Commonplace •

Community and culture-centered code

Henry Zhu, Sarah Kearns

Published on: Jul 15, 2021

DOI: 10.21428/6ffd8432.819aab48

License: [Creative Commons Attribution 4.0 International License \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/)

Introduction

In this new and experimental podcast, KFG acquisitions editor Sarah Kearns talks with Henry Zhu, who maintains Babel and hosts his own podcast. They highlight the importance of people-centered code and online communities rather than metric-motivated ones, stress that maintaining code is just as (if not more) important than adding features, and open source is composed of many different cultures and practices.

Take a listen here:

Music composed by Dawit Tegbaru

Transcript

(Automatically transcribed by otter.ai, lightly edited for clarity)

Sarah Kearns (SK): *Welcome all to the first ever Commonplace podcast recording. We are here with Henry Zhu who stewards and maintains [Babel](#) and then hosts the podcast [‘Hope in Source.’](#) So with that, welcome. Maybe we could start off with you introducing yourself a little bit more. And then you can share sort of how you got into open source and open access.*

Henry Zhu (HZ): Cool. So I guess I'm in New York. I started off as a software engineer. And then just three years ago, I left my job at Adobe to work on Babel full time. And I guess there's even a backstory to that too. In short, I actually got my job at Adobe, even through open source, too. So I was kind of doing that for fun when I was working, and then they kind of reached out send an email, saying: “Hey, would you want to work with us.” I think it was relatively rare to get a job through open source, but I think it was five years ago, got me interested in joining, because then I was like, “oh, maybe I'll work on it [open source] work more.” And then my team and my boss were nice enough to convince the company that I should get paid for half time to do open source, which is still rare.

And then eventually, I guess, I had this conflict, internal conflict of, well, I still had to spend half my time doing work. And then maybe that's not what I'm into. And then it also felt weird, because everyone else in the company was focused on, obviously, the products, and I'm also doing open source. So then it got to this point where I'm like, "I think it might be better, just like mentally to focus on open source entirely, or just go back to work and just do source for fun on the side." But, you know, obviously, I chose to do this full time. And now I'm at this point where I'm trying to figure out how we do funding and the future of this project, and in my role in it as well. So yeah, I'm in this place where I don't know, what's next either, which is could be fun, too. But yeah, I guess, for me, too, is it can be scary.

Sustainable Infrastructure in OA

SK: Yeah, we think about this too a lot at the Knowledge Futures Group about how to have sustainable funding or regenerative practices in these spaces and having secure, and long term funding for these things is, maybe not hard to come by, but kind of! Having the infrastructure to have something that lasts when people come and go does seem to be a challenge.

ZH: And it takes someone like me to have to do that. That's not available for a lot of people in terms of like finances. The uniqueness of where I'm at and all that I just happened to work out. I don't think that someone could replicate, replicate what I did.

People want to be able to just quote unquote, write code, and then get paid for it. I don't even know if that's the right ideal anyway. But I think that's how you think about when you're going into as an engineer, right, the thing you're doing his code and what I realized over the years was that the thing I appreciate the most about open source isn't even the code. And then also, I realized that was a lack in a lot of our projects. And so I kind of stepped up to do that work, you know, fundraising, or rolling all these things, and it's a real job, it's a full time thing.

The more you write code, that doesn't necessarily lead to more money, if anything might even lead to less because people finding you might think that you're okay with wherever the project is at, right? If, because they're not going to go out of their way to wonder, "Oh, where's the money coming from?" They just assume you figured it out. It's like, of course, we have to tell them that we need money. So that's why we [Babel] had that [blog posts](#) like I guess a month ago and then it actually helped get a lot of funding. Even though I don't like being the kind of person that like tries to be like the influencer and

hashtag we need funding like every day or every week. I don't know how to make that more organic.

People-based and well-maintained codebases

For me, it's more like, I want to build a relationship with people meet them in person, like that kind of thing that is appealing to me because it's just like a real way of knowing people rather than just trying to get their money. That's not a sustainable way. Cuz, you know, if they're not actually getting any value then, why donate? It's a one time thing. And that's exactly it's not gonna last.

SK: Yeah, I really like that like having it be community focused and like organic partnerships that make the projects last because like that having that like inroads with people.

HZ: Yeah, like you're working closely with them. And you can make sure that you're both getting something out of it. And to that the products are, like, interesting to people who would be using them, but that's hard to win, like your team is small or, I think the sustainability of code and things are important, but also the sustainability of the people who do the stuff behind it, you know, people are so overburdened. And I think people are resources, and their time is critical. That's why we like to say that code is the easy part. But in the end, if you don't have any people working on the code, it's, well, we call it dead code. It's like, it still works, it still runs. But all code has bugs, all code. If you're going to use it in a way that's changing, like the use cases change, so people are going to want more features. Even if they don't need a feature, it still needs to be maintained regardless.

Part of me wants to argue that you should get paid just to maintain the project, even if you don't make new features. I think it's easy to get into this mode of adding features, just because that's what products do. But Open Source doesn't have to copy the industry. Especially if it's infrastructure level, you know, we've seen with a pandemic, all this infrastructure is failing us. Open source is also infrastructure. And you know, the more features you add, the more bugs. And so it's there's this trade off: you add all these features, but in my mind it feels like a Frankenstein sort of thing where you kind of just tack on things, then the whole core of it seems really hard to maintain. Maybe it's good enough to just leave it as is. It's sort of like, less is better, in a way. We always want to put like more money, more people more whatever into this problem. Or maybe the problems aren't because of those things. Escalation essentially, like adding more rather than, like

keeping as is. But maybe no one wants to pay money for that, right? Because we need it to like be better or something.

SK: Yeah, I feel like that definitely rings true with the stuff that we're thinking about, especially in terms of having public digital infrastructures, right? Because things need to be paid for, but it's the infrastructure, not some new tool that you're paying for. It's not some new, like, fancy feature.

So I guess that makes me wonder what you think the role of a maintainer to guide towards infrastructures and practices rather than like features and stuff, if that's, if you're seeing sort of that those types of conversations happening in open source and tech, in general, sort of like how to shift that focus to infrastructure?

ZH: Yeah, I think this is something that maybe some of this is at the core of a lot of the problems within, like, say JavaScript itself, or, you know, we're so focused on moving forward, maybe even referencing like Facebook's old, motto, “move fast and break things.” I mean, they've changed that, which is funny. Especially in JavaScript, so it's really fast in terms of always having new ideas, which is great. But that means a lot, there's a lot of churn, there's a lot of versions that you have to upgrade. I think I was talking to another one of my podcast guests. And he mentioned this the cult of newness, where it's like everything new is we have to move on to the next thing, the next tool that replaces the other tool.

And, you know, most of our code [at Babel] is legacy. And of course, it'd be great to start over. But just like in real life you can't really do that in most cases. Or it's like a privilege to do that, where you make a whole new project, you can start over. But most of these other projects, I don't think anyone's going to want to spend time to rewrite their whole code base in a new language or a new framework.

So like, how do we get Yeah, the questions like how do you kind of let people experiment with the new ideas that will be the potential futures, what keeping what we have in the past, like tradition or not, because it's better but because that's the way to move forward. And we need to keep what we've had before. Like things will last for a long time. If people don't like using will still be there, right? Or at least whenever any kind of technology, not just coding.

Versioning and metrics of success

Yeah, so it's like interesting cultural change. I think a lot of is looking at the past and then seeing how people that have been doing it for a long time do things. So I think about upsetting open source projects have been around for like decades. It's funny saying a decade feels like eternity in JavaScript land, because JavaScript itself is only like 20 years old. And most of these projects maybe last for like one or two years. And you move on to the next thing. There's a project called LaTeX, it's like the thing you use for writing papers, right. And it was made by Donald Knuth. And he's like a famous programmer. But the funny thing about his versioning system: his versioning system is based off of pi. So it'd start with three, then 3.1. So interesting enough, like, you know, it goes to infinity. But it's saying that over time, it gets less and less, you know. For us, Babel would be version six, version seven, version eight, and new, each called a major version, which could have breaking changes. But for that, it's like almost like ossifying over time, where we realized that this thing doesn't need features anymore. If anything will change and less and less over time. And to the extreme of he said, “when I die, you should change the version to the symbol pi,” instead of like 3.14. Which is like just so cool to just think about it in that way.

We're always focused on, “Oh, it's version 100 now,” and you know, that's like a marketing thing to get people to like upgrade or switch. And this [alternative versioning system] is more like, “hey, this thing is going to last for a long time. It's okay, if people make a totally different thing. But for our thing, we know what we want. And we're going to stick with that.” I think that's really cool. Because I feel like this tug of needing to rewrite everything or make everything faster, just because someone else did it. And maybe that's where it's like passing the baton of like, maybe it's Babel's time, it's not over. I don't need to try to become everything. Seeing what place you are in the ecosystem like for in our case, Babel is ubiquitous, it's being used by like every, you know, website, I guess, in some sense, even you might not even know it, right infrastructure. But that doesn't mean because some other tool came out that's being written in a language that's “faster,” that we need to go ahead and spend a year rewriting it just seems like a waste of time for us, and also, maybe ultimately bad for the ecosystem. But I could go on.

SK: *What sort of other lessons are there beyond versioning? Are there sort of like — in this LaTeX model of, even though it's not being like, super updated, in the sense that a lot of*

other code bases are, but it's still like propagating and being used? Like, I guess, sort of like, while they're left to be seen in like, that sort of analysis of more traditional open access open source?

HZ: Yeah, I guess, unfortunately, I haven't spent that much time in that. That was just like one that someone else told me. But you could think about protocols as well. So there's HTML and, or HTTP, like, those are all really old. And even JavaScript itself, everyone knows it's not a good language, because it was designed in 10 days. And so it's like, it doesn't matter. And it's not something like an insult to Brendan Eich, the creator. It's more just like, it doesn't matter how smart you are, if you only do so much in that period of time. And it's a success in the sense that, you know, we've still been working on it through this committee called TC39. And it's the most widely used language in the whole world.

But, and there's that whole phrase versus better, where you know how much do you need something to be perfected, versus just taking the practical steps and make it work. I think in most things that plays out where it's like, the thing that was like, “better,” doesn't really win out in terms of like, popularity or usage. And we just have to deal with the fact that, there's going to be trade offs.

Different coding cultures and practices

In terms of other lessons, another project I think about is SQL Lite. It's a database and it's also relatively old in terms of how long people have been using it. And they do something interesting called, or they don't do what we call open collaboration. One of the problems with problems with GitHub is that everyone assumes that everyone does everything the same way, just because it's on GitHub. And maybe, especially if you're coming from a certain language community, it's almost like every language has their own culture or way of doing things. The JavaScript community, tries to be very inclusive and welcoming, not that other people aren't, but almost to a fault, where we try so hard to get people to help us. We care about like, our logos, and emojis and like that kind of stuff, right. And that's not bad. It's just the culture. And I think, for them, they're not open collaboration in the sense where the source code is open, like, the code itself, is free to use and access however people want, but they don't actually accept people to write code for them. So they'll accept your ideas, but they're the ones that want to implement. And that's just how

they that's their policy. That's how they do things. And that's, that's just different. And their reasoning is, well, they've been working on it for like, you know, decades, maybe one, two or three people. They want to keep their code, And they want to make sure that the random person that sent their poor question didn't take code from someone else. So they in order to one way of making sure of that is just to say that I'm they write the code all of it.

That's just different, a different way of doing things. And I think that it doesn't mean that they're not welcoming people. Obviously, they want to get your feedback, but maybe they just don't want to accept code. And I think being able to see that upfront is very interesting, because on say, GitHub, you go to a random repository, you just assume things are as they are. Because maybe I think maybe because the UI looks the same, the layout of everything, you just assume the way people do things is the same.

I've used this analogy of cities in the past of like, or countries. Each repo or org is sort of like that. But the problem is that, because it's online, it's not physical, you might not see the people behind it, you don't know what they use to organize themselves versus when you go to a different country, it's pretty obvious, you know, you get off the plane, or boat or car or whatever, and people are different, they act differently. You, you might even ahead of time find the travel guide that tells you like kind of culture or watch a YouTube video or something. But in GitHub, they don't really have that in the same sense, or people aren't expecting that. So then that becomes the conflict between the the ideals or desires of the maintainers and the community. And then new people that come in, like how do you kind of on board not for code, but for culture? in a way that's respectful for everyone?

SK: Yeah, I like that. And I really thought about how those different types of cultures are sort of, like hidden behind, like a more broad interface, I guess, let me see wonder how different, like open source and open access communities like coding maybe should interact, you know, like to try and if you're having like this, like bigger movement of making more open source or making more like public infrastructures, like how, how can areas? Is it? Have you seen it be possible? Or how in ways Have you seen it be possible to sort of overcome those, like, cultural barriers?

HZ: One thing I think of is just the fact that like, it was possible, to differentiate your projects, in terms of the look, you know, I was thinking more of like MySpace sort of thing where you can change things (maybe I' m dating myself there).

In some sense, it's the point of GitHub to standardize, right. And that's a good thing because it makes it so that anyone can be familiar with how things work when you want

to join a project. But the problem is, essentially that because it looks the same, everyone assumes that and I think ultimately, that's worse than allowing people to express themselves in and kind of upfront show who people are, what the project is about, and stuff like that. For example, when you go somewhere, like the mall or whatever, you don't have to read some pamphlet to figure out what it's about. It would be better if you had welcomers or people that greet you, and you can talk to them and ask for information, that kind of thing.

Maybe in tech, we might be like, oh, what if we made a bot that like, you know, automate and that's just like, I feel like the wrong people to like doing something like this. Of course you should have documentation and you could have a bot that has a question and answer thing. But, ultimately, if you really want to get to know someone, you kind of have to be on a personal basis. Of course, that doesn't scale.

But maybe it shouldn't scale in a sense, where I was like, we really want this thing. We're like, oh, one person can like onboard 1000 people, and then somehow they'll just work out. But we know at a company that doesn't work like that either. So why would we expect it to work in open source, we have even less people and resources and money.

Maybe this slow and steady approach is better early in terms of trying to onboard people through like mentorship, stuff like that. The approaches that we have are more of like a, watch this video, participate in this hackathon with a lot of other people and and that will bring a lot of people in, it's just that we have this second issue phenomenon, or whatever you want to call, it was like, the first thing to get in is really easy. A lot of people get in, but for them to stay there is really hard. What will convince someone to continue doing open source or whatever it is, it doesn't even have to be in the same project.

I think a lot of people get their start in [open source], but they're afraid. And that's understandable. Because it's very daunting. You just have no context into how how things work outside of like talking to someone and I think the maintainers would like to do that. But of course, they have limited time as well. And they're always going to struggle with writing features and growing this community while handling everything within the project, which is a lot more than just writing code. Spending time creating these initiatives to get people in might feel draining to people, but I like to think it's fun and all that it is but like, it takes time, like being in meetings is work as well. And I speaking of some stuff that happened to me, like a few weeks ago, like, I think that people assume that if you don't write code, then you're not doing anything. Right. And so that's another thing I think is, I don't know how that's not really something needs to change necessarily, but like, acknowledging like, what I guess what efforts can help a project, and it could be

a lot of things. And a lot of those things aren't measured, or maybe shouldn't be measured.

SK: I guess there were a few things there. I like what you said about having to be patient about the relationships and community that you're building. It takes time to, to make and have those relationships with the people that you're working with into a growing community is in terms of like, outreach, but also, like we said, There towards the end about the things that count and like should be counted. Coming from like an academic track myself, the only thing that is measured or really cared about is how many talks you give, but things like outreach or volunteering middle school science show or something like that isn't, and in fact takes it takes time and effort away from the thing you're supposed to be doing. But [outreach] is something that's useful. And I guess I'm curious what should or shouldn't be sort of measured in terms of those?

HZ: I feel like I've changed in that instance, where I'm finding myself like, drawn to more like liberal arts is funny, kind of funny. I used to not care about history, or English or writing or like that kind of stuff before. And now it's like, I think those things are very valuable. Given the background, technical background, I have, maybe before I didn't appreciate as much and are just the arts in general. The art is around this idea that you can't sometimes you can't explain or say or express the things that you see, or you hear right, within those things. And I think that in programming is not that different. Even though people like to think that it's just like, a bunch of text or like the ones and zeros and stuff like that, but to me, it comes down to like people matter, more than the artifacts that they create.

I think a lot of this is I'm trying to think a lot bigger and a lot longer in scope than what we typically think in terms of. I think that the liturgies and habits that we have in our industry causes us to be very short term minded, you know, scaling VC money, you know, like that kind of stuff, rather than be like, “Okay, why is going to last for 10 years, 100 years?” I don't, we're not really thinking of that. And, you know, we're not like incentivized to, in some sense, you know, we're gonna die and blah, blah, and the community money, all this stuff. But if we want these things to last, and to sustain ourselves, we're gonna have to think that way. Unfortunately, that's not necessarily something that is going to get paid for by a sponsor, or even like, even in academia, because it needs to have some kind of “practical application.” But I've, I don't know, I guess I'm drawn to this idea that like these more philosophical things can lead to what I would say like, if you get inspired, or you inspire someone, that will be a lot more work than just fixing some bugs, right? Like getting someone on board and thinking about the

future is like what we're all about, like, that's what science is supposed to be about, like getting people to hope for something different, that can be better. And so that's what I'm drawn to personally. And I think that, yeah, unfortunately, that's not something that is.

I think that obviously, we need to, get stuff done. But if we only write code, or write papers, or whatever, that's just like, you're in this, when you call it, like, local optima, or whatever, on the the h-index, right, or like, how many lines of code like, those are all measurements that we have. But of course, there's — I forgot the name of this law, or whatever you call it — the thing that becomes the target, ceases to become a good measure of what is success, or whatever it is. And I think that's very true in coding, where the GitHub becomes a resume, how many lines of code is like your worth, as a programmer, that's like, totally not true. And we all know that. But like, you can't help but feel that way. Or even judge people, because they didn't do as many lines of code as you or something like that. And everything becomes like this number game of like, you didn't write enough slack messages, or tweets, or whatever, it's like, we all know, you can fake all those things. And that, and you can also write one of them, and that has so much more impact than 100 of them.

And it just, I think the attitude that we can even try to measure that is maybe wrong. Like, it's impossible to know the impact of a talk that you gave to some person, maybe it was a student that I've given some talks at some colleges, just telling them, you don't have to work at Google, you know, and that's kind of the gist of it. But through my story and my experience, maybe they'll change a way that is helpful. Maybe it's bad, I don't know. I guess I just see that there's a different way of doing it. And I want to pursue that. I mean, I don't know the answers either. But I feel like it's worth working on.

SK: Yeah, I like what you about having it be really difficult to measure impact. At the KFG we recently have been thinking about our goals and the strategies of how we get there, we kind of need that structure to have an aim, and some sort of measurement of progress. So it's been a challenge for me as well, just to think about which metrics are we using to like gauge growth and impact? Like, what does that even mean? That is a really tough question. And it is, you know, ephemeral, but it it is sort of intangible. So trying to make those intangible impact meaning into into numbers does seem like so limiting. But what else? What else is there to sort of like engage or even beyond like open access, open source, just I think in life, too, it's just like, oh, like, do you have a house yet? Like, do you have enough money yet? Do you have like a family yet? Like, whatever, whatever the metric is, like, How high are you up in your job? Like, these things are metrics to that maybe aren't gauging like happiness or fulfillment, but I think those are still hard.

HZ: Yeah. I mean, yeah, exactly. I think not that measurement itself is bad. It's just so I guess, so easy to get corrupted. I think that would be the word is like, we use good things. And then they become if I want to use them.

In a more religious terminology, good things become ultimate things. And so that's where when we talk about idolatry in a religious sense, that's all it means it's when you put something ahead of God, and it doesn't have to. You could apply that to anything, right? It's like my happiness or whatever it is, if it's based on how many lines of code I have, then maybe lines of code has become my identity. And maybe we shouldn't have that. And it's really easy to like, mentally be like, “Oh, I don't care about that.” But like, culturally and environment, what we do, it shapes us, it changes us in ways that we can't willpower our way out of, you know, because we're so like, strong or whatever, like, the things will change you thinking openly about me. A lot of people are like, “Oh, I should just quit Babel,” or because we don't have that much money, or it's not as much as I could be making if I was like an engineer at like a big tech company.

Beyond economic impacts

And it's like, well, what's the anything? What's the opportunity cost and all this stuff, but it's like, maybe that's not everything. Just because I get a job and make lots of money isn't going to? Well, that's definitely not just gonna make you happy. If I mean, it could, depending on what situation you're in, but yeah, it does, it comes down to knowing yourself and like your own goals, too.

***SK:** Right. And sort of, like you mentioned, liturgy, and religion, and faith, too, is you just have to trusted and believe in the things that you're doing. And that they are meaningful beyond ways that make it ultimate in that religious sense. It doesn't matter, the numbers; it doesn't matter, the money, These things are more important or more impactful, or like they mean something beyond whatever cultural thing is, right?*

HZ: I think I want to be able to think beyond like, essentially economic value. It's like, you know, I know, assuming you're at a point where you feel okay, and maybe you're never gonna feel okay, because we're in a culture of the rat race, and like, everything is economic. Now, everything's a transaction, you know, I guess, and I see some value in crypto, but you know, some of the problems with like NF T's and making everything

something you can sell, might cause an attitude of like, oh, if it doesn't give you money, then why even do it? Right?

But I'm like, isn't that kind of going back to art is that that's kind of the whole point. Of course, you can sell art for a lot of money. But you know, we talked about how it's priceless, or there's, like poetry is like where you can't reduce the words to like, simplify them, because that's the whole point, the form of it is the words themselves. And so, yeah, maybe need to rediscover what is quote unquote, beautiful, because it's like the opposite of this thing where like, oh, everything has value, there's there are things that have their own duty, and then in themselves, like the actions that we take, or what we do.

So I've been reading up on what's called Shadow Work. And it's like work that is necessary to be done, but it's behind the scenes, like commuting to work, even though we don't really do that as much anymore. But it's like a part of the job, it has to be there, but you're not paid for that time. Maybe you should, maybe you shouldn't. Parenting is probably the obvious one, right? It's a very important part of our society. And it's necessary for anything to happen, but you're not paid for it necessarily. And maybe getting paid for it doesn't mean you shouldn't get paid for it. But I think it's important for us to think through. If we do that, what will change in terms of like how people act? When that happens?

It's not as simple as so long as we put more money into the things just like open source, like just because we have a lot of money, we just inject money into open source, it's not going to solve the problems, deeper problems of what what's happening with open source in terms of like. Maybe you guilt that you feel for not doing it and maybe you get like for me like I'm getting paid now. Or you're getting donations from people. And then you feel bad because you didn't help someone or you feel the need to help everyone write anything, maybe money amplifies the problem for those things.

***SK:** That's probably true. But I like what you said there about being intentional about the things that you're investing into. And it sounds like a really cool concept: the things that are behind the scenes is sort of like the prop and like, again, like maybe like the structure of any good work to begin with. But it sounds like you're maybe hesitant to say that — say, we're using money as the thing that you give to show that you have value — it seems like you're hesitant to say that we should put money towards towards, like that type of Shadow Work behind open access or open source.*

HZ: Yeah. I don't think that is wrong to do it necessarily. But I guess it's, again, it's really easy to turn it in a bad direction. And so and, you know, I totally understand, like, you know, you don't have any money you need to live, you need to survive. But I think maybe

one of the problems is that we live in a society where you feel like, even when, and I can't say that people have enough, or I have enough or anything like judge other people for that, but it's like, we create a society where you never feel like you have enough.

And I think that's the problem, not that there's a certain limit, or there's a number where there's enough and there's not enough, but like the fact that we can't come up with one is, I think, a problem where we can't have self imposed limits on ourselves, or in our own culture or community where, you know, enough is enough, in terms of how much money we have, how many people are in the project or whatever.

Like, it's like, the only thing that we can do is add, right, we can't like subtract in terms of like, you know, it's like saying, like, you know, when you're a software engineer, you can make a lot of money, right, but then you always want to get promoted, you want to get more and more, you can't stop. And so then there's all this, this thing called the golden handcuffs where it's like, I don't know if you heard this before, but you get like, I guess, stock in the company. But you have to kind of like basically stay there for a certain amount of time for it to get to you. And so then every month or a year, you get more, and so you never feel like you can leave. And it's this funny first one problem, basically, where they're rich, but they never feel like they can leave. I'm assuming they have desires to do something else, you know, make a company or just like get out of tech or something, I don't know. But like, it's just interesting you can be in a situation when you have more money than you could ever need. But then feeling like trapped. Right? And I'm not talking about people that are not like they're privileged. Right. And it's like, the fact that those people feel trapped, I think is a symptom of like, everyone, but you can't escape from this.

SK: I guess that's interesting, especially being like in that sort privilege space to be in code and to be in the spot where like, you have all this like, wealth and power in this space. But like, I guess what we'll maybe like, what's like a better way to sort of structure like this question, right? Like, this is probably the big question like, how do you structure tech? Or like any any, like, big infrastructural thing that doesn't, like allows for more people to come in, like, allows for more like, dynamic, like, positive outcomes from these things that like, run our whole world? Like, the whole reason, like they were having this conversation right now? is like, because of tech and like, probably because of some person, right? feeling trapped? For some for like, for whatever reason.

HZ: I have no idea what the answers are. But I think that if I think the some of the symptoms of this are of the fact that we have to continue adding. My friend put it as our only solution is escalation, you know, in terms of like, he mentioned, like, say, the wars,

like, you know, more bonds, or more military or more police or more, whatever, more money. How do we tell ourselves? No, right? How do we say no to these things? When the possibilities out there for more, and being able to say no, I think is, is a very powerful, right?

Even in my case, I don't want to it's not because I'm necessarily like, morally better than anyone. Like I think I'm better because I chose not to, like take the tech job, but it's just for me, like I feel like it would be better for me if I didn't do it. That the everyone's always wonder like, oh, why did you quit your job? Like why wouldn't it's like a no brainer to just go work at a big tech company. And I would like to hope that that's not right. It's possible to not go that route. I might still go that route, you know, maybe I don't have enough money or whatever. But like, it's okay to not go that route. I guess it's like that we shouldn't, I guess impose one way of doing things. And I my way isn't the best or anything or any any of that or like maybe it's bad, but like at least it's another option.

And so I guess, the tech, maybe part of the problem is that it's so in compensating of everything. It's like the whole, like, I don't know if you've heard that phrase like, tech is like eating the world or whatever. And maybe, maybe we don't want that, like, I don't want to be like, I don't want to come out as like some anti tech person, I obviously, like technology, it's just like, I think it's good to acknowledge the limits of it, and then also where the harms are. And I think we're maybe we're really bad at counting the costs essentially, are really good at like, so there's a, there's a phrase I like where it's like, it's really, it might be easy to predict, like the car, but it's not easy to predict traffic jams. Right? So or, like, that's a negative, that was a result of us using it. And I don't think anyone can predict it. And doesn't mean that just because someone came up with something new, that we should poopoo it and say that we shouldn't go read that route. But I think people are so enamored by like, Oh, we should do it just because we can. There's also another quote like, it's time to build, right? Well, the question is, what are we going to build? And why not just that we should like that that's just like a engineering mindset of like, well, just because it's possible, I might as well try it instead of thinking about the implications a little bit. Right at all, like, we're not really thinking we're just like moving because we're curious, and, and then not realizing that, you know, some technologies can have a lot more harm than others. Not. Like, it's like, technology's not good or bad, but it's not neutral, either. You know, it's not just that it's a tool that it's in the hands of someone bad or good, and then they can use it for bad, but like, like, just think about, like a nuclear weapon. So I think that it's mostly bad, it's like, it can be used for good and a lot of things can use for good, but like, maybe one has an affordance or leans towards being used for harm that maybe we should be more careful about it.

SK: Yeah, I feel like having that, like what and why focus maybe ties back into that sort of community driven aspect of maybe like anything, like having having that very explicit tie into. And like knowing, like really being embedded in that like, meaning of why you're making and doing the thing to begin with, but maybe gets into like the lack of that community, maybe in the big tech world, like the lack of that maybe like real authentic community in the big tech role. This is sort of why these things right, like, are very, like tool driven, just like doing things for the, for the for the sake, I guess something else I was thinking of maybe like, in terms of being at escalation, I've definitely tinkered with the thought of becoming a Luddite myself, just because there's so much, there's so much it's not good about tech, and things like that. But I guess that makes me wonder like, does that mean that like, sort of how does that tie into like needing to start over like wanting to like sort of scrap everything like no, you talked earlier on about like, it doesn't, it doesn't always make sense to like, go back to the beginning and like write your whole code base, but sort of when when is it maybe appropriate? Or when do you think it's appropriate to actually have that like, starting over and just like getting rid of, or throwing the baby out with the bathwater, just to actually have a good space?

HZ: Yeah, actually, I, my, I, my instinct now is that I don't, I don't even know that that is ever a good idea. But it's like really rare. So even in this case, I'm so tired. Like I don't, we're not going to go back to like non tech society, I don't think we should do that. Like illage, who's known for being like this anti tech person was not trying to say that either. Otherwise, you'd become like a, like a traditionalist or someone that like, wants to go back to the past, what we should do is more like, try to move towards a new future, by learning from the past. Things that we forgot, we used to think we should rediscover them, and then reapply them. And I think that goes back to this. You actually had a question, like, in the email about, like doctrine and how that relates to like, progress. And I think that, you know, when you hear the word doctrine, it's a very, like, rigid thing or whatever, and like, everyone hates it, especially if it's in terms of religion, but um, I really liked this thinking from Gk Chesterton. He's a Catholic, and he has this quote i and there's like a longer quote part of it, but he basically said, tradition is the democracy of the dead and I thought that was like mind blowing. And so I just thinking about that more deeply that he basically said that there's like an oligarchy of people that are alive, and we get to impose what the future will be. But the people of the past can't say anything anymore. They're dead. And so the way that they have their vote, their say, is through

tradition through the Pass through artifacts in our liturgies. And you know, history. And we should acknowledge and take that into account, it doesn't mean that they were right. And we shouldn't do things just because they were old. But old things that last last for a reason. And so there's this thing He has called Chesterton's fence, basically, that if there's a fence in front of you the tradition that you shouldn't just get rid of it just because there was a fence, if it's a man made fence, they put it there for a reason. And so it doesn't mean that you should get rid of the fence just because it was there, or you should keep it just because it was there. But that you should think through why they put it there. And if you can understand why it was a bad reason, you should get rid of it. If you realize that it was a good reason, maybe you should keep it. And so that's just thinking more deeply about why things were there in the first place. like throwing out the baby with the bathwater is negative, because you don't understand why things were put there in the first place. And so even from this is kind of niche, but in the religious sense, like say, for a Protestant, and then like Catholic, a lot of price since we got rid of our liturgies. Because we threw out the baby with the bathwater you thought that like those liturgies became idols. And so we're like, no, let's get rid of that. Let's all turn everything into belief by getting rid of all physical liturgies in some sense. And then now, we're in this place where a lot of Protestantism is like super modern, in the sense of like, everything is about what you think and what you believe. And it's not about your body anymore. But the Christian faith is a very embodied faith that the Incarnation is like, cry, God becoming a person. And so liturgy is important. And I think that we learn through the actions that we take, not just what we think. And so that's like one example of how we've lost something by like, throwing out the baby with the bathwater, and like, not realizing that tradition has its place. And maybe we should renew that. So I think that tradition is the place where you can quote unquote, innovate, even though it seems like the opposite. It's like you take the past and you can readapt it to the future. That's how we move forward, not starting over, because you're gonna make the same mistakes as before, like that kind of thing, right?

SK: Yeah, I definitely love that example of like, returning back to or like finding something that like had been lost in maybe throwing something out to preemptively are moving on too quickly, just to just maybe for the sake of innovation, but I mean, obviously, Parkinson's, Heather had good reasons. For or they say that they have good reasons for breaking away from the Catholic Church. And many of them I think were valid at the time. But I think the, like, I like one of my luthers things was like, oh, like, the Bible is not in language. People can read and like, indulgences are like non biblical, but like, those things have significantly changed in the Catholic Church, like, Where are

those things? Like, are those problems still valid? Like, but anyway, to maybe rein it into the conversation more about, like, tech and open source and things like that? So I guess it makes you just wonder, what sort of practices have we lost in this sort of push for progress?

HZ: Yeah, I guess that's my interest in like, the, how program was done before? And I don't really know, honestly, um, I mean, I know, you know, some of the things that we do now, like I can think of like some practices, like, these are very recently there's like, hacktober Fest, right? It's a thing sponsored by GitHub and Digital Ocean. So you could say it's like, someone's called in, and this is a negative tag on it, but they call it a corporate funded DDoS, which is like, you know, paying spending money for teachers to get people to contribute to open source so that they can help their companies. And that's like a negative way of putting it, of course, does bring people to open source. And there's also like, like, there's a Twitter hashtag 100 days of code to get people every day they I guess they post something that they write, we have like, good first issues. And then we Everyone has their memes and their phrases that we tell each other. I think about like crypto, there's like, the whole hold or huddle thing where you try to don't sell your Bitcoin or whatever. In coding, we have like, this thing called don't repeat yourself dry, where it's like, oh, we want to automate everything. So you shouldn't write something two times you should write some code that do it for you. So I don't Yeah, the practices from before I feel like I have no idea if it's, if it was more like a form of art before or like writing. You know, I mean, you could take code in so many different ways. But for me, it's just You know, if we're losing, over over measuring everything, we're losing the humanity of it. And so I think of also the, like, say the movie, like Hidden Figures where they talked about how like, people, like women used to be the computers, right. And so I think if you thought that before, they were just resources that were used to compute, quote, unquote, then like, that is a, I guess, a perspective on on programmers that was really productive. That might still, I don't know if it's still carried today, but it was like that for a long time. And so it's almost like blue collar in some sense, right? Or it was just like some expendable outsource of or labor. And then now I think people realize more that like, okay, the people really matter that outsourcing doesn't really make sense. Because, you know, if someone spent all their time creating the initial product, right, the things are going to change features is going to get added. And if they leave, and you try to bring new people in, they have no idea how it works. And so it'd be nice to retain those people. And that's why like, I guess, software, people are getting paid so much. But now it's almost too extreme, where now they think all the programmers think they're the hot shot, and like, we're, everything will be solved by tech, you know, it's always like funny that we have these two

things, we're like, first programmers are just, they don't matter. The code is what matters. And now it's like, sometimes they think that there's too important, but I'd like to think that, yeah, obviously the people in any job matter, because of essentially the tacit knowledge that they have about what they do, which is knowledge that is retained, but isn't written down, it's like in their head, it's in their body, whatever that means. So when they leave, they take something with them, right? They could have tried as hard as they could to like, um, you know, the whole onboarding process where you try to tell people like, this is why I did this, I think you can ever explain some things because maybe you don't even know how to explain those things that you did. I think that that's like the core of it is like, people really do matter. And like there's you can't extract every piece of information in their mind because they might not even be able to express it other than through working with them essentially.

SK: That gets back to you that we're talking about sort of at the beginning of having like this whole culture that you're sort of embedded in maybe that's the be like a helpful way to just like if you're if if you really believe in this culture of the company, or whatever it is, like that, sort of that intuitive pneus becomes through a little bit more like sort of the how and why sort of shines out on that maybe like the particular I mean, the people do matter, because like, that's where ideas come from, it's the people who actually do the work. But in terms of like moving forward, just like having this like bigger culture and community like having that. Having that be the thing that's like driving and connecting people together. Like that's probably that is like, what I'm just sort of echoing what you're saying there. Like that probably is like the better way to continue transmitting information. Like you were saying that you're, like, more interested in like, more of the humanities type thing.

HZ: That definitely, like, seems to apply. And this sort of, yeah, I think, um, I mean, it's, I guess this is a greater problem of the whole, like, individualism versus community, sort of dynamic and East versus West, or what are all these different things? I like how do we recognize the importance of each individual person, but then know that like, I don't want anyone to become like the, like a god or like a cult leader, right? It's like, we worship celebrities or influencers and stuff like that. But at the same time, we don't treat people that maintain things well, as people at all, they're not really people. Right? The fact that we have something called human resources, I think it's almost like insulting in a way. It's like, we're just, each person is just like some, like, I don't like the word resource. sounds bad. It's like, it's like a piece of metal or whatever, right? Yeah. So like, how do we acknowledge the value of people without turning them into like these figures that are like

so amazing, that can't do wrong? Yeah. Yeah, that balance i think is like, and yeah, it's nothing to like, open source. It's like every every life is about that too.

*SK: Where do you think you'll be going with Babel or otherwise, like, where do you
Where do you like sort of see yourself or where do you see? Or what do you maybe hope to
see, um, with, like, the things that you're working on in the spaces that you're in?*

HZ: I mean, yeah, to be honest, I don't, I'm literally still, I've been trying to think about, like, for real the whole year, honestly, but like, even last month a lot. And, you know, I mean, if I, like I think that it's good for me to think about what it looks like to leave. And maybe that would be like if we posted it online, and people are like, wow, how can you say that? Or whatever. But like, I think it's healthy. I don't think I'm going to leave anytime soon. But I think I don't want to leave when things are going bad in the sense, like, I think it's, it's almost, like kind of cool to be able to do that when it's in a good place. You might think that's the worst time, but actually the opposite, right? It's like if things are going bad, and you leave, that's like, I don't know why you would do that. But yeah, I think that if we get to a point where it feels good that it's like I maybe I did what I wanted to do, I'm in this place where like, we have money, we have people things like, and of course, if I leave this there is going to be some gap there. But, you know, if Yeah, I don't Yeah, part of me was, I thought I would need to find what I need to do next. But I don't really know what that looks like I, I it's not like I'm going to leave tech I'm talking about this stuff is really interesting. To me being a part of different projects is interesting. I want to be able to do that. I just I don't even know if there's like a job for that. Really, there probably isn't. So I'm open to explore what that looks like. But yeah, I don't know. I just wanted to figure out like, how do we? How do we like make open source? more sustainable, more human? I guess I don't know. Like, it's like that. I think open source has its own culture that is separate from Tech. And I think that that's interesting. And the part that is issued to me, of course, is how it intersects with like, faith and stuff. So like, where do those fit together and within community, and also like governance and stuff like that? Those are the kinds of questions or field that I want to continue to explore. And then relating that to, I guess, maintenance as a whole. Like, I always come back to like, say, a gardener, like they're a maintainer to what do they know that we don't, and not that I can't be a gardener, but like I haven't had an experience in that. So like, there's so many different ways that we people maintain things that they don't even recognize maybe that will only come out through conversation, right? And that's kind of what I want to continue doing.

*SK: Well, I hope you feel like with the gardening, I think that's a super good analogy of
like, organic growth and like community driven and like needing time and effort to make*

things grow and make things

HZ: yeah, no, no, definitely. And I think the one that I resonate the most, is actually the fact that you can't make it faster, right? You can't there's a level of like, lack of control, essentially, that you kind of have to, to the winds of weather and environment. And how's it being parenting is actually the same thing, right? You can't make your kids do anything. You can only create the environment unfortunately. But that's like it's almost like the beauty of it. It's like it we we really want this control over like people, the world and all this stuff, but like I think we have to like let go of that and like know that you can only do so much. And maybe that's better.

SK: Well I think with that. Oh, thank you again for your time. This is awesome conversation. And yeah, thank you. I'm excited to see what comes next.