EPISODE 1278

[INTRODUCTION]

[00:00:00] JM: Coinbase is a popular and well-trusted cryptocurrency platform for buying and selling digital assets like Bitcoin, Etherium, and others. With Coinbase, you can manage your portfolio of crypto in one place like you would for other investments. And there are other features for scheduling recurring purchases of assets, time-delayed withdrawals and mobile apps with sleek UIs for mobile access to the markets. And the mobile apps are the focus of today's episode. Coinbase recently moved to React Native, and today's episode has Brent Walter and Jacob Thornton discussing the move to React Native. It's a great show about a technical migration. And it runs in contrast to the decisions by Airbnb that we explored several years ago about Airbnb's decision to move off of React Native. I hope you enjoyed this episode.

[00:00:55] JM: A few announcements before we get started. One, if you like Clubhouse, subscribe to the Club for Software Daily on Clubhouse. It's just Software Daily. And we'll be doing some interesting Clubhouse sessions within the next few weeks. And two, if you're looking for a job, we are hiring a variety of roles. We're looking for a social media manager. We're looking for a graphic designer. And we're looking for writers. If you are interested in contributing content to Software Engineering Daily, or even if you're a podcaster, and you're curious about how to get involved, we are looking for people with interesting backgrounds who can contribute to Software Engineering Daily. Again, mostly we're looking for social media help and design help. But if you're a writer or a podcaster, we'd also love to hear from you. You can send me an email with your resume, jeff@softwareengineeringdaily.com. That's jeff@softwareengineeringdaily.com.

[INTERVIEW]

[00:02:00] JM: Guys, welcome to the show.

[00:02:02] BW: Hey, thanks for having us.

[00:02:04] JM: Today is an exploration of React Native at Coinbase. And this deserves a call back to a few conversations that I've had on the podcast. The first one that comes to mind, and I know you guys are familiar with this, is the case study of Airbnb adopting React Native and then moving off of React Native. And I think this is quite an interesting case study with regards to this conversation because Airbnb is in many ways like Coinbase. You could even point to the fact that the founder of Coinbase came from Airbnb. But Airbnb is this kind of consumer-facing application where there're financial transactions and a lot of sensitivity. And there're a lot of employees working there. So I'll just start off with the straightforward question of if React Native did not work for Airbnb, why does it work for Coinbase?

[00:03:00] JW: Yeah, it's a great question. Brent might have a different answer than me. Do you want to go first, Brent?

[00:03:04] BW: Yeah, I'm happy to dive in on this one. I think circumstances are very, very different. Like when you go through the history of Airbnb and React Native, and our team actually spent time with those folks at Airbnb getting an insider's look into the decisions they made, why they made those choices, why it worked, why it didn't work both technically and organizationally. So, in many ways, the differences, the comparison is really a false comparison, really, we took a lot of the learnings from the folks who migrated to React Native than away from it when we made our decision. And so comparing the companies is very different than comparing a brownfield approach and a greenfield approach. In the way we did our migrations and integrations.

[00:03:48] JT: Yeah, so a couple of things. Are you familiar with like brownfield versus greenfield? Kind of what Brent's talking about there?

[00:03:54] JM: Yes.

[00:03:54] JT: Okay. Great. The second thing I would say is just a lot of time has passed, right? So if you think of back when Airbnb was first kind of experimenting with React Native, since then, we have like a completely new engine in Hermes. We have a handful of new patterns. We're doing like functional components in React. We have hooks. Like it’s just a completely different world. And so, yeah.

[00:04:21] JM: Okay, I'd like to go a little bit deeper on this question, because if I recall, the Airbnb struggles were – A lot of it was around tooling and debugging issues and kind of a sense that React Native was this idea that was flashy from the outside, but was not yet ready for primetime. Because I just remember, I think the guy I interviewed was Gabriel Peal. And I think I remember him just talking about indecipherable error messages that were really problematic. And you had to have the right people who were familiar with both React Native and whatever native platform they we're building on, whether was Android or iOS, and it was just too much. Is that consistent with why it didn't work for Airbnb and just an issue of those things being ironed out over time?

[00:05:15] JT: Yeah, I mean, yes and no. Like realistically, there's still a lot of ways to go. I remember I spent a ton of time looking into this issue, where we had a request failing when the debugger was open. But when would close the debugger, the request would work. And we couldn't figure out what was going on. And it took us a while to understand that like when we were opening the Chrome inspector, it was actually changing out the entire like JavaScript runtime engine to a completely different interpreter. As a result, like that radically shifted the way we were thinking about it and the way we were thinking about debugging things, and a handful of other things. And there are still a number of those kind of gotchas. And really, on our end, when they bite you, they definitely bite you. And we've done a lot to kind of document these, share them with the team and kind of like figure out processes and like tooling and scripts to work around that. But realistically, like there's still a ways to go for sure.

[00:06:14] JM: Let's take a step back. What motivated Coinbase to want to use React Native?

[00:06:21] BW: So if you look back at the history of applications at Coinbase, originally, there was just a web application, then a fairly meaningful amount of time after that they brought in mobile applications. And so the company already had deep roots in React and web technology. And the company was trying to scale over time and ran into just difficulties hiring mobile engineers to scale over time. We were on a double web team when we got maybe 50 or less percent of those native mobile engineers. And then when you consider you have iOS and Android that you're hiring for at the same time, it just compounds itself. And so when we started this journey a little over two years ago, there was this theory that if this could be effective, if we

could have as high quality of an application as we have with our native technology, then we could reduce our code, like our code complexity and our higher needs from three to two going from iOS, Android, and web to mobile and web. And then even more so, with additional investment, we can get those platforms down to one and a half.

And so the theory was you'll get the efficiency gains of having reduced services to support if you can maintain that level of quality. And so that was the big question when we started down that journey was if, if we can have level of quality, and if we can realize the velocity gains, and if we can win the hiring game that way. And so when you look at the journey, it was very incremental, starting with a new application that was for new market, which is for our pro customers and our pro app, and then testing that out, seeing if that was successful, if that can meet our needs. And then after that we looked at, "Well, what would brownfield be like?" What if we redid our onboarding flow with React Native and integrated that in native and React Native? And sort of incrementally stepping up the complexity as we prove to ourselves that the technology would support our needs, the quality would land where it needs to be for business metrics. But that was really the journey of moving in that direction and why we made that choice, and the slow methodical testing along the way to verify our assumptions.

[00:08:38] JM: Jacob, anything you want to add?

[00:08:40] JT: No. I mean, yeah, so many kind of things just needed to come together just perfectly for us to do or to make a bet on React Native. I think that we had Harry Tormey join the company, who was the engineer that wrote up the React Native post for us. And he had a lot of React Native experience. And he had just joined the pro team, which was just starting to undergo kind of like a Greenfield native app, right of the pro app. And so kind of side-by-side with that, it was very little risk for us to also experiment with like, "Okay, what would this look like in React Native at the same time and kind of having his expertise?"

I think what we found really quickly was just like how fast he was moving, like the quality of being like really just as high, if not higher than we really expected it. And just so much like React is just such an enjoyable product development kind of software suite at this point that we thought we'd get – And we were really like a react shop up into that point, I would say also. We

had a lot of experience. Like our web app at the time was massive, and we had a lot of really good engineers there that we thought we could leverage a lot of learnings.

[00:09:59] JM: And let's just talk a little bit about the fundamentals here. Maybe it's obvious to us talking about it, because we're kind of domain experts, or I'm definitely not a domain expert in this. But I have some domain understanding, the benefits of React Native, as I understand it. Just setting the table here. Traditionally, you had iOS and Android development that basically took place in silos. And the parallel tracks of those applications were basically managed by design teams, and product management teams, and senior level mobile managers. And that sounds okay, but it's actually not enough to synchronize the development of iOS and Android to an extent that alleviates this partitioning. And you had these really chronic problems that developed from people working essentially in isolation. You had the iOS team, and then the Android team, and then they just have weird different ways of accessing backend data models, and you would have all these in consistencies. And it created all kinds of headaches. React Native doesn't solve the problem completely, but it is a really nice solution, because it gives you component reusability, code reusability across the entire mobile stack. And therefore it gives a communication, essentially a communication layer between the iOS and the Android team because they can share components among each other. Am I encapsulating the motivation for React Native correctly?

[00:11:35] JT: Yeah. I mean, definitely. I think you hit on a lot of the like sort of things. And what's interesting is actually when we started out on this kind of journey, even especially with like retail, which is if you think of Coinbase, Coinbase is like app today. We weren't even sure that we were going to get a lot of those benefits, because we weren't sure that we were going to build anything other than the Android app. When we set out at this, we were like, “Let's just build Android. And let's start there.” And we know that there's potential for us to like expand to other platforms if we think that it makes sense. But at the time, like we had a pretty great iOS app. We had a really strong iOS team. And so we kind of just narrowed in on like even in the world where we're just replacing Android, is there enough win here and like the speed at which we could execute the quality of the app that we could produce? Like are we able to build an Android app with React Native against Android native engineers? And our conclusion was yes.

[00:12:41] JM: When should React Native be used? Like people are listening to this, and maybe they run a small engineering-led software product that has mobile applications, should they always be using React Native at this point? Like if I run a small engineering team with one iOS developer and one Android developer, should they be sharing everything over React Native?

[00:13:06] JT: I think, at least for me, so much goes into that decision, like staffing, like what your goals are, the type of app you're trying to build. There are things still that React Native has like historically struggled with. So in React Native, a lot of things go over this thing they call the bridge. And so real time interactions where you're like having to update and send things across the bridge to update an animation are very hard to pull off like with high-performance. That said, there's a lot of things in the pipe like this thing called Reanimated 2, which is a library built on a lot of the new kind of like fabric architecture that's rolling out slowly with React Native right now.

And so I think that the gaps there are going to be closing. But I could just say, me personally, if I were to leave Coinbase today and start a company, I would want 100% – It would be really hard to convince me to go native. Even as like a swift UI fanboy, it would be like very hard to convince me not to use React Native for a business.

[00:14:03] JM: What about the typical case where I'm developing a consumer mobile application and I'm starting with iOS? Like I'm a startup, I built my first version of a consumer product. And I know I want to go to market with iOS first and validate iOS before I move to Android. Should I be using React Native even in that case?

[00:14:25] JT: Yeah. For me, yes still. I mean, I think that there's a real cost if you have already built the app in iOS. You're a small dev team. You've already staffed around iOS and you want to learn and improve things out on iOS. Makes sense to like stay the course and reevaluate later based on like how you want to go. But if you're starting from scratch, in particular, if it's much easier, and my opinion to hire really strong web people is to hire strong mobile people. There's just a lot more people that write TypeScript today. And I feel like the learning gap to ramp up if you already know TypeScript, if you're already proficient into React Native, is not that large. And so, yeah, again, even if I was going to one platform, which we did essentially at Coinbase, absolutely, React Native.

[00:15:13] BW: Yeah, little context on the one platform. We're talking about starting with iOS or Android. At Coinbase, we did a rewrite of the main app. We started on Android, because it was clear that React Native on the Android platform was much less mature, much more buggy, much more difficult to make performant. And so to keep ourselves honest, we also started on Android to test out the technology, to make sure we could hit the bar quality we needed on that platform with the assumption that when we then launched it on iOS, it'll be that much better, which was what we found. And the process of going from Android to iOS was a handful of months. It was relatively simple and easy. So I would also like take Jacob's approach, which is if you're starting on a greenfield today, absolutely recommend React Native.

[00:16:08] JT: Yeah, I think there's also – Just to pile on that. There's a lot of other like sort of multi-platform technologies that are coming down the pipeline. You have like Kotlin multi-platform, you have Flutter, a handful of other technologies. I think at the very least, I would be evaluating those alongside React Native, especially for the small teams. I just think there's so much potential for a win there. And then my bias is I just really love like the React ecosystem. And I like the kind of developer ergonomics of developing in React.

[00:16:40] JM: Yeah, I mean, this is a pretty softball question. But how do you feel about Flutter versus React Native?

[00:16:47] JT: Yeah, I think it's a great question. Like Flutter has – We started this journey like two years ago it feels like, and Flutter has come a long way in that time. I think they've gained a lot of momentum with the developer community. They have some really impressive performance metrics. Like their development suite is really good. I think that if we were starting this project today from scratch versus when we did, I think that Flutter would have been a much bigger piece of kind of the puzzle as far as like weighing the pros and cons, whether or not we should go with Flutter or React, React Native. However, when we were doing this, like Flutter was still so new. We weren't really sure on like how flutter would become. Also because we had so much expertise in React already. We have some pretty heavy hitters from the React community on our team. We just felt like really confident in our ability to architect a scalable React Native app.

[00:17:49] JM: When you think about this paradigm in its most mature form, obviously impossible to fully imagine. But when you think about like Flutter versus React Native and React Native sort of having this fundamental idea of the JavaScript bridge, which I think is being fixed or changed over time, but I'm not super up to date on that. But versus Flutter, which I believe operates in a more close to the metal path, because I think it compiles directly to GPU code or something. Can you clarify a little bit more for me, or if you know this area well enough, the fundamentals of Flutter versus React Native?

[00:18:37] JT: Yeah, I mean, I'm a little bit out of my depth, because, obviously, I spent way more time with React Native than Flutter here. But I think you're exactly right. I think that's one of the really big sort of advantages with Flutter. I.E., for React Native, I think the advantages that you get out of the box today are a much larger community, especially if you include React at large. Most of our app is just straight TypeScript. And so we're able to utilize a lot of React libraries, not even necessarily React Native libraries.

The second thing with the bridge traffic in particular that is something that has been addressed with like Hermes and Fabric and the new sort of architecture of React Native internal. So the bet here essentially is like betting on Facebook and their like pedigree of open source and putting out really, really, really strong technical solutions to hard application development problems. And just kind of like we have a few like really close contacts on that team. Eli, who's been really, really great. Nicolas Gallagher and the other one has been really awesome. That just kind of like taking us along, like listening to feedback, working with us on these sort of things. And so that, again, combined with the community, combined with our experience with React and like our ability to ramp up on it really fast just makes it like a no brainer for us. But that's not to take away anything from Flutter. I think they're doing really cool stuff. And I think competition is like really, really good for open source frameworks.

[00:20:09] BW: And for whatever it's worth, we always have the conversation about performance when you're choosing a technology. And like if you're going to take swift on iOS and Kotlin on Android, and compare it to React Native performance, just like bare bones numbers, like you're always going to win on native on native, right? Less abstraction, there's no bridge. Even if you make the bridge less and less, still you're operating in that abstraction. And that abstraction has its benefits and its cost. And so the way we always look at this is it isn't

about optimizing performance. It's about optimizing customers experience and the business metrics and value.

And so when we ran the numbers, and we ran our tests, and we ran all the performance, like it was as fast and even faster in some situations using React Native versus native. And from a customer standpoint, like they could not tell the difference, and business metrics made no difference. So at the end of the day, absolute performance was not nearly as important as perceived performance, and customers' version of what performance means to them.

[00:21:12] JM: What I think is interesting about the Flutter versus React Native approaches, is they're really iconic of how Facebook and Google respectively solve problems. Like I've done a lot of interviews with people from Facebook. I actually have a book coming out about Facebook engineering pretty soon. But one of the things that's highlighted in the book is the fact that like Facebook is not this carbon copy of Google. It's culturally very different. And it's where basically hacky solutions that work quickly went out over these carefully designed solutions. And you think about React Native. It's sort of a hacky solution at its core. Just this idea that you're going to write like JavaScript-like code and it gets compiled over this JavaScript bridge, versus the idea that your code is getting compiled down to GPU code in the Google world on Flutter I think is just iconic of these how these different companies solve problems. But I want to know more about the actual process of building this stuff out. We've kind of talked abstractly at this point. But I'd like to start with Brent, your point of view managing the move to React Native. At the beginning, you've got this iOS app. You've got this Android app. They're written natively. You want to migrate them both to React Native. What is the process? What's do you start with? How do you architect the overall process? How do you build sustainable strategies for the teams working together? Tell me your holistic overview for how to manage this process.

[00:22:50] BW: So as you can imagine, there's many, many parts of it, right? There're native developers transitioning to JavaScript developers. There're two applications in production that are incredibly valuable to the company that you cannot slow down and you cannot pause or distract in any way. And on top of it, you're betting on a technology that the industry has questions about, and companies like Airbnb, like you mentioned earlier, have moved away from.

And so we took approach of being deliberate, methodical, data-driven, and incremental. So what it means is as you look at the situation, you look at where the opportunity lives and the risk is the lowest. For example, we started with Android, because we knew the performance would be the hardest to do well, and it's because hiring, iterating on that platform with our native team was already the hardest. And our iOS app was the flagship app for Coinbase users. And so we took a very incremental approach. Put guard rails around the metrics. Made strategic decisions about the number of resources that are going to work on the React Native app versus continuing to make features on the native application and have clear checkpoints along the way of like, "How's the quality? How's the performance? Are we meeting our deadlines?" And that's talking with engineering leadership, product leadership through that whole journey. And I'm sure Jacob has a very interesting perspective on that about that too, bootstrapping that process on the ground floor.

[00:24:29] JT: Yeah, it's hard, right? Because it's not just a technical problem. It's like an emotional problem for a lot of people or like a personal thing. Probably one of the harder things to do organizationally is like telling a group of folks, especially folks like mobile engineers who identify very strongly with the platforms that they work on, right? Like iOS engineers write iOS oftentimes because they love the Apple ecosystem. They love everything about that world. They I love Swift. They love Swift UI. They love that whole thing. And the same can be said for Android. And so coming to them and saying like, "Please keep an open mind and come on this journey with us as we go away from this and try a different framework."

Ultimately, we're still building on to these devices. But we're less like in the apple ecosystem and less in the Android ecosystem. It's definitely was very, very, very challenging. And kudos to the team who took like really a lot – Basically, everyone took a huge leap of faith with us. Put a lot of trust in us that we'd be able to get somewhere that we're all super proud of. But it took a lot of listening, a lot of hard conversations. A lot of figuring out like what is the acceptable bar? What is success? What is failure? Getting all that really black and white front?

The actual technical bit of just like writing the app at the end of the day was the easy bit, right? Granted, there's like some performance gotchas here or there, or some hard bugs and some different tooling things that we had to work through. But the people piece, easily the hardest part about like re-platforming an entire engineering org.

[00:26:09] BW: Yeah. And if I had one takeaway from our approach that I thought was beneficial was being upfront and transparent about the migration, the journey, the metrics, the guardrails, the decision points, and having those honest conversations upfront and incrementally along the way so that there wasn't any bait and switch or like unexpected surprises. And like Jacobs said, I can't express it enough that the hard work, the trust, the dedication of our native mobile engineers on Android and iOS can't be understated. Like this is only a success because they were willing to take the leap with us. They're willing to do the hard work to learn a new technology stack and invest and change their careers and to stick with Coinbase. And that is in no one's hands but their own, and they deserve full credit for that.

[00:27:03] JT: Another thing I would just quickly mention is one of the things we heard and learned very early on was just how important native platform experts are to making React Native look and feel really, really good on those respective platforms. You need someone who's an iOS engineer, and you need someone who is an Android person first in order to like really pull off things that look and feel Androidy or iOSy, and can help hold that bar kind of culturally on the team. I think if we had just only asked all of our web engineers to get together and build a React Native app from scratch, it probably would still have been good. But I think that we would have missed a lot of the key details that we're able to bake-in that make it feel a little bit more native. So whether it's like the headers growing on iOS when you like over scroll, or different rubber banding in certain places, just like little tiny details like that that disguise like the abstraction of TypeScript on top of what we're trying to do here.

The only other thing also is – Other thing is I think we spent a huge amount of time taking these folks along with us and a lot of energy into that. So we didn't just come to them and say like, "Alright, we're doing React Native. Good luck. Next week, we expect you to be writing TypeScript." Instead, we came up with like a very involved onboarding process that like had folks going through a React school that we hired a contractor to teach and lead alongside of a few of us internally. Then we had people pair with like mentors on web and do kind of like a tour of duty type thing on web to kind of get their hands dirty, a little bit writing React code. And then once they were comfortable in that sort of environment, given that they have native backgrounds already, and they've written some production React code, it was only then that they actually moved back into the actual React Native code base. And we have a pretty massive

org. So I think in the article we said there's like 100 people commit in. I think, just in retail, there're probably about 60 or 70 native engineers. So we split that up into a couple of different groups. So we had a first group of about 10 or so folks move through this program and kind of then send a second stage through and a third stage through. So if you kind of trickle these people in, like improve the processes around like code reviews, mentorship, answering questions, etc., and kind of leveled the team up gradually. It was, yeah, no small feat, for sure.

[00:29:35] BW: Yeah. And even today, we have three to five full-time React Native contractors pairing with engineers all day every day, right? Like holding that bar of quality very high includes education, huge part of education.

[00:29:50] JM: Have you guys built any internal tools around managing React Native?

[00:29:56] JT: Yeah. So we have done some basic sort of like scripting and tooling around our like build and release process as far as like when we cut release branches and how we distribute our builds and stuff like that. But, honestly, we've relied on a lot of the tooling that we had already been using before. So like Crashlytics, and bugsnag, and all that kind of good stuff. We are sort of just beginning to invest more into this thing we're calling mobile engine, which is kind of this abstraction that we're creating hopefully to enable over their updates, which will be a whole tool and a whole suite of things. But we haven't fully executed on it yet.

[00:30:42] BW: I guess the only area of custom tooling we did roll out, we did roll out some performance analytics and performance monitoring pretty early on that was like I think a wrapper an extension of some of the tooling that already existed.

[00:30:55] JT: And from our conversations with Facebook is like some of the more impressive or like mature tooling around that stuff. It was such like an obvious failure case for us where it was like everyone has the stigma that React Native can't be performant. Everyone has this stigma that React Native can't like look and feel good. And so because of that, we made really concerted efforts to make sure our performance was pinned as high as possible. We have like a frame per second like widget that's like on every single page as you cruise around the app internally. We invested in a design system extremely early and staffed it with some of our like best engineers to make sure that the visual polish and look of things was really, really high. And

then we enforced kind of design system coverage up around – I think it's like 97% of the app is like powered by the design system, which is super high. So a lot of stuff like that we kind of made sure to land early on.

[00:31:52] JM: What are some of the more hard-earned lessons from managing this migration?

[00:32:01] BW: Yeah, I'd say one of the hard-earned lessons is that the transition from native engineering to React and React Native engineering requires large investment and continuous investment and continually improving that investment. Learning new language, learning a new ecosystem, learning a new set of tools is not easy. Like it's very hard to do. It's a huge mountain to climb. And making sure that you invest strongly and heavily in that to support your people in that journey is critically important.

[00:32:35] JT: Yeah, I guess I would say from the engineering side maybe, this project more than anything I've been a part of in a while just really reaffirmed to me that the technology stack matters less than your engineer's attention to detail when you're setting that bar of quality. I had a suspicion, or I've had this suspicion for quite some time that that's the case. Like I still feel like some of the best UI I've seen are like from Flash way back in like early 2000s. But just people's stubbornness and ability to sweat details on how things should work and why they should work that way and taking the time to implement them is like 80% to 90% percent of the battle and actually getting something that looks and feels good. And it's just about convincing people that they should spend that time. And the technology, almost all technologies that I've seen for writing software these days, will allow you to get there if you're willing to sweat those details.

[00:33:35] JM: Coinbase is such a sensitive application. Like a security hole can be really, really dangerous. How does the sensitivity of an application like Coinbase affect your approach to security. Testing, general infrastructure, code review? Give me your perspective on sensitivity.

[00:34:04] JT: Yeah. I can take this from the technical side. Security here is like 10 out of 10 times. Having worked at other really great companies like Twitter or Medium, security here is on a different level. I think back when we still had a headquarters office, I think during the security onboarding, like our office is secured as though we have like – it's something like 10 or 50 just like semi-trucks worth of gold bars sitting in the office room at any time. And so we take this stuff

and our security team takes it stuff just outrageously seriously, which is awesome. We have a bunch of different things built into our CI process, starting from things like CI, to security reviews. And we have like a dedicated security member on all of our product teams. We have – Gosh! Like we don't Use any third-party anything basically. So, for example, right now we're like trying to – I'll just say it makes it difficult working with a lot of software where we see something and we're like, "We want to use this, but they don't have a self-hosted enterprise version. And so we can't use this." And so we have to reach out and work with teams to see like, "Can we get an enterprise version or a self-hosted version? And like what does that look like?" And then it has to go through a bunch of security audits and everything else. So, yeah, I would say security is very serious. It's very difficult. Everything is on behind a VPN. Everything is authenticated. code reviews are authenticated.

[00:35:42] BW: I'll give some context. So in former lives, I worked on the Bank of America homepage. I worked on user and identity flows at PayPal, and the security of Coinbase makes those look like they're open playgrounds. And I think it's less about technology, less about React Native, or React, or iOS, or Android and more about processes rigor and attention to detail. And I've been blown away with my time at Coinbase, just the rigor of security.

[00:36:11] JM: Yeah, I think I did an interview like three or four years ago with the – I can't remember if it was the head of security at Coinbase. I think his name was Philip. Does that sound familiar?

[00:36:22] JT: Yeah. Yeah. Yeah. Yeah. Yeah.

[00:36:23] JM: I think he's like ex-Palantir? He was a serious guy.

[00:36:29] JT: Yeah, he is a character for sure. There're definitely some characters on the security team. Yeah, Philip is great. Phil. Yeah, very lucky that he's doing a lot of the stuff he's doing.

[00:36:41] JM: This is less about a React Native question. But, I mean, Coinbase is a novel company. It's a generational company. What else is unique to the Coinbase mobile platform?

[00:36:56] JT: So I'll give this question a shot. So my whole career I basically worked on like social media apps, right? So I mentioned before, like Twitter and Medium, then a few startups. And I would say there's more things similar than dissimilar to like startup technical things like this – Oftentimes it feels like we're using GraphQL, but it feels like a REST sort of app. Like we're just playing lists and tables, we allow you to buy things.

There's obviously this very terrifying, looming threat of just like true business impact. So, for example, at Twitter, we would have incidents where you take the site down, or you take whatever down, and we would retro them and learn from them. At Coinbase, like a lot of the way that we quantify things like that is like true like business impact as far as like bottom line revenue. Like how do that like change the way we make or lose money? One of the first things I did at Coinbase was I accidentally broke the way that the UK deposits money into Coinbase, which was horrifying. And was like on literally day two I was trying to fix a bug and just like deleted wire transfers for the UK or whatever.

And so I would say there's just this existential threat that there's something that you could do, which could cost the company or customers like millions of dollars, which feels a lot different than customers not being able to fav a tweet, or like customers not being able to like access a draft. I feel like there's like still some real like consequences there on social apps where it's like, “Oh, like you lost your draft, and you're working on it. And like that really sucks.” But it's a whole another thing I think when like people's livelihoods are attached to it.

Security is the other really, really massive one. And then beyond that, honestly, it feels like building like, yeah, just like an app. I don't know. It's a pretty simple app in a lot of ways, right? Like there's a lot of complicated stuff going on. But if we're strictly talking about the frontend of Coinbase mobile, it's not a very technically challenging app.

[00:39:12] BW: Yeah. I'd say from my perspective, what's unique about Coinbase, the app, is exactly what Jacob described on the front. Like it's a consumer-focused, very accessible, simple looking application. But as you drill down through the layers upon layers upon layers, you're sitting on like numerous nodes of blockchain integration, pushed up through simple API's, pushed up through GraphQL, pushed up to like beautiful UI. And so as a UI developer, it feels like you're iterating on a startup with beautiful API's and simple integrations. But you just keep

going down the stack. It's just much, much more complicated. That's sort of an interesting, interesting place to be sitting.

[00:39:54] JT: Yeah. The one thing I would say is if you start looking beyond just like Coinbase's like core app and you start looking at pro and other things like that, you do have some interesting things coming with like streaming, dealing with streaming API and like chart rendering performance. I know there's a lot of energy and time that goes into showing real-time order books, and candlesticks, and all that sort of stuff. But it isn't something that I've directly worked on. But I do know that there're some really smart folks doing that stuff performant for everyone.

[00:40:26] JM: Given how impactful Coinbase has become, very little is known about how engineering works at Coinbase, as opposed to a company like Facebook or a company like Amazon. Are there any distinctive engineering processes at Coinbase that lend structure or that fuel the innovation at the company?

[00:40:56] JT: Yeah. So I would say we have a handful of different mechanisms. Some of which are like pretty common. So like we have a weekly engineering review where people like bring and present technical design docs to get feedback from a larger group before proceeding on some of the like more challenging architectural processes and projects. We have a handful of like working groups I would say that kind of spin up with people that are very interested in specific areas and improving them. So we have a release working group. We have a product excellence working group. We have folks working on standards. We have a handful of different things there. Organizationally, we're split kind of between product engineers. And then the team that Brent and I are on, which is the client foundations team, which is a more traditional kind of infrastructure client, infrastructure team, where we're doing lower level kind of data layers, or design system, or code sharing, or tooling solutions.

But beyond that, I mean, we're a handful of product pods built on top of that, that are running like pretty standard Agile processes. And we have code reviews or work on GitHub. We use Slack in a huge amount. I'd say one thing that everyone says when they joined Coinbase is we're an extremely documentation-heavy company. So almost all of our thinking and decisions run through documents. And so whether that is we have this mechanism called PPS, which is

like problem proposed solution, or technical design docs, or anything like that. Like basically anything that we do, there's a written paper trail for like what the problem is. Why we think this is the right direction to go and how we got there? And then a date and an author, which makes ramping up on sort of where things are and like kind of pointing to why things are the way they are helps a ton, especially in this like remote world.

[00:42:57] BW: And I can pile on that. From my perspective, I've been at Coinbase a little over a year. Jacob has been at Coinbase a little over three years. And so Jacob has seen a lot more of the evolution. But from my standpoint, like when I landed over a year ago, the thing that hit me right in the face was the strong alignment and decision making culture, right? So we're talking about this React Native transition. We took the entire organization from native to React Native in a little over two years, fully aligned. We're working on unifying our web and mobile architecture with data layers and a UI layer that is fully aligned, like debate, decide, deliver. That level of alignment, decision rigor and follow through is, for me at least who's worked at numerous software companies over the years, is incredible. The fact that you can make a decision and teams will follow that decision and all get on the same boat rowing in the same direction is amazing, and is what, in my opinion, enabled us to make this type of transition, and to continue innovating technically the way we are, because you can make these decisions, and you can drive them across the company and support them fully.

[00:44:17] JM: As we begin to wind down, tell me about the near future. What's on the roadmap for Coinbase mobile? And also any predictions you have about the crypto ecosystem?

[00:44:31] BW: When I think about the technical future of Coinbase application platforms, I think about it in three waves, where wave number one was creating a unified JavaScript platform across web and mobile. And that was achieved when we launched the React Native application on iOS earlier this year. That was sort of the last piece of that part of the wave. The second wave is, now that we have a unified JavaScript, TypeScript platform, how do we unify the UI layer, the data layer the tooling so that whenever you like improve build times by five seconds, it's across hundreds of engineers? Whenever you improve the caching of the data layer, it's across all of our application surfaces. Whenever you do a rebrand and you change the look and feel of the navigation, it's across all of our surfaces. And so that adoption across web and mobile is where we're focused today, sort of enabled by React Native, and having react on

web is that unification. And then three is once you have a unified JavaScript platform and you've unified UI layer and data layer and tooling, then you start focusing on developer effectiveness, developer velocity, and quality, and how you continually iterate and improve those over time and drive key business metrics. And so that's what's on top of my mind is continuing that progressive growth that we started with React Native.

[00:45:56] JT: Yeah. And then I guess I can talk to the crypto. What's going to happen with crypto? I mean, gosh, wouldn't that be great to know? I think what I've learned is like the crypto markets and the cryptosphere is just absolutely unpredictable in every way possible. It's extremely volatile. And yet, it's like so super exciting. I think the only thing I feel like pretty confident in is like we're still just really early. Like we're still seeing kind of people figuring out a lot of this really cool infrastructure. I think DeFi in particular has been really interesting to me. But it's also powered through a Chrome extension, and a lot of ways with MetaMask, which feels like just wrong, fundamentally.

[00:46:45] JM: You guys just build something new though, right? Wasn't there a Coinbase wallet thingy that's supposed to replace a –

[00:46:49] JT: Yeah. Yeah, we do have the wallet team, and they are working on DeFi stuff. But I do still think there's a ways to go there. And our team is working on some cool secret stuff that we can't share yet.

[00:47:02] JM: MetaMask is like the jQuery of the crypto –

[00:47:06] JT: Yeah, a little bit. I mean, yeah. Yeah, that's interesting, where can probably spend a lot of time unpacking. But yeah, I mean, I have to believe that we're just still super early. I think when things work with DeFi, when you're earning like a bunch of interest, like 10%, 15% interest on something that just feels unreal, some of the like initiatives and some of the things that are happening feel just super exciting. NFT's were like kind of blowing up. And now they're kind of – Maybe they like went through the media hype loop already. But I think there's just a lot of interesting potential here, that people are still uncovering and figuring out and we're so early. I know everyone says that, but it truly feels like – In a lot of ways, like were going to be looking back at this time being like, "What are we even doing?"

[00:47:59] BW: Yeah, I have an analogy that really kind of stuck with me, is when I think of like where crypto technology is today, I think back like the origins of Amazon, right? Like, if you look at Amazon, I'm like, "That is the weirdest book selling company I've ever seen, right?" If you look at Amazon today, but you go way back, you're like, "Yeah, they sold books online. They're a bookstore." And so I think about where we are today, it's like we're a bookstore. But eventually, the ecosystem will be like Amazon. Like as the Internet changed and grew, Amazon changed and grew with it. Same thing with crypto, as the technology matures and grows and evolves, like who knows what people are going to build on top of it? But obviously, I'm biased like working at Coinbase and believing in a crypto technology that, "Just wait." And then you're going to have the weirdest crypto trading company in the world is what you're going to about Coinbase and about the ecosystem.

[00:48:56] JM: Yeah, unless Binance is the weirdest company in the world.

[00:48:59] JT: Yeah, Binance is definitely – I mean, they do a lot of stuff where it's like you see it and you're like, "Oh, my God! Why are we doing this? This is so obvious." And then you're like, "Oh, okay. That's super illegal to do in the US." That makes sense. I swear I have that like all the time, where I'm just like, "Oh, my God! We're so behind." And then it's like, "No. No. No. We have this idea. But like the US is not that," and we're like, "Oh, okay. It makes sense."

[00:49:23] JM: Okay. Well, a lot of interesting stuff. We'll sign off for now. Guys, thanks for coming on the show. It's been a real pleasure talking to you.

[00:49:30] JT: Yeah. Thanks for having us. Appreciate it.

[00:49:31] BW: Yeah. Thanks, Jeff.

[END]