So you want to be a blockchain developer?

Revised 2022-01-18.

There's a lot of people that have a lot of reasons for wanting to "get into" blockchain development. Which is great! Learning is always good! But a lot of people decide this without *really* knowing what that means. But that's ok.

So where do we start? A lot of people have the same questions, so I'm writing this document to be a kind of infodump, reading list, and FAQ. I'm not trying to teach you everything, but my goal is to provide some direction and "less-biased" information. I'm not going to pretend that I'm *not* biased, but I'm going to do my best to cut through the nonsense. I'm also going to come off like I'm shilling Rust here, but there's a lot of good reasons for that and it really does deserve it (for reasons that I won't bother going into here). This is also pretty US-centric, so if you're not from the US then I'm sorry, but hopefully you'll still find value in this.

Before we start, if you're looking into launching a token or your own cryptocurrency, make sure to see the FAQ question at the end about it.

Note: This document is a bit of a work-in-progress, and might evolve over time. If you have questions on particular topics which I might not have given all the detail that they deserve, reach out to me somehow.

- So you want to be a blockchain developer?
  - What?
    - Dapps & "Web3"
    - Exchanges, Custody Providers
    - Infrastructure (or "Core Dev")
    - Wallet development
    - Security & Auditing
    - Research
    - Hardware development
  - How?
    - From Zero
    - YouTube tutorials
  - Why?
    - "Hard money"
    - Alternative power structures and modes of interaction
    - Don't go insane
    - Useless projects
  - Where?
    - On proper decentralization
    - L2s and Sidechains
    - Other ledgers
    - Differences in approach
  - Further Reading
  - FAQs, misconceptions, and other topics

- Is proof-of-work bad for the environment?
- What's the most efficient proof-of-stake network?
- I want to launch a (ERC20/BEP20/etc.) token / cryptocurrency network
- I want to launch an NFT
- I want to create a "fan token"
- Is proof-of-stake better than proof-of-work?
- Are soft forks more coercive than hard forks?
- // TODO more questions

What?

The first question we should ask ourselves is what do we really mean by "blockchain developer". This can mean a *ton* of different things. I'm going to list off a few different areas that the term "blockchain developer" might refer to.

Before we get started, Python is pretty convenient to know in *all* of these areas. It's just so versatile that *not* knowing it can be seen as out of the ordinary, even if it's not strictly necessary for the job. You don't have to be an expert by any means, but feel comfortable reading/writing it.

Dapps & "Web3"

This might be what you were thinking of when someone linked you this article.

DeFi is in vogue at the time of writing (1 December 2021), as well as NFTs and other things. I'll discuss these more in depth later.

At the moment, this involves a lot of web development. Personally, I *despise* the current state of the "web" industry. I find the tooling a huge pain to work with, the languages and libraries really dumb and half-assed, and the software pretty bloated and inefficient in general.

If you're going to do this, you'll most likely need to learn Solidity. People will also say that you need to learn JavaScript, which *might* be true, but please do yourself a favor and at least use TypeScript instead. Typechecking is good, not typechecking is bad, wibbly-wobbly types are worse.

And no, **you cannot *just* learn Solidity**. You always need to have some other software to interact with the contracts you deploy.

If we're lucky we'll get to live in a future where all of the business logic for DeFi apps is built in Rust and someone only needs to make a thin web UI wrapper layer if they want to use it from a browser.

Exchanges, Custody Providers

There's a ton of jobs at exchanges. These are often fairly traditional software development jobs, but you're still exposed to a lot of neat crypto concepts and topics. If you want to get into these kinds of jobs, the skills you'll need vary depending on where in particular you're looking to work. Sometimes web skills are useful, sometimes not. Trading engines are usually in C++, but there's a ton of other kinds of jobs in the middle.

I'm not the best resource on these kinds of jobs as I've never worked somewhere like that, so you'd best ask someone else about it.

Infrastructure (or "Core Dev")

This is also a pretty broad category, but what I'm referring to here isn't the stuff you build *on* blockchains, but the stuff blockchain systems are *made out of*. Not just the blockchains but also "Layer 2" systems like Lightning or rollups (more on these later as well).

In another vein, light client development is extremely important for being able to support users on low-end devices.

This kind of work is primarily done with Rust, which *does* have a fairly steep learning curve but it absolutely pays off. Go used to be and still is fairly popular, but Rust is much more mature now. C/C++ is also used in some places, especially around Bitcoin-like chains.

This is *roughly* the area that I personally specialize in because it's a very nice mix of applied cryptography, game theory, and economics.

Wallet development

Related to infrastructure, *designing good wallets* is not an easy problem.

Bitcoin wallets are fantastic, there's good ones on every platform that support all kinds of features. Ethereum on the other hand is *severely lacking* in good wallets. Wallets are not like Dapps so the methodology when building them is very different. All of the Ethereum GUI-based wallets rely entirely web technologies which present a very large attack surface for vulnerabilities to be found in and demand a lot of system resources from the hardware that runs them. This is not acceptable for mass adoption.

For an example of what happens when this does go wrong, see the Mist audit report. This disaster is what ultimately led to the Mist wallet being discontinued. There was a tweet about it but I can't find it now.

What Ethereum needs (among other things) is a good, rollup-agnostic mobile and desktop wallet using native UI toolkits. And it doesn't *have* to be specific to Ethereum, ideally it'd support anything that isn't a shitcoin, but Bitcoin already has wallets that are pretty good so we don't *really* have to go that far.

Security & Auditing

There are *all kinds* of unique classes of vulnerabilities that occur in smart contracts and other blockchain systems. Since these systems are directly handling value, there's a very strong incentive to find and exploit them.

Auditing is very expensive but very important for any serious wallet software or smart contracts. If you already have experience pentesting other kinds of software, you'll probably be able transfer over a lot of your existing skills.

Research

These are the people that do the really hard stuff, like making proof-of-stake even work. Some of it is more practically oriented, bringing theoretical ideas to practical implementations (which sometimes overlaps with building infrastructure). But some of it is definitely more on the math side, like designing cryptosystems such zero-knowledge proof protocols.

To get an idea of the kinds of topics being discussed, read through the bitcoin-dev archives or ethresear.ch. On the more theoretical side read some of the blockchain-related papers on the IACR ePrint archives to see if that interests you instead.

Hardware development

You can't securely store funds on a regular desktop computer, which is what most people use as their daily-driver. Even a server (which have lower attack surfaces), if it's connected to the internet, it's going to be a target. The way that we solve this "the right way" is by storing the bulk of funds on dedicated custom hardware like Trezor or Ledger hardware wallets. You (as the manufacturer) get to control *all* of the code running on the microcontroller in these devices and the hardware itself is also very controlled and standardized (with no need for things like northbridges, storage controllers, wireless adapters, etc. that might steal your data), which gives you a very strong sense of security in what's running on them.

But these microcontrollers don't even running a multitasking operating system like Linux and you have access to very little system resources, so there's a lot of unique challenges, even for embedded developers. This is usually all done in C or assembly, but Rust is positioned to come into popularity here.

A lot of the skills involved involved in hardware development overlap with those used by software auditors, but focused more on the end-to-end complexity.

As an aside, there's experimental projects like OpenDime that are working on making new kinds of hardware wallets more cheaply.

How?

Getting to the meat here, regardless of what you choose to pick up, it's hard to go into the field without being fairly adept at writing software. Crypto is *technically* fintech and it comes with a lot of responsibility, moreso than in traditional finance where in some cases mistakes can be reverted. If you screw up you *can absolutely* lose all your money, or worse, *other people's* money. This is why it's so important to use statically typed languages, write good tests, and *do your due dilligence*.

But keeping all that in mind, regardless of what you're working on it's important that you have a fairly solid foundation on how blockchains work. It's very common for people to skip this and get straight into writing Solidity contracts, then have a very hard time and write very poor contracts.

So here's a short list of things you should definitely read, roughly in order:

- Basic cryptography: how hash functions are used, merkle trees, public-key signatures, etc. (also to prompt you to "think about the bytes" instead of treating data types as magic collections of data)
- this document on p2p networks (ignore the particulars here if you aren't

interested)
- The original Bitcoin paper: make sure that you have a fairly solid understanding of how blockchains actually work and why as a core foundation (annotated version)
- The Bitcoin Wiki: there's a lot of good articles here, read at least the FAQ, Myths, etc.
- Ethereum Whitepaper (linked as webpage): gives an introduction to how it implements a more stateful smart contract ledger (note that some of the issues it raises about Bitcoin's expressiveness are no longer accurate)
- The Lightning Network paper: this one is pretty dense but L2 systems are the future (see below) and it's important to have a good model for thinking about them, if you can understand enough to have an educated conversation about it then you're learning a lot
- Vitalik's Incomplete Guide to Rollups: goes into how state of the art stateful L2 systems work on account-based ledgers

After going through some of these fundamentals (and whatever other specific material you personally need to understand it), you will probably have, in my subjective opinion, a good enough working background to be able to jump into a lot of other blockchain-related topics and go wherever you might want to go. There's a lot you might have to learn about particular projects built on top of these networks, but that'll be much easier for you when you have the basics well-understood.

If there's any topics introduced in these links that you feel that you don't have a thorough understanding of, then find another resource on that particular topic somewhere else to fill in the background you might be missing.

Don't read these *yet*, read the rest of this doc first. :)

From Zero

If you have absolutely no programming background, that's ok! Everyone should have a better understanding of how the tech around them works, and it's never too late to get started.

Before trying to get involved in crypto, spend some time learning about:

- **Python:** it's a great teaching language, if you want to do "web" things go look at FastAPI
- **algorithmic skills**: stuff like leetcode might help you here, but I don't really like it
- **computer networking**: to understand how the internet *actually* works
- **Linux:** it's what pretty much all serious software relies on, you don't have to switch to it as your daily driver OS but I highly recommend it, as doing anything with crypto on Windows is a bit of a liability due to all the malware/spyware that's built into it (and no, WSL is not really a full substitute)

I also tend to recommend going throug these steps to get a good all around knowledge and get better at DIYing things.

- buy a domain (Namecheap)
- rent a VPS (DigitalOcean or someone, just not AWS, GCP, or Azure)
- point the domain at the VPS

- install a web server (NGINX, Apache, etc.) from a shell over SSH
- get a certificate for that domain (Let's Encrypt via Certbot)
- put something on your new website (put files into `/var/www/html`)

For bonus points, set up a git `post-receive` hook and add an `alias` directive in your NGINX config so you can just do `git push prod` and update the website easily, like I have. If you can figure all that out then you're on a good track.

YouTube tutorials

There's a lot of tutorials on YouTube that claim they can teach you everything you need to know about X, Y, and Z technologies. Some of these are good, but they often railroad you down a specific path and don't leave you prepared to make your own decisions. They prompt you to just copypaste code and not "learn how to learn" things on your own. If they help you, then that's great! If not, then don't be disappointed.

There's one many-hour-long video I've seen a few people share before that covers a lot of topics, but you can't expect to go from 0 to being proficient just by sitting down and watching that. The best way to learn *is by doing*. But don't bite off more than you can chew as that can make you feel discouraged.

On the other hand, YouTube *is* a great resource on deep dives, usually recordings of conference presentations or certain podcasts.

Why?

Ok so, *reeeally* big picture stuff.

A lot of people have a lot of good reasons for being interested in crypto. And of course, a lot of people have a lot of bad reasons.

"Hard money"

A certain category of political ideologies has a major focus on hard money. The narrative is that by uncoupling the US dollar from the gold standard and putting issuance into the hands of the Federal Reserve it changed the incentive structures, leading to a lot of long-term inflation that slowly eroded buying power. If you subscribe to this line of thought, then if we ensure that the base monetary supply is rigid and keep money "hard", then our buying power can be maintained. And at the surface level that makes a lot of sense.

But it turns out economics isn't a hard science and there's a lot more to that story. There's a few problems with this reasonsing that I (and others) object to:

- Issuing currenties are often the only option for governments when trying to recover in times of crisis. At several points in its history, the US decided to issue fiat currencies in order to get shit done when the economy was struggling. The last major time being during the Great Depression, which (although it was partly reinstated for some time after) is essentially where we are today. The other thing that very hard money makes a problem is that it causes the "real value" of debt *grow* over time, faster than their interest rates, which is the opposite of what you want if you're not a massive lender

like a bank.

- It cuts down on investment since capital holders would be happier just to sit on their wealth instead of giving it to people to do stuff. Ideas for how a modern financial system built around a very hard form of money would operate (and not lead to very negative outcomes for those that don't have much of it), are still very theoretical. If the Hyperbitcoinzation hypothesis end up happening then the level of economic disruption it would cause would probably outweigh the benefits we'd get out of it.

- Printing money alone doesn't actually automatically cause inflation. One example is that Japan has been aggressively doing QE (which is really about providing liquidity rather than simply printing money, but it's treated as essentially same by a lot of people) for the last 25 years and JPY is *still* suffering a modest level of deflation. The *circulating supply* is more relevant, but there's other factors. What we're seeing right now in December 2021 much more of the issue is just supply chains falling apart due to shocks of the pandemic and understaffing. Especially with people suddenly gaining a small amount of self worth and realizing they deserve more than ~$11/hr to work a shitty retail job.

- Inflation can actually somewhat of beneficial thing in that they gradually erode debts over time (assuming it doesn't have a very high interest rate). If you're someone with a lot of debt and you have a job with enough leverage to increase your pay according to inflation, it actually makes more sense to just pay the minimum you're expected to and wait until you have more cashflow. If you're a recent college graduate, this probably applies to you. I don't know why so many people are obsessed with being debt-free, since our economy *runs* so heavily on debt. Conversely, with a fixed monetary supply, debts *grow* over time in real terms at a faster rate than just the interest on the loan.

There's also arguments about the Federal Reserve system existing just to bail out corporations, or to print money and destroy the wealth of the middle class, or some other things along those lines. The political position of the Fed is a little bit of complicated history. While, yes, it's *technically* true that the Fed is privately owned by banks, it *does not* function as a usual financial institution and it largely held to the direction of the executive branch. Should the general population have more oversight into the operation of the Fed system? Probably. Democracy is generally good. But the reason it's given a relative level of autonomy is to help avoid scenarios where it's just used as a political tool in the months before an election to the detriment of the broader economy. But perhaps there's ways around this? I don't know. I'm not trying to be an apologist here because *obviously* there's a lot of things about the US government that could be improved, but not everything is a grand conspiracy and sometimes the incentive structures of people in power aren't the most obvious to us on the streets.

So my perspective is that Bitcoin will probably go a long way to replace gold as a wealth-hoarding asset, but fiat currency is here to stay in one form or another. Sacrificing control over a country's monetary supply in the interest of some ideological purity about how money *ought* to work is shortsighted when it means you're getting rid of a tool that could actually be used for good in a time of need. The term "Modern Monetary Theory" is the big word that roughly describes what the west has been doing for the last several decades, although if you ask someone

in /r/Bitcoin what it means they probably won't be able to define it. And I'd argue that the US is mainly just doing *an impressively bad* job of it rather than MMT fundamentally being flawed because it worked pretty well for a few decades after WWII.

A major critique of MMT is that it allows the government to run up a massive deficit and "spend taxpayer money" that they "don't have" like nobody cares. This is true by definition, but government deficit is a harmful myth used to deflect from real political issues. Here's a video on *that*, if you're interested.

I hope that makes sense, it's a little bit rambly.

// TODO want to give some links for further material cite some more sources for this section

Alternative power structures and modes of interaction

What's more interesting to me is the potential that distributed ledgers have for building alternative kinds of power structures.

As it turns out, humans are really good at coordinating. We kinda *evolved* in order to work together. This can be very good because it allows large teams of people to very quickly exert a lot of effort on large projects, but it takes time to set up the relationships necessary to coordinate at a large scale. This is what a business is, it's a group of people operating under some form of hierarchical task delegation. But there's other ways to organize large teams, like in FOSS projects.

Markets with a moderately large set of parties involved and accurate information readily available, can be an effective and efficient substitute for more organized forms of coordination. But it's important that these markets remain at a high level of efficiency, that weird behavior is recognized, and that parties aren't forced to engaged in it coercively, because otherwise, it's really easy for the market to become a game where strong parties exploit or push out weaker ones. Unfortunately, in today's economy it seems like a lot of markets that people engage in in a day-to-day basis aren't actually that free or efficient.

Distributed ledgers can allow us to build interesting new kinds relationships and modes of interaction between people to accomplish tasks in more efficient ways than we could when relying on traditional forms of economic coordination. There's a lot of potential to do this much more tranparently and expressively through use of rich domain-specific programming languages than we could with traditional legislation or pure market economics. And it's much more participatory than having to rely on electoral politics to make things happen.

Carbon credits are a really shitty and limited example of this kind of thinking. If you're unfamiliar, they're are a mechanism devised to incentivize use of carbon-sequestering practices and disincentivize carbon-emitting practices. The legislation creates a tradeable notion of "credits", where large scale emitters are required to find some way to purchase credits on a market from other businesses that have them to sell. This mechanism isn't entirely effective, but it's the kind of thinking where we can shape incentive structures to accomplish a public good. In this case it's by introducing an artificial market (yay neoliberalism I guess?) to enable business models that sequester carbon become economically viable. (one article on issues)

These ideas haven't been completely conceptualized because it's important to get it to work right. But there are at least some people working in this direction.

Vitalik has shared a few of these ideas previously:

- Coordination
- Moving beyond coin voting governance
- Retroactive Public Goods Funding | ETHOnline 2021

Don't go insane

It's really easy to get sucked into a rabbit hole of ancap nonsense. Remember that everything (even this document) has at least somewhat of an ideological slant, so instead of being obsessed with finding "unbiased sources" it's more worth it to get different perspectives on an issue and analyze it in terms of the incentives that authors of those sources have.

I don't really want to have to include this next paragraph but I see it enough that it worries me so I feel like I should bring it up.

Cryptocurrency communities tend to have a very large population of rightwing libertarians (see the hard money section above), and so that can attract people that that talk a lot about preserving the sanctity and purity of a few certain things other than just free markets. It's everyone's responsibility to call people out if they spread hateful rhetoric and challenge them on their basic premises if they're fundamentally unjust to ensure that we can maintain a space that's well respected and accepting of people of all walks of life.

Try to avoid conspiratorial thinking and be aware of the pardox of tolerance. There's some other interesting topics on this if you're interested in more socio-political stuff, message me directly and we can have a conversation about it.

Useless projects

There's a huge number of useless projects out there. They're mostly cash grabs of one form or another. But it's important to understand why.

A lot of people are here just to make a ton of money.

There's a couple of ways to do this, but a few major categories are:

- effectively creating a heavily abstracted form of gambling where they initially hold all the cards and play them to extract wealth from people who show up to play
- taking advantage of speculative investors and issuing tokens (both the usual kind an NFTs) that they sell to people who have the expectation of being able to sell it to someone else again, or just to straight up con people saying that
- metaphorically selling shovels in a gold rush

I find this to be generally stupid. It's just moving value around in ways that usually benefit the wealthy and extract a lot of value from people in these desperate times to feel like they have a chance of "making it".

The third one of these is more okay if you're working to ensure that people have

more equal access to quality investment advice, which is good, but it can also more be working perpetuate and legitimize the game and might end up with the net effect of causing more harm than good. But there's also a lot of freelance developers out their that just charge other con artists that don't want to learn Solidity themselves for doing the dirty work for them (usually copypasting existing contracts, tweaking a few things, and redeploying), which I think is about as ethically sound as working for the Lockheed Martin. (Although the latter is obviously far more objectionable.)

This might sound like I'm criticizing the entire concept of DeFi, but that's not what I'm trying to do here. It's more specific than that.

For more perspective, see the "I want to launch a …" subsections at the end.

Where?

There's a ton of different cryptocurrencies out there! Great right? Well, only kinda. There's a lot of tradeoffs in all things.

Bitcoin is the original, and the tooling for it is far more mature than anything else. But it's limited in what it can do on the base layer, but in exchange it's a lot more lightweight and cheaper to use than Ethereum (even for basic use cases at the time of writing). It's also much easier to upgrade it to introduce new opcodes due to how its scripting system works, so long term Bitcoin is still very viable. And due to its much more constrained scope, if you're trying to build *cheap* payments, Lightning is *far* more mature than any Ethereum L2 is, just go look at all the wallets there are.

You might think that the UTXO model is overly restrictive. And it is, to an extent, if you're trying to do very stateful contracts on the blockchain. But blockchains are for *verifying* things, not *computing and storing* things, and since we know that any computation in NP can be verified in P time… you do the math.

Ethereum is much more expressive and fits into more similar kinds of mental models about programming that people already have. But the price you pay for that in the much higher cost of doing anything with it in comparison: a single L1 tx on Bitcoin at the time of writing is on the order of $1 (sometimes as low as 1 sat/vB (~16 cents) right now!), but a simple L1 ETH transfer on Ethereum at the moment is $9.60, with Uniswap swaps costing at much as $90 *or more*.

That's not supposed to be FUD. There's solutions to these problems, see below.

On proper decentralization

People often misunderstand why transactions on blockchains even *have* fees in the first place.

When you publish a transaction, *every node on the network* must verify it and the persistent state it creates (utxos on Bitcoin and storage on Ethereum) must be stored indefinitely. The transaction history itself also has to be stored by a large subset of nodes on the network. Certain mechanisms like fast sync can alleviate some of the cost of downloading and replaying the transaction history, but even ignoring the one-time costs, the *ongoing* costs of maintaining the current ledger

state replicated across every node in the system is a pretty high, including bandwidth costs. In order for the network to continue to be decentralized, which is important since a healthy population of full nodes that fully validates blocks is essential for any light clients to be able to be able to be used safely, we must keep these costs low. It should be possible to run a full node on a consumer-grade laptop *and* continue to be able to use it for regular tasks. On Ethereum the overall ledger state is already far higher than what's reasonable, at a few hundred gigabytes with a full archive node requring multiple terabytes. Bitcoin is not too bad, but Bitcoin currently lacks a fast sync mode so you have to resync from genesis which is still around 380 gigabytes.

So there's limits that are put in place for those reasons. When the network use reaches those limits, a market appears where block producers are "selling" block space and users are paying for it. As demand increases further, these fees can get pretty high. We could increase the block size and/or gas limits, but this extra capacity would get consumed sooner or later by induced demand or just natural increasing demand for the technology. But increasing network capacity causes all those problems to get worse, and if that becomes the norm it turns into a kind of a feedback loop. We can't do that, because decentralization is important to us.

Here's a pretty good talk that Vitalik did on it and why it matters that the network is decentralized and what kinds of restrictions that places on this. You don't really have to watch this if you are already convinced, but if you aren't then it's worth considering what can happen if users don't run their own nodes and are subject to the will of the people, as happened in this story that gave rose to the Hive network. But that's also a lesson in why centralized dPoS consensus algorithms are not particularly safe.

This section got longer than I meant for it to, you could argue for hours over this, but let's move on.

L2s and Sidechains

So how do we solve this problem? If transactions are large and expensive because everyone has to verify them, then we have two options that we can try to do either or both of:

- make transactions smaller and/or easier to validate, reducing the burden of them
- reduce the number of parties that need to be aware of them, therefore reducing the overall cost of a certain level of network activity (while trying to preserve security)

This is exactly what off-chain "layer 2" solutions try to do.

Lightning does this by making a network of bilateral channels between peers that can be updated just by passing some signatures between nodes without touching the chain and use this to relay payments through the network. By batching channel open/close transactions the overall impact of managing these channels is relatively low and individual payments have 0 impact on the chain. That way peers that aren't directly involved in relaying a payment don't even know that it even happened!

Rollups do this by doing some careful bookkeeping to be able to reduce the size of

a transaction to on the order of 12 bytes, and leave validation to either a fraud game (optimistic) or a zero-knowledge proof (zk). Users can reconstruct their part of the rollup state by following the history on-chain. Through a scheme like this, a large “batch” of transactions can be “validated” in a single operation.

“Validiums” take the idea of a zk-rollup farther, and omit the on-chain data entirely but keep the zero-knowledge proof that the old and new state roots are valid, which means that the per-tx cost is lessened farther. Instead, users can reach out to a network of data providers that have an incentive mechanism to provide data about the state they care about on request.

What’s common about these systems is that they don’t introduce any new major assumptions. You get scaling but you don’t have to trust that another consensus system is secure and decentralized in order to have confidence in the system. *They derive their security from the L1 chain (and math).* To go more into detail on this, go through the reading list in the “How” section above.

Rollups/validiums are very general and support fairly general and stateful computation, just with a higher level of complexity. The long term goal for all (serious) networks is for most economic activity to happen in L2 systems. This might seems weird and restrictive, but that’s just a product of the tooling not being quite ready yet, the fundamentals are solid.

Rollups are the future, for Ethereum. But the tooling just isn’t there yet and in my opinion there’s a lot of ecosystem fragmentation that may prevent these kinds of systems from dominating in the short term. Maybe you can be the change we need to make that happen?

Another maybe viable option are sidechains. Sidechains come with other assumptions in that they are usually their own consensus systems and it’s usually possible for the operators of the sidechain to steal your funds from the main chain they are “pegged” out of. This is a different kind of system than what’s considered an L2, as the security of the network is more disconnected from that of the network they’re based around.

Examples of these are Liquid on Bitcoin (which uses a multisig quorum of a group of reasonably crypto organizations) and Polygon and xDai on Ethereum (which use some kinds of dPoS). Liquid is pretty centralized all things considered so it’s not widely used. Polygon is more widely used, but it’s pretty centralized since they essentially took Ethereum and turned the numbers with it way up in order to “scale” better. xDai isn’t very widely used but there are a handful of projects using it including POAP and Dark Forest.

Other ledgers

The vast majority of other general-purpose blockchains out there are scams. There’s a few projects out there that are more application-specific and/or have very principled goals in mind, but there’s many more that aren’t.

Good examples of good application-specific and goal-oriented chains:

- **Siacoin:** doing decentralized storage the right way
- **Litecoin:** it used to be a “testing in production” for Bitcoin, they even activated SegWit first, but it’s started to fall out of favor in recent years

- **Zcash:** very strong privacy guarantees, better than Monero, which is still good but not *as* good

But there's more examples of networks that are absolutely, unequivocally shitcoins:

- **Binance Smart Chain (BSC):** it's not fucking decentralized, if it's not decentralized then there's no point, nobody cares about how low the fees are if it's basically just using 1 corporation's database and you can't sync the chain yourself, it's a cash grab for Binance and I won't be surprised if it becomes illegal to use in the US at some point in the mid future, it's also literally a copypaste of Ethereum
- **Tron, EOS:** it's just silly really, it's not remotely decentralized either
- **Bcash, BSV:** CSW is a charlatan, the worshipping of "what Satoshi would have wanted" is shortsighted because Satoshi is not god and is not infallible
- **Ripple (XRP):** they issued themselves 90% of the supply at genesis and have been selling that out whenever they need to raise funds again, which sounds a lot like an unregistered security
- **ICP / DFINITY:** "it's aws if it was a franchise", to apply run a "datacenter" you have to go to them and open a support ticket, it's not decentralized at all
- **IOTA:** I'm not sure if much of the original team is there but they harassed security researchers for exposing collision issues in their weird ternary hash function and did other weird "quantum" buzzwords, since then it seems like they've started to drop the buzzwords but it's silly
- **Nano:** "zero fees", *spam attack*, pikachusurprised.jpg, *adds fees* (still nobody actually uses it except shills on reddit)
- **Secret Network:** the working principle of TEEs issecurity through obscurity, so really all you're doing is just shifting the trust to *Intel* of all places

In general, something to consider here is this graph that shows the distribution of token allocations. The red "Insiders" slices are the parts to pay attention to, where bigger is generally worse.

And there's a few others that are more of a grey area and don't really want to endorse but I can't say *hard* that they're shitcoins and nothing but:

- **Solana:** it's really popular because it arose around the time that Ethereum started to become really expensive to use, and there is some genuinely interesting concepts underlying it, but it's not really much more decentralized than Tron/EOS since (at the time of writing) it only required 19 validators to misbehave or be coerced into halting the chain and the original team has a magic stop button that they've pushed before when things broke, it's also still pretty expensive to run your own node even if you aren't a validator
- **Cardano, Tezos:** Haskell and OCaml are cool languages and good choices for smart contracts but they've been overpromising and underdelivering for ages now

And there's plenty of others that could fit into these categories.

Differences in approach

Something I've noticed from different kinds of people is how they decide to evaluate projects.

If you're coming at a problem from a investor-oriented mindset, you probably are looking for platforms and ecosystems that are relatively small and have a lot of potential for growth. The expectation here is that you can invest and make huge profits later on, but this carries risk. This mindset can also lead you to ignore fundamental problems with an approach with the expectation that they'll be addressed before it actually matters.

If you're coming at a problem from a technically-oriented mindset, you probably are looking for platforms and ecosystems that are already well developed and have a colorful ecosystem with good tooling and an already-strong network effect. This makes it easy for new users to come and interact with your project and you get the benefits of being able to integrate more easily with the projects that are already there.

Sometimes those fundamental problems that you overlook can only be addressed by changing the system to be more like another one that already exists (like: limiting computation requirements and shifting capacity to L2 systems), but then there won't be any reason for the other one to exist as it'll just be needless unconstructive ecosystem fragementation!

This difference highlights to me the need to emphasize taking principled approaches to problems and trust the nerds when they say things. This isn't to say the first one is strictly wrong, but it can lead you into doing things that are really dumb, like doing *anything* on BSC. And since we're on the topic, since so many things that *do* launch of BSC are just trying to be scamcoins, any project that *is* trying to be honest and above-board that launched on BSC gets immediately treated with extreme suspicion simply because of the environment it exists in. You can call this unfair, but network effects are important.

Further Reading

- Jeremy Rubin has been writing some good stuff on the technological values underlying Bitcoin
- Vitalik Buterin has a really good blog about some really big picture things and specific technical deepdives

FAQs, misconceptions, and other topics

Is proof-of-work bad for the environment?

To an extent, but that's the wrong question to ask.

First of all it's important to split up the relevant points here. Merely using electricity, in the general case, is morally neutral. *Emitting carbon dioxide* into the atmosphere is the problematic part. *Sometimes*, part of your energy supply may be produced through fossil fuel plants that *do* emit CO2. In *that* case it *is* indirectly morally problematic.

So what do we do about that? You can't be expected to just *not* use electricity. The path of human development demands it, and it's especially silly considering that there's *plenty* of green sources of energy that aren't being utilized to their maximum possible capacity. The reason for why they aren't is long and

complicated, but the short answer is: ***blame oil and coal lobbying***. And if you're in opposition to nuclear power, know that it's actually *far* safer than oil and coal (measured in deaths per TWh) if you include the respiratory issues suffered by people who live/work in/around fossil fuel plants, and that a major reason people hate it is due to fearmongering by our friends in the fossil fuel industry.

Different groups like to do some math and figure out some really wild estimates for the number of tons of CO2 that get released "for each Bitcoin tx". Ignoring that the error bars on these estimates are pretty large and hard to figure out exactly, measuring emissions in terms of CO2/tx is completely the wrong way to do it. The argument that follows is usually along the lines of "if Bitcoin handled as many tx/s as Visa then it'd be emitting as much CO2 as $reallybigcountry", which is nonsensical and not how it works. The implicit assumption with that is that you just *deploy more Bitcoin blockchains* and independently throw the same amount of hashrate at them to get the same level of security as the one Bitcoin blockchain we have now. But that design is completely impractical to make decentralized and would (if you did it lazily like this) get you the same level of security as just increasing the block size by 100+ times and then keeping the same level of hashrate, which is also a nonstarter.

It also ignores that Bitcoin txs are not 1:1 with how bank transfers work and ignores the possibility that Lightning has to scale *really efficiently and cheaply*.

The other thing that's worth considering is that mining operations also can support a price floor for solar farms, where at peak hours normally must sell electricity onto energy markets *at a loss*. This incentivizes deploying new solar generation capacity, which makes it actually a good thing! This is only a small example and doesn't apply in every location, but it's something to consider.

What's the most efficient proof-of-stake network?

This is fighting over pennies. It's pointless to compare because in the grand scheme of things it's pretty much negligible.

There was a paper that claimed that Cardano was the most efficient PoS network, but the methodology of that paper was really silly. They did the thing where they measure capacity in terms of transactions per second, which is stupid for the reasons I outlined above, but it's especially stupid if you take the very likely scenario that most people are tranacting in rollups and other L2 systems, there just *isn't any upper bound* to throughput on *any* network that can support them. You could probably process thousands of transactions per second on a single raspberry pi sipping less than 5 watts.

I want to launch a (ERC20/BEP20/etc.) token / cryptocurrency network

First question to ask here is "why?". What goal are you trying to accomplish with the token/currency?

To speak more generally, there's few different kinds of (useful) cryptoassets:

- store of value: this is the direction Bitcoin is probably headed, these assets must have a strong use case and be foundational to the system, you're

probably not going to be able to build one of these yourself and there might not even *be* space for a new one in the space
- utility tokens: some protocols have one or more tokens with programmatic issuance/burning in order to achieve a specific goal, MakerDAO with its MKR token to regulate and govern the DAI stablecoin is an example of this
- bridge tokens: aren't *really* tokens in their own right, but are issued on one ledger as a proxy for an asset issued on another ledger (note that things like WBTC are completely custodial, it's not a cryptographic system that *you* can trust)
- stablecoins
  - custodial stablecoins: someone gets people to give them a bunch of dollars, puts them in a bank account, then issues a token 1:1 for them, and you trust they won't play games with the dollars they put in it
  - algorithmic stablecoins: see MakerDAO
- DeFi tokens: used in new kinds of financial instruments and can have a variety of different goals

There's others, these are just a few examples. The point I'd like to highlight here is that you should not try to make a token just because you want one. You can make one on testnet for learning purposes, of course, there's nothing wrong with that. But if you launch a token and try to get people to buy it just because you think you can use it to get rich, then you have to know that you're getting rich off of these people you're convincing to buy your token. If there's not any real-world value that's being created through issuing this token, then all you're really doing is moving money around. So if you're profiting off this, then that means others are losing out, which isn't far from just stealing from them but with extra steps.

And I'd like to end this bit with this quote from Wikipedia:

> A Ponzi scheme (/ˈpɒnzi/, Italian: ['pontsi]) is a form of fraud that lures investors and pays profits to earlier investors with funds from more recent investors. […] A Ponzi scheme can maintain the illusion of a sustainable business as long as new investors contribute new funds, and as long as most of the investors do not demand full repayment and still believe in the non-existent assets they are purported to own.

If you're building a system where the only real use for the things people buy from it is to sell it later for a profit, then you're basically building some kind of Ponzi scheme and you're not building anything economically productive, and you're probably building something to facilitate extracting wealth from desperate and misinformed people.

I want to launch an NFT

I don't consider NFTs *as a concept* to be completely useless but the current iteration of them that we're in in 2021 I think is pretty suspect.

A major argument in favor of them is that they can be a new vehicle for funding artists through royalties. What people don't understand is that these royalties are completely voluntary, and there's no way to *actually* enforce them in a meaningful way on the ledger. I'm of the opinion that artist should be funded primarily through

patronage and commissions. This is the model that artists operated on for hundreds of years (maybe thousands if you include theater in ancient Greece) up until the rise of modern capitalism, and it worked pretty well for them. With the technological infrastructure we have today it should absolutely be able to recreate this model in a more fair and equitable way.

So that leave the kind of NFT where the NFT *is the art itself*. As it turns out, it seems that the vast majority of "generative art" sales are actually just rich people shuffling around money to inflate volume and create a false sense of value to outside users. (// TODO find that paper) These "pure NFT" tokens that are supposed to have value in their own right are especially ridiculous in my perspective, because they're *purely speculative*. I haven't seen any examples where they have any demonstrable external use value. And since they're intrinsically niche and nonfungible, they cannot ever possibly function as a currency in any reasonable scenario.

People sometimes claim that NFTs can be a new kind of store of value, but then also say that royalyies from NFT sales are a way to fund are are just wrong. Since they're unenforceable it's up to the holders to voluntarily pay them, but then that makes it a worse store of value because it effectively loses some of its value when you sell it, which has the effect of making it less liquid and being a shittier store of value.

And survivorship bias is a thing, too. The small subset of NFTs that sell for millions that you see in the news make up for the vast vast majority that sell for nothing that you don't hear about.

There is probably a space for blockchain games that happen to use NFTs as a way to represent items in the game, just make sure to run that in a rollup. I wouldn't be surprised if I turn out to be wrong one this one though. Be careful not to make it into a Ponzi but with extra steps, which is what seems to have happened to Axie Infinity, probably not even intentionally. It turns out that economics is hard.

Another point that I'd like to get to is "the only digital art worth paying for is art that doesn't exist yet". Because it can be freely copied and distributed, nothing is lost or stolen when that happens. If this was universally applied then that would kinda suck and make sites like Bandcamp that provide a lot of starving artists with some income unviable, but in an ideal world we'd have better infrastructure for funding artists without having to rely on purely transactional mechanisms.

I want to create a "fan token"

I saw this quote the other day (modified for brevity)

> Rally provides social tokens which uniquely combine elements of patronage (to support the artist), fandom (to provide a closer connection to the artist), and investment (financial upside from the appreciation of the digital asset).

There's a few of problems with this, in part because there's better ways to do these and in part because we shouldn't be encouraging these in this way.

- **patronage:** patreon.com already exists and it works really well, all we have

to do it make a decentralized alternative to it (and doing this way by having artists tokens of themselves setting them up to make them function as unregistered securities, which is illegal in the United States)
- **fandom:** the fun part of being part of a fandom is a social act rather than an economic one so you can't mimic it with financial instruments in a fulfilling way, and this can promote parasocial relationships which isn't healthy
- **investment:** speculative assets don't create any new value (they just move it around in ways that usually benefit wealthy people) and doesn't really do anything to bring people together

You could debate these points, of course, but keep this in consideration.

Is proof-of-stake better than proof-of-work?

(setting aside point above about environmental concerns)

This is an impossible question to answer because the premise is poorly conceived.

There's a fundamental difference in perspective between the proponents of each. Proof-of-work gives you a very strong and very mathematical sense of an objective "best chain". Generally proof-of-stake proponents (especially when accepting "weak subjectivity") argue that this level of objectivity tend to argue that this isn't necessary and that it's more useful to be able to use the best live chain that the users around you are using than it is to be on some objectively correct chain that nobody is using.

This disagreement is irreconcilable and it's kinda pointless to try to have a debate over it. The rationale is that since ledger can't steal your funds, so if you go into a cave and come back 10 years later you'll still have access to your funds regardless (unless there's another DAO-style fork, in which case yeah sure). And if you want to interact with other people using the ledger you'll have to have some kind of mutual social agreement with them in the first place, so it's kinda a logical extension from that.

Objective truth about social constructs is up for debate in the first place, so it's overkill to try to adopt a more mathematical view of it that PoW can give you. Some use cases may really demand PoW, like those that Bitcoin is *probably* going to be the most appropriate tool for long-term.

Also, there's work into VDFs to regain some confidence about a "proof of history" that can be lost with PoS alone.

Are soft forks more coercive than hard forks?

TL;DR: No

This is kinda a weird question that I'm including here because it brings up an interesting discussion.

The line of argumentation here comes out of how the SegWit upgrade works, so we have to go over some background. Bitcoin transactions are a set of "inputs" and "outputs". Outputs include a script, usually including a pubkey's hash, that specifies the requirements to spend it, which usually includes a signature of the transaction. To spend an "previout output", you have to provide a "witness" for it,

usually includes a signature, as part of an input. Addresses are pubkey hashes encoded in some format. When you spend coins to an address, you construct a “pay to pubkey hash” output, using this address. (Skimming over some details here.) But there are other kinds of outputs, namely “anyone can spend” outputs. These outputs, as the name suggests, can be spent by any transaction without providing any witness.

So where does the arguement that soft forks are more coercive than hard forks come in? The way SegWit works, is creating a second “witness structure” that accompanies blocks. Signatures and pubkeys tend to be big, so we want to be able to ignore them when syncing older blocks until we have to verify them. It also has the side effect of making it so that modifying the transaction witnesses won’t impact the txid, making off-chain contract systems like Lightning safe to build. The way that this was made as a soft fork is by having “segwit outputs” be these `ANYONECANSPEND` outputs with extra data added to them. SegWit-aware nodes (which in 2021, is basically all of them) are aware of what the extra data means and use it to verify the witness stored in the extra “witness transaction” structure instead of embedded in the primary witness-less transaction. This is why SegWit requires a new address type. Nodes that are unupgraded see the outputs and just skip verifying them, because from their perspective there is no extra rules about how the witness should be structured. (Again, skimming over some details here.)

So the argument is that for users that are running unupgraded nodes, that they should be able to spend those outputs that, from their perspective, don’t have any rules about how to spend them. And that because when they publish a transaction that doesn’t have the extra SegWit witness data for those outputs, their transactions get rejected by peers, that SegWit is “censoring” their activity.

What this line of logic highlights to me is the difference between the core cryptographic protocol, and the use cases of the protocol that software guides users to with respect to the social consensus around the protocol.

Take this statement:

> I should be able to spend funds sent to addresses I control.

That seems reasonable, and you could write a similar one about addresses for multisigs and swaps, but we can ignore that. And its contrapositive:

> I should not be able to spend funds sent to addresses I do not control.

This also seems reasonable.

Pre-SegWit, wallets would usually give you a (now-)legacy `1` prefixed address. Due to p2p tx relay policies, it was also hard to publish a tx in the usual way that included an output that wasn’t a “pay to pubkey hash”, “pay to script hash”, or “pay to multisig” output, defined in very strict ways, as you would have to directly deliver the tx to a miner that agreed to include it, bypassing mempool policies. These factors alone make it difficult for any standard use of `ANYONECANSPEND` outputs, but what really adds onto this for me is those above statements. While *technically*

it's perfectly possible for a user to spend those funds, the *user expectation* that they can only spend funds sent to addresses they control means that they *don't really* control `ANYONECANSPEND` outputs, as since there isn't any wallet that would be able to generate any address that a sender would construct an `ANYONECANSPEND` output for. It's also kinda ridiculous, as you're just throwing away funds if you created an output like that, so why would you want to in the first place?

The social contract within the Bitcoin ecosystem is fairly conservative when it comes to potentially infringing upon use cases. There was *a few* individual cases of users creating `ANYONECANSPEND` outputs, but this use cases is utterly insignificantly small that holding back progress to cater to maybe potentially using it in the future (and any use case can be done in SegWit still) is shortsighted. Soft forks in Bitcoin will never make an established use-case impossible, so they are forced to rely on the mechanisms they have to add additional functionality. You are still free to ignore the upgrade and maintain using your existing clients, and still keep in-sync with other users. This is unlike hard forks, which usually *do* compel an upgrade to maintain sync with other users.

When the overwhelming majority of other users are neutral or in favor of the upgrade, insisting that you should be able to spend a nonstandard output that's exceedingly rarely used you're being entitled and purist. So you're free to hard fork off if you don't like it, which some people did.

SegWit was only seen as contentious because of active misinformation campaigns carried out by Bitmain and other mining corporations that saw Lightning as a threat to their profitability. If you don't want to accept this then you're not paying attention and need to get off the internet and meet crypto people in real life.

To better illustrate this scenario, if you happen to look a little further into how SegWit works, you'll notice that a similar logic could be made where segwit script version numbers >1 are spendable by anyone currently, in order to allow a subsequent soft fork to add more rules to. The mempool policy rejects txs that create these outputs but it is valid for these txs to exist in blocks. You can see in advance that this mechanism would allow this argument to be made again, but framing it that way does it really seem like it's really unfair to say that soft forks when carried out like this are coercive, because if you go out of your way to do something weird and don't tell anyone about it you're asking to have your use-case made more complicated down the road.

// TODO more questions