101 Dogecoin Tricks

chromatic

101 Dogecoin Tricks

Editor: chromatic
Logo design: TBD
Cover design: TBD

ISBN-10:
ISBN-13:

This book uses the Onyx Neon toolset. See https://onyxneon.com/ for more details.

Onyx Neon typesets books with free software, especially Ubuntu GNU/Linux, Perl, PseudoPod, and LaTeX. Many thanks to the contributors who make these and other projects possible.

First edition coming late 2023

This is a draft edition of a book under active development and editing. Please do not share this file with others.

For more details, see https://ifdogethenwow.com/books/dogecoin-tricks/. Please *do* share this link with your friends and colleagues.

Thanks for reading!

Contents

Preface

Dogecoin is the Internet's original meme-based cryptocurrency.

That sentence may read like a bunch of nerdy nonsense to you. Don't worry; it is! Fortunately, this book will make it clearer. It'll still be a bunch of nerdy nonsense, but you'll understand it and be able to have fun, do cool things, and show off your new knowledge to friends at parties[1].

Let's explain that first sentence.

Dogecoin is what we're talking about.

The Internet is a globally-distributed network of computers that lets people and machines communicate in words, pictures, video, audio, and all kinds of data.

A meme is an idea, usually clever or ironic or sarcastic or joking, that's easily spread between people. It often takes the form of an image or video that can be shared on the Internet.

Cryptocurrency is a mechanism to record digital transactions over the Internet between people who don't necessarily trust each other as individuals but, as a group, agree to specific ways to communicate that ideally make the entire network of people trustworthy enough to serve as a mechanism of financial exchange.

Whew.

The crypto stuff is interesting, and there's a lot to learn if you want, but most of it summarizes to this: you don't have to take someone's word that they're trustworthy. You should be able to verify it.

If that sounds like a lot of work with a bunch of serious nerds doing serious nerd stuff, remember that there's a friendly dog mascot with a bunch of silly memes and the whole point is to have fun.

That's what this book is about.

What do I need?

This book assumes you have access to a computer somewhere (though some of the tricks will work on mobile devices such as phones and tablets), have access to the Internet, and can download and install and run software.
At the time of this writing, you can visit dogecoin.com for links and information about how to download and run the Dogecoin Core. Don't take our word for it though; check online if that's still the home of the Core. Test that against other sites such as https://github.com/dogecoin/dogecoin, the Dogecoin Reddit at https://reddit.com/r/dogecoin/ and other locations. If something seems off to you, then ask questions. Honest, helpful people will give you good answers and help you verify them for yourself.

[1] Talk about other things too; we want you invited *back*.

How to Read This Book

A cryptocurrency with a friendly dog mascot is supposed to be *fun*, so you should read this book in a way that you enjoy. Maybe that's front to back or back to front or skipping around. Maybe you read a couple of pages before going to sleep to relax you or teach your brain a lesson. The only *wrong* way to read this book is a way that you don't enjoy.

In general, the book starts by assuming you're new to cryptography, cryptocurrency, and Dogecoin. Earlier pages introduce concepts that later pages build on, but it's totally okay to skip around. Skim the table of contents. Look at the names of every chapter and every tip. If something grabs your attention, read that first. Repeat.

This book also tries to express links between concepts, ideas, and tips. Sometimes something introduced earlier will show up later, and sometimes you'll see similar themes repeated throughout the book. That's on purpose. It's just as valid to breeze through this book on a lazy summer afternoon as it is to skim through it over the period of weeks, months, or years. Feel free to set it down and come back to it later; all of the information will be waiting for you, and ideally each tip has enough breadcrumbs to point to other places that you can remind yourself where you were in your journey at the time.

Put this book on your bedside table. Face it out on from the bookshelf behind you when you're on video calls. Carry it when you're walking, riding the train, or enjoying a cannolo from a park bench downtown. Lend a copy to your friends and donate a copy to your local library. Above all, embrace the fun. Your author believes that learning things can be enjoyable and being in control of your own coins is liberating and powerful. There's a lot to learn, but it doesn't have to be intimidating. It can be fun and playful.

With that said, take note of a couple of things.

Cryptocurrency Isn't Only Dogecoin

Although this book *focuses* on Dogecoin, the basic concepts and a lot of the ideas apply to multiple cryptocurrencies, especially Litecoin and Bitcoin. Many of the examples will work with small or minor modifications. If you prefer one of those other coins over Dogecoin, that's totally okay! You can still learn a lot and have the appropriate amount of fun. We all share some common ground, including code, so we can all learn from each other.

Programmable Money Brings Risks

Even if the only thing we used cryptocurrency for were to track scores in a massive, multiplayer online game of some kind, there would still be risks. The fact that it's *money* means those risks are more serious. Scammers, thieves, grifters, and other bad actors are, unfortunately, a part of the landscape. Where possible, this book attempts to identify risks to you, the people you care about, and the other people in the community. It's up to you to keep yourself and other people safe, but it's only possible to ameliorate risks you know about. Read the risk sections carefully and think about other things that could go wrong.

If you're not sure, ask. After you ask, verify. After all, this is a book about ways to prove things. Knowledge is power.

A Little Programming Can Be Fun

There's code in this book, and if you're familiar with programming in a language such as Python, Ruby, JavaScript, Perl, or something else, you're in a position to get the most out of it. If you've never done this before, that's okay too! The code is there to help you understand and take advantage of things, but you don't have to be a programmer to use programmable money effectively.

With that said, you will get the most out of this book if you're comfortable installing and configuring software, especially working with text files and the command line. If you have a trusted friend to help you, you can learn a lot and take control of your system in ways you previously didn't understand.

Basically if you're comfortable working a spreadsheet, you can build up your skills!

You Can Grow with This Book

Sure, you picked up this book because it has cute dogs on the cover and you're not yet ready to build a pinball museum where people pay for credits with cryptocurrency. That's okay! If you read one tip today and another every week after that, you'll get through this book in two years. That's not a bad investment of lazy Sunday afternoons–and at that point, you may be full of other ideas no one has thought of before as well as new skills to bring them to life.

Credits

This work, as expected, has turned out to have required the assistance of many people.
Thank you to RNB[2] and Rachel[3] for editorial questions and suggestions which led to several of these tips!
Thank you to Mishaboar[4] for advice, perspective, and helping promote a deep empathy for other people.
Thank you to Patrick Lodder[5] for technical advice and always-trenchant corrections.
Any remaining errors are the fault of the sometimes-too-clever author.

[2] https://twitter.com/RNB333

[3] https://twitter.com/meta_rach

[4] https://twitter.com/Mishaboar

[5] https://github.com/patricklodder/

This is a Preview of the Full Book

This is a preview of 101 Dogecoin Tricks. The full book will/does contain 101 tips and tricks about Dogecoin.

This preview gives you a taste of what that book is about.

If someone gave you this PDF and you're wondering what's happening, please visit https://ifdogethenwow.com/books/dogecoin-tricks/ to learn more, perhaps even preorder your own copy! Share and enjoy!

Tip #1 Forge a Chain

Isn't it interesting how the word “forge” in English can mean either “create something successful and strong” or “produce something for purposes of a deception”?

Imagine you're trying to keep a group of rambunctious young goats closed up in their pen to enjoy the sunshine (and out of the garden, because they will eat your baby tomato plants in the blink of an eye). You want a strong fence with a gate, and you need a strong chain to keep that gate closed. In a chain, every link connects to another link. The strength of every part of the chain depends on the strength of every link in the chain.

That metaphor applies to your bank account too. If you're not sure about the transactions from a month ago adding up correctly, how can you trust that the transactions of yesterday and the balance of today make any sense?

Given that this is a chapter of a book about cryptocurrency, you've probably already guessed that cryptography has a few ideas on how to answer that question.

Validate Data

Assume you have a niece or nephew who wants to babysit your goats to earn some spending cash. We'll call this kid Kid A, because that's a nice gender-neutral name and baby goats are also called kids. You don't have a lot of time to keep records[6], and Kid A is really bad about writing down their time, so you want to avoid arguing over when you paid and how much. You need to track this in a way that doesn't mean you have to go back into your phone's text messages every couple of weeks to reconcile things.

You can fix this! First, agree on the format of an invoice. Kid A will report their hours in the form:

```
Goat Sitting
March 5 - 18 2023
12 hours
```

You will report in the form:

```
Goat Sitting Payment
March 5 - 18 2023
12 hours
1200 Dogecoin
```

That gives you the base data format you can agree on. If you use a cryptographic hash (**??**, pp. **??**) on each set of data, you can get a unique fingerprint to prove no one tampered with the payment request/report forms. Use SHA-256 hashing, for example, and you'll get hashes like `24a0ae...` and `84ab55...`. If Kid A provides the hash of their invoice and you provide the hash of your payment, you have a record you can keep in multiple places.

That can help reduce the possibility of accidental or deliberate tampering, but validation of invoicing and payment is a different story.

Even better, let Kid A keep their own hash of their invoice and you keep your own hash of your payment and use them to validate each other. When you receive an invoice, calculate your own hash. When Kid A receives a payment, let them calculate their own hash. This can work together really well.

Validate and Chain Data

What if the validation for one invoice depended on the previous invoice? For example, if March 5 - 18 is the first invoice, you know the hash of that. What if you add the hash of the previous form as input in the next form? For example, the payment for the invoice could change to be:

[6] Or, in the case of Kid A, you prefer streaming individual songs.

```
Goat Sitting Payment
March 5 - 18 2023
12 hours
1200 Dogecoin
Follows: 24a0ae...
```

Now the hash of the payment becomes `eadfd8...`. Add that to the next invoice:

```
Goat Sitting
March 19 - April 2 2023
20 hours
Follows: eadfd8...
```

...and the new hash is `fd13b3...`. If you keep this going, you can validate two things. First, that no one has tampered with any individual invoice or payment. Second, that no one has tampered with the history and lineage of the invoices or payments.

In this mechanism, validation is easy. If you've both kept your own hashes when you've submitted your forms to each other, you can look at the response message and check against your own hash. If anything's different, raise a red flag.

What's Really Going On

As it turns out, that latter property is essential to all sorts of applications, including cryptocurrency and managing the source code for computer programs. By publishing the history of changes, or even only the hashes for each piece of data, anyone with access to the data can verify that every piece of data is in the correct place and everything before and after it belongs where it is.

If you dig into computer data structures or cryptography, you may hear this referred to as a *Merkle Tree* or *Merkle Chain*.

In this payment example, you don't even have to validate the entire chain of invoices and payments to figure out if something went wrong; if you've been validating as you go, all you have to do is validate the most recent message and you can be confident everything else before now is still untampered and accurate and pristine. You can–and sometimes *should*–still go back throughout history and prove the entire chain is valid, but you don't have to assume it is and you don't have to verify everything every time you make a change.

Tip #2 Roll Over Your Odometer

Imagine it's 1979 and you're really, really good at the arcade game Asteroids. Every 10,000 points you gain a new ship. What's the high score?

99,990.

At that point, the score resets to zero, although apparently you can buy a special add-on which increases this limit to 9,999,990[7]. This mod still has an upper limit, at which point the score resets to zero and starts over. The previous sentence has a pun, and you're about to understand why.

This feature turns out to be really important in cryptography.

Romulus, Remus, Modulus, and Remainder

Think of the high score counter as an odometer in a vehicle or the hands on a clock. There's an upper limit (12 or 24 hours on a clock, hundreds of thousands of miles or kilometers in an automobile, a hundred thousand or a hundred million points in Asteroids). Once you exceed the limit, you start over.

This is an expression of a mathematical concept called a *modulus*. You may have heard of it as a *remainder*. You can even demonstrate it by counting on your fingers, from one to ten.

Suppose you want to add the two prime numbers five and seven. Count on your fingers. Raise seven fingers, then raise five more. When all of your fingers are up[8], lower them all again but keep counting. The number of fingers you have raised at the end is 7 + 5 mod the number of fingers. If you have 10 fingers, 7 + 5 mod 10 is 2. If you have 9 fingers, 7 + 5 mod 9 is 3.

What's Really Going On

Why does this matter? If you're doing something like calculating a hash of data (**??**, pp. **??**), you want the output to have two properties. First, you want the output to have the same number of bits regardless of the size of the input. Short inputs and large inputs should produce output that's the same size. Second, you want attackers to have a high difficulty predicting what kind of changes to inputs produce what kind of outputs.

In both cases, dealing with really large numbers but constraining the results to a narrow range helps.

That's what modulus does. You'll see it in all sorts of cryptographic applications. It happens in non-cryptographic applications too, including days of the week, minutes in the hour, degrees of a circle, months of the year, names of notes in the 12-tone Western scale, and the Roman nundinae, or 8-day market/work week.

For our purposes, it has one other essential property. Given arbitrary inputs and a modulus, you actually *lose* information. Multiply 2 by 7 mod 10 and you get 4. Also multiply 6 by 4 mod 10 and get 4. Multiply 12 by 2 mod 10 and get 4. Given the answer of “4” and a modulus of “10”, you can only guess at many, many integers you could multiply together to get that answer.

In other words, you're going to have a lot more difficulty *reversing* the calculation to figure out its initial inputs, even if you have part of the equation. That irreversibility turns out to be an essential property of cryptography.

[7] The things you learn researching cryptography for a book about funny dog money! Search for “Asteroids High Score Kit”.

[8] If you have more or fewer than ten fingers; that's okay. You'll get a different answer, but the same concept applies!

Tip #3 Run a Node

A subtle but essential truths of Dogecoin is that Dogecoin is what we all agree it is. While developers have some ability to add features, fix bugs, and release software for other people to examine, run, modify, and redistribute, it's the *network*–the collective behavior of everyone who participates–that decides what actually happens. By design, everyone who runs a node gets a say in which behaviors are valid or invalid.

For example, the Dogecoin Core 1.14.5 release lowered the default fee per transaction from 1 Doge per transaction to 0.01 Doge per kilobyte of transaction size. You can now take advantage of these fee changes because enough nodes have adopted the new version that the network supports these lower costs.

The power of the network–what we call *consensus*–is in the hands of people who run nodes. That could be you.

What You Need

What do you need to run a node? A computer. Some memory. An Internet connection. Persistent storage. The willingness to do some research and keep things up to date. Patience. A kind heart.

In specific, you need a relatively modern computer (a decent desktop or laptop machine manufactured in the past 10 years will work). As of this writing in April 2023, a full node requires over 60 GB of free hard drive space. The size of that requirement will grow in the future. 4 GB of RAM will help. Windows, Linux, and Mac OS X are all supported. Other operating systems or architecture combinations may require more work on your part.

Are you a node-half full person?

What's the difference between a full node and a partial node? If your node stores the entire blockchain history from the first block mined until today, you have a full node. Otherwise you have a partial node.
While running a partial node saves disk space and provides some benefit to the network by validating and transmitting transactions, full nodes are essential to verify the validity of *every* transaction and to help other full nodes come online. If you can spare the space, running a full node is a great contribution.

Find the Right Software

As of this writing, `dogecoin.com`[9] and `github.com/dogecoin/dogecoin`[10] are two reputable sources which both announce new Dogecoin Core releases and provide download links to Dogecoin Core releases. Depending on when you read this, one or both of those URLs may have changed–probably not, but it's possible–so do some research to figure out what's reputable.

From a verifiable source, download the distribution that best matches your operating system. For example, if you're running Windows on Intel or AMD hardware (not ARM), look for a `win64` bundle. For Mac, look for the `osx` bundle. For Linux, look for the `linux` bundle that best matches your processor bundle. Then verify the download (see **??**, pp. **??**). If something looks suspicious or seems off, stop! Do some research. Figure out what went weird, then decide if and how you want to try again.

After you unpack the software (Windows has an installer or a zip file, Mac OS X has a DMG file, Linux users get tarballs), you'll end up with a couple of alternatives. Do you want to run a GUI or a background process?

This book assumes you'll run the GUI but have access to the other files: a command-line interface called `dogecoin-cli` and the background process `dogecoind` (see **??**, pp. **??**).

[9] See https://dogecoin.com/.

[10] See https://github.com/dogecoin/dogecoin.

Are There More Details?

If these rules seem like they assume too much knowledge you don't have yet, or if things look very different in the future, look for a Dogepedia entry called "Operate a Dogecoin Node"[a]. That guide goes into more detail and, because it's a website, can be updated when things change more quickly than a printed book can.

[a] Currently at https://dogecoin.com/dogepedia/how-tos/operating-a-node/.

Configure Your Node

When you first start your node, you may see a prompt asking you where to store configuration information, logs, and persistent data such as your wallet and the blockchain. Take note of this.

- On Linux, BSD, and other Unix-like systems, the default is *$HOME/.dogecoin/*
- On Windows, the default is *%APPDATA%\Dogecoin*
- On Mac OS, the default is *$HOME/Library/Application Support/Dogecoin*

You can change these if you like, but you run the risk of confusion. If, however, you need the hard drive space elsewhere, changing this directory can be useful. The choice is yours.

After you have your node up and running, you'll have to wait a while to download the entire blockchain and store it on your disk. This can be a good time to start a new hobby or polish your skills with an existing one, such as reading a book, knitting a warm pair of socks, or baking a delicious pear galette. The secret to the latter is ginger and brown sugar.

You can also spend some time configuring your node.

For example, in the GUI, click on Settings then Options then Network. Click the box for "Allow incoming connections". Without this, your node will only receive data, never transmit. That can be useful to you, but it's not useful to the network. Be aware that you may have to do some work on your own network setup to finish the task, however: your node needs to be reachable from the public Internet on port 22556, and your machine needs a reliable IP address from your DHCP server.

Sometimes clicking the "Allow UPnP" box in the network tab in the Core GUI will fix this. Sometimes it won't. If this seems like Star Trek-style technobabble to you, that's fine. It's okay to stop at this point for a while; you can still *use* a node running on your own even if you don't or can't allow incoming connections. Ask a friend, do some research, learn the implications of what this means, and then decide if you want to continue.

Skim some of the other configuration options. They all have their uses. Other tips in this book will cover some of the most important. Do be aware that you will have to restart your node to take advantage of them, so pick a time when that's least disruptive.

Do make note of your configuration and data storage directory however. You'll use this a lot throughout the rest of the book.

Alternatives

You don't need a desktop computer or laptop. You could run a VM in the cloud. Various service providers such as Amazon AWS, Google GCP, Microsoft Azure, and Oracle Cloud offer free introductory packages as well as modestly-priced services, depending on your definition of "modest". You'll need some degree of system administration and automation skills to set up and run a node with these services, especially if you're sensitive to price caps and spending limits.

At the risk of referring to something that seems exciting (at the time of writing) but hasn't shipped a tangible result yet (at the time of writing), a do-it-yourself hardware project under the umbrella of very dot engineer[11] is attempting to assemble a known-working combination of hardware and software to run a full node with small form-factor computers, solar power, and goat-resistant networking[12].

[11] Seriously, see https://very.engineer/.

[12] Possibly other livestock too. Perhaps even birds.

If you're good with a soldering iron or flashing bootloaders or comparing spec sheets for all-in-one processor boards, check that project for more details.

Understand the Risks

Running a node has a few risks and costs.

First, it's a commitment of time and resources. Depending on the speed of your hardware and network connection, it could take a few days to download all of the blockchain. Depending on how well-connected your node is to other nodes, you could send dozens of gigabytes of data over your connection every month. Depending on your hardware, you could see a measurable increase in your power bill. Before you start running a node, consider what you can commit to now and measure carefully the effects (see **??**, pp. **??**); take care of yourself first before the network.

Second, remember that running a network service like a Dogecoin Core node means you'll receive traffic from all over the Internet and you'll send traffic all over the Internet. While the Internet is full of wonderful, selfless people like everyone running Core nodes, it has its share of malicious people (some deliberately, some unknowingly), so keep your security practices up to date. Invest time and effort into a good firewall, monitor for suspicious activity, and don't perform any actions you haven't researched and vetted.

Third, keep your node secure (**??**, pp. **??**, for example) and your wallet safe (**??**, pp. **??**, for example). You can do a lot of interesting things with a Core node, but many of those interesting things are a lot more interesting when only you can do them. Otherwise they're more scary than interesting.

Finally, remember to keep your node up to date. Core releases add new features, of course, but they also fix bugs, add more configuration options, and improve security. One important thread in development philosophy is to give users–people who run nodes like you are considering–the ability to shape the behavior of Dogecoin and its network. To do this responsibly, you need to understand what and how you can contribute to your vision of Dogecoin.

Tip #4 Set Your Node Comment

Look at the network peers tab in your core wallet (Help -> Debug -> Peers, in the Qt GUI as of 1.14.6). You'll see a list of other nodes you've connected to, along with like their IP addresses (IPv4 or IPv6), the amount of data sent and received, and more.

That additional data includes a node version string. By default[13], every Dogecoin Core reports itself as `Shibetoshi/version`. This applies whether you're running `dogecoind` as a headless network service or `dogecoin-qt` as a GUI application or some other way of running the Core that we haven't invented yet.

The Core allows you to *append* text to this version string to customize your node, amuse yourself, and, potentially, brighten someone's day through a feature called `uacomment`.

Setting `uacomment`

To customize this feature, you need to edit your *dogecoin.conf* configuration file and add an entry:

```
uacomment=My Cool Node
```

Of course, you can and should pick something more interesting than that. As it currently stands, you have 256 characters to play with here[14].

Save this file, then launch or restart your node. If you're running the Qt GUI, look at the debug information window under "User Agent" (Help -> Debug -> Information). The text you see there will be broadcast to other nodes whenever they connect to your node, so this is your chance to make a mark and do something meaningful.

Setting a *Secure* UA Comment

Of course, this is your node broadcasting something potentially specific to you, so remember that you're broadcasting it to an entire Internet full of people, some of whom may not have your best interests at heart.

When you're choosing what to say, keep at least two important points in mind:

- Be Kind
- Be Safe

If you're thinking of setting a comment that might hurt someone's feelings or cause drama like "Patrick is a curmudgeon![15]", think twice. There are human beings on the other side of the Internet, and our funny dog money only works if we all work together and cooperate.

If you're thinking of setting a comment that might reveal personal or private information about you or someone else, like "Proudly hosted at 1 Chome-2 Dōgenzaka, Shibuya City", think twice, then three and four times. Personal information can make you or anyone else a target. A visit in person or electronically from someone who doesn't have everyone's best interest at heart goes against the spirit of fun and collaboration.

What makes a *good* comment? If you follow those three rules, then anything goes.

You could use a dopey joke, like "Why does the Swedish Chef love Dogecoin? Because it goes bork bork bork!"

You could advertise your Doge-related project, like "Check out IfDogeThenWow!"

You could brag a little about your uptime, like "Proudly serving the network since 2013."

The sky is the limit, as long as you're smart about it.

---

[13] In the Dogecoin Core source code, look for a value called `CLIENT_NODE`.

[14] This is defined as `MAX_SUBVERSION_LENGTH` in the source code. Check the source for the version you're running to see if this has changed.

[15] He'll probably agree, so this is the worst thing I wanted to put in print.

Tip #5 Add a Wallet Address to a DNS Record

If you have a website, like https://ifdogethenwow.com/, how can people find you and send you a tip or a payment for something awesome you've done? You need some way to associate a Dogecoin address with an Internet property.

Fortunately, the Internet already has a well-understood mechanism to add this kind of metadata to a domain name. It's the same way any device can turn that domain name into an IP address or addresses to look up a web page, send email, make an SSH connection, or do other things: DNS, the domain name system.

At the inagural Dogecoin hackathon[16], Timothy Stebbing showed off this idea[17].

How to Add a TXT Record

Assume you have a website. How do you allow people to send you tips? First, generate a new, unique Dogecoin address (see **??**, pp. **??**, for example). Use the Dogecoin Core, `libdogecoin`, or a non-custodial wallet you trust. Keep the private key safe, as always.

Second, take a domain name you control. Maybe that's `mycoolsite.example` or something else. You've registered this through a domain registrar, and you have the ability to configure DNS.

Keep Your Domain Running!

If you didn't set up your own DNS, talk to the person or people who set it up for you. If you edit this on your own and make a mistake, you could break your website, email, or other network services associated with that domain.

Go into your DNS configuration and add a ‘TXT‘ record. This is an arbitrary text string associated with your domain name and propagated through the globally-accessible, cached DNS system. For now, stick with the defaults of *which* DNS entry, TTL settings, et cetera.

The contents of this new record should be the literal string ‘dogecoin:‘ (yes, include the colon) plus the new wallet address you generated, with no spaces in between. This conforms to BIP-21[18], so other applications can use this. Hold onto that thought.

Save the configuration and publish it. In a few minutes, when your DNS changes propagate to the rest of the Internet, you'll be ready to go. Anyone will be able to look up your domain name, find this new record, and do something with this wallet address.

Specific DNS Configuration Details See. . . Elsewhere

Looking for a visual explanation of how to do this? The process varies depending on your domain registrar and DNS host. Again, this has the potential to render your domain *temporarily* unusable, so be cautious, read the documentation, and ask for help if you need it.

Understand the Risks

At this point in the book, you should be a little bit wary of the words *anyone* and *something*, because publishing an address like this means anyone can do *anything* with both the address and the knowledge that the address is associated with the domain name.

[16] https://foundation.dogecoin.com/announcements/2022-09-08-dogeathon-downunder/

[17] . . . and, inadvertently, kicked off the writing of this book. Thanks, Timothy!

[18] See https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki

DNS entries often contain contact information for the person managing the domain. While many registrars allow domain security and privacy to hide your personal information, some don't. Before you add this record, check your DNS settings with a DNS lookup tool like the `whois` command-line tool or a reputable website. If you see your name, address, phone number, and email address available, then so do other people.

You *can* of course add this record even if you have contact information recorded, but be mindful of the fact that any transaction to and from this address can be associated with the domain contact permanently.

You should also be aware that DNS as originally designed has potential security flaws, including spoofing. If you're concerned about this potential flaw, make sure you've configured DNSSEC correctly. Otherwise, someone could pretend to be you, change this DNS information, and swap their Dogecoin address for yours.

Finally, if you fail to renew your domain or someone snatches it up out from under you, they can swap their address for yours in their configuration, in the same way that they could swap their website for yours if you lose control of your domain name.

What Can You Do With This?

Apart from being an interesting forehead-slapper of a trick[19], the value of this trick is in the integrations it enables.

Imagine a web browser extension that could show you a Dogecoin tip address whenever you visited, if the site owner had one configured. If you like the site, throw a few Doge their way, right from your browser or mobile device.

Or imagine we repurposed this record to point to a text file or structured data file on the server, served over HTTPS, that gave access information, purchase data, or other suggestions, like "If you send 10 Doge to this address, you'll get 24 hours of access" for a news or other subscription site.

Consider also an alternate approach. `ifdogethenwow.com` has, as of this writing, the main domain and a well-known[20] subdomain of `blog.ifdogethenwow.com`. Any and every subdomain could have its own TXT record with a unique wallet address. An enterprising shibe could start a site called `shibetips.whatever` and, for a small fee or out of the goodness of their hearts, allow users *optionally* to associate an address with a subdomain. As always, keep in mind the security and privacy implications, but think of the possibilities.

[19] In the sense that, "Oh yeah, you could totally do this!"

[20] Okay okay, hear me out though!

Tip #6 Take Actions on New Blocks

Dogecoin is an ever-increasing network of transactions. Each transaction occurs as a specific item in a block. Each block has a specific order in the blockchain, and each block comes in at a specific time. If you think of Dogecoin that way, you can see it as a network of transactions that's also a series of events.

This is a powerful idea you can use to do many interesting things. Consider payment processing. When a block gets mined, look for any payments to any address you're interested in, then do something based on the source address, the destination, the amount, any script in the transaction, or whatever.

You could also monitor the health of the network as a whole by looking at time between blocks, number of transactions in a block, number of coins transferred in a block, difficulty change in blocks over time, or any other piece of data available.

Dogecoin Core gives you options to treat these events as events, so you can do these interesting things and more.

Configuring `blocknotify`

Dogecoin inherited a Bitcoin feature called `blocknotify`. This is a configuration option available from the command line or set in *dogecoin.conf* which allows you to ask the Core to launch an external command whenever the Core processes a new block.

You can pass two options to this command: the *number* or *height* of the block and the *hash* of the block. With that, you have everything you need to do much, much more.

How do you make this work? First, configure your node for RPC. Be aware of the risks of doing so, and follow all the security guidelines to your degree of comfort and safety. Second, write a command that does something useful. Third, launch or re-launch your node.

That's it.

Processing a New Block

Let's start by writing a simple command that shows basic block statistics, such as the number of transactions, difficulty, time, and size of a block. This code re-uses some example code to perform authenticated RPC (see **??**, pp. **??**), so you can focus on the behavior:

```
#!/usr/bin/env perl

use v5.038;

use JSON;
use Path::Tiny;
use RPCAgent;

exit main( @ARGV );

sub main( $height, $hash ) {
    my $config = get_config( $ENV{CONFIG_FILE} );
    my $rpc    = create_rpc( $config );
    my $block  = $rpc->get_block_by_hash( $hash );

    my $stats  = analyze_block( $block );

    say <<~END_HERE;
    Found block $height
    Processed at $stats->{time}
    Contains $stats->{num_tx} transactions
    Size of $stats->{size}
```

```
    Difficulty of $stats->{difficulty}
    END_HERE

    return 0;
}

sub analyze_block( $block ) {
    my $num_tx = $block->{tx}->@*;
    my $time   = localtime $block->{time};

    return {
        num_tx     => $num_tx,
        time       => $time,
        difficulty => $block->{difficulty},
        size       => $block->{size},
    };
}

sub get_config( $config_file ) { ... }
sub create_rpc( $config )      { ... }
```

If you haven't read much Perl, don't fret at some of the syntactic details. The first handful of lines set the version of Perl (the latest stable release, as of this writing, to use a few newer features) and load a couple of useful modules, including `RPCAgent`, the secure authentication code. All *that* code does is wrap calls to the Core's RPC listener in a Perlish interface, so that the line `$rpc->get_block_by_hash( $hash )` doesn't have to manage the details of providing the right HTTP headers. Everything's already set up.

The interesting work is in `main()` and `analyze_block()`.

`main()` starts the action. It reads a configuration file (more on that in a moment) containing data to set up an object to perform RPC requests.

When this code runs, it gets two arguments, the height and hash of the new block. The RPC call gets all of the block's data with that hash. The Core returns that data as a JSON data structure, but the `RPCAgent` code turns that into a Perl data structure (a nested hash, or a dictionary as you might call it in Python, or a hashmap in Java, or an object in JavaScript, or…).

From there, `analyze_block()` extracts useful data from that data structure. Difficulty and size are obvious fields. The block's time is in a field named `time`, but that records seconds since the epoch, so the code uses Perl's `localtime` to turn that into a textual representation. Finally, the *number* of transactions is interesting, so the code uses the standard Perl idiom to access the items in the `tx` field (representing transactions) as a scalar value, representing the count of items in the array.

But I Don't Read Perl!

"Context? Scalar? That's unique! Also, wow, look at the sigils!" *De gustibus non est disputandum*, but your author *also* wrote a book called *Modern Perl* that explains all this stuff and more, so head over to http://modernperlbooks.com/ and download a free copy. You'll have one more interesting tool in your toolkit!

When `main()` gets the results, it prints the transction's description. It will look something like:

```
Found block 4485057
Processed at Wed Nov 23 15:15:28 2022
Contains 12 transactions
Size of 3979
Difficulty of 12696685.226509
```

To use this code yourself, you'll need to set up the configuration file and create an `RPCAgent` object. If you have much programming experience, you can probably imagine what they look like anyhow.

Launching Your Program

How does this get executed? Start by adding a line to your *dogecoin.conf* file to add this command. This shell wrapper sets up the execution environment correctly[21]:

```
#!/bin/bash

cd "${HOME}/dogecoin-tricks-book/"
export CONFIG_FILE="./chapter_3/env.json"

perl -Ilib bin/show_block_stats.pl $*
```

This file sets the path to the RPC credentials configuration file and makes sure Perl where to find the `RPCAgent` module. All that's left is to add a single line to *dogecoin.conf*[22]:

```
blocknotify=/bin/bash \
  "${BOOK_HOME}/chapter_3/bin/launch_listener.sh" \
  "%i" "%s" >> blocks.log
```

If you've set up a `cron` job, this will also look familiar.

The `%i` parameter is the height of the new block and the `%s` parameter is the hash of the new block. Whenever the Core processes a new block, it'll launch a system process and pass those two values as variables. In this case, the system will invoke the `bash` shell which will itself invoke `perl`.

As with other changes to *dogecoin.conf*, you'll have to restart your node before the Core will start executing this command.

Understand the Risks

Any time you use RPC against a local Core, you should enable authentication and authorization (see **??**, pp. **??**) and keep unauthorized activity away from your node. This is especially true if you have a wallet attached to your node. Anyone who can connect to your node via RPC can get a lot of data you might not want to expose and take network actions you might not want to permit.

Second, be aware that you could write buggy code[23]. The abstraction here of a shell script calling another program lets you test your RPC code in isolation by calling it with block heights and hashes you already have on hand. That leaves you free to test the *dogecoin.conf* integration later, rather than having to do both at once–especially as any change to the *dogecoin.conf* code means relaunching your node.

If you do get your program into a crash loop, you might harm the stability of the system and the experience for you and others on the machine. Don't let that stop you from doing interesting things but do be careful to think about what could go wrong.

What Can You Do With This?

Perhaps you can light up a lava lamp whenever you make or receive a transaction.

You could update a counter on a webpage or post a message to Slack or Discord (see **??**, pp. **??**) with every new block mined.

You could plot the size and fullness of blocks over time.

You could keep a list of the busiest wallets over the past day, week, or month.

What interesting ideas do *you* have now? Be creative!

---

[21] As you might do for a `cron` command.

[22] Linebreaks are for formatting; they're not needed.

[23] Your author's first few attempts had bugs!

Tip #7 Follow Core Development

How do you manage a global network where anyone can participate and the success or failure of your personal financial transactions relies on your ability to trust the work of countless strangers?

You have to trust that everyone's playing by the same rules. To do that, you have to verify that you understand the rules and that someone's not trying to sneak something past you.

In the world of Dogecoin and cryptocurrency, this means that we must:

- follow rigorous, tested, and well-understood mathematical principles and formulas
- adhere to well-defined rules and parameters for participation
- trust the software we use to implement all of these things correctly

That's one of the reasons the Dogecoin Core is so important. The source code is available, the released built from it are done so transparently, and development and organization occurs in the open.

Where is the Core?

At the time of this writing, the source code is available on a site called GitHub, at https://github.com/dogecoin/dogecoin[24].

This website and the `dogecoin/dogecoin` project in particular allow people to collaborate in the development process, whether submitting bugs, asking for new features, improving documentation, translating text into multiple languages, adding new features, or having design discussions.

This is a good place to report a potential bug or ask for a new feature.

Productive Discussions and Contributions

While anyone *can* participate, please be respectful of the time and effort of countless other people. Creating an issue saying "Devs please make price go up!!!" won't help anyone, because Core development deliberately resists any activity that could influence the price.

Navigating the Core

If you look at that page on its own, you may find it a little overwhelming at first (so many features!) and a little underwhelming (wait, where's all the activity?). To navigate all of this, you need to understand a little bit about Git and GitHub.

Git is a software tool that lets developers manage the source code: the instructions on what the Dogecoin Core does and how it does it. All of this source code is stored in a Git *repository*.

GitHub is a website that lets multiple developers share their Git repositories with each other. While Git itself is a distributed system that allows people to collaborate, it doesn't inherently enforce any "official" prime repository. GitHub does; only those people allowed to make changes in the `dogecoin/dogecoin` repository can change the code that makes up the software that you're running.

[24] Though it's unlikely this will change any time soon, if you're reading this in 2099 or even 2029, check to see if this is still true before blindly downloading anything.

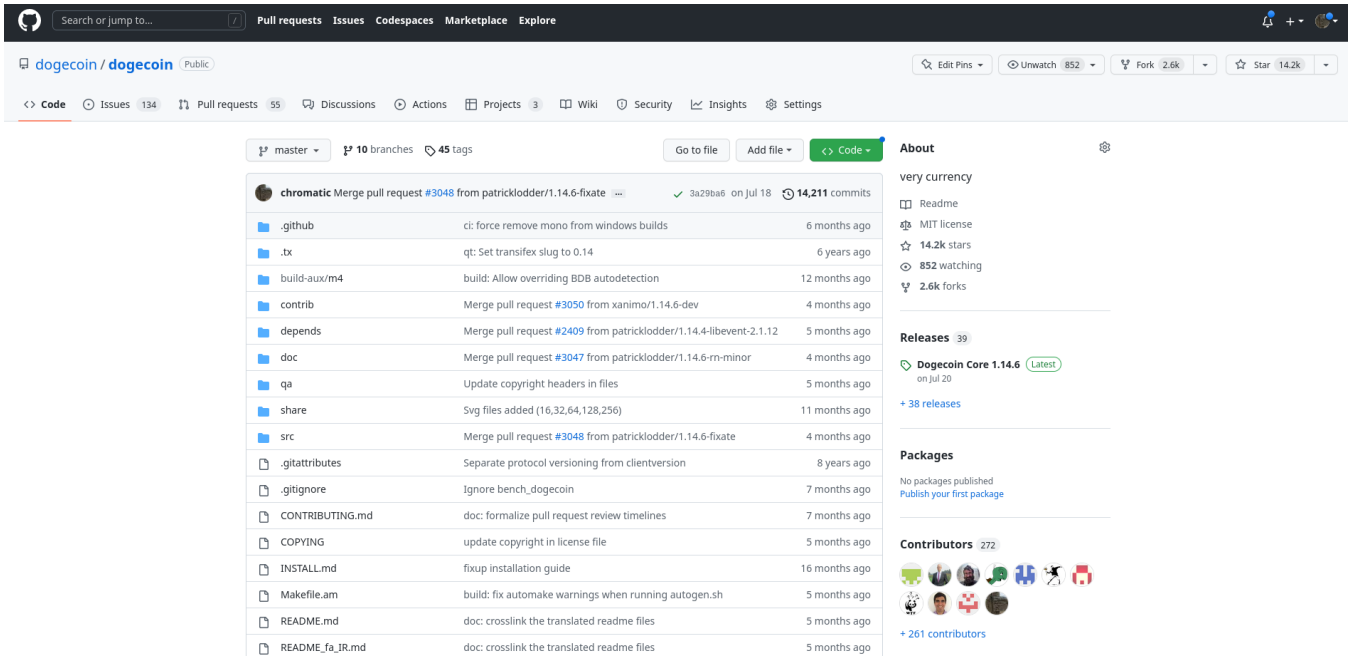

Figure 1: Dogecoin Core GitHub

However, anyone can *access* GitHub and create (or *fork*) their own repository to make their own changes. Then, if they choose, they can request other people *pull* those changes into their own repositories (or even the main repository itself, `dogecoin/dogecoin`).

That forking metaphor is really important.

If you think of everyone's individual repository as a fork off of the main repository, you get the idea of a tree or a river or some sort of organic entity. This holds true in the main repository itself, but rather than forks it holds *branches*.

In the same way that a fork represents something that's different from the main repository, so branches can be different.

In the same way that a fork can produce a pull request (to merge changes back into the main repository), so can branches.

If you've never used Git or GitHub before, this isn't always obvious. However, now that you know, you know what to look for.

So where does development happen?

Dogecoin Development Process

If you only ever looked at the main branch (called `master`), you'd think not much ever happened. That's because all development occurs on other branches.

In Dogecoin Core Development Branch, pp. 16, you can see a little widget saying `1.14.7-dev`. This switches the web view to show a branch other than the `main` development branch. This particular version number names a branch that represents “the code that will be released as version 1.14.7”.

You can switch that view yourself to see what *was* released in previous versions and what *will be* released in future versions. As of this writing, there are two future versions in active development, 1.14.7 and 1.21.

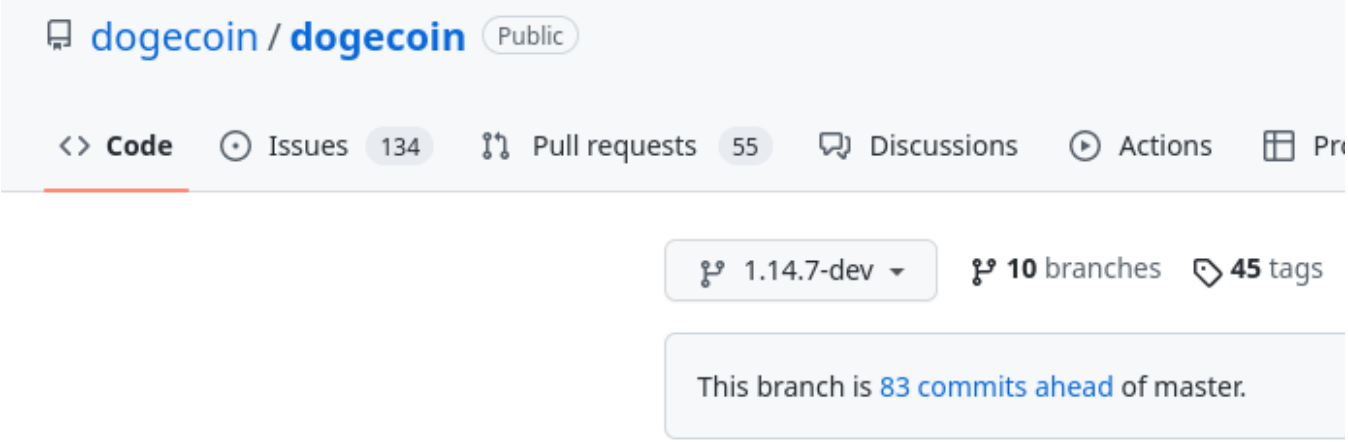

Figure 2: Dogecoin Core Development Branch

In this image, you can also see links for Issues and Pull Requests. The former is bugs or feature requests and the latter is code that's in progress. Pull requests always point to a specific branch, so that the developers can keep track of what they intend to merge where and when.

At any point, you can use the Code link, or the list of files and directories, to search through the code as it existed at any point in time (or, in the case of pull requests, code as it would exist if the pull request were merged to a branch).

This is really powerful stuff, even if you're not a developer.

What Can You Do With This?

There's a lot more you can do with GitHub and the code, even if you're not a developer. Feel free to click around, read issues, think about discussions, and look at pull requests. Contribution is open to anyone willing to put in the time and effort to help their fellow shibes.

As always, be respectful, understand the rules, and treat other people with respect. Remember that this tool is essential to the Core developers, so please be mindful of their time and resources.

Tip #8 Generate a QR Code

If you're taking payments in Dogecoin in some kind of in-person setting, you probably want an easy way to allow people to send you payments without talking through an entire wallet address out loud[25].

This is even more important when you want to remove some degree of human intervention from the process. Suppose you have a pinball arcade and you want to allow people to pay with quarters *or* with Dogecoin. What can you do to accept payments with as little friction as possible?

> **Payments, Part One**
>
> There are other logistical problems to overcome with the arcade example, but those are the subjects of other tips. Keep reading!

A Bitcoin Improvement Proposal has the answer.

Dogecoin-Aware Wallet Links

As defined in BIP-21[26], a hyperlink starting with `bitcoin:` and followed by an address tells your computer, phone, tablet, or other device to open some sort of wallet application and prepare to send a transaction to the given address.

Dogecoin adopted this obviously good idea.

If you are, for example, selling electronic copies of a book about Dogecoin online, you might create BIP-21 address like `dogecoin:DAY5wNkebzEyqUXCkN9koKNBuzXRKRTjcL` to receive funds. Then if someone clicks on that link on a device that understands that format, their wallet will pop up and allow them to decide whether to complete the transaction.

That works pretty well on a web page, but if you're in this amazing Doge-aware pinball arcade, do you really want to pull up a webpage and search for the right link when you turn the corner and see the incredibly rare Baby Pac-Man! machine you weren't sure if you imagined in a dream?

Wouldn't it be more fun to pull out your phone, snap a pic, and then start playing?

Dogecoin-Aware Images

You're probably already familiar with QR codes, the three-dimensional barcode looking images that contain arbitrary data and take you to real estate listings or menus in restaurants that don't want to hand out grubby menus. Why not make your own?

Okay, you've probably also already seen that the Dogecoin Core creates both a BIP-21 address and a QR code when you ask it to produce a receiving address. That's not always 100% convenient (What if you're using an offline wallet? What if you don't have the Core available? What if you don't have the Core GUI running?)

It's easy enough to make your own QR code with your own address.

While there are multiple websites that purport to generate QR codes given a link, you shouldn't have to trust a third-party to do it right. Yes, you can and should always double-check with your own phone or device that the code contains the link you want, but it's important to know how to work these tools on your own as well.

You can generate QR codes from lots of programming languages and tools. One good tool is the Python module `qrcode`[27]. This module includes a command-line utility called `qr` that makes generating images easy:

```
$ qr 'dogecoin:DAY5wNkebzEyqUXCkN9koKNBuzXRKRTjcL'` \
  > book-address-qr.png
```

[25] "That's D as in Doge, K as in Kabosu, no I don't want to say Donkey Kong, just trust me."

[26] See https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki.

[27] See https://pypi.org/project/qrcode/ for more details.

This will write a PNG file as output containing the QR code encoding the address. If your terminal supports image output, you can test the command by skipping the file redirection and pointing your phone at the screen, as shown in in Generate a QR Code from the Command Line, pp. 18.

Figure 3: Generate a QR Code from the Command Line

Easy enough?

Install Python and qrcode Simply

IF you already have Python 3 installed, use `python3 -m pip install qrcode`. If you don't have Python 3 installed, I've created an installer at https://platform.activestate.com/chromatic/Python-QR-Codes for Linux, Mac, and Windows systems.

What Can You Do With This?

You can take payments from a variety of devices with reduced human interaction.

Understand the Risks

We've already discussed the risk of using someone else's QR code generator to generate your QR codes. Always check and double-check that the embedded link is a BIP-21 link that contains your address.

If you do print out these QR codes and put them on your pinball machines, be sure to do so in a way that doesn't hurt the paint. Attach them to glass or metal surfaces. You'll thank yourself later.

Printing out QR codes does run the risk of re-using addresses. While it's much, much safer to generate new addresses for each transaction, you may find this risk is worthwhile to reduce the need for human interaction for every pinball play. If you do go this route, rotate your addresses and generate new QR codes frequently (weekly? monthly?).

Finally, be aware that a payment system that relies on people scanning codes with their phones is susceptible to the risk that someone may print out their own QR codes and stick them over the top of your own codes, sort of like an ATM skimmer. Watch your pinball machines carefully.

Tip #9 Practice a New Language

Some people think English is the language of the Internet. Someday they'll discover the language of the Internet is becoming memes. Until then, it's up to all of us to find ways to communicate with other people in ways they understand.

The Dogecoin network communicates in terms of numbers: big numbers, small numbers, all bundled together in blocks and transactions and addresses and Dogecoin and fractions of Dogecoin. We humans bring meaning to those numbers by agreeing on what they mean and presenting them in ways we can understand: labels, QR codes, addresses, secret keys, et cetera.

When it comes to the Dogecoin Core itself, we do the same thing. Whether you're a goat farmer in western Canada, a small business investor in eastern Africa, a polar ice researcher in Antarctica, or a student studying European finance in Austria, you balance two important things when you use the Core: what the underlying information *means* and how you *prefer* to consume that information.

In other words, without all this highfalutin' talk, you should be able to read use the Core in any written language you prefer[28]. Good news: more and more people have that option!

Language Setting

If run the `dogecoin-qt` program and ask for its help output, you'll see an option called `--lang`:

```
$ dogecoin-qt --help
...
UI Options:

-lang=<lang>
     Set language, for example "de_DE" (default: system locale)
```

If you've configured your computer to use German spoken in Germany as your default language, the Core should do the right thing *if* the developers have provided the appropriate translations. You should have to do nothing special with Dogecoin for the software to behave normally.

Suppose instead you're about to go on a trip to Portugal and Brazil and want to brush up on your Portugese, so you'd like to manage your transactions in one or both languages. How do you figure this out?

How Language Codes Work

The example code, `de_DE`, has two parts. The first is `de`, which represents Deutsch, the German language. For more details, see ISO-639-1[29]. The second part is `DE`, which represents the country of Germany, known as Deutschland to its inhabitants. For more details, See ISO-3166-1[30]. The combination of language and country code means “German, as spoken in Germany”. Similarly, `en_US` refers to “English spoken in the United States of America” and `en_GB` refers to “English as attempted by the residents of that island we defeated back in the 18th century”[31].

This is the ideal behavior, anyhow. If there's no specific UK variant of English or Canadian variant of English, it's okay for the translation system to fall back to generic English. Similarly, if there's a Portugese in Portugal language code `pt_PT` but no Brazilian Portugese `pt_BR`, it's okay to fall back to Portugese, even though you'll probably embarrass yourself with country-specific slang.

How do you know what the Core actually supports? Look at the Core's source code[32] to find a list of all current translation files. Look for your language code and any specific country code. If there's a file present, you can use that language. Otherwise, you'll have to choose something else.

---

[28] Your author recognizes the irony of writing this whole book in English.

[29] Wikipedia explains this at https://en.wikipedia.org/wiki/ISO_639-1

[30] Wikipedia also explains this at https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

[31] “You'll Be Back” not withstanding.

[32] See https://github.com/dogecoin/dogecoin/tree/master/src/qt/locale

Understand the Risks

The biggest risk you face with this technique is fleeting disappointment. Test this with the `--help` command to `dogecoin-qt`:

```
$ dogecoin-qt --help --lang=af_ZA
Gebruikerkoppelvlakopsies:

...

-lang=<lang>
     Set language, for example "de_DE" (default: system locale)

-resetguisettings
     Alle instellings wat in die grafiese gebruikerkoppelvlak gewysig is,
     terugstel
```

You may find, as in this case, that only some of the text has translations. Alternately, you may have no translations for your language and/or country. Think of this as an opportunity, however. This could be your chance to contribute to the Core and help countless other shibes and potential shibes use Dogecoin in their own preferred languages!

The other risk is that you inadvertently set a default language to something you don't understand well enough to disable[33]. If this happens to you, examine your configuration file for the `lang` setting and change it to your preferred locale setting. Be aware that, as of 1.14.6, using the `--help` command does not process your configuration file but instead will use your system's default locale setting.

[33] Your author does not admit to setting the default language on a fleet of French laser printers to Bulgarian from the United States on accident once upon a time.

Tip #10 Decode Transactions

If you find yourself doing something complicated more than once, think about how you can avoid repeating yourself–especially if the complicated steps are easy to get wrong. This could be anything from transcribing one set of data between systems or copying and pasting information from multiple processes, windows, or machines.

If you read the previous tip (**??**, pp. **??**) closely, you noticed that the Core as of this writing provides no direct way to go from a transaction hash to the decoded transaction, or a list of inputs or outputs, or the scripts for outputs, or anything else.

You can do the two- or three-step shuffle to get this data, but why do that more than once?

Chaining RPC Commands

Another tip discusses RPC command stacking (**??**, pp. **??**), where you feed the output of one RPC call into that of another. This allows you to act as if the Core supported only one call to perform a bunch of behavior, such as decoding a transaction into a data structure given only the transaction's hash.

If you were to do this manually, you'd have to call `getrawtransaction` with the hash, then feed the result of that into a call to `decoderawtransaction`. In the ideal case, this works perfectly.

If either RPC call fails, you need to account for that failure. That code will look something like this *pseudocode*[34]:

```
def decode_tx_from_hash( tx_hash ):
    tx := RPCcall 'getrawtransaction', tx_hash

    if tx.error != nil:
      throw tx.error

    decoded_tx = RPCcall 'decoderawtransaction', tx.result
    if decoded_tx.error != nil:
      throw tx.error

    return decoded_tx.result
```

You can write this in any language you like. Alternately, install the `dogeutils` library and toolkit, which does this for you.

Understand the Risks

The risks here are minimal. Adding extra calls through another language will add latency to the process, though you can add whatever error checking or convenience features you like if you provide your own implementation. This can be especially useful if you wrap calls which take optional parameters.

The greater risk is long-term maintenance. If the RPC calls you use change between releases, your wrapper will have to adapt to them. If the Core adds a call with the same name as yours, your code will continue to work, but you may have a divergence between your code and the Core's implementation, and you'll have to consider the maintenance costs of doing something different from the Core.

Relying on anyone else's wrapper, especially if it performs authentication, means giving up some control to someone else's code. Audit it yourself (or have someone you trust audit it very carefully for you) before using it.

[34] Did…did you just invent a weird new notation for this? Yep!

Tip #11 Calculate Dogecoin's Maximum Market Stats

In 2013, your author received his first Dogecoin tip, probably for saying something insightful or clever on a social media forum. It was worth a fraction of a penny but it meant something–it gave a warm glow that your author brought a little bit of value into the world.

In 2021, that tip was worth hundreds of dollars. At the time of this writing, it's worth tens of dollars. Who knows where it will be in a week, a year, or ten years?

Some people care a lot about that. If you're using Dogecoin to buy a meal or donate to pay for medical costs of a young child with illness or to support a friend on the opposite side of the world, the fluctuations over time aren't important. If, on the other hand, you see an asset that may appreciate over time, you owe it to yourself to do the math to figure out what *could* happen, if fortunate favored you.

Count All the Doge in Existence

How many Dogecoin are in the world? To answer that question, use any reputable block explorer (or ask a search engine to find one). As of this writing in April 2023, multiple sources suggest there are about 139 billion Dogecoin in the world. This number grows by ten thousand with every block minuted, which happens about once per minute, so, on average, you can expect this number to grow by 5.26 billion Dogecoin every year.

As of the current writing, there are about 7.96 billion people in the world.

If you do the math, there are, at most, about 17.5 Dogecoin available for every person in the world. Maybe you have a few more. Most people have fewer. If the population stayed flat at 7.96 billion, everyone could get 1.5 more Dogecoin in the next year.

That's not a lot of coins to swap between people if you want to buy cool art or delicious beverages, or pay for someone to write you a fun computer program or reward someone for answering a difficult question. Let's think of Dogecoin in a different way.

Count All the Value in Existence

If you had 100 Dogecoin and sold them at the highest possible valuation Dogecoin has reached (as of this writing), you'd have made about $70 USD. If you had 1 Bitcoin and sold it at the highest possible valuation it has reached (as of this writing), you'd have made about $67,500 USD. That's a big difference between the two currencies.

Part of the difference is that there can only ever be 21 million (that's million with an m, not billion with a b) Bitcoin. Right now, if all of the Bitcoin that could ever exist existed, there'd be 6,620 Dogecoin for every 1 Bitcoin. If Dogecoin had the same market capitalization as Bitcoin at its all time high, that $67,500 per Bitcoin would be equivalent to about $10.20 per Dogecoin. (The math goes like this: 139 billion Dogecoin divided by 21 million Bitcoin gives 6,620 Dogecoin per Bitcoin. $67,500 divided by 6,620 gives $10.20.)

What if there were no Bitcoin though? What if all other cryptocurrencies and currencies in the world vanished, and there were only Dogecoin?

The World Bank estimates the total Gross World Product–the value in dollars of all of the economic transactions of all of the countries of the world–at $96.5 trillion (that's trillion with a t) US Dollars at the end of 2021. Let's subtract a year of Dogecoin from the current number of coins to give 133.74 billion.

If *every one of those dollars worth of GWP turned into value in Dogecoin*, each Dogecoin would be worth $721.55 USD.

Understand the Risks

If you're investing in Dogecoin, hoping it reaches $10,000 per coin so you can buy a nice island and shoot rockets at Mars, you're going to have to wait a while longer.

These calculations are very rough. They don't take into account things like Dogecoin that were sent to addresses with forgotten keys, or addresses that don't exist. They don't account for changes in mining strength where there may be more or fewer blocks mined than we expect every 60 seconds.

What they *do* accomplish is to set expectations. The entire world economy may never switch to Dogecoin as a world currency in the lifetime of anyone reading this book. All 8 or 10 or 20 billion people in the world may never get their average of 17.5

Dogecoin per person. Your wallet of 100 Dogecoin may never be worth $6,750,000 and you may not be able to sell it and buy the goat farm you've always dreamed about nestled between the wheat fields and mountains of Alberta, Canada.

Dogecoin can still be fun and useful, though. In fact, if anything, realizing that a billion-dollar moonshot is mathematically and economically impossible frees up this goofy cryptocurrency to do interesting and quirky and amusing things. It doesn't have to be the buttoned-up, why-so-serious currency of Very Serious People who do nothing more than look at charts and talk up their bags.

It's free to be the currency of people who enjoy fun and creativity and doing useful things.

Have fun. Be creative. Do useful things.

Index