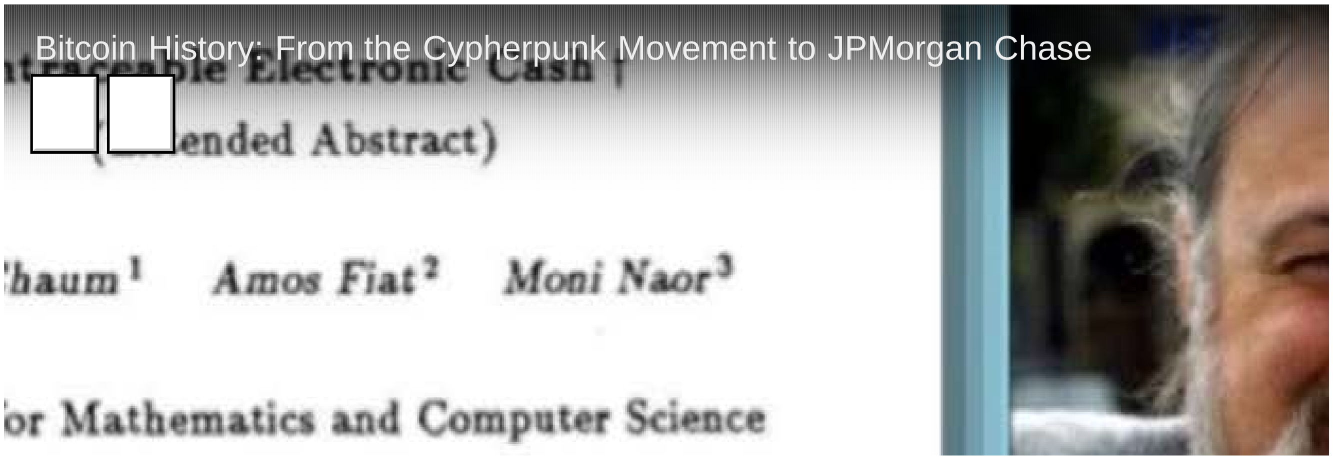
There are also other blockchain implementations you can find, written in various programming languages. Go on and build your own, and satisfy yourself that it's mostly functional.

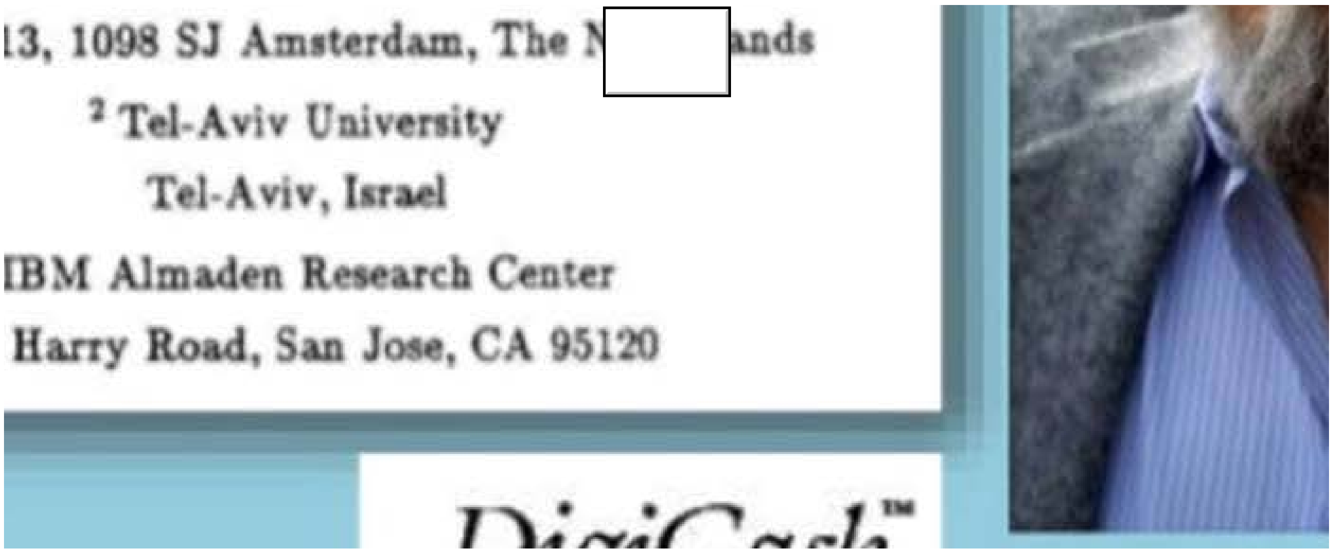
Once you've made it this far, you should have a good grasp of how to implement a simple payments application atop a blockchain (i.e., Bitcoin). You should also by now have enough background that you should be able to read and understand the original Bitcoin whitepaper.

To understand the economics and mechanics of Bitcoin mining, I recommend watching the lecture on Bitcoin mining in the Bitcoin and Cryptocurrencies Princeton course.

If you've gotten this far, you should understand Bitcoin well enough to walk through a Bitcoin block header and understand what each of its components mean. You should also be able to play around with a Bitcoin block explorer and navigate raw Bitcoin transactions.

Now is a good time to study up on the history of Bitcoin and cryptocurrencies. The below video, offered by a UC Berkeley Decal, gives a good overview.


Bitcoin History: From the Cypherpunk Movement to JPMorgan Chase


13, 1098 SJ Amsterdam, The N ands
2 Tel-Aviv University
Tel-Aviv, Israel
IBM Almaden Research Center
Harry Road, San Jose, CA 95120
DigiCash™

Some more extra credit resources:

- Academic precursors to Bitcoin
- Mechanics of Bitcoin: UTXOs and Bitcoin script (Bitcoin script is not super important, just know roughly what it can do)
- Short guide to Bitcoin forks
- Soft forks and miner signaling
- Double spends, 51% attacks, and selfish mining
- Replay attacks
- Bitcoin scalability problems, which is the source of most of the contentiousness in the Bitcoin ecosystem. You should have an idea of why Bitcoin folks argue so much about the block size.
- Segregated witness, a.k.a. SegWit, not essential but it comes up a lot.
- Lightning Network, one of the more important scaling solutions for Bitcoin, also generalizes to other blockchains
- Bitcoin full nodes, Bitcoin fee statistics, charts, charts and more charts
- Bitcoin energy consumption index (at the time of publication, Bitcoin mining consumes as much energy as all of Peru)
- Insightful essay by Gwern on the scrappy inelegance of Bitcoin

- Jameson Lopp has a wealth of other resources on Bitcoin if you want to go deeper down the rabbit hole.

Ethereum and smart contract programming

Now that you've built a blockchain and understand the dynamics of Bitcoin, it's time to delve into Ethereum.

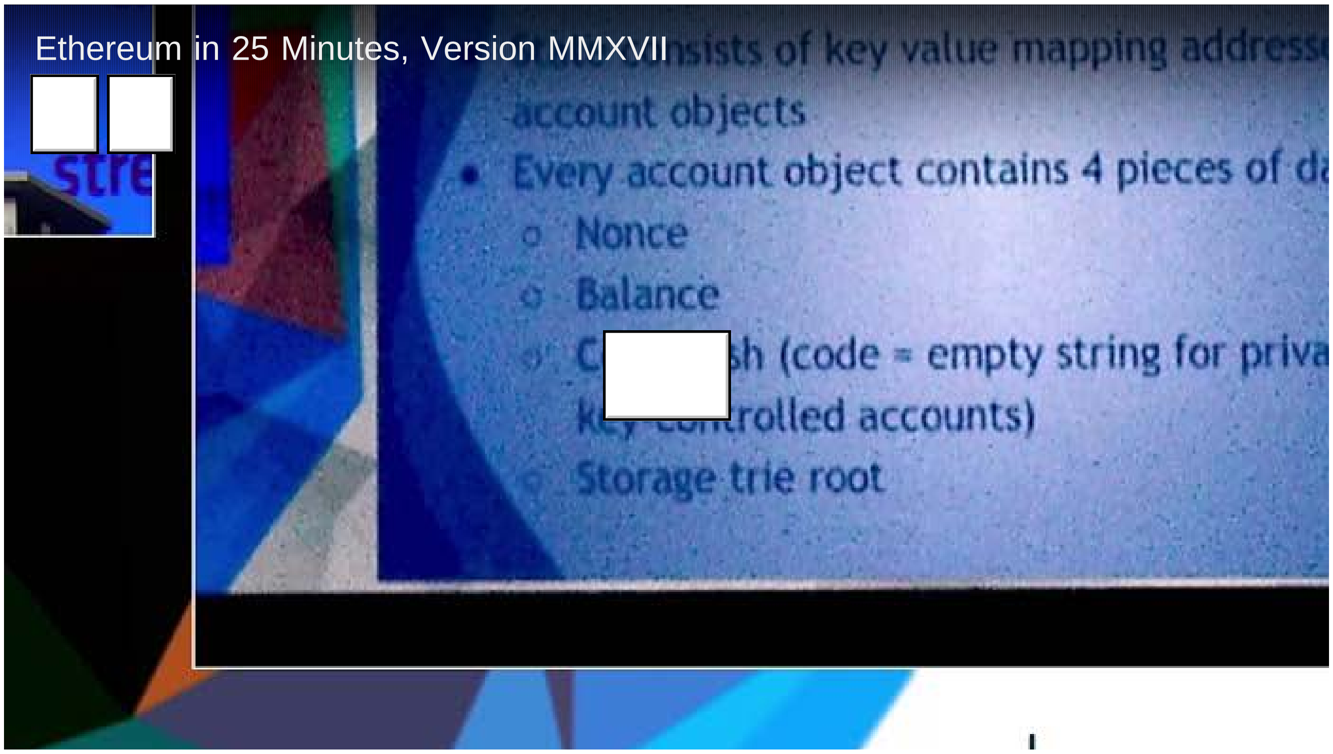
You understand how blockchains and proof-of-work can achieve distributed, Byzantine fault-tolerant consensus within a peer-to-peer network. But a payments network is just one application you can run atop such a blockchain. In 2013, Vitalik Buterin, the creator of Ethereum asked: what if you used a blockchain to implement a decentralized computer?

In Ethereum, you pay miners to execute your programs on this distributed virtual machine. This means you can perform arbitrary computations, using a Turing-complete programming language (unlike Bitcoin script). Obviously that includes payments-related applications, so Ethereum enables a superset of Bitcoin's functionality and has birthed a renaissance of innovation.

This brings us to smart contracts — the name for programs that run on such a virtual machine. A smart contract can interact directly with the blockchain's cryptocurrency in accordance with the execution of a program. In other words, you can create financial contracts that automatically enforce themselves. It's a wild idea, and all sorts of sci-fi futuristic stuff you can do once you embrace this programming model.

Ethereum has enabled the wave of ICOs and developers building atop the blockchain. It is the second largest cryptocurrency behind Bitcoin, it has more than 10x the developers of the next most popular platform, it has the strongest developer team, the most mature tooling, and the majority of ICOs and projects atop it. It also has the most industry support, which goes a long way. In all likelihood, if you're doing blockchain development, you'll be writing code for Ethereum smart contracts. (Even if you're not, it's essential to understanding what's going on in this space.)

First, a more detailed high-level explanation of Ethereum:


Ethereum in 25 Minutes, Version MMXVII

nsists of key value mapping address
account objects
- Every account object contains 4 pieces of da
  - Nonce
  - Balance
  - C sh (code = empty string for priva
    rolled accounts)
  - Storage trie root

The ideas behind Ethereum have also spawned a wave of innovation in cryptoeconomics. You should dip your toes into the ideas around DAOs, and all of the sci-fi fever dreams that they hint at.

Okay, that's enough fantasy, let's dig into the tech.

Here's a good overview of the Ethereum yellow paper and its internals, by Preethi Kasireddy. Ethereum uses an account model rather than Bitcoin's UTXO model — you'll soon see why this makes it easier to write smart contracts.

As with any technology, the best way to get acquainted with Ethereum is by building a few small projects.

The dominant programming language for Ethereum is Solidity, which is a statically typed JavaScript-esque language. It's a language with a lot of warts, and many questionable design choices. More robust languages like Viper may replace it once they're production-ready, but for now Solidity is the lingua franca of smart contract programming. It's basically Ethereum's JavaScript, so you'll need to learn it (and its pitfalls).

To get your first exposure to Solidity development, I recommend working through all of the CryptoZombies tutorial. It's a delightful and high-quality Codecademy-esque tutorial that will teach you the basics of Solidity programming.

Learn to Code Ethereum DApps By Building Your Own Game
CryptoZombies is an interactive code school that teaches you to write smart contracts in Solidity through building your own crypto-collectables game.
Get Started, It's Free
Learn More
CRYPTOZOMBIES

Now that you've whetted your appetite, it's time to develop on your own.

The "hello world" of Ethereum is building an ERC-20 compliant token. I recommend this guide as a first tutorial to walk you through the process.

Remix is an in-browser Solidity editor and compiler — it's basically the training wheels of Ethereum development, so I recommend working through the rest of your practice in Remix. But it's also worth setting up a local blockchain and getting a sense of the Ethereum tooling. This tutorial does a good job of walking you through an end-to-end blockchain stack and explaining the pieces as they go along.

Next I'd recommend building a voting system. I'd call this the Todo App of Ethereum. Karl Floersch has a great tutorial where he walks through how to build a secure commit-reveal voting system.

Great, now for your mid-term exam: build a secure coin toss game, where two players can securely bet on the coin flip. No tutorial this time, do it on your own. Think about possible attacks — how can the players cheat? Can you ensure that they play honestly?

Here are some hints.

Smart contract security

Security is absolutely essential to blockchain development. Smart contracts have been plagued by disastrous hacks, including the DAO hack, the Parity Wallet hack, and the affectionately named Parity Wallet hack 2 (now with its own T-shirt). You absolutely must read analyses of all three of these hacks if you're going to be writing production smart contracts.

The truth is, **smart contracts are extremely hard to get right**. Though the programming toolchain will improve to make these exact attacks harder, they were ultimately all due to programmer error. There are also many subtler bugs that arise from smart contract programming, such as in frontrunning or secure generation of randomness.

As a smart contract developer, you must treat security as paramount. There's no "move fast and break things" in smart contract programming. That means any code that handles significant flows of money should be run through static analyzers like Oyente or Securify, tested thoroughly, and then audited by an experienced smart contract auditor. You should also try to rely on pre-audited components, such as OpenZeppelin's open source contracts.

To strengthen your security chops, I recommend working through The Ethernaut by OpenZeppelin, a game where you find and attack vulnerabilities in smart contracts. Many of them have you replicate real attacks against smart contracts that have occurred in the wild.

Phil Daian also has an excellent set of smart contract hacking challenges called Hack This Contract.

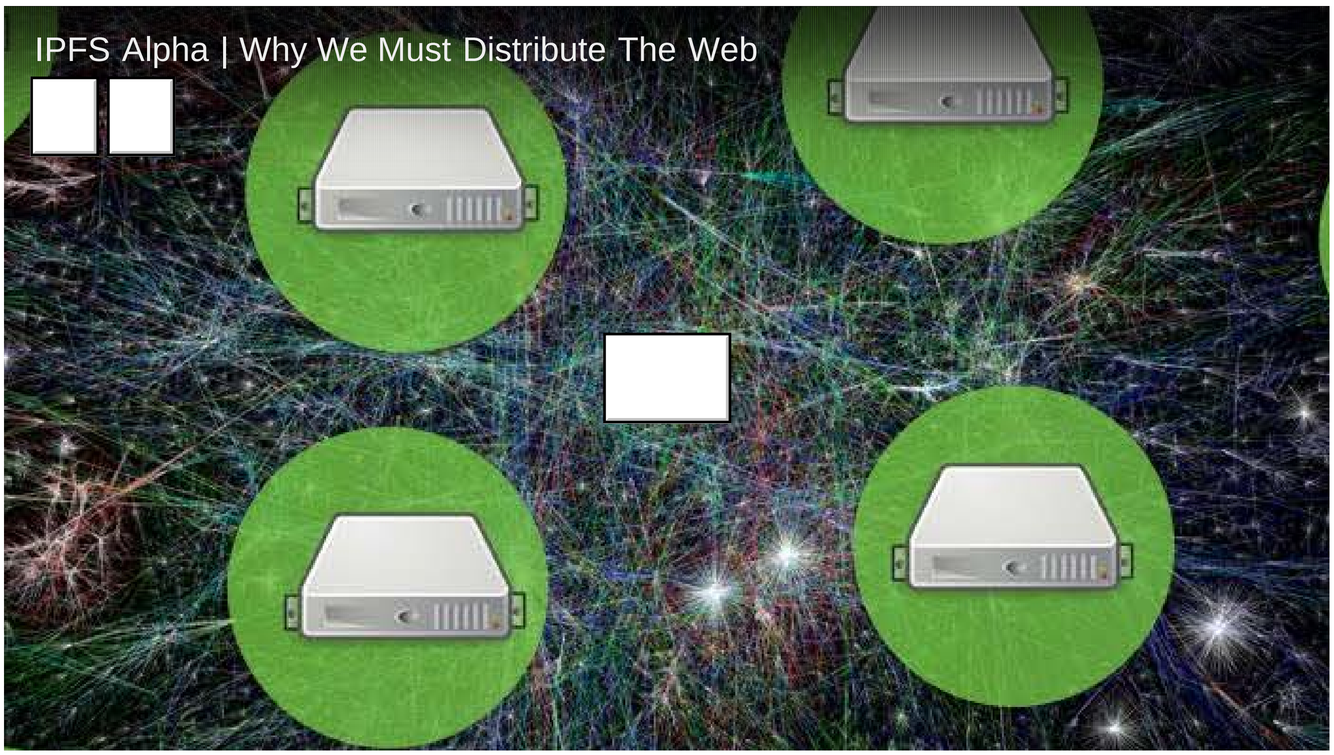
Once you make it past that, I strongly recommend reading the entirety of Smart Contract Best Practices, compiled by ConsenSys. Expect to revisit this document many times over in your smart contract programming career. The bibliography is also worth exploring for further reading by security experts.

Taking off the training wheels

If you’ve made it this far, you should now be ready to move past Remix and start using a serious Solidity development stack.

Most developers recommend VSCode or Atom for your text editor, since they have decent Solidity plugins. For interacting with a local blockchain, you’ll want to use Ganache (formerly TestRPC), and you’ll want to use the Truffle framework for your (JS-based) tests and configuring your build pipeline.

Now is a good time to look into IPFS, which you can use as a fully decentralized filestore at much cheaper cost than the Ethereum blockchain. Here’s a brief explainer by the creator, Juan Benet:



For interacting with Ethereum and IPFS full nodes, Infura is what most devs recommend. Etherscan and ETH Gas Station provide useful real-time stats on the Ethereum network.

Once you have your full Web3 stack set up, try deploying an end-to-end Dapp (decentralized application). This tutorial provides a nice full-stack overview using Node and Postgres for the backend, and this tutorial will show you how to create a fully decentralized application, using IPFS as your persistence layer.

Building your own projects

You should now be comfortable with most of the tech — what's left is to start building stuff and going deeper into the blockchain community.

First, start building your own projects. If there's some great idea that you're excited about, go build it, and convince others to hack on it with you! If you don't have an idea yet or aren't comfortable getting your hands dirty, there are many high-quality open source projects that welcome contributions. OpenZeppelin might be a good place to start for smart contracts.

Better yet, I'd recommend starting by finding an actively developed project that you're a fan of. Get on their Slack or Rocketchat — the devs are usually readily accessible. Tell them you'd like to contribute and ask for some small tasks (or find unresolved issues on their Github).

Note that while I've been focusing on protocols and smart contract development, blockchain companies need web developers to build their core functionality. These roles will often require interacting with blockchains, so it's essential to have a good mental model of how blockchains work — but for many engineers at blockchain startups, most of your work will be in building a Python webserver, or designing a React frontend, and interacting with the blockchain may be a small part of that job. You don't have to specialize in smart contract development — in reality, that's only one part of a working blockchain stack.

Beyond open source contributions, there are also many blockchain hackathons constantly popping up. Most projects have a free public Slack you can join, and there's a very active Gitter channel for Ethereum itself where lots of devs hang out. As you go deeper into the space, you'll eventually find your peer group, whether it be in a Slack channel, Telegram group, or Gitter channel. Wherever it is, find your people and continue learning.

Navigating the blockchain community

The best way to really understand the blockchain world is to immerse yourself in it. Read and listen to the smartest people, especially stuff they've written in the past. This has always been my strategy when trying to learn a new domain, and it's paid dividends for me.

There's lots of good blockchain content out there, but there's also a lot of crap. Here's the information diet I recommend.

Media

The three fantastic podcasts I recommend are the Software Engineering Daily Blockchain interviews, which provide good technical intros to many topics and cryptocurrencies. From there I recommend Epicenter and Unchained — you'll want to go back and listen to many of the older episodes. Another interesting up-and-coming technical podcast is Conspiratus. I'd recommend subscribing to each of these.

There are a few good Youtube channels (though there's tons of trash on Youtube). Subscribe to the Ethereum Foundation and watch Devcon3 presentations. Blockchain at Berkeley records many of their lectures, most of which are excellent technical overviews. Decypher Media also posts talks, whitepaper reviews, and tutorials. Jackson Palmer has engaging weekly overviews, these are on the less technical side but very evenly presented.

Online reading

For realtime blockchain chatter, it lives mostly in two places: Reddit, and Twitter. For Reddit, most subreddits are very low quality and dominated by noise. r/Ethereum is consistently decent quality (and there are a few okay subreddits for specific cryptocurrencies). Most subreddits though are primarily dominated by speculators, and are not a good use of your attention. Stay away from Bitcoin-related subreddits. Bitcoin notoriously has one of the most toxic communities, and Reddit only magnifies that.

Twitter is more of a mixed bag. For better or for worse, most blockchain people live on Twitter. Blockchain Twitter was somewhat of a mystery to me at first, but eventually I developed an informal ontology of Twitter blockchain people. From my experience, there are five types of blockchain personalities: the builders, the entrepreneurs, the journalists, the traders, and the "thought leaders."

Avoid "thought leaders" like the plague. Entrepreneurs can be okay, though they mostly act as hype men or tweet about their own projects. Investors mostly tweet about prices and hype-y projects, so if that's your thing, that's your thing. Journalists tend to tweet about major news items of the day—I recommend staying away unless you need real-time analysis, which you probably don't. If you're an active trader it might be important, but if you're trying to build on the blockchain, most real-time stuff is a distraction.

Pay the most attention to the builders. They're the people who matter most right now, and who are pushing the technology forward.

A few representatives from each category (do a breadth-first search of who these people follow if you want to fill out your Twitter feed):

Builders

- Vitalik Buterin, Ethereum
- Zooko Wilcox, ZCash
- Nick Szabo, inventor of smart contracts
- Vlad Zamfir, Ethereum
- Marco Santori, Cooley LLP
- Riccardo "fluffypony" Spagni, Monero
- Matt Liston, Gnosis

Entrepreneurs

- Balaji Srinivasan, Earn.com
- Erik Voorhees, Shapeshift

Investors

- Naval Ravikant, MetaStable
- Ari Paul, Blocktower Capital

- Linda Xie, Scalar Capital
- Chris Burniske, Placeholder

Journalists

- Tuur Demeester, Adamant Research
- Laura Shin, Forbes

(You should also follow me, though I definitely don't belong on this list.)

All that said, I recommend minimizing your exposure to Twitter and Reddit. If you're not a journalist or a daytrader, chances are, you don't need a firehose of realtime chatter. Important information will bubble up to you asynchronously. There are several good news digests that will summarize the most important news of the day/week that you can consume on your own time without being at the mercy of attention markets.

I recommend subscribing to Inside Bitcoin for daily digests of the most important crypto news pieces (it covers more than just Bitcoin). For token projects, Token Economy has excellent weekly writeups, and Week in Ethereum has good digests of developer-focused happenings in the Ethereum ecosystem.

Beyond this, you probably don't need to be monitoring for real-time news. Focus on building stuff and learning.

You'll want to follow the best blogs. Long-form content tends to be the best bang for the buck. I recommend following these:

- Vitalik Buterin for excellent blockchain and cryptoeconomic analysis (read all of his older blog posts as well, Vitalik is widely regarded as a once-in-a-generation thinker)
- Hacking, Distributed for blockchain security analyses by Cornell researchers
- Unenumerated, Nick Szabo's luminary blog with challenging and eclectic essays about the role of cryptocurrencies in society
- Money Stuff, Matt Levine's Bloomberg syndication, with cutting and insightful

analysis that touches on the intersection of markets, finance, and blockchain news

- Vlad Zamfir for tempered and cautious perspectives on the state and public blockchains
- Chris Burniske for a string of excellent blog posts on how to value crypto assets
- Jameson Lopp for his great technical posts from the perspective of a software engineer building for the blockchain ecosystem
- Great Wall of Numbers by Tim Swanson, for his sober and unflinching deconstruction of blockchain mania, especially in the enterprise space

(You should also read my blog, though again, I don't quite belong on this list.)

Books and courses

If you want a more structured approach to learning this material, there are a few high-quality books and courses out there (and a lot of low-quality ones).

The best overall textbook for blockchains is Bitcoin and Cryptocurrency Technologies (which accompanies the Princeton Coursera course). The only other books I'd recommend in this space are Mastering Bitcoin by Andreas Antonopoulos, and his upcoming Mastering Ethereum, co-authored by Ethereum cofounder Gavin Wood (both published by O'Reilly). The one nontechnical book I'd recommend is Digital Gold by Nathaniel Popper. Pretty much everything else worth reading will be in blogs, not books — this space is moving so fast that the most important figures seldom have the time to write books, and books are often outdated by the time they're released.

If you want a more structured approach to learning this material, there are a few high-quality courses out there (and a lot of low-quality ones). I've already linked to a couple lectures from the Princeton Coursera Course (the videos are on Youtube as well), and the UC Berkeley Decal. I've also heard good things about Consensys Academy for folks who want to get into smart contract development.

I'm also teaching a 4-week seminar on cryptocurrencies for software developers at the Bradfield School of Computer Science in SF. The course is in-person in SF only and seats are limited, since it's a small and in-depth seminar-style class. But if you're a software engineer in SF and want to learn more about the theory and practice behind cryptocurrencies, it might be a good fit for you.

Getting a job

As I said before, blockchain startups are hiring like crazy. If you've actually gotten this far and have done even half the things I suggested, you are probably already employable in this space. AngelList did a great writeup on how to get a job in the crypto space.

There are several good aggregators for blockchain-related job postings:

- AngelList crypto startups
- BlockchainJobz
- Ethereum Jobs
- Be in Crypto
- Blockchain Job Board
- Crypto Jobs List
- Google jobs (blockchain search query)
- ConsenSys jobs (Ethereum venture studio with many projects under their umbrella)

Some particularly promising blockchain startups that I know are hiring devs:

- 0x
- Dharma Labs
- Civic

There are also a number of larger companies in the market for crypto devs:

- Coinbase, the Google of crypto, is always hiring like crazy
- Stellar and Ripple if you want to work directly on more enterprise-friendly cryptocurrencies

- Square has integrated some blockchain, though not sure if they're hiring externally
- IBM, Visa, or JP Morgan if you want to kick it old school

(Note that this specific company list is super Bay Area-centric, because that's where I live, so your mileage may vary. The job aggregators are more global though.)

But to my mind, the best way to get involved in a company is to find a project you're excited about and reach out to them directly. Most blockchain teams are willing to hire remote for the right talent. Many devs are readily accessible on Twitter, Github, or on their public Slack channels. If you have a solid portfolio and can demonstrate technical chops, most people will be impressed if you show some initiative.

And that's as far as I've got for you. If you've done all of the above, you should be set, and you'll probably be even farther along than me before long.

The rabbit wormhole

What I've shown you is just the beginning. Cryptocurrencies are still in their infancy, and I really believe the is the most rapidly evolving space you can be working in. I'm sure this guide will be out of date within a year, and there are so many amazing projects I haven't had the opportunity to talk about. But if you get into this space, you'll find them in due time.

Keep exploring. Keep getting better. Keep learning.

And I hope to see you come join us.

Haseeb

Thanks to Nadav Hollander and Ivan Bogatyy.